Question-driven Text Summarization with Extractive-Abstractive Frameworks

Mahsa Abazari Kia

A thesis submitted for the degree of
Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

September 2022

Abstract

Automatic Text Summarisation (ATS) is becoming increasingly important due to the exponential growth of textual content on the Internet. The primary goal of an ATS system is to generate a condensed version of the key aspects in the input document while minimizing redundancy. ATS approaches are extractive, abstractive, or hybrid. The extractive approach selects the most important sentences in the input document(s) and then concatenates them to form the summary. The abstractive approach represents the input document(s) in an intermediate form and then constructs the summary using different sentences than the originals. The hybrid approach combines both the extractive and abstractive approaches. The query-based ATS selects the information that is most relevant to the initial search query. Question-driven ATS is a technique to produce concise and informative answers to specific questions using a document collection.

In this thesis, a novel hybrid framework is proposed for question-driven ATS taking advantage of extractive and abstractive summarisation mechanisms. The framework consists of complementary modules that work together to generate an effective summary: (1) discovering appropriate non-redundant sentences as plausible answers using a multi-hop question answering system based on a Convolutional Neural Network (CNN), multi-head attention mechanism and reasoning process; and (2) a novel paraphrasing Generative Adversarial Network (GAN) model based on transformers rewrites the extracted sentences in an abstractive setup. In addition, a fusing mechanism is proposed for compressing the sentence pairs selected by a next sentence prediction model in the paraphrased summary. Extensive experiments on various datasets are performed, and the results show the model can outperform many question-driven and query-based baseline methods. The proposed model is adaptable to generate summaries for the questions in the closed domain and open domain. An online summariser demo is designed based on the proposed model for the industry use to process the technical text.

Acknowledgements

This work was supported in part by the Computer Science and Electronic Engineering Department, University of Essex, and in part by BT Group/Openreach. My Ph.D. study at the University of Essex is wonderful, cheerful, and unforgettable, and I want to thank everyone who has been with me during this journey. Furthermore, I want to express my appreciation to those who have been especially helpful. I want to thank my supervisors, Dr. Shoaib Jameel, Dr. Jon Chamberlain, Aygul Garifullina, Dr. Mathias Kern, and Professor Ansgar Scherp who have always supported me with their knowledgeable guidance and passionate encouragement. It will always be an honor and fortune for me to be their student. Finally, I thank everyone at the School of Computer Science and Electronic Engineering at the university of Essex and BT for providing a friendly work environment and good vibes, and my parents for all of their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 History of Text Summarisation

Automatic Text summarisation, the reduction of a text to its essential content, is a very complex problem that, despite the progress in the area thus far, poses many challenges to the scientific community. Given the rapid expansion of textual material online and the requirement to swiftly evaluate the contents of text collections, it is also a vital application in today's information society [11]. In the late 1950s [12], there was a particular interest in the automation of summary for the development of abstracts of technical documentation, which attracted the attention of the scientific community to automatic summarisation. The interest in the area declined for a few years until Artificial Intelligence started to show interest in the topic [13].

It has long been assumed that summarisation requires comprehension of the source text, which necessitates calculating a text's explicit (semantic) representation to recognize its essential content. Therefore, text summarisation has become an appealing application for evaluating the intelligence of artificial systems. Due to the complexities of the task, however, interest in this approach to text summarisation gradually subsided, and text comprehension developed an open area of research. In the 1990s, there was a resurgence of interest in summarisation due to the organization of several scientific conferences [14]. This interest peaked in the year 2000 with the introduction of evaluation programs such as the Document Understanding Conferences (DUC) [15] and the Text Analysis Conferences (TAC) [16] in the United States. How to identify the fundamental content of a document and how to compress the selected content are two fundamental challenges in text summarisation [17, 13]. Text summarisation research has focused predominantly on the output, the summary, and less on the cognitive foundations of text comprehension and production of human summarisation. A

greater comprehension of the cognitive basis of the task might alleviate some of the shortcomings of existing systems. However, formalizing the content of open domain documents is still a matter of research. Therefore, most systems are based on a selection of sentences from the set of original documents.

Concerning the transformation of the original text content into a summary, there are two main types of summaries: an extractive summary, which is a collection of sentences from the input document, and an abstractive summary (i.e. an abstract), a summary in which some of its material is not present in the input document [18]. Summaries can also be characterized as either indicative or informative, depending on whether they are designed to alert or inform. Early research in summarisation focused on the summarisation of single documents (i.e. single document summarisation). However, in the modern Web environment, several systems concentrate on the summarisation of multiple related documents (i.e. multi-document summarisation). Most summarisation methods currently target the production of extracts due to the challenges associated with the automatic generation of well-formed texts in arbitrary domains. It is generally agreed that there are a variety of elements that affect the content selection from the source document and the type of output to make [19], for example, factors such as the audience (e.g. expert versus non-expert reader) surely impact the information to select. Several publications and books [18, 20, 21] provide more overviews of text summarising systems and methodologies.

Text summarisation has become necessary because of the rise in online publication, internet users, and the rapid development of electronic government (e-government). Due to the rapid development of information and communication technologies, a huge number of electronic documents are available online, making it difficult for users to locate pertinent information. In addition, the Internet has made available vast amounts of text on a range of subjects. This accounts for the redundancy in the texts available online. Users get so exhausted reading a large amount of texts that they may skip reading many important and interesting documents. Consequently, a comprehensive text summarisation system is necessary for this generation. These systems may condense information from a variety of documents into a concise, readable summary [22, 23]. Huang et al. [24] address four primary objectives: information coverage, information importance, information redundancy, and text coherence.

Summarisation is a challenging task even for people, and many professionals in the industry write summaries as part of their job. This challenge arises from the fact that summary involves a number of complicated Natural Language Understanding (NLU) components, in-

cluding information selection, assessment, aggregation, and rearrangement, followed by information compression, generalization, and/or paraphrasing. Additionally, this must happen at several (potentially abstract) levels, including those of sentences, paragraphs, sections, and papers. At present time, modern summarisation systems are still far from performing and speaking fluently as humans do. However, they might still provide useful summaries that can be time-saving [25]. Due to the subjective nature of this task, where various summaries might be regarded as appropriate, evaluating the quality of output summaries is a crucial problem. Summarisation has resulted in a number of practical applications. Early examples include the summarisation tool in Microsoft Word [26], systems that summarise voicemail messages for users, to help them determine the priority of a call [27], as well as systems that can provide a digest of user forums [28].

1.2 Question-driven Text Summarisation

The goal of this research is to summarise the source document with respect to a specific question. Distilling question-relevant information from a text document reduces reading time and accelerates the process of research about a specific question. A question-driven summary needs to satisfy three goals, answerability, understandability, and persuasiveness. The question-driven summary constitutes one answer sentence and reasoning sentences. The answer sentence indicates the answer to the question (answerability) and the reasoning sentences provide more information about the detected answer and the question to make the summary more understandable and persuasive for the reader. The answer sentence will be justified and explained by the reasoning sentences (understandability and persuasiveness).

There have been several attempts, in recent years, to develop methods for question-driven automatic text summarisation [29, 30, 31, 32, 33]. A summary based on a question is supposed to contain a reliable answer for that question and some further details that justify and explain the answer, which is the key challenge in this work. Query-based document summarisation aims to produce a compact and fluent summary of a given document that answers or is relevant to the search query that leads to the document [34]. An example of query-based and question-based text summarisation is provided in Table 1.1, the bold text shows the relevant text to the proposed query and question and the gold summary is the abstractive summary generated by humans. As it is shown in this example the query-based summary has summarised the text given the query “foods for lower blood sugar”, it contains all the information

Table 1.1: Query-based and question-driven text summarisation example

Text: ... **According to Powers, your eating plan should focus on the type and amount of carbohydrates you eat throughout the day. Choose low-carb vegetables such as mushrooms, onions, eggplant, tomatoes, Brussels sprouts, and zucchini, as well as low-carb squashes. To add flavor and texture to a meal, serve them with low-fat souces, hummus, guacamole, and salsa, or roasted with herbs and spices like rosemary, cayenne pepper, and garlic. Sweet potatoes, when combined with other meals, can successfully slow food digestion, increase satiety, and moderate blood sugar swings. Thus, individuals who are hyperglycemic can have some sweet potatoes, which will not only not elevate blood sugar but will also aid in blood sugar control.** ...
Query: foods for lower blood sugar
Query-based Gold Summary: The amount and type of carbs you put in your diet throughout the day should be seriously considered. Low-carb and tasty veggies, like mushrooms, onions, eggplant, tomatoes, Brussels sprouts, and low-carb squashes, like zucchini with dips such as low-fat dressings, hummus, guacamole, and salsa, or roasted with different seasonings such as rosemary, cayenne pepper, or garlic could be included to the meal for better flavor and texture. Sweet potatoes can help to slow down food digestion, increase satiety, and stabilise blood sugar levels which not only do not raise blood sugar but also help to control blood sugar.
Question: How sweet potatoes helps people with hyperglycemic?
Question-driven Gold Summary: Sweet potatoes can help to slow down food digestion, increase satiety, and stabilise blood sugar levels. As a result, persons with hyperglycemia can eat sweet potatoes, which not only do not raise blood sugar but also help to control blood sugar.

about the diet for hyperglycemic people, but the question-driven summary is shorter and only contains specific information, the answer and its explanation, to the question.

Table 1.1 shows an example of query-based and question-driven summaries. Many related studies focus on query-based summarisation to summarise the query-related content from the source document [35, 36, 37, 38] . However, these approaches are not suitable for tackling question-driven summarisation problems in Question Answering (QA) scenarios, whereas the query-based summarisation process is commonly based on semantic relevance measurement but for question-driven summarisation, answer detection and the reasoning on the document regarding the detected answer is needed as it is shown in Table 1.1.

Currently, question-driven text summarisation and answer summarisation are ineffective due to using recurrent neural networks and perform weakly in answerability and persuasiveness. Recurrent neural networks have a number of disadvantages that narrows their ability to solve more complex problems (e.g. Question-driven text summarization). Vanishing and exploding gradients, slow training, limited memory, and high data dependency are the drawbacks in recurrent neural networks causing deficiency in the performance for variant Natural Language Processing (NLP) tasks.

Besides, compared to extractive summarisation, the content generated by abstractive

methods often suffers issues such as poor readability, data redundancy, and large semantic deviations from the source. Most of the recent abstractive summarisation models are based on sequence-to-sequence (seq2seq) neural networks [39, 40, 41, 42, 43, 44]. They are made up of encoders to understand input sequence and decoders to generate output sequence. How- ever, there are four key problems with using seq2seq neural networks to generate reasonable text: (1) out-of-vocabulary (OOV) problem (2) generating a particular word or phrase repeatedly which brings in redundancies, (3) exposure bias at test time, and (4) non-optimized learning for evaluation metrics used by models in fields such as text summarisation and machine translation. As a result, they cannot generate appropriate abstractive summaries since they cannot convey the semantics of the document [45, 46]. Abstractive summarisation needs advanced natural language techniques for interpreting and understanding the text to reproduce the important material in a new way. While the extractive summaries may contain repeated words, a high frequency of certain words, and redundancy in some sentences [7].

1.3 Applications for Industry

We are currently witnessing an exponential increase of data that emanates from varied sources such as different types of records in companies. It is very challenging to process this sparse, noisy, and domain-specific data. For instance, BT, a technology company in the UK and the Ph.D. project sponsor, has a significant workforce of field engineers, desk-based agents, and customer support services who generate, collect and manage large volumes of temporally organized unstructured and semi-structured information every day. The problem that they face is tracking the progress of the order and finding out about the problems causing delays.

BT require computational models that could effectively and efficiently distill relevant information for the technical and non-technical users at BT from various technical order record documents which are very noisy and follow no structural pattern. To this end, what would be useful is to automatically summarise and derive meaningful information in the form of answers to questions from this vast source of distributed occurring data.

The information about orders at BT is in structured and free text format which is captured by and stored in different internal systems. These are being used by teams handling the orders. This volume of text about orders is an invaluable source of information that needs to be summarised in a way containing the information most relevant to:

- Why is the order not complete?

- What does the order require?
- Where has the order been passed to?

It will help the desk agents to have a clear picture of the latest status of the order journey at the point in time t, instead of checking the order information from several places for the time that a customer calls in to find out about the progress of their order. In other words, generating summaries at the point in time t would help humans comprehend the text content effectively and efficiently. BT orders are structured in such a way that there is an update at regular intervals of time. These updates are required to be input to the summariser as data over time.

The current question-driven text summarisation approaches utilized WikiHow [47], PubMedQA [48], and MEDIQA [49] which include a question, an article, and an abstractive answer which summarises the context corresponding to the question. Only one question and a plain text document are the inputs to the summarisation system and there is no timestamped attribute or data over time. Here, an example for BT (Table 1.3) and MEDIQA (Table 1.2) inputs and the desired summaries are provided.

The different nature of BT orders' text (notes) and public dataset, lack of training dataset and gold summaries for BT orders require a model that could be fine-tuned with small training datasets. Also, the nonexistence of the gold summaries causes the evaluation to be challenging for the BT domain since a human evaluation method should be proposed for the generated orders' progress summaries.

The progress summaries for BT orders should address three questions despite the public question-driven text summarisation that is based on only one question. Recognizing the DateTime attribute in BT orders text documents and finding the appropriate piece of text based on its DateTime attribute is another challenge for the BT domain which does not exist in public datasets. For the BT domain, the recent text pieces (regarding their date-time) have priority over the older text pieces for generating the question-driven text summarisation.

1.4 Thesis Contribution

To overcome the problems and challenges mentioned in the section 1.1 above and obtain a reliable summary of the text document, a novel two-stage, hybrid extractive and abstractive summarisation approach has been proposed which combines the advantages of the two methods. Firstly, the extractive model selects the answer sentence and its supporting sentences,

which provide details or explanations for the answer sentence. After obtaining the question-driven extractive summary, a novel abstractive model transforms the extractive summary into an abstractive summary. The advantage of doing this is that the extractive phase helps reduce the amount of redundant information from the data, which helps improve the effectiveness of the abstractive summariser. A multi-hop QA system based on a Machine Reading Comprehension (MRC) model and a reasoning process has been proposed for selecting the answer sentence and supporting sentences to construct the question-driven extractive summary. The abstractive model consumes the generated extractive summary and transforms it into an abstractive by utilizing a novel paraphraser model and sentence fusion mechanism. In this work, a question-driven abstractive summarisation model is proposed and described in detail, and its main contributions can be summarised as follows:

1. Extractive: A novel multi-hop QA system comprising a hierarchical CNN attention network MRC model and a reasoning process for generating the question-driven extractive summary.

    - The CNN-multi-head attention model captures the relevance between the question and context sentences at different levels of granularity.
    - A candidate answer identifier module and a question expansion modules are complementary components in the MRC model, designed for selecting question-relevant sentences and question rewriting, respectively.
    - The proposed MRC model is compared against state-of-the-art comparative methods for closed-domains and open-domain QA and MRC tasks.
    - A novel reasoning approach is proposed for analysing the document regarding the detected answer sentence and searching for relevant supporting sentences.

2. Abstractive: A novel paraphrase framework based on GANs, Q-learning, and transformers is proposed to rewrite the generated extractive summary from the previous stage and then a sentence fusion mechanism is proposed for fusing relevant sentences and generating an abstractive summary.

    - The paraphrase generation model is designed based on transformers architectures with GAN and Q-stepwise evaluation to regenerate the extractive summary.
    - A sentence fusion mechanism is developed based on the next sentence prediction and the Pointer Generator (PG) network to produce shorter high-quality abstractive summaries.

- The next sentence prediction is utilised for detecting the sentence pairs (relevant sentences) and singletons to be consumed by the PG network.
- The PG network gets a sentence pair and generates a fused sentence.
- After fusing the sentence pairs, the singletons and fused sentences construct the question-driven abstractive summary.

1.5 Research Questions

The overall research questions of this thesis include:

1. What steps are humans taking for question-driven text summarisation, and what is an effective way to simulate these steps?
2. How proposing a hybrid extractive and abstractive approach could assist with the efficient automatic generation of question-driven extractive and abstractive summaries?
3. What strategies and techniques can be utilized for transforming extractive summaries into abstractive in order to reduce the complexity in abstractive summarisation?
4. How will the training procedure's design affect the summarisation model's adaptability to different domains ?
5. How to effectively track the progress of orders at BT with a question-driven text summarization approach?

Table 1.2: An extractive question-driven text summarisation example from MEDIQA dataset.

Text: Peppermint Oil Background: The herb peppermint, a natural cross between two types of mint (water mint and spearmint), grows throughout Europe and North America. Both peppermint leaves and the essential oil from peppermint have been used for health purposes. (Essential oils are very concentrated oils containing substances that give a plant its characteristic odor or flavor.) Peppermint is a common flavoring agent in foods, and peppermint oil is used to create a pleasant fragrance in soaps and cosmetics.Mint has been used for health purposes for several thousand years. It is mentioned in records from ancient Greece, Rome, and Egypt. However, peppermint was not recognized as a distinct kind of mint until the 1700s.Today, peppermint is used as a dietary supplement for irritable bowel syndrome (IBS), other digestive problems, the common cold, headaches, and other conditions. Peppermint oil is also used topically (applied to the skin) for headache, muscle aches, itching, and other problems. Peppermint leaf is available in teas, capsules, and as a liquid extract. Peppermint oil is available as liquid solutions and in capsules, including enteric-coated capsules. How Much Do We Know? A small amount of research has been conducted on peppermint oil, primarily focusing on IBS. Very little research has been done on peppermint leaf. What Have We Learned? Peppermint oil has been studied most extensively for IBS. Results from several studies indicate that peppermint oil in enteric-coated capsules may improve IBS symptoms.A few studies have indicated that peppermint oil, in combination with caraway oil, may help relieve indigestion, but this evidence is preliminary and the product that was tested is not available in the United States.Peppermint oil has been used topically for tension headaches and a limited amount of evidence suggests that it might be helpful for this purpose.There's not enough evidence to allow any conclusions to be reached about whether peppermint oil is helpful for nausea, the common cold, or other conditions.There's not enough evidence to show whether peppermint leaf is helpful for any condition. What Do We Know About Safety? Peppermint oil appears to be safe when taken orally (by mouth) in the doses commonly used. Excessive doses of peppermint oil can be toxic.Possible side effects of peppermint oil include allergic reactions and heartburn. Capsules containing peppermint oil are often enteric-coated to reduce the likelihood of heartburn. If enteric-coated peppermint oil capsules are taken at the same time as antacids, the coating can break down too quickly. Like other essential oils, peppermint oil is highly concentrated. When the undiluted essential oil is used for health purposes, only a few drops are used. Side effects of applying peppermint oil to the skin can include skin rashes and irritation. Peppermint oil should not be applied to the face or chest of infants or young children because serious side effects may occur if they inhale the menthol in the oil.No harmful effects of peppermint leaf tea have been reported. However, the long-term safety of consuming large amounts of peppermint leaf is unknown.
Question: Is it safe to add peppermint essential oil in mouth wash?
Question-driven Extractive Summary: Peppermint oil appears to be safe when taken orally (by mouth) in the doses commonly used. Excessive doses of peppermint oil can be toxic. Possible side effects of peppermint oil include allergic reactions and heartburn. Capsules containing peppermint oil are often enteric-coated to reduce the likelihood of heartburn. If enteric-coated peppermint oil capsules are taken at the same time as antacids, the coating can break down too quickly. Like other essential oils, peppermint oil is highly concentrated. When the undiluted essential oil is used for health purposes, only a few drops are used. Side effects of applying peppermint oil to the skin can include skin rashes and irritation. Peppermint oil should not be applied to the face or chest of infants or young children because serious side effects may occur if they inhale the menthol in the oil. No harmful effects of peppermint leaf tea have been reported. However, the long-term safety of consuming large amounts of peppermint leaf is unknown.

Table 1.3: An extractive question-driven text summarisation for an order at BT

Text: 03/10/2019 13:07:00 - <A-Name> Complex planner required. No plant to unit 9a. Nearest useable DP is approx 120m away. 21/10/2019 11:12:00 - <B-Name> I cannot complete this task because the end customer or their representative is unaware of the order. Please see additional information On side representative unaware of order. Premises already has 2 working lines and tagged D side in PCP. Business owner not present away on holiday. Cp to contact customer next week to see if they want this line. 06/11/2019 16:00:00 - <C-Name> I cannot complete this task because I donot have the skills required to complete the task and could not obtain assistance on the day. I am passing to an engineer with track and locate equipment to complete the order. Please see additional information EU's property has a buried lead in. No sign of BT socket or a POI in the property. Passing to TandL to trace lead in.
Questions:Why is the order not complete?, What does the order require?, Where has the order been passed to?
Question-driven Extractive Summary: The update that our engineer <C-Name> has reported on 06/11/2019 16:00:00 about your order: *** I can not complete this task because I donot have the skills required to complete the task and could not obtain assistance on the day . I am passing to an engineer with track and locate equipment to complete the order . planner required . [03/10/2019 13:07:00] *** Please Continue to review fault tracker for further updates.

1.6 Publications and Presentations

- "Text summarisation of Customer Order Journey Data" has been presented in Tommy Flowers Network 2020, a poster indicating a brief overview of this Ph.D. project from BT's perspective.
- "Automated Multi-document Text summarisation from Heterogeneous Data Sources" is published at ECIR 2021 (doctoral consortium), presenting the Ph.D. proposal.
- "Adaptable Closed-Domain Question Answering Using Contextualized CNN-Attention Models and Question Expansion" is one of the published papers based on the proposed approach. In this paper, the novel versatile reading comprehension style approach for closed-domain QA is presented, which is thoroughly described in chapter 4. Paper Link DOI: 10.1109/ACCESS.2022.3170466
- "Question-Driven Text summarisation Using an Extractive-Abstractive Framework" is under review in the Computational Intelligence journal. This paper presented the extractive stage (chapter 4) and paraphrase generation (section 5.2.1) for question-driven text summarisation.
- The online demo described in chapter 6 is presented to the Openreach people and deploying team. The evaluation procedure has been discussed and agreed upon, and now the demo is in the deployment procedure.
- "Using NLP to understand complex technical notes - a telecoms case study" is presented in AI-2022 Forty-second SGAI International Conference on Artificial Intelligence Cambridge Workshops, England 13-15 December 2022. This speech was based on chapter 6.

Chapter 2

Background and Literature Review

This chapter begins by introducing relevant background concepts in NLP and deep learning. Following that, the text generation, automatic text summarisation, and Question Answering related works and approaches and their strengths and weaknesses are presented. In order to propose an efficient approach for question-driven text summarisation, analysing the drawbacks of the existing text summarisation and Question Answering approaches could be beneficial which is discussed in this chapter and chapter 3.

2.1 Neural Networks

An artificial neural network (ANN) is a paradigm for information processing that takes its principles by how information is processed by biological nervous systems, such as the brain. Multiple layers of basic processing units known as neurons comprise an ANN. The neuron carries out two tasks: collecting inputs and producing output. The application of ANN provides an overview of the theory, learning rules, and applications of the most prominent neural network models, definitions, and computational styles [50].

2.1.1 Recurrent Neural Networks (RNNs)

The design of RNN encoder-decoders is based on the sequence-to-sequence paradigm. The sequence-to-sequence model translates the input sequence to a similar sequence of characters, words, or phrases in the neural network. Several NLP applications, such as machine translation and text summarisation, employ this approach. As indicated in Figure 2.1, the input sequence for text summarisation is the document to be summarised, and the output

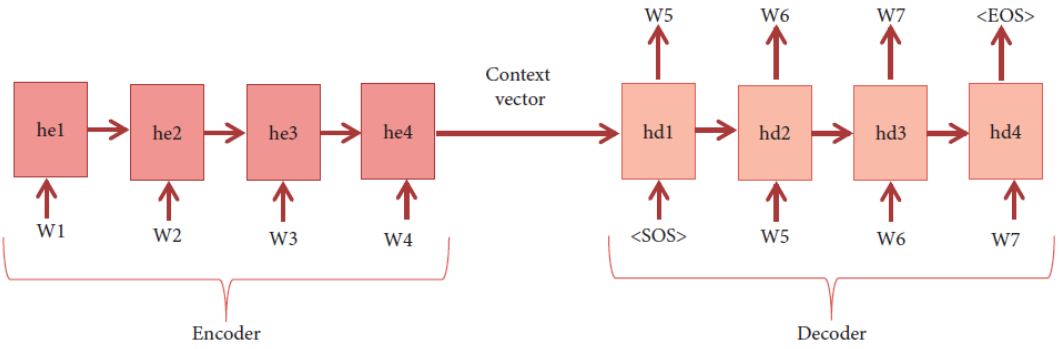

Figure 2.1: Sequence-to-sequence; the last hidden state of the encoder is fed as input to the decoder with the symbol EOS.

sequence is the summary [51, 52]. An RNN is a deep learning model used to sequentially process data in which the input of one state is dependent on the output of the previous state [53, 54]. For instance, the meaning of a word in a phrase is tightly tied to the meaning of the preceding words. An RNN comprises a set of hidden states that the neural network has learned. An RNN may be composed of several layers of hidden states, where the states and levels acquire distinct features. The final state of each layer indicates the layer's total inputs since it aggregates the values of all preceding stages [1]. For instance, the first layer and its state can be used for part-of-speech tagging, whilst the second layer learns to construct phrases. In text summarisation, the RNN's input is the embedding of words, phrases, or sentences, and its output is the embedding of the summary's words [1].

At specific hidden states on the encoder side of the RNN encoder-decoder model, the vector representation of the current input word and the output of all previous hidden states are merged and sent to the next hidden state. As seen in Figure 2.1, the vector representation of the word $W3$ and the outputs of the hidden states $he1$ and $he2$ are merged and provided as input to the hidden state $he3$. After all the words of the input string have been supplied to the encoder, the output created from the final hidden state of the encoder is fed to the decoder as a vector known as the context vector [51]. In addition to the context vector, which is sent to the decoder's initial hidden state, the start-of-sequence symbol <SOS> is provided to construct the first word of the summary from the headline (assume $W5$, as shown in Figure 2.1). In this instance, $W5$ is given to the following decoder hidden state as its input. Each created word is supplied as an input to the next decoder's hidden state in order to generate the subsequent summary word. The final produced word is the sequence-ending symbol

<EOS>. Each output from the decoder will be transformed into a distributed representation prior to being delivered to the softmax layer and attention mechanism [51] to build the next summary.

2.1.2 Bidirectional RNN

Bidirectional RNN comprises of both forward and backward RNNs. After reading the input sequence from left to right, forward RNNs create a list of hidden states. After reading the input sequence from right to left, however, backward RNNs create a sequence of hidden states. The input sequence is represented by concatenating the forward and backward RNNs [55]. Therefore, the representation of each word is contingent upon the representation of the preceding (past) and subsequent (future) words. In this instance, the context will consist of the words to the left and right of the current word [56]. Using a bidirectional RNN boosts performance [57]. For example, for the following input text "Sara ate a delicious pizza at dinner tonight" in this case, assume that the goal is to predict the representation of the word "dinner", using bidirectional RNN and the forward neural network represent "Sara ate a delicious pizza at" while the backward neural network represents "tonight". Considering the word "tonight" when representing the word "dinner" provides better results. Using the bidirectional RNN at the decoder size, on the other hand, reduces the likelihood of an incorrect prediction. This is because the unidirectional RNN only analyses the prior prediction and only considers the past when reasoning. Therefore, if there is a mistake in a prior forecast, the error will accrue in all subsequent predictions, and the bidirectional RNN [58] can tackle this issue.

2.1.3 Gated Recurrent Neural Networks

Gated RNNs are used to alleviate the problem of vanishing gradients that arises while training an RNN on a long sequence. This problem may be addressed by allowing the gradients to back-propagate along a linear path with weighted and biased gates. Controlling and modifying the amount of information that passes between hidden states is a function of gates. During training, the gate weights and biases are modified. The most prevalent gated RNNs are Long Short-Term Memory (LSTM) [59] and Gated Recurrent Unit (GRU) [60], both of which are RNN variations.

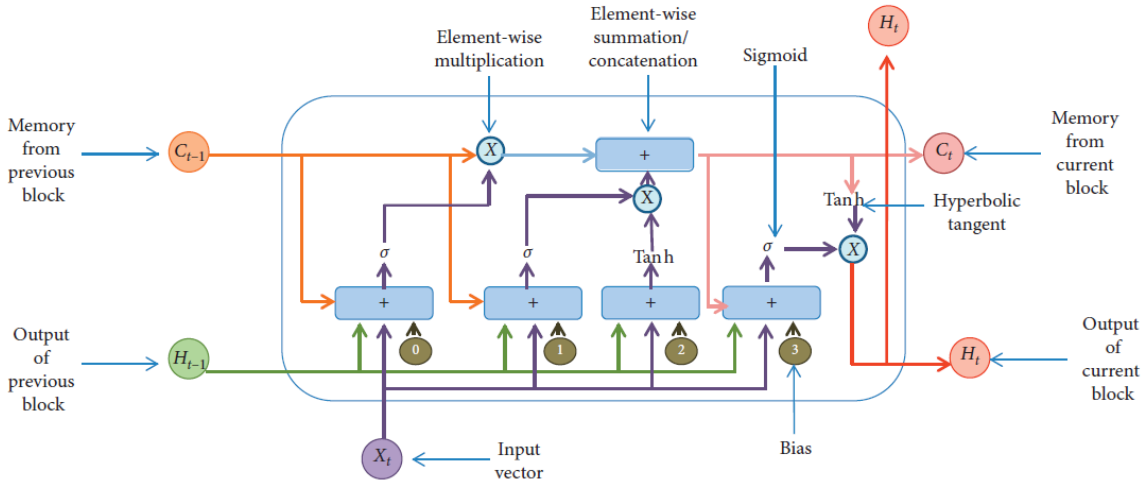

Figure 2.2: LSTM unit architecture [1]

LSTM

The LSTM architecture's repeating unit consists of input/read, memory/update, forget, and output gates [1, 58], although the chaining structure is identical to that of an RNN. Due to the fact that the four gates communicate information with one another, information can flow in loops for an extended length of time. The four gates of each LSTM unit are depicted in Figures 2.2 and 2.3.

- Input Gate: In the first timestep, the input is a vector that is initialised at random, but in following steps, the input is the current step's output (memory cell content). In all circumstances, the input is multiplied element-by-element with the output of the forget gate. The result of the multiplication is added to the current output of the memory gate.

- Forget Gate: A forget gate is a single-layer neural network with a sigmoid activation function. The value of the sigmoid function determines whether the previous state's information should be forgotten or remembered. If the sigmoid value is 1, the prior state will be remembered, but if it is 0, it will be forgotten. In language modelling, for instance, the forget gate remembers the subject's gender to output the correct pronouns until it encounters a new subject. The forget gate accepts four inputs: the output of the previous block, the input vector, the information remembered from the previous block, and the bias.

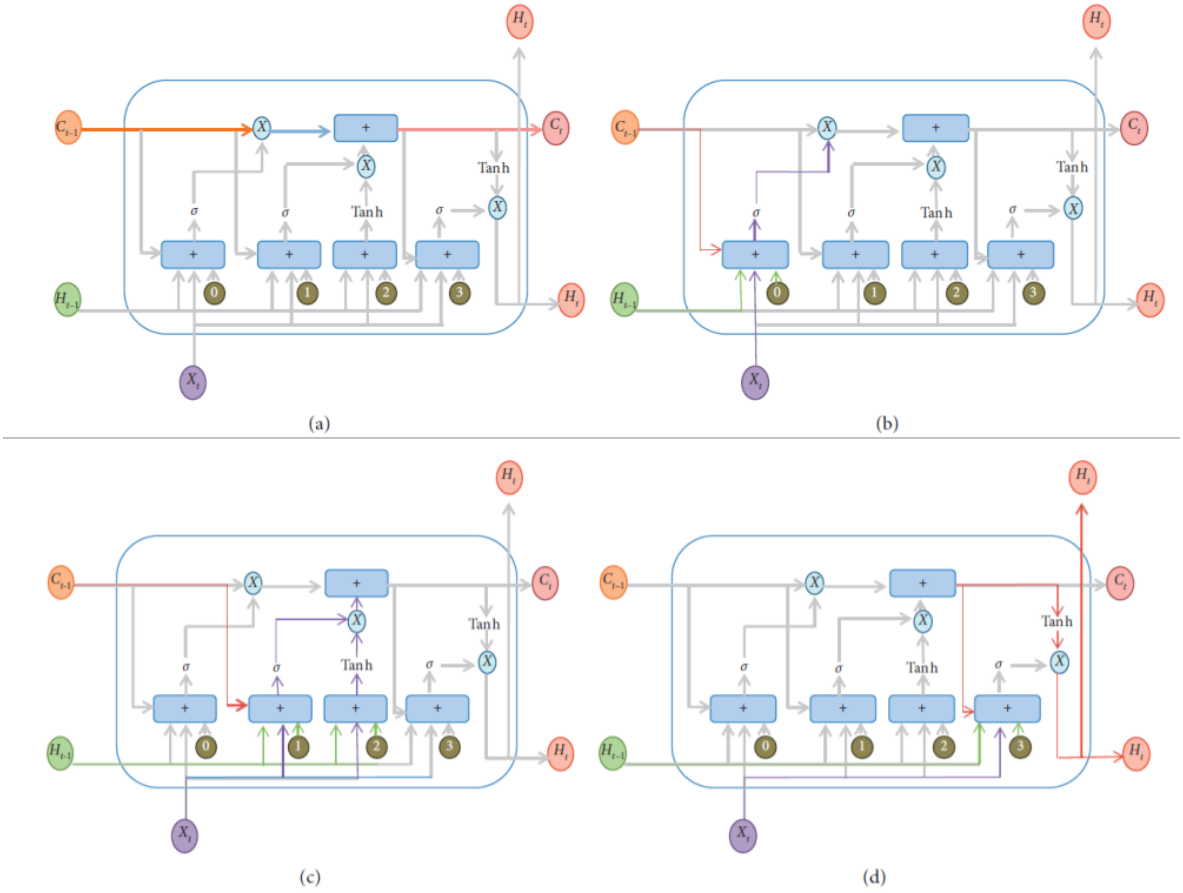

Figure 2.3: LSTM unit gates [1]: (a) input gate; (b) forget gate; (c) memory gate; (d) output gate.

- Memory Gate: Memory gate regulates the impact of remembered information on the new information. There are two neural networks within the memory gate. The first network has the same structure as the forget gate but a different bias, whereas the second neural network with a $tanh$ activation function is used to produce new information. The new information is created by combining the old information with the result of element-wise multiplication of the output of the two memory gate neural networks.

- Output Gate: The output gates determine amount of new information passed to the subsequent LSTM unit. As inputs, the output gate is a neural network with a sigmoid activation function that takes into account the input vector, the prior hidden state, the incoming information, and the bias. The output of the current block is the product of the output of the sigmoid function multiplied by the tanh of the new information.

The formulations for The LSTM units input gate, forget gate, memory gate, and output gate are as follow. The hidden state h_t given input x'_t at time t is computed as follows:

$$i_t = \sigma\left(W_i\left[x'_t, h_{t-1}\right] + b_i\right) \tag{2.1}$$

$$f_t = \sigma\left(W_f\left[x'_t, h_{t-1}\right] + b_f\right) \tag{2.2}$$

$$\tilde{C}_t = \tanh\left(W_c\left[x'_t, h_{t-1}\right] + b_c\right) \tag{2.3}$$

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \tag{2.4}$$

$$O_t = \sigma\left(W_o\left[x'_t, h_{t-1}\right] + b_0\right) \tag{2.5}$$

$$h_t = O_t * \tanh\left(C_t\right) \tag{2.6}$$

Among them, f_t ; i_t ; O_t are states of forget gate, input gate and output gate at time t; respectively. W_f ;W_i; W_o and W_c are weight matrices corresponding to each components, and b_f ; b_i; b_o and b_c are bias vectors corresponding to each components. $\tilde{C}_t$ is candidate state value at time t of memory cell and is calculated by $tanh$ function. C_t is memory cell state at time t. σ is $sigmoid$ function. For BiLSTM, the hidden layer outputs $\vec{h}_t$ and $\overleftarrow{h_t}$ of forward LSTM unit and backward LSTM unit are calculated by formulas above. Then $\vec{h}_t$ and $\overleftarrow{h_t}$ are spliced together to get hidden layer output ht of BiLSTM at time t. The splicing formula is shown below:

$$h_t = \left[\vec{h}_t, \overleftarrow{h_t}\right] \tag{2.7}$$

Finally, the output of BiLSTM is $h = \{h_1, h_2, \ldots, h_n\}$.

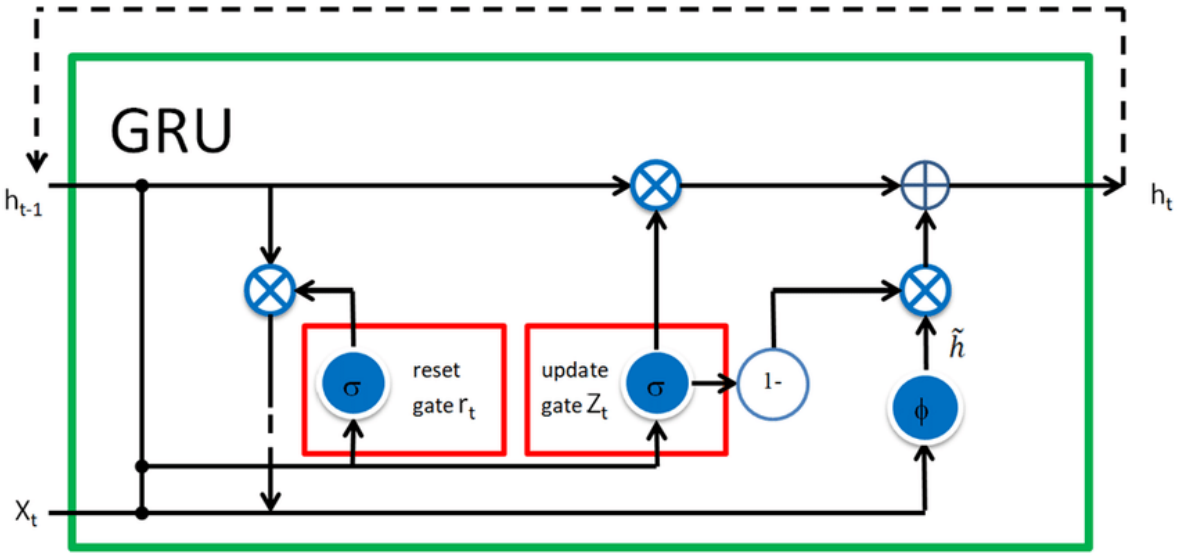

Figure 2.4: The diagram of a GRU cell [2]

GRU

A GRU is an LSTM with only two gates, a reset gate and an update gate, and without explicit memory (Figure 2.4). When all the reset gate elements approach zero, the prior hidden state information is discarded and only the input vector influences the candidate hidden state. In this instance, the update gate also serves as the forget gate. LSTM and GRU are frequently used for abstractive summarisation because LSTM contains a memory unit that gives additional control; nevertheless, the calculation time of the GRU is decreased [61]. In addition, whereas LSTM parameters are easier to modify, GRU requires less training time [52].

2.1.4 CNN

CNNs are one of the most used computer vision methods. These were the driving forces behind picture categorization and the majority of computer vision systems [62], from Facebook's automatic photo tagging to autonomous vehicles. Recent research on CNNs used for NLP tasks has shown intriguing outcomes [63]. CNN comprises of a convolutional layer and a pooling layer for text processing that captures significant features. Convolution is the central layer of CNN which is made up of a set of convolution kernels (filters). For convolution

computation, the convolution kernel and the local window of input data are employed.

$$c_i = f\left(F \cdot h_{i:i+l-1} + b\right) \tag{2.8}$$

Among them, F represents convolution kernel. $F \in R^{l \times d'}$; where l is height of convolution kernel, d' is width of convolution kernel, and its size is output feature dimension of BiLSTM unit. b represents bias parameter, f is $RELU$ nonlinear function, $h_{i:i+l-1}$ represents BiLSTM hidden layer output vector from i to $i + l - 1$; . is convolution operation, and c_i is result of convolution calculation. The convolution window slides down with step size of 1 to obtain local feature vector $C = (C_1, C_2, \ldots, C_{n-l+1})$. Multiple convolution kernels are used to perform convolution operations, the number is N; so N local feature vectors are obtained. Pooling layer's goal is to sample the output of convolution, lower the size of the convolution vector, and prevent overfitting. Maximum pooling and average pooling are two types of the pooling method which maintain the key information of text. All sampled feature values are combined into $M = (M_1, M_2, \ldots, M_N)$ as output of the CNN.

$$M_i = \max\left(C_i\right) \tag{2.9}$$

In contrast to computer vision problems, where image pixels are used as input, NLP tasks employ phrases, words, or often characters, depending on the classification of the problem. Therefore, each row is a vector representing a word. Typically, word embeddings or one-hot vectors are used to index a word inside a vocabulary. For 10 sentences utilising 100-dimensional embedding, the input matrix will be 10 * 100. In computer vision, filters pass over limited regions of an image; in NLP, filters pass over a whole column of words (matrix). The width of the input matrix and the width of the filters are same. As an example in Figure 2.5, CNNs are used to classify sentences, which illustrates how CNNs are applied to text. Each of the three filter regions, which are numbered 2, 3, and 4, has two filters. After performing the convolution operation on the sentence matrix and obtaining variable-length feature maps, the max-pooling algorithm is applied to each map, producing the largest number from each feature map. From these six maps, univariate feature vectors are formed, which are then concatenated to form a single feature vector for the penultimate layer. Finally, the softmax layer gets this feature vector as an input, and then sentences are classified, assuming a binary classification with two potential outputs [64].

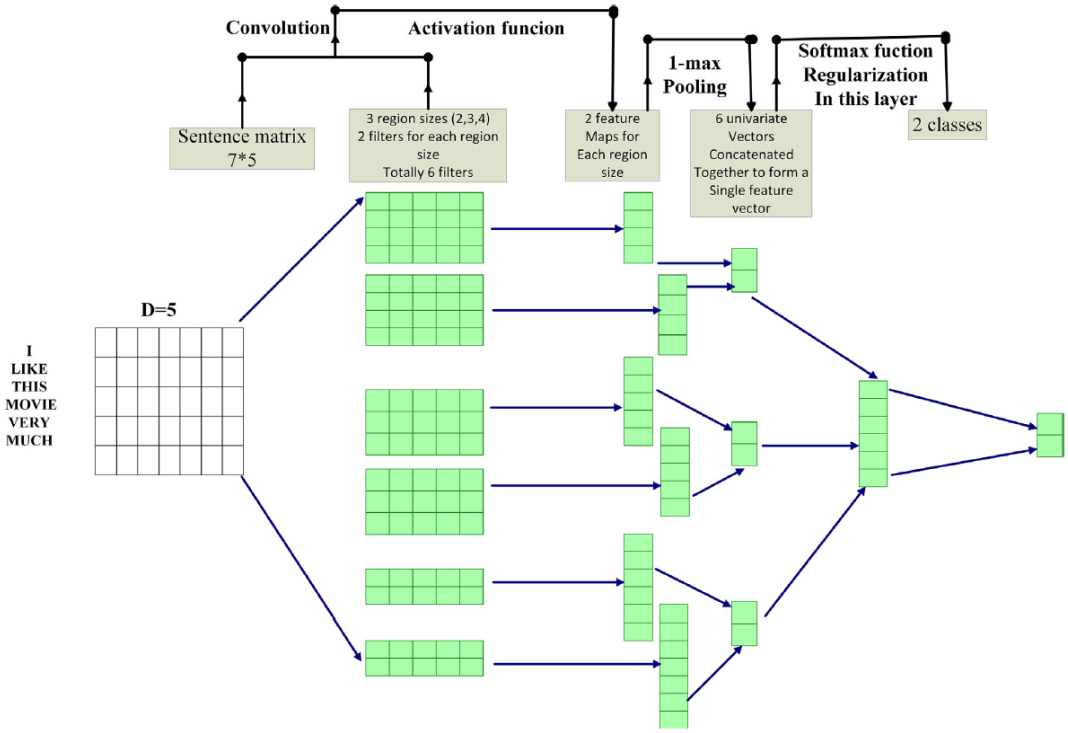

Figure 2.5: Convolutional Neural Network for NLP [3]

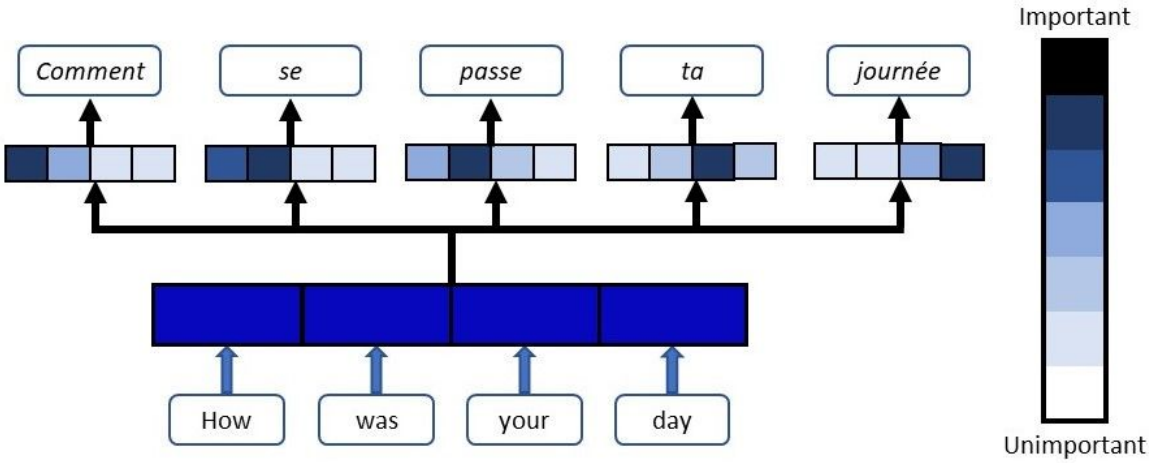

Figure 2.6: Attention mechanism for machine translation [4]

2.1.5 Attention Mechanism

Before being used for NLP applications such as text summarisation [65], the attention mechanism [55] was applied for neural machine translation. Figure 2.6 shows an example of how Attention works in a translation task. The sentence “How was your day” is here to be translated to the French version - “Comment se passe ta journée”. What the Attention component of the network will do for each word in the output sentence is map the important and relevant words from the input sentence and assign higher weights to these words, enhancing the accuracy of the output prediction.

Attention is a mechanism that helps the network remember certain aspects of the input better, including names and numbers. The attention mechanism is used when outputting each word in the decoder. For each output word, the attention mechanism computes a weight over each of the input words that determines how much attention should be paid to that input word. The weights sum up to 1, and are used to compute a weighted average of the last hidden layers generated after processing each of the input words. This weighted average, referred to as the context, is then input into the softmax layer along with the last hidden layer from the current step of the decoding [51].

2.1.6 Transformers

Transformers are multilayered structures comprised of Transformer blocks stacked above one another. Transformer blocks are distinguished by a system for multi-head self-attention, a position-wise feed-forward network, layer normalisation [66] modules, and residual connections. Typically, the input to the Transformer model is a tensor with the form $\mathbb{R}^B \times \mathbb{R}^N$, where B is the batch size and N is the sequence length. The input initially traverses an embedding layer that transforms each one-hot token representation into a d_model dimensional embedding, i.e. $\mathbb{R}^B \times \mathbb{R}^N \times \mathbb{R}^{d_{\text{model}}}$. The resulting tensor is then constructed additively using positional encodings and fed through a module with multiple headed self-attention. Positional encodings can be sinusoidal inputs (as described in [67]) or trainable embeddings. Multi-headed self-attention module inputs and outputs are connected by residual connections and a layer normalising layer. The output of the multi-headed self-attention module is then forwarded to a two-layered feed-forward network with similarly linked inputs/outputs and layer normalisation. The expression for sub-layer residual connections with layer norm is:

$$X = \text{LayerNorm}\left(F_S(X)\right) + X \tag{2.10}$$

where F_S is the sub-layer module which is either the multi-headed self-attention or the position-wise feed-forward layers.

Multi-Head Self-Attention

The Transformer model employs a mechanism for multi-headed self-attention. The key concept underlying the method is that each token in the series will learn to collect information from other tokens in the sequence. The procedure for a single head is defined as follows:

$$A_h = \text{Softmax}\left(\alpha Q_h K_h^\top\right) V_h \tag{2.11}$$

where X is a matrix in $\mathbb{R}^{N\times d}$, α is a scaling factor commonly set to $\frac{1}{\sqrt{d}}$. N_H is the number of heads, and $W_q, W_k, W_v \in \mathbb{R}^{d\times\frac{d}{H}}$ are the weight matrices (parameters) for the query, key, and value projections that project the input X to an output tensor of d dimensions. Softmax is implemented row-by-row. The outputs of the heads $A_1 \cdots A_H$ are concatenated and sent to a dense layer. Thus, the output Y may be represented as $Y = W_o\left[A_1 \cdots A_H\right]$, where W_o is a linear projection of the output. Note that A is often computed in parallel by taking into account tensors of $\mathbb{R}^B \times \mathbb{R}^N \times \mathbb{R}^H \times \mathbb{R}^{\frac{d}{H}}$ and performing the linear transforms for all heads in parallel. It is the responsibility of the attention matrix $A = QK^\top$ to learn alignment

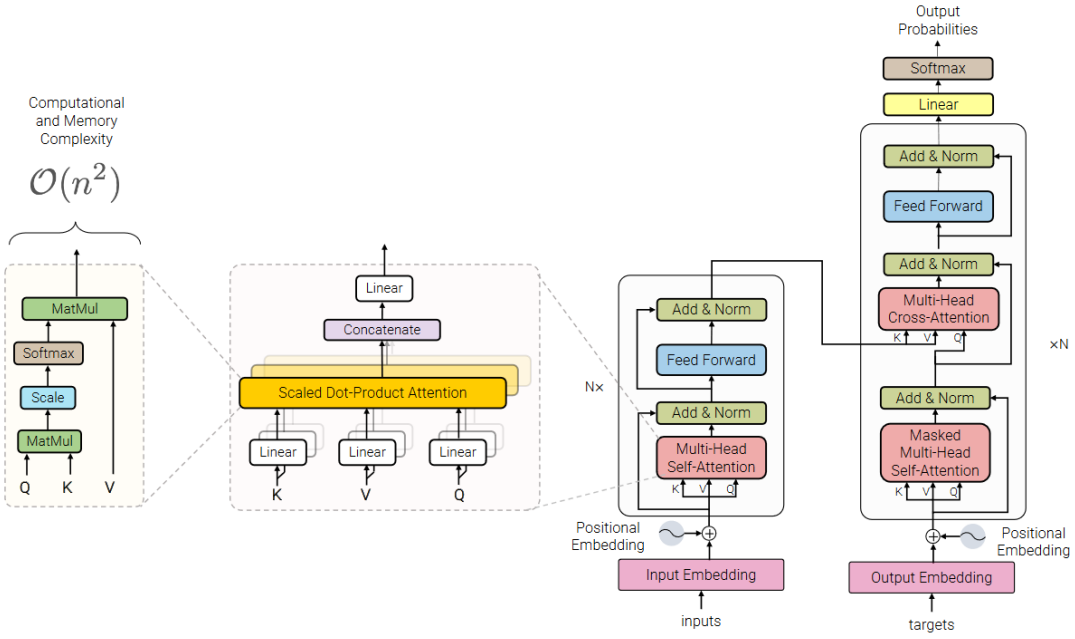

Figure 2.7: Architecture of the standard Transformer [5]

scores between tokens in the sequence. In this formulation, the dot product is calculated between each element/token in the query (Q) and the key (K). This promotes the process of self-alignment in self-attention through which tokens learn to gather from one another.

Position-wise Feed-forward Layers

The outputs of the self-attention module are then fed into a two-layered feed-forward network including $ReLU$ activations. This feed-forward layer acts independently on each position. The expression is as follows:

$$F_2\left(\mathrm{ReLU}\left(F_1\left(X_A\right)\right)\right) \tag{2.12}$$

where F_1 and F_2 are feed-forward functions of the form $W_x + b$.

Puting it all together

Each Transformer block can be expressed as:

$$\begin{aligned} X_A &= \text{ LayerNorm }\left(\text{ MultiheadAttention }\left(X, X\right)\right) + X \\ X_B &= \mathrm{LayerNorm}\left(\mathrm{PositionFFN}\left(X_A\right)\right) + X_A \end{aligned} \tag{2.13}$$

where X represents the input to the Transformer block and X_B represents its output. Note that the MultiheadAttention() method receives two tensors as arguments, one for the query and one for the key-values. This is the MultiheadSelfAttention method if the first and second arguments represent the same input tensor.

Transformer Mode

The variations in how the Transformer blocks are utilised should be taken into consideration. There are three main applications for transformers: (1) encoder-only (for classification, for example), (2) decoder-only (for language modelling), and (3) encoder-decoder (e.g. for machine translation). There are often many multi-headed self-attention modules in encoder-decoder mode, including a conventional self-attention in both the encoder and the decoder and an encoder-decoder cross-attention that enables the decoder to use information from the encoder. This affects how the self-attention mechanism is created. There is no requirement or limitation in the encoder mode that the self-attention mechanism is causal, i.e. entirely reliant on the current and past tokens. Since each auto-regressive decoding step in an encoder-decoder system may only depend on prior tokens, the self-attention employed in the decoder (i.e. across decoding positions) must be causal. In contrast, the self-attention used in the encoder need not be causal. For many effective self-attention systems, meeting this condition might be difficult. A Transformer model's manner of use often relies on the intended application. Encoder-Decoder architectures like T5 have been used for classification in recent research [68]. Generation generally employs decoder-only models that are trained with a language modelling purpose (of predicting the next token). Due to the nature of the loss, these models are frequently more advantageous for open-ended generation [69]. A causal decoder-only model and an upper triangular mask are required to prevent tokens from peering into the future.

2.2 Text Embeddings

Modern natural language processing systems typically employ distributional word representations learned in an unsupervised manner on large-scale corpora [70, 71, 72]. However, these approaches only obtain a single global representation for each word, ignoring their context. Contextual embeddings differ from conventional word representations in that each token is connected with a representation that is a function of the complete input sequence which goes

beyond word-level semantics. These context-dependent representations can capture many syntactic and semantic properties of words under diverse linguistic contexts. Previous works [73, 74, 68, 75] have shown that contextual embeddings pre-trained on large-scale unlabeled corpora achieve state-of-the-art performance on a wide range of natural language processing tasks, such as text classification, question answering, and text summarisation. Further analyses [76, 77, 78] demonstrate that contextual embeddings are capable of learning useful and transferable representations across languages.

Pre-training contextual embeddings can be divided into either unsupervised methods (e.g. language modeling and its variants) or supervised methods (e.g. machine translation and natural language inference). In this section, only unsupervised methods are described. Language modeling is the standard method for learning distributed token embeddings. A probability distribution over a list of tokens is a language model. The probability of a sequence of N tokens, $t_1, t_2, ..., t_N$, is factorised by a language model as follows:

$$p\left(t_1, t_2, \ldots, t_N\right) = \prod_{i=1}^{N} p\left(t_i \mid t_1, t_2, \ldots, t_{i-1}\right) \tag{2.14}$$

Maximum likelihood estimation (MLE), which is often penalised with regularisation terms, is used in language modelling to estimate model parameters. A left-to-right language model estimates the conditional probability by taking into consideration the left context of t_i, which includes $t_1, t_2, ..., t_{i-1}$. Large unlabeled datasets are typically used to train language models. The conditional probabilities are most commonly learned using neural networks, and the learned representations have been proven to be transferable to downstream natural language understanding tasks [79, 80].

2.2.1 ELMO

In order to generalise conventional word embeddings, the ELMo model [81] extracts context-dependent representations from a bidirectional language model. The left and right contexts are encoded using a forward L-layer LSTM and a backward L-layer LSTM, respectively. The contextualised representations are the left-to-right and right-to-left representations concatenated at each layer j to provide N hidden representations $(h_{1,j}, h_{2,j}, ..., h_{N,j})$ for a sequence of length N. To use ELMo in downstream tasks, the (L+1)-layer representations modeling the global word embedding) for each token k are aggregated as:

$$\mathrm{ELMO}_k^{\text{task}} = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \, \mathbf{h}_{k,j} \tag{2.15}$$

where s^{task} are layer-wise weights normalized by the softmax used to linearly combine the (L+1)-layer representations of the token k and γ^{task} is a task-specific constant. Given a pre-trained ELMo, it is straightforward to incorporate it into a task-specific architecture for improving performance. As most supervised models use global word representations x_k in their lowest layers, these representations can be concatenated with their corresponding context-dependent representations $ELMO_K^{task}$, obtaining $[x_k; ELMO_K^{task}]$, before feeding them to higher layers. The effectiveness of ELMo is evaluated on six NLP problems, including question answering, textual entailment and sentiment analysis.

2.2.2 GPT

The two-stage learning approach used by GPT [82] consists of (a) unsupervised pre-training with a language modelling objective and (b) supervised fine-tuning. Learning universal representations that are applicable to a variety of downstream tasks is the purpose. To this end, GPT trains the language model using the BookCorpus dataset [83], which includes more than 7,000 books from a variety of genres. The language model is implemented using the Transformer architecture [67], which has been shown that performs better for capturing global dependencies from the inputs than its alternatives in a variety of sequence learning tasks, including machine translation and document generation. During fine-tuning, GPT applies task-specific input adaptations driven by traversal-style techniques to inputs with multiple sequences [84].

2.2.3 BERT

ELMo concatenates representations from the forward and backward LSTMs without considering the interactions between the left and right contexts. GPT [82] uses a left-to-right decoder, where every token can only attend to its left context. These architectures are sub-optimal for sentence-level tasks, e.g. named entity recognition and sentiment analysis, as it is crucial to incorporate contexts from both directions. BERT proposes a masked language modeling (MLM) objective, where some of the tokens of an input sequence are randomly masked, and the objective is to predict these masked positions taking the corrupted sequence as input. BERT applies a Transformer encoder to attend to bi-directional contexts during pre-training. In addition, BERT uses a next-sentence-prediction (NSP) objective. Given two input sentences, NSP predicts whether the second sentence is the actual next sentence of the first sentence. The NSP objective aims to improve the tasks, such as question answering and natural

language inference, which require reasoning over sentence pairs.

2.2.4 BERT Variants

Recent work further studies and improves the objective and architecture of BERT. Instead of randomly masking tokens, ERNIE [85] incorporates knowledge masking strategies, including entity-level masking and phrase-level masking. ERNIE 2.0 [86] further incorporates more pre-training tasks, such as semantic closeness and discourse relations. SpanBERT [87] generalizes ERNIE to mask random spans, without referring to external knowledge. StructBERT [88] proposes a word structural objective that randomly permutes the order of 3-grams for reconstruction and a sentence structural objective that predicts the order of two consecutive segments.

RoBERTa [89] makes a few changes to the released BERT model and achieves substantial improvements. The changes include: (1) Training the model longer with larger batches and more data; (2) Removing the NSP objective (3) Training on longer sequences; (4) Dynamically changing the masked positions during pre-training.

ALBERT [90] proposes two parameter-reduction techniques (factorized embedding parameterization and cross-layer parameter sharing) to lower memory consumption and speed up training. Furthermore, ALBERT argues that the NSP objective lacks difficulty, as the negative examples are created by pairing segments from different documents, this mixes topic prediction and coherence prediction into a single task.

2.2.5 ELECTRA

ELECTRA [91] proposes a more effective pre-training method compared to BERT. Instead of corrupting some positions of inputs with [MASK], ELECTRA replaces some tokens of the inputs with their plausible alternatives sampled from a small generator network. ELECTRA trains a discriminator to predict whether each token in the corrupted input was replaced by the generator or not. The pre-trained discriminator can then be used in downstream tasks for fine-tuning, improving upon the pre-trained representation learned by the generator

2.2.6 T5

T5 (Text-to-Text Transfer Transformer) proposed by [68], for unifying natural language understanding and generation by converting the data into a text-to-text format and applying

an encoder-decoder framework. T5 introduces a new pre-training dataset, Colossal Clean Crawled Corpus by cleaning the web pages from Common Crawl. T5 also systematically compares previous methods in terms of pre-training objectives, architectures, pre-training datasets, and transfer approaches. T5 adopts a text infilling objective (where spans of text are replaced with a single mask token), longer training, multi-task pre-training on GLUE or SuperGLUE, fine-tuning on each individual GLUE and Super-GLUE tasks, and beam search.

2.2.7 BART

BART model [92] introduces additional noising functions beyond MLM for pre-training sequence-to-sequence models. First, the input sequence is corrupted using an arbitrary noising function. Then, the corrupted input is reconstructed by a Transformer network trained using teacher forcing [93]. BART evaluates a wide variety of noising functions, including token masking, token deletion, text infilling, document rotation, and sentence shuffling (randomly shuffling the word order of a sentence). The best performance is achieved by using both sentence shuffling and text infilling. BART matches the performance of RoBERTa on GLUE and SQuAD and achieves state-of-the-art performance on a variety of text generation tasks.

2.3 Text Generation

Deep generative models are useful for studying not just how well the model has learnt, but also for learning the domain of the problem. In the deep learning era, the most common text generation approaches are GANs [94] and Variational Auto-Encoders (VAEs) [95].

2.3.1 GANs

A GAN provides a technique that overcomes these issues encountered by generative models [94]. GAN is a popular deep learning method that uses an adversarial strategy, unlike the typical neural network. Two adversarially trained models are incorporated into GANs. First, the generator creates the data samples and discriminator that classifies these data samples as real (training data) or fake (produced by the generator), as seen in Fig. 2.8.

The objective of the generator is to produce samples that closely resemble the genuine data in order to trick the discriminator, whereas the objective of the discriminator is to properly distinguish these two types of data samples. The goal function is represented as a component of the minimax function, similar to a game-theoretic method. The discriminator D

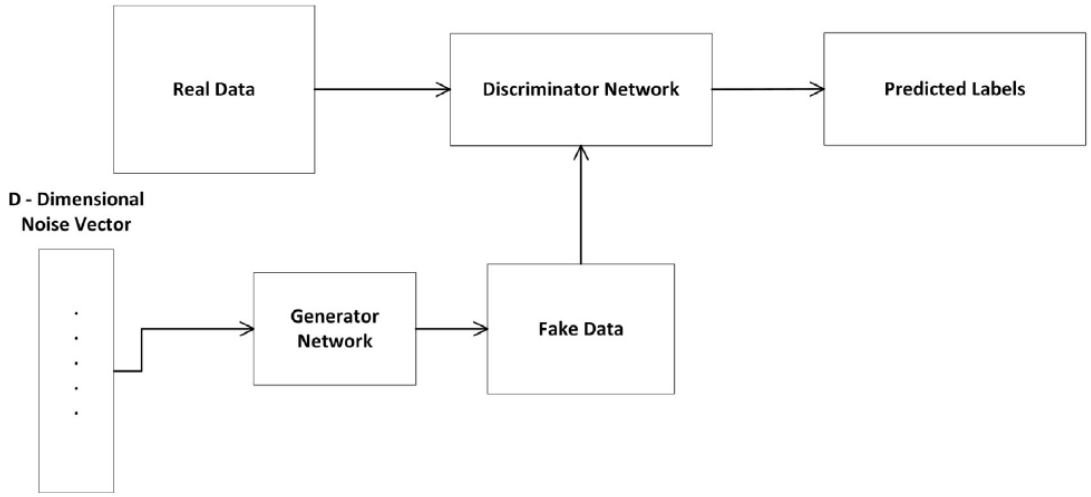

Figure 2.8: Block Diagram of GANs [3]

attempts to maximise the objective function, whereas the generator G attempts to reduce it. Thus, D and G attempt to play the minimax game with value function $V(G, D)$.

$$\min_G \max_D V(G, D) = E_{x->p_{\text{data}}(x)}[\log D(x)] + E_{z->p_z(z)}[\log(1 - D(G(z)))] \tag{2.16}$$

Since GANs have demonstrated excellent outcomes for picture creation (Pix2Pix GANs, Conditional GANs, Westerrrian GANs, etc.) but training GANs for text is more complex due to the indistinguishability of discrete symbols. The initial attempts at creating meaningful sentences [96] utilised Maximum Likelihood Estimation (MLE) (a technique for determining the parameter values that maximise the likelihood of the process outlined by the model). Despite its success, this training objective has some drawbacks, including shallow, repetitive, and limited responses. For this reason, clarity is required in several areas, such as which concepts constitute normal communication and approaches and how they might be combined with deep learning, in order to resolve these challenges. Numerous approaches have suggested physically characterising a few qualities (acknowledgment, informativeness, and consistency) and incorporating reinforcement learning frameworks to train and create highly rewarded words [97, 98]. However, manually constructed reward functions cannot account for all valid information and may result in low-quality statements. A successful generative model should produce phrases that are indistinguishable from those produced by humans. The use of GANs to NLP tasks has not met with equivalent success. Because the text production process is discrete, it is difficult to trace output errors back to the generator. How GANs may be utilised

to improve text generative models and how the associated challenges might be solved is a concern of ongoing research. In recent years, a proposal entitled Professor Forcing was made [99] on variable-length inputs and comparison of the distributions of two sequences (Training Sequence and Generated Sequence). It was demonstrated that the discriminator not only considers single-step predictions, but also the statistics of the behavior. Providing the discriminator central hidden values of the generator produces a differentiable model and achieves good results in a variety of NLP tasks, such as sequence creation and sound generation.

Maximizing the likelihood of each word in labelled data, given previously determined output, is a well-known strategy for addressing NLP challenges. The issue with this strategy is exposure bias (where the system gets more exposure to ground truth data and fails to generate meaningful sequences at test time). Bengio et al. presented a solution to this problem using the notion of schedule sampling (a training approach in which the model is explored more during the training stage, making it more resilient to errors committed during inference) [100]. The plan is to partially feed the generative model with synthetic data while predicting the following word during the learning stage. This methodology was deemed an inconsistent training method, as errors were not back-propagated through sample decisions, and no progress was shown in addressing the issue of exposure bias.

Reinforcement learning

Deep generative models employed reinforcement learning to overcome the aforementioned issues [101, 102]. The use of reinforcement learning to deep generative models has produced encouraging results. Reinforcement learning is a subfield of deep learning in which a model learns by interacting with its environment and being rewarded for performing actions. The algorithms for reinforcement learning may be comprehended utilising the concepts of agents, environments, states, actions, and rewards:

- Agent:- An agent is someone who takes actions;
- Action:- All possible moves, an agent can make is considered as an action, like in a video game action can be moving right, moving left, standing still.
- Environment:- The space where an agent learns, where the agent receives its current state and outputs its reward and its next state. State: It might be a solid spot and point, an unrestricted design that sets the agent in relation to other important items,

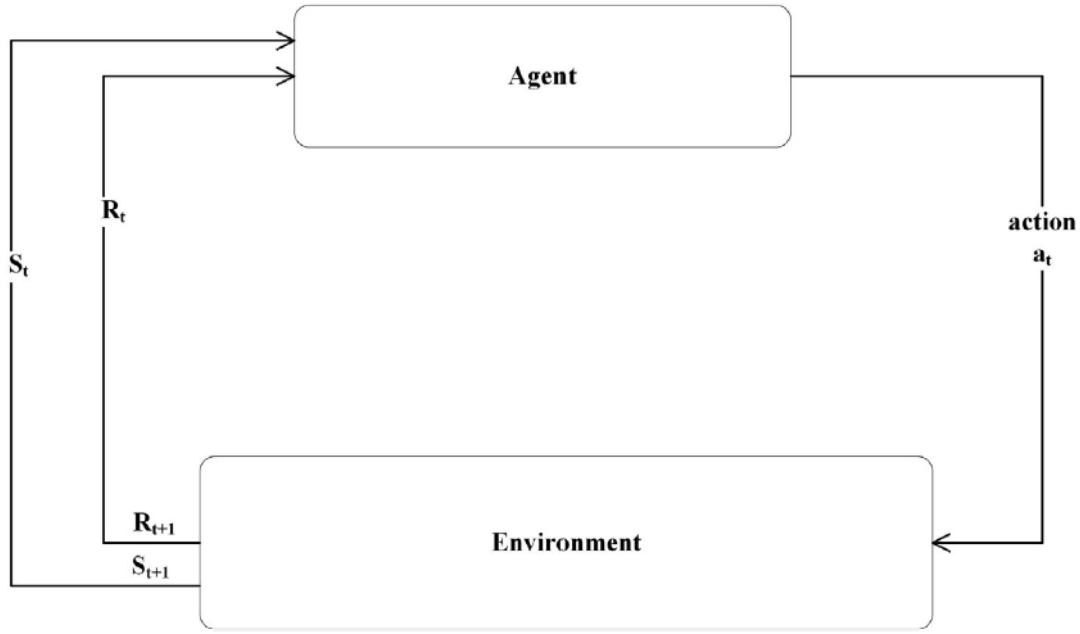

Figure 2.9: Block diagram of Reinforcement Learning [6]

for example, apparatuses, impediments, enemies, or rewards, the present circumstance returned by the environment, or any future condition.

- Reward:- A reward is an input used to evaluate the success or failure of an agent's actions; for example, in a computer game, when an agent makes contact with the coin, he receives points. The agent from a random state transmits output in the form of activities to the environment, which rewards the agent with a new state. The process through which subsequent actions are determined depending on the existing condition. As seen in Figure 2.9, it connects states to actions, actions that promise the greatest reward.

This is the key concept utilised to represent text generation as the reinforcement learning problem. This topic was initially investigated by [103], who concluded that sequence generation issues may be phrased as sequential decision-making challenges. Informed by this and the notion of reinforcement learning, a new training method for generative models was presented [104]. The extra network "Critic" is trained to produce the predicted task score for each token. These projected outputs are used to train the "Actor" main sequence prediction model. Under the assumption that the critic generates accurate output values, the explana-

tion used to train the actor is a neutral measure of the predicted task-specific score gradient. However, the use of reinforcement learning in GANs for text production must address any concerns like how awards may be produced to control the generator. Methods for answering this question are an ongoing object of research.

A basic strategy would be to utilise the task-specific score as a reward [104]. However, such task-specific scores might be difficult to locate at times. The sequence GANs (SeqGAN) proposed by Yu et al. [105] are guided by the discriminator's prediction score. The sequence $s - p_G$ is generated by the generator G_θ, and the discriminator predicts whether it is real (high reward) or not (low reward). The current purpose of the generator model (policy) is to generate a sequence from the beginning state s_0 that maximises the predicted final reward. R_T is the ultimate reward governed by D as indicated by the equation:

$$J_\theta = E\left[R_T \mid s_0, \theta\right] = \sum_{y_1 \epsilon Y} G\left(y_1 \mid s_0\right) \cdot Q_D^G\left(S_0, Y_1\right) \tag{2.17}$$

Y stands for vocabulary. Now, the expectation of attaining the end reward R_T conditioned on starting state s_0 and θ (generator parameter) is the product of all possible reward values and their occurrence probabilities (Ist term). The second term denotes the action-value function, which yields the reward value for action Y_1 in initial state s_0 with policy G. This action-value function predicted by the discriminator D rewards just the sequence that has been completed. However, it is vital to assess the suitability of both prior and created terms. In order to calculate the reward, the current policy is rolled out using the Monte-Carlo search (a technique used in game theory that begins with random acts and repeatedly iterates until the final state) [106]. At the end of the sequence, the discriminator D predicts the cumulative score for each network node. The generator employs the current learnt policy network to roll-out several times until sentences are finished and the estimated reward is obtained. Training techniques for the generator and discriminator have a significant impact on the performance of GANs. SeqGAN's training technique differs from that of conventional GANs since SeqGAN involves pre-training of the generator on the target corpus prior to adversarial training.

As discussed previously, it is difficult to back-propagate the gradient in a network when the output is discrete. To overcome this issue, the generative model is viewed as a stochastic parameterized approach in which Monte Carlo is utilised to estimate the state value. The concept utilised to train the policy gradient lowers the difficulty of differentiation in conventional GANS for discrete data. In fact, it has been seen that positive and reliable reward signals from the discriminator might be difficult to get, even with thorough preparation. MaliGAN

(Maximum-Likelihood Augmented Discrete Generative Adversarial Networks) was developed to overcome these challenges [96]. Inspiring by Norouzi et al. [107], the normalised maximum likelihood optimization is employed to overcome the challenge of back-propagating rewards.

Lin et al.[108] presented the novel adversarial training known as RankGans to create high-quality text descriptions. Instead of categorising the output between 0 and 1, it is ranked between 0 and 1 according to the output's diversity. The model learns from this relative ranking of information produced by humans and machines. The adversarial framework precisely consists of two networks: Generator and Ranker. The ranker is educated to rank machine-produced sequences lower than human-made sequences, while the generator is trained to produce phrases that trick the ranker. To address the issue of non-differentiability, the gradient approach is utilised. Despite its improved BLEU score [109], its practical use is still limited by a few obstacles.

1. The use of a scaler as a score may not be sufficiently descriptive to instruct the generator, as it cannot adequately describe the transitional structure of text during creation.

2. Estimating the reward for intermediate sentences tends to be noisy and imprecise, particularly in long text production where the generator receives reward only after the full sentence has been completed.

The primary difficulty in creating lengthy text sequences is the sparsity of the binary guided signal, which is only delivered after the entire sample has been formed. Zhang et al. [110] recommended that instead of maximising the rewards from the discriminator, the generator G should be trained to learn the feature representation of the real text and the produced text to be matched. To address these issues, LeakGAN [111] was developed, which combines feature matching with hierarchical reinforcement learning [112]. Manager (LSTM which is utilised as a mediator to receive the feature representation from discriminator D) and Worker (which uses these features received from discriminator as the guiding signal for generator) make up the hierarchical generator G. This information from D is preserved internally and is identified as material that has leaked. A significant outcome of LeakGAN is the ability to examine if the generator generates data using the leaked information from the discriminator. All of the aforementioned strategies attempt to extract more information from the discriminator in order to generate text of higher quality. Using adversarial training to generate realistic-looking samples is a fast expanding field of study. The concept of reinforcement learning

proposed by Yu et al.[105]; Li et al. [97] posits that text production is a sequential decision-making process. Despite the effectiveness of these strategies, their practical use is limited by the framework's two basic flaws.

- Mode-Collapsing: Mode collapse is an analogous and related failure mode of Generative Adversarial Networks. In mode collapsing, the generator only learns one mode in a multi-modal distribution and chooses to always use that mode to exploit the discriminator. For instance, if the training set contains both dogs and cats, the generator will attempt to generate wild cats but no dogs in order to trick the discriminator. This sort of issue is known as mode collapse [113].

- Vanishing-Gradient: The issue comes when training the two models in an adversarial manner, as the discriminator quickly converges and the generator ends up learning nothing, a phenomenon known as vanishing gradient [114].

Inspired by the approach of hierarchical feature representation [115], Zhang et al. [110] created the TextGAN model, which employs LSTM as the generator and CNN as the discriminator. Utilizing kernel-based moment-matching (a technique for matching the moments of two distributions), the produced and genuine sentences are forced to have identical moments. The generator was taught to generate data that fits the empirical distributions of real data in the feature space by reducing Maximum Mean Discrepancy (MMD), also known as minimising the moments of two distributions. This method improves the model's ability to acquire features that are both instructional of initial sentences (using the autoencoder) and discriminative with respect to produced sentences (through the discriminator). To facilitate the training of generative adversarial networks, initialization strategies (weights of LSTM generator were initialised from a pre-trained CNN-LSTM Auto-Encoder) were also presented. This method mitigates the mode-collapsing problem inherent to conventional GAN training. As previously stated, RNNs are the most common generating model for text and many NLP applications. The issue of using GANs for discrete data is a highly active field of research.

Mask-GANs [116], a version of GAN in which the model is trained on a sequence infilling task, was introduced in order to lessen the effect of the issues observed in standard GANs when used for NLP tasks. The purpose of the model is to insert the sequence's missing words when a few words have been deleted or altered. This model's purpose is to fill in the missing words of the sequence such that it can be distinguished from the original sequence. While infilling this missing part of sequence, the model works auto regressively over the words it has

so far filled in, as in traditional language modelling constrained by actual context knowledge. If the entire sequence is altered, language modelling is the only remaining option.

Due to the fact that deep learning models are trained on enormous datasets, the data created by generative models is the main mechanism of addressing this issue. The produced text is a potential technique for data augmentation, as demonstrated by [117, 118], which used GANs to create text and reached state-of-the-art results. To address the challenge of small datasets and to train stronger models, the generated data must be categorical (labelled). For the production of labelled sentences, CS-GAN [119] was developed, where RNNs are employed as a generator and the generator behaves as an agent that predicts the next character based on the current character, similar to the process of reinforcement learning. Thus, the combination of RNNS and reinforcement learning specifically addressed two difficulties.

1. They generate realistic sentences with GANs given the discrete nature of the text.

2. Incorporating category information into GANs in order to produce labelled synthetic data.

Recently, a new method was proposed by Shi et al.[102] to address the main two challenges (Mode Collapse and Reward Sparsity) in GANs for text generation. This method uses the concept of Inverse Reinforcement Learning (IRL) [120] to treat text generation as an IRL problem, where the reward function learns to explain the expert behaviour and the generated policy is learned to maximize the expected total rewards. The objective of the reward function is to raise the payouts for the genuine texts in the training set while decreasing the rewards for the created texts. Intuitively, the reward function in SeqGAN serves a similar purpose as the discriminator. Unlike SeqGAN, the reward function provides more dense reward signals by rewarding each step and action immediately. The generating policy samples one word at a time to build a text sequence. The optimal policy is learned by the "entropy regularised" policy gradient [121], which inherently results in a more diverse text generator. The similarity between two summaries (system summary and reference summary) is determined to evaluate the performance of generative models. Numerous measures for measuring the performance of deep generative models have been presented ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [122], BLEU (BiLingual Evaluation Understudy) [109] , etc). The issue with the assessment measure BLEU is that if the same sentence is repeatedly generated, the BLEU score will be perfect.

GANs were viewed as a promising strategy for text creation. In order to evaluate the

potential of GANs for text production, however, a precise assessment criteria was required. Some text evaluation methods include:

1. ROUGE is used to evaluate the resemblance between the reference summary (Golden Summary) and the system-generated summary. The Rouge is a collection of summary evaluation metrics [122]. Using a comparison between the golden summary and the system summary. Rouge Recall:- It means how much of the golden summary this system summary captures. It is calculated as:

$$\text{Recall} = \frac{\text{Similar} - \text{Tokens}}{\text{total} - \text{Tokens} - \text{in} - \text{Training} - \text{Summary}} \tag{2.18}$$

   Rouge Precision: It actually gives us the system summary which is relevant or needed and is calculated as:

$$\text{Precision} = \frac{\text{Similar} - \text{Tokens}}{\text{total - Tokens} - \text{in - Generated} - \text{Summary}} \tag{2.19}$$

2. BLEU is the measure that compares the produced sentence to the reference sentence. BLEU [109] was initially used for machine translation systems and operates by assessing the similarity between machine-translated text and reference material, where unigram or 1-gram is viewed as a single word and Bi gram as a word pair. It operates by comparing Ngrams of machine translation with reference material and counting the number of matches. Consequently, more matches result in an improved machine translation.

In addition to the above-mentioned text evaluation techniques, several evaluation approaches have been offered [123, 124, 125]. However, new research has shown that existing approaches have a low correlation with human judgement. Evaluation of text creation systems is an open area of research. Since there was no evaluation metric for GAN-based text production and the metrics stated above, which are based on N-gram overlapping, are deemed to have low correlation and low robustness, there was no evaluation metre for GAN-based text generation [126]. Tevet et al. [127] provided a technique for evaluating Generative Adversarial Networks with standard probability-based evaluation metrics, where the prediction of the generative model is viewed as language modelling (LM) and a basic Monte-Carlo approach is used to approximate it. The estimated probability distribution is then assessed using LM measures like as Perplexity (the evaluation of how accurately a model predicts a sample) or Bits Per Character (BPC) (average number of bits needed to encode on a character). A major worry for the text generation strategies discussed in this study is whether

Table 2.1: Comparison of recent GAN models on text generation.

	BLEU2	BLEU3	BLEU4	BLEU5
SeqGAN [105]	0.724	0.416	0.178	0.086
MaliGAN [96]	0.755	0.436	0.168	0.077
RankGAN [108]	0.686	0.387	0.178	0.086
MaskGAN [116]	0.265	0.165	0.094	0.057
TextGAN [110]	0.205	0.173	0.153	0.133
MLE [96]	0.205	0.173	0.153	0.133
LeakGAN [111]	0.835	0.648	0.437	0.271

the model suffers from mode-collapse in addition to text quality (lack diversity). Counting unique n-grams is a modern method for calculating text diversity. As explained in MaskGAN, improving such an n-gram measure does not inevitably increase variety. The subject of the variety of text creation has received less attention in recent literature, but some relatively recent research is begun to investigate it [3].

The new evaluation procedure [128] proposes to use the sweep of temperatures for each model to compute the temperature curves in quality-diversity space, which provides researchers with information regarding which model should be used for generating samples with high quality-diversity. The notion of temperature sweep might be considered a cross-validation technique (which means early stop once the best curve has been achieved). However, the assessment criteria stated in the preceding paragraph produce mistakes since they are are based on N-gram overlapping that have low correlation and low robustness [126]. Table 2.1 displays the BLEU score of the GAN-based generating models described before. Except for MaskGAN, all the models suffer from mode collapse.

2.3.2 Variational Auto-Encoders (VAEs)

Most deep learning models rely on well labelled data to function effectively. Since the majority of data is unlabeled or unstructured, training popular deep learning models is impossible since they require vast amounts of structured data. Labeling the unstructured data takes a great deal of time. Using unsupervised techniques to train on data without labels is one strategy for addressing this issue. Variational Auto-Encoders [129] is one of the most effective unlabeled deep generative models. It includes an encoder that encodes data into latent variables, followed by a decoder that decodes these latent variables in order to reconstruct

the encoded data. The encoder accepts input x and generates output latent space $p_\phi(z \mid x)$, where ϕ represents the encoding operation's parameters, whereas the decoder performs the exact reverse. It determines the probability distribution $q_\theta(x \mid z)$ of data on a specified latent distribution, where θ indicates the decoding operation's parameters. Additionally, ϕ and θ can be considered weights for encoding and decoding operations. The loss function that compels the model to acquire a rich representation of latent space may be stated roughly as the sum of two terms: $LossFunction = Reconstructionloss + Regularisationterm$. Reconstruction term is the squared mean error between input and output data. However, the regularisation term reduces the gap between the latent distribution $p_\phi(z \mid x)$ and a previous distribution $p(z)$. With the use of latent space, VAEs learn the probability distribution of data, which makes it suited for producing new data. Kullback–Leibler (d_{kl}) divergence is utilised to assess the divergence between the encoder's distribution $p(z \mid x$ and posterior distribution $p(z)$. The mathematical expression for the loss function is:

$$L_i(\theta, \phi) = -E_{z\_>q\theta}\left(z \mid x_i\right)\big) \left[\log p_\phi\left(x_i \mid z\right)\right] + D_{KL}\left(p_\phi\left(z \mid x_i\right) || p(z)\right) \tag{2.20}$$

$L_i(\theta, \phi)$ defines the reconstruction loss at data point i, $-E_{z\_>q\theta}\left(z \mid \chi_i\right)\big)$ measures the expectation of encoder's distribution over representation $D_{KL}\left(p_\phi\left(z \mid x_i\right) || p(z)\right)$ describes the divergence, $(p_\phi\left(z \mid x_i\right)$ and $p(z))$.

The block diagram of Variational Auto-Encoders is seen in Figure 2.10. The input data x is sampled in latent space as the standard deviation σ and the mean μ, and then the stochastic sample of z is predicted based on this distribution. Finally, the sample z is decoded in order to create the output $x\prime$. VAEs have become one of the most prevalent methods for unsupervised learning of complicated distributions. There are few uses for VAEs in the creation of discrete data (text). The primary issue with utilising VAEs for text creation is KL collapse (when the decoder becomes more strong than the training objective, the problem can be handled with a false strategy), which occurs when the decoder outputs output regardless of latent space. If the KL term is zero, then the posterior probability is independent of the input data [130]. Bowman et al. [131] suggested a text generating model for VAEs that use recurrent neural networks to capture the universal characteristics of sentences (e.g. subject, style) as continuous variables. The problem of the collapse of the posterior was also seen. Bowman et al. offer the notion of KL-annealing (the complete phrase is included into distributed latent space) and word dropout (removal of some information during learning) as potential solutions to this problem [131]. This factorization helps to represent sentence attributes such as style and high-level semantic characteristics. In particular, the weights of the network

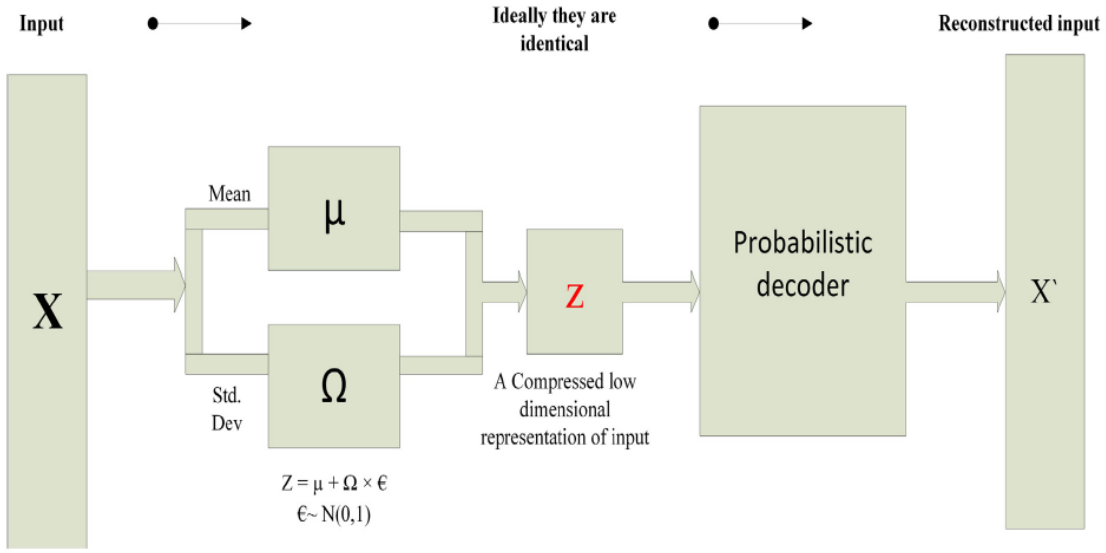

Figure 2.10: Block diagram of VAE [3].

are raised during the training phase, and the random replacement of word tokens causes the decoder to rely on the global representation z rather than the learnt language model. However, this methodology was insufficient in resolving the KL collapse issue; hence, several efforts were made to identify more effective methods. Text generation models frequently employ the x_t approach to produce text from previously generated tokens.

This method's output could not accommodate the diversity (subject, style, semantics, etc.) of created sentences. The most current solution for text creation and resolving the KL collapse was provided by Yang et al. [132], which substitutes the RNN decoder with a dilated CNN [133] and simplifies the management of contextual capacity by varying dilation. Specifically, earlier text generation algorithms were dependent on modelling the joint probability $p(x)$ directly. This work proposes that the marginal distribution be modelled as $p(x)$. They produce continuous latent space z initially based on the previous distribution $p(z)$ (multivariate Gaussian). Decoder then parameterized the creation of sequence x from a conditional distribution $p_\theta\left(\frac{x}{z}\right)$. This helps to incorporate the latent variable to balance the development of the entire discourse and makes it easier to get high-level characteristics of data variance. Due to the recurring nature of RNNs, they are regarded as a beneficial text creation method. However, a different method is required to address the underlying issue of relieving posterior collapse. Semeniuta et al. [118] introduced the new model called Hybrid VAE, where Conv and De-

Conv neural networks were utilised in place of LSTMs as encoder and decoder, respectively. The auxiliary $\text{loss}\, J_{aux}$ represented as:

$$J_{aux} = -\alpha E_{q(z|x)} \log p_\phi(x \mid z) \tag{2.21}$$

J_{aux} forces the decoding process to depend on latent representation z for the optimization of ELBO (Evidence Lower Bound). α controls the penalty of auxiliary loss, ϕ represents weights used in decoding process. It was shown that hybrid VAE converges faster and more effectively than LSTM VAE. However, the model was able to handle the issue of latent loss, although this technique had trouble creating lengthy text sequences. Training deep learning models is always complex, and training VAEs is no exception. Nevertheless, several attempts have been made to mitigate this issue. Kim et al.[134] presented the Variational Inference (VI), a technique for controlling network computation. Traditional VI techniques involved iterating through observed data and updating model parameters in closed form. However, this approach of parameter updating needs conjugate models. The implementation of this strategy for non-conjugate models was very difficult. Stochastic Variational Inference (SVI) [135] and Amortized Variational Inference (AVI) [136] are two recently suggested approaches for parameter selection that are scalable to massive training sets and non-conjugate models. This method of predicting model parameters is beneficial to VAEs when used as generative models. In the case of SVI, however, locating the local optimum was simple, whereas optimising individual data point is challenging. On the other hand, AVI's inference is too rapid, but the structure of input data as a parametric function makes it too restrictive; AVI's learning was designed so that the parameter update could be performed on sub-optimal variational parameters. Semi-Amortized Variational Auto-encoders are a recent development in the field of text production that utilises both AVI and SVI [134]. The inference network was used to choose the initial variational parameters, and then SVI was used to refine them. This method, however, exceeds existing autoregressive generating models, although no improvement was shown in resolving KL Collapse the primary issue in the field of text production. Even for humans, producing complicated prose from scratch in a single pass is challenging. Inspired by this strategy, Guu et al. [137] suggested Prototype then modify model, a second way for text generation. It first samples a random model sentence from the training corpus, then conjures a neural editor that generates a random edit vector and generates a new sentence by applying the model to the edit vector. However, the created samples were competitive, but this strategy made no contribution to the problem of "posterior collapse." The solution to the problem of KL collapse is still a subject of current research. Kim et al. [134] introduced

Table 2.2: Comparison of recent VAE models on text generation.

	Perpelexity	Negative log likelihood	KL
VAE [131]	60.1	380	15
Improved- VAE [132]	63.9	332.1	10
Hybrid- VAE [118]	*	*	12.5
Semi Amortized-VAE [134]	60.4	327.1	7.19
Neural Editor-VAE [137]	26.78	*	*
Skip-VAE [138]	60.55	*	22.54

a new method for initialising the parameters of a variational auto-encoder using amortised inference and subsequently refined it using stochastic inference. Dieng et al. [138] created skip connections between latent variable z and decoder that reinforce the link between latent variables and reconstruction loss. Due to the fact that trials validated the superiority of both approaches over their predecessors, it can be concluded that both methods are superior to their predecessors. Guu et al. [137] have presented an intriguing new development in which Von Mises-Fisher distribution [139] is employed instead of the Gaussian distribution. Xu et al. [140] investigated this technique further. Variational Auto-Encoders with this integration are referred to as Hybrid VAE, and they may regulate KL term by hyperparameter k, so addressing KL collapse. Table 2.2 displays a comparison of several text generation models based on VAEs.

The table illustrates enhancements made to VAE for text production jobs in which its mechanism has been significantly enhanced. These models' parameters consist of perplexity, negative log-likelihood, and the highlighted KL term in this table. Comparing the performance metrics of different models makes it difficult to pick one. Therefore, research on VAE for text creation problems remains active.

2.3.3 Paraphrase Generation

The task of paraphrase generation refers to rewriting a given sentence to a new paraphrase sentence, which requires new text generation. The paraphrase generation approaches are analysed in this section for the idea of transforming an extractive summary into an abstractive described in section 1.4. The paraphrasing task requires that the generated sentence and input sentence be different in expression form, but have the same expressed meaning [141]. Li et al., [142] proposed DNPG to dissect a sentence into sentence-level pattern and phrase-

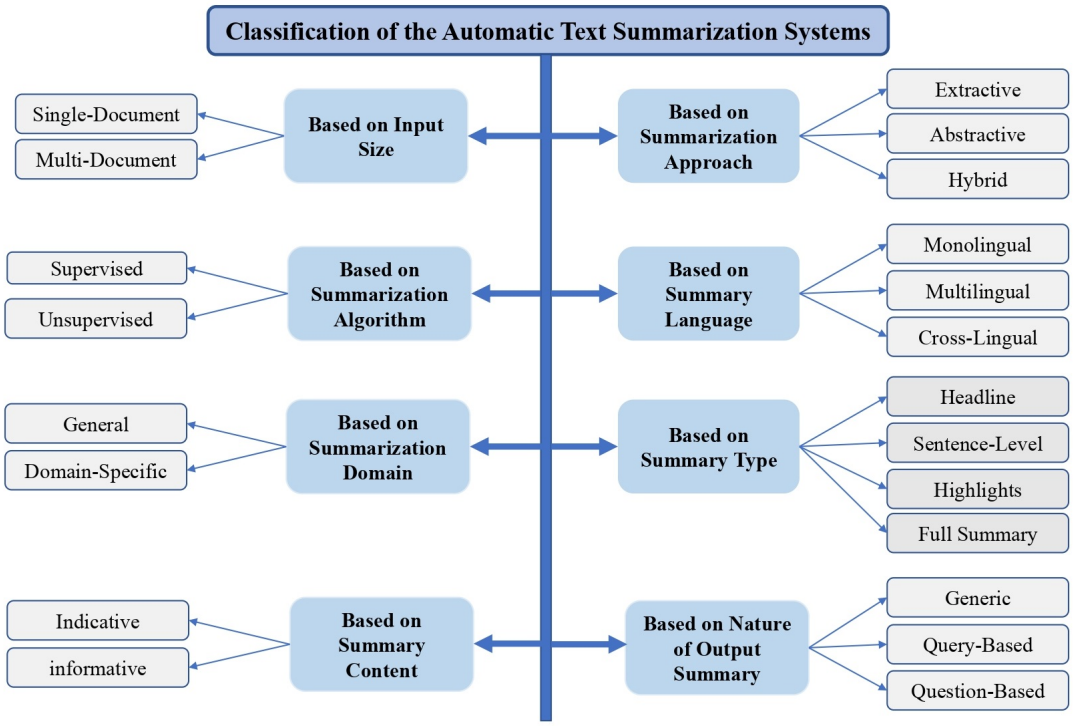

Figure 2.11: Classification of ATS systems [7].

level pattern to make neural paraphrase generation more understandable and controllable, and they discovered DNPG can be adopted into unsupervised domain adaptation method for paraphrase generation. Fu et al. [143] proposed a new paraphrase model based on latent bag of words. Siddique et al., [144] proposed an unsupervised paraphrase model with deep reinforcement learning framework variational autoencoder. Liu et al., [145] regarded paraphrase generation as an optimization problem and designed a sophisticated objective function. All methods above focus on the generic quality of paraphrase and do not care about the diversity of paraphrase. Yang et al., [146], Cao et al., [147], Vizcarra et al., [148], and Tuan et al., [149] proposed paraphrase generation models based on GAN are considered as the baseline methods and are discussed in detail in section 5.3.4.

2.4 Automatic Text summarisation (ATS)

There are many classifications for Automatic Text summarisation (ATS) systems as illustrated in Fig.2.11. ATS systems can be classified based on any of the criteria below.

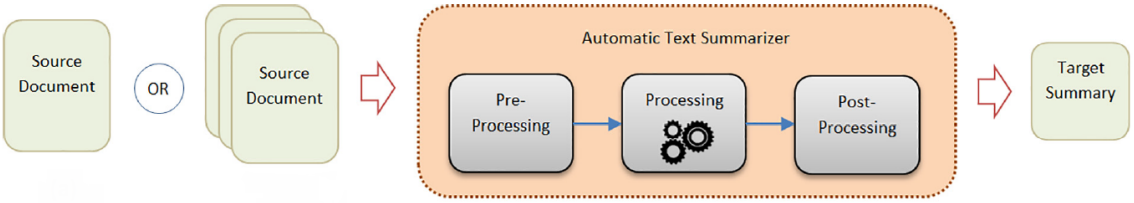

Figure 2.12: Single-document or Multi-document, automatic text summariser [7]

Classification based on the input size: Single-document or Multi-document. Input size refers to the number of source documents taken to build the target summary. As seen in Fig. 2.12, Single-Document summarisation (SDS) employs a single text document to construct a summary, with the goal of condensing the source document while preserving its essential content [150]. Multi-Document summarisation (MDS) generates a summary based on a series of input documents, with the goal of removing repeated information from the input documents [150]. MDS is more complicated than SDS and is plagued by challenges like as redundancy, coverage, temporal relatedness, compression ratio, etc [151].

Classification based on the text summarisation approach: Extractive, Abstractive, or Hybrid. The extractive text summarisation method selects the most significant sentences from the input document(s) and composes the output summary by concatenating the chosen sentences. The abstractive text summarisation approach provides an intermediate representation of the input document(s), from which the output summary is created. In contrast to extractive summaries, abstractive summaries consist of sentences that differ from those in the original source. In other words, sentences in the abstractive summary convey the same content but in a different form than their original sentences in the document. The hybrid text summarisation approach combines the extractive and abstractive methods.

Classification based on nature of the output summary: Generic, Query-Based or Question-Based. A query-based summarisation indicates that the multi-document summariser works with a collection of similar documents extracted from a big corpus in response to a query [152]. The resulting summary then contains items linked to the query. A query-based summary provides the information most pertinent to the initial search query, whereas a general summary provides an overview of the document's content [153]. The query-based summary is also known as a query-focused, topic-focused, or user-focused summary [154]. Question-

driven summarisation approaches answer a question and also provide additional informative content to that answer from the source document(s) to make it more understandable and convincing [29].

Classification based on the summary language: Monolingual, Multilingual, or Cross-Lingual. A summarisation system is monolingual if the language of the source and destination documents are identical. A summarisation system is multilingual if the original content is produced in many languages (such as English, Arabic, and French) and the summary is likewise created in these languages. A summarisation system is cross-lingual when the original content is provided in one language (for example, English) and the summary is created in another language (Arabic or French) [154].

Classification based on the summarisation algorithm: Supervised, or Unsupervised. An annotated training set of data is necessary for the supervised algorithm's training phase. The training data must be manually annotated, which makes it difficult and expensive to produce. The unsupervised method, on the other hand, does not need a training phase or training data [155].

Classification based on the summary content: Indicative or Informative. A summary that is indicative includes just the essential ideas or information from the source text [156]. In order to inform the user about the original source, it is utilised to determine what the input text is about (i.e. what topics are discussed) [157]. The objective of an indicative summary is to inform readers about the breadth of the input content in order to assist them in deciding whether to read the original text. On the other hand, an informative summary incorporates the essential facts and concepts from the original text [156] to cover all topics of the text [158]. The objective of an instructive summary is to convey, without going into detail, the essential points of the original text [157].

Classification based on the summary type: Headline, Sentence-Level, Highlights, or Full Summary. Depending on the ATS system's objectives, the length of the produced summaries varies. The headline created via headline creation is often less than a sentence [159]. A sentence-level summary creates one sentence, often an abstract one, from the supplied material [159]. A highlights summary generates a telegraphic-style, intensely condensed summary that is often presented as bullet points [160]. The highlights summary gives the reader a quick rundown of the key details in the source document(s) [160]. Lastly, the development of a full summary is typically directed by the needed summary length or a compression ratio.

Classification based on the summarisation domain: General, or Domain-Specific. The

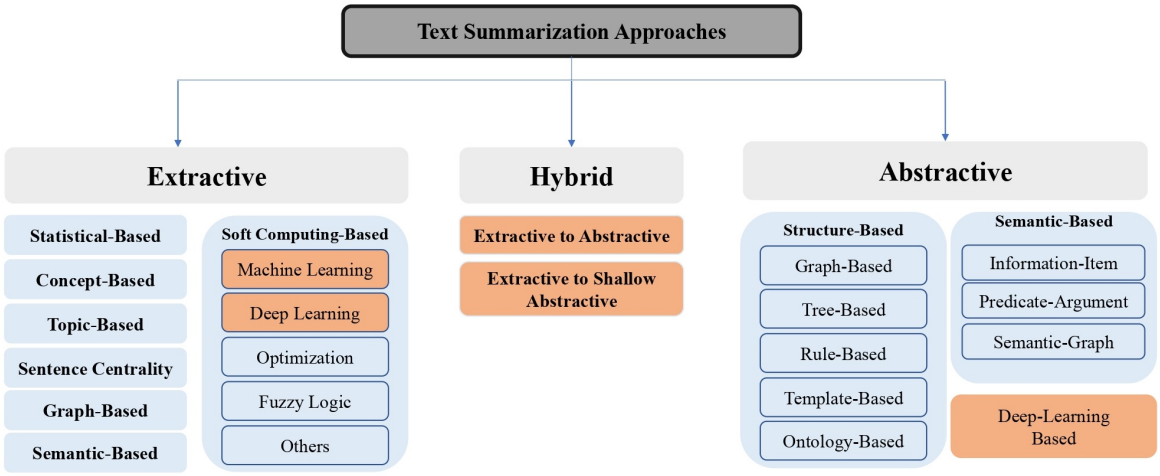

Figure 2.13: Automatic text summarisation approaches and their associated methods [7].

general or domain-independent ATS system provides summaries of documents from several domains. In contrast, the domain-specific ATS is designed to summarise documents from a given domain (e.g. medical documents or legal documents).

There are three main text summarisation approaches: extractive, abstractive, or hybrid. Each approach is applied using different methods as shown in Fig. 2.13. The following sections will provide a detailed overview about machine learning and deep learning-based extractive approaches, deep learning-based abstractive approaches and hybrid approaches.

2.4.1 Extractive Approaches

Cao et al. [161], among the supervised deep learning extractive approaches, suggested a recursive neural network for rating significant sentences in multi-document summarisation. They generated summary prior features for extractive text summarisation using improved CNNs. Denil et al. [162] provided a ConvNet model for documents with an architecture designed to facilitate document structural introspection. Furthermore, they have demonstrated that the visualization approach may be used to locate and extract task-specific relevant sentences from documents. Cheng et al. [163] approach consists of a hierarchical document reader or encoder based on neural networks and an attention-based content extractor. The reader is responsible for deriving the document's meaning from its sentences and their constituent words. To extract sentences or words, their algorithms use a type of neural attention. Nallapati et al. [164] proposed a highly interpretable neural sequence model that provides

intuitive visualization for extractive document summarisation. In addition, they suggested a unique abstractive training process to eliminate the necessity for extractive labels during training, yet this strategy is still a few ROUGE points below their extractive training on the majority of datasets. Wu et al. [165] proposed an approach with two stages. In the first stage, they suggested a neural coherence model that uses the distributed representation of sentences rather than sparse, manually crafted features. The proposed neural coherence model is completely independent of entity recognition systems and may be trained from scratch. Multiple layers of convolution and max-pooling enable the neural coherence model to capture cross-sentence entity transitions and discourse relations. Second, they created a unique Reinforced Neural Extractive summarisation (RNES) model that combines reinforcement learning with neural extractive summarisation and coherence. During the training of RNES, the output of the neural coherence model is utilized as immediate rewards so that it can learn to extract coherent summaries. Narayan et al. [166] stated in their study that cross-entropy training is not ideal for extractive summarisation. Models trained in this manner are prone to producing overly verbose summaries with excessively long words and redundant information. They proposed a model for overcoming these challenges by globally optimizing the ROUGE evaluation measure and learning to rank sentences for summary creation using a reinforcement learning objective. Liu et al. [73] produced Wikipedia articles by presenting the challenge as a multi-document summarisation task and used decoder-only networks for summarising. The suggested model by Mehta et al. [167] is comprised of four primary blocks: an LSTM-based sentence encoder, a topic modelling-based context encoder, an attention module, and a binary classifier. Overall, they seek to find the probability $p(y \mid s, d)$, where $p(y)$ is the likelihood that sentence s in document d is summary-worthy. Using a sentence encoder, they represented s as an embedding vector of fixed dimensions. Next, they represented each document by the topic extracted by LDA and used those topics to generate an embedding context. The attention model then utilises the sentence and context embeddings to learn how to give weights to various sentence segments. The classifier then utilises the attention module's output and the original context embeddings to determine whether a sentence is summary-worthy or not.

Kobayashi et al. [168] proposed a summarisation method based on embeddings and document-level similarity (i.e. distributed representations of words). A word's embedding represents its meaning. A document is a bag of sentences, whereas a sentence is a bag of words. The task is described as the challenge of optimizing a submodular function defined by the negative sum of the distances between nearest neighbors on embedding distributions (i.e.

a set of word embeddings in a document). They concluded that document-level similarity can determine interpretations that are more complicated than sentence-level similarity. Chen et al. [169] present an ATS system for the summary of a single document based on a reinforcement learning algorithm and a RNN sequence model with encoder-extractor network architecture. Using a sentence-level selective encoding approach, the summary sentences are retrieved once the significant characteristics have been identified. For the machine-learning-based summariser, the summary problem is transformed into a sentence-level supervised classification problem. Using a training set of documents, the system learns by example to categorise each sentence in the test document as summary or non-summary (i.e. a collection of documents and their respective human-generated summaries). Moratanch et al. proposed a machine learning-based summariser with the following sentence scoring steps [170]: 1) extracting features from the preprocessed document (i.e. based on multiple features of sentences and words); and 2) feeding the extracted features to a neural network that generates a single output score.

2.4.2 Abstractive Approaches

Deep linguistic analysis was employed by Baralis et al. [171] to generate abstract summaries by analysing semantic graphs utilising neural networks. Minimal Recursion Semantics (MRS) was utilised for the semantic modelling of grammars. Using the maximum entropy model, they accomplished disambiguation. This model was created to address the alignment issue of AMR graphs. MRS may be used for both parsing and generating text. Niu et al. [172] suggested an abstractive summarisation method for multiple documents utilising chunk graphs and neural networks. They utilised a Recurrent Neural Network Language Model to evaluate the linguistic quality of each sentence, which aided in the production of understandable abstract summaries. The input sequence is mapped to the output sequence via a simple sequence-to-sequence model. Jobson et al. [61] created the summaries using encoder-decoder RNN and LSTM. They employed word-embedding for training and the attention function to generate the context vector at every time step. The attention model and RNN were utilised by Nallapati et al. [173] to model keywords and capture the hierarchical structure between the sentence and word. They utilised a bidirectional encoder with GRU-RNN (GRU, which is used to address vanishing and exploding gradient issues) and a unidirectional decoder using GRU-RNN. They used the attention model to the source's hidden states and the softmax layer to the target. Rush et al. [65] designed a feed-forward neural network

which worked on sentence-level text summarisation. For sentence-level summarisation, the attention-based encoder and beam-search-based decoder were utilised. Similar to Rush et al., Chopra et al. [174] employed the encoder-decoder model and the conditional RNN to address the problem. Rossiello et al. [175] created grammatically valid abstractive summaries using neural networks, RNN, and probabilistic models. They utilised both prior knowledge and neural networks to analyse the problem. Liy et al. [176] generated abstract summaries using the sequence-to-sequence encoder-decoder paradigm. They examined the text's latent structured information to enhance the quality of summaries. The summary was generated using the recurrent generative decoder to transform the source code into hidden states and then back to original word-sequences. Song et al. [52] introduced a deep learning-based method called the Long Short-Term Memory encoder-decoder model, in which phrases were utilised as input to generate abstract summaries rather than words. Using a sequence-to-sequence encoder-decoder based Convolutional Neural Networks model, Fan et al. [177] developed a customised controlled abstractive summarisation approach. They constructed the summary based on the user's selections, such as the entity whose information they desire, the size of the summary, and the portion of text whose summary they desire. Even though deep learning has been effectively deployed and has emerged as one of the most promising methods for creating abstract summaries, the availability of large, high-quality corpora for training purposes remains a difficulty. In addition, the majority of corpora are old and lack modern morphological, semantic, and syntactic properties. In addition to this, the majority of corpora are only available in English. Modaresi et al. [178] have developed a method for creating a single document corpus in response to the aforementioned issues. Lin et al. [179] employed the Seq2Seq model in conjunction with the attention mechanism to address the issue of repetitions using Seq2Seq models. Abstractive summarisation was performed using convolutional gated units together with global encoding on the encoder side and unidirectional LSTM on the decoder side. In [10], an approach of abstractive summarisation was presented; it addressed sentence repetition and incorrect information. The model suggested by See et al. is comprised of a single-layer bidirectional LSTM encoder, a single-layer unidirectional LSTM decoder, and the sequence-to-sequence attention model presented by [173]. The See et al. approach creates a long text summary as opposed to one or two-sentence headlines. In addition, the attention mechanism was utilised, and the attention distribution aided in the generation of the next word in the summary by informing the decoder where to explore in the source words. This approach created the weighted sum of the encoder's hidden state

to assist the production of the context vector, which is a representation of the input with a defined size. The decoder's generated probability (Pvocab) was used to build the final prediction utilising the context vector and the decoder's last step. In addition, the value of Pvocab for Out Of Vocabulary (OOV) words was equal to zero. In [180], RL was used to abstractively summarise text. RL is combined with supervised word prediction which consisted of a bidirectional LSTM-RNN encoder and a single LSTM decoder. In [58] LSTM-RNNs were used to construct abstract summaries. Using a bidirectional RNN, incorporated past and future context while on the decoder side making predictions and solving the summary imbalance issue is considered. Two LSTMs comprise the bidirectional decoder: the forward decoder and the reverse decoder. The forward decoder decodes information from left to right, whereas the reverse decoder does the opposite. The beginning input of the backward decoder is the final hidden state of the forward decoder, and vice versa. In addition, a method is presented for bidirectional beam-search that creates summaries from the proposed bidirectional model. Bidirectional beam search combines previous and future information to provide a more accurate summary. Consequently, the output summary was well-balanced by incorporating both past and future information and a bidirectional attention mechanism. In addition, the input sequence was read in reverse based on the conclusion [181, 182] that LSTM learns better when reading the source in reverse order while remembering the target's order. The output of the decoder was used as input for a softmax layer on the decoder side to determine the likelihood of each target word in the summary over the vocabulary distribution. The decoder output is dependent on the internal representation of the encoder, i.e. the context vector, the current hidden state of the decoder, and the previously created summary words by the decoder's hidden states. Training is intended to increase the likelihood of alignment between the sentence and summary in both directions. During training, the forward decoder receives the preceding reference summary token as input.

Liu et al., [183] proposed an adversarial framework that the generator, bi-directional LSTM encoder and attention-based LSTM decoder, takes the original text as input and generate the summary. They used RL (i.e. policy gradient) to optimize the generator and implemented the discriminator as a text classifier that is trained to label the generated summaries as machine or human-generated. Scialom et al., [184] introduced an approach using Discriminative Adversarial Search (DAS), their method differs from GANs in that the generator parameters are not updated at training time and the discriminator is based on a seq2seq architecture that is trained to distinguish human-generated texts from machine-generated ones.

The discriminator is integrated into a beam search that obtains a label at each generation step to refine the probabilities and select the top candidate sequences. They utilized the Unified Language Model for natural language understanding and generation (UniLM) proposed by [185] based on BERT for generator. Rekabdar et al., [186] applied generative adversarial networks with an attention mechanism for abstractive text summarisation task. The data generator is modelled as a stochastic policy in RL. The generator is based on LSTM encoder-decoder with attention mechanism and the discriminator is based on CNN aims to distinguish that a summary came from the training data rather than the generator. Hence the generator model is updated by employing a policy gradient and Monto Carlo search based on the discriminator's expected end reward. Dang et al., [187] presented a custom-built GAN model which contains one generator and two discriminators. The generator consists of an encoder and a decoder based on LSTM that is responsible for encoding the input sentences in shorter lengths. Whereas, out of the two discriminators, one is the similarity discriminator which is a four-class (Similar class, Incomplete class, Redundant class, Irrelevant class) CNN based text classifier. The other one is Readability Discriminator, a CNN-based model that tells whether the summary is generated by the generator or human.

Sentence Fusion and Compression Techniques

Sentence compression [188] and sentence fusion are the two most often utilised abstraction approaches [189]. Sentence compression is the process of reducing the length of a sentence by eliminating or omitting irrelevant components. In contrast, sentence fusion is a text-to-text generation approach that fuses sentence fragments from numerous sentences to generate a sentence that is more informative than the individual sentences under consideration. Multiple-Sentence Compression (MSC) is another name for sentence fusion [190].

Sentence Fusion Approaches For aligning and condensing numerous sentences into one, discourse structure and dependency graphs have been proven to be beneficial [189, 191, 192]. Mehdad et al. [193] build an entailment graph over sentences for sentence selection to combine. Success in sentence fusion has also been demonstrated via graph-based representations and abstract meaning representation [194, 190]. By concentrating on discourse connectives, Geva et al. [195] employ a Transformer to combine sentences in pairs.

The importance of remaining faithful to the source text has been highlighted for sentence fusion in recent summarisation studies. Seq-to-seq models are used by Cao et al. [45]

to rewrite templates that are prone to include irrelevant details. It has been successful to include more information, such as entailment and dependence structure, into a seq-to-seq model [196, 197]. The study by Falke et al. [198] appears to be the most similar to the human judgement, they discover that the PG model, while having a lower ROUGE, is more accurate than Fast-Abs-RL and Bottom-Up. They demonstrate that 25% of the results from these cutting-edge summarisation methods do not accurately reflect the source text. Similar research by Cao et al. [199] reveals that 27% of the summaries produced by a neural sequence-to-sequence model include errors.

Sentence Compression Approaches A line of research for sentence compression has focused on neural network methods due to the advances in computational power and data amount. These techniques utilise labelled data to automatically extract characteristics, providing promising results. Filippova et al. [200] used sequence-to-sequence learning to apply LSTMs to sentence compression for the first time. They created nearly two million sentence-compression pairings, with promising outcomes. Andor et al. [201] presented a neural network architecture for deletion-based sentence compression utilising over 2.3 million instances, similar to [200]; they represented words with word embedding, dependency labels, and part of speech tags, and slid a window left-to-right to make decisions based on the local context. However, neither of their large deletion-based sentence compression datasets are publicly available. In the same year, however, Klerke et al. [202] used eye-movement information as external knowledge in a multi-task learning manner to attain equivalent performance. In the case of small training datasets, their research suggests that external information often promotes sentence compression.

Using auto encoders and reconstruction objectives is a commonly suggested unsupervised framework for sentence compression [203, 204, 205]. These techniques are predicated on the premise that a good sentence compression may be inferred from the original text. Wang et al. [206] is an example of prior work on unsupervised sentence compression using reinforcement learning. They employ a Deep Q-Network to maximize a reward that incorporates probability from an n-gram language model and grammatical restrictions. This model deletes a token repeatedly until it terminates. Zhao et al. [207] employ RL to maximize the score of a syntax-focused language model and it is initialized by a supervised sentence compression model.

2.4.3 Query-based Approaches

Much research has been conducted on query-based summarisation. A variety of conventional non-neural network-based approaches with query dependent features are proposed for query-based summarisation, including tf-idf cosine similarity [208], WordNet similarity [209], word matching [210], word co-occurrence [211], etc. Since a deep-learning-based approach is proposed in this thesis, recent approaches based on neural networks are described. Nema et al., [37] introduced a typical encode-attend-decode model (based on LSTM) for query-based abstractive summarisation which first computes a vectorial representation for the document and the query, and then the decoder produces a contextual summary one word at a time. Each word is produced by feeding a new context vector to the decoder at each time step by attending to different parts of the document and query. Li et al., [212] designed a neural network architecture that consists of a sentence encoder, a query filter and a document encoder. A bi-GRU sentence-level encoder is proposed to encode a sentence in a document and then a query filter component attention model upon the sentence encoder is designed to inject such information into sentence encoding and computing the new sentence encoding including the query information. In the end, a feed-forward neural network is applied to compute a salience score for each sentence. Cao et al., [213] used CNNs to project the sentences and queries onto the embeddings and pooling layer with the attention mechanism are applied to combine the sentence embeddings and form the document embedding in the same latent space. In the end, Ranking Layer Rank a sentence according to the similarity between its embedding and the embedding of the document cluster. They proposed two bi-directional LSTM encoders for generating vectors, one for query and another for source document, and an LSTM decoder that outputs the summary. Ishigaki ei al., [38] introduced three copying mechanisms designed for query-based abstractive summarisers: copying from the source, copying the overlapping words, and copying the overlapping words and their surroundings. In the copying mechanism, two different probabilities for every word in the vocabulary are considered, the generation probability and the copying probability. Zhao et al., [214] have designed a dataset for Chinese query-based document summarisation and also implemented three solutions for the task of query-based document summarisation, which are 1) relevance-based summarisation that selects the 6 text pieces and then the top 3 text pieces of the document according to the query are selected. 2) ranking with dual attention that first prepares the representations of the query and the document with pre-trained word embeddings then calculates five various feature scores with respect to different aspects of the samples. 3) a simple BERT-based en-

coder and a text-pair classification are used to determine whether each text piece belongs to the summary which performed better than the other two methods. For query-oriented multi-document summarisation, Zhong et al. [215] suggested an unsupervised deep architecture based on Restricted Boltzmann Machines. Yousefi-Azar et al. [216] suggested an unsupervised query-oriented extractive text summarisation technique employing deep autoencoders and term-frequency (TF) characteristics. Additionally, they developed an ensemble of noisy auto-encoders that adds noise to the input vectors and picks the highest-ranked phrases from an ensemble of noisy runs. Using a Relevance Sensitive Attention-based model, Baumel et al. [217] produced multi-document query-specific abstractive summaries.

Recent query-based text summarisation approaches have been studied in this section to examine their ability for being used for question-driven text summarisation. The main goal in query-based text summarisation approaches is to summarise the retrieved relevant information to the query but in the question-driven text summarisation, answer detection and explaining that answer in a summarised form is desired. Accordingly, the query-based text summarisation approaches are not adaptable for a question-driven summarisation problem. A number of question-driven summarisation techniques have been proposed recently which are described in chapter 4 and chapter 5.

2.4.4 Hybrid Text Summarisation

The hybrid approaches combine both the abstractive and extractive approaches and their advantages [7]. Wang et al., [218] proposed a hybrid system for long text summarisation "EA-LTS". Their system consists of two phases: 1) the extraction phase for selecting the key sentences using a graph model, and 2) the abstraction phase is a RNN based encoder-decoder in addition to a pointer and attention mechanisms to generate summaries. Bhat et al., [156] proposed a single-document hybrid ATS system called "SumItUp" consisting of two phases as follows: 1) Extractive Sentence Selection: uses some statistical features, 2) Abstractive Summary Generation: the extracted sentences are fed to a language generator (i.e. a combination of WordNet, Lesk algorithm and part-of-speech tagger) to convert the extractive summary to the abstractive summary. Subramanian et al., [219] designed a method to produce abstractive summaries for long documents. They designed a simple extractive step using a hierarchical bidirectional LSTM seq2seq sentence pointer before generating the summary. This step reduces the amount of context for a subsequent abstractive step before utilizing a single trained transformer language model for formulating abstractive summarisation. Chen

et al., [220] presented a novel multi-task learning-based abstractive summarisation approach that incorporates extractive summarisation as a supplementary task. Their framework is composed of five components: : (1) Word-level bidirectional GRU encoder for encoding the sentences word-by-word, (2) Sentence-level bidirectional GRU encoder encodes the document sentences, (3) Sentence extractor for labeling each sentence, (4) Hierarchical attention facilitates generating the sentence-level and word-level context vectors to be consumed in the decoding steps, (5) GRU-based decoder for decoding the output word sequence with a beam search algorithm.

Jin et al., [221] proposed a multi-granularity hierarchical structure for their hybrid text summarisation model. They unified extractive and abstractive summarisation into one architecture. Extractive summarisation works on sentence granularity and directly conducts the sentence representations while abstractive summarisation is designed for operating on word granularity and their representations. They exploited the attention mechanism to encode the connections between the same semantic granularity and hierarchical relationships for the different semantic granularity. The decoding part comprises a sentence extractor and a summary generator. The guided generation approach put forward by [222] used a mix of abstractive and extractive strategies. In order to represent important information, the extractive technique creates keywords that are then encoded via a key information guidance network (KIGN). The suggested technique also used a prediction guidance mechanism to predict the ultimate summary of the long-term value [223]. A feedforward single-layer neural network that anticipates the essential details of the final summary during testing is referred to as a prediction guidance mechanism. The suggested model's basic encoder-decoder architecture is comparable to that given out by Nallapati et al. [173], who used both the bidirectional LSTM encoder and the unidirectional LSTM decoder. The attention mechanism and softmax layer were used in both models. Additionally, by suggesting KIGN, which takes the keywords retrieved using the TextRank algorithm as input, the process of producing the summary was enhanced. The final forward hidden state and the first backward hidden state are joined to represent critical information in KIGN. The attention system and pointer mechanism are used by KIGN. The output of KIGN will be sent to the attention mechanism in order to detect keywords because, in general, the attention mechanism hardly ever recognises the keywords. As a result, the keywords will have a significant impact on the attention mechanism. The encoder context vector and hidden state of the decoder will be sent to the pointer network, and the output will be used to compute the soft switch, in order to allow the pointer network

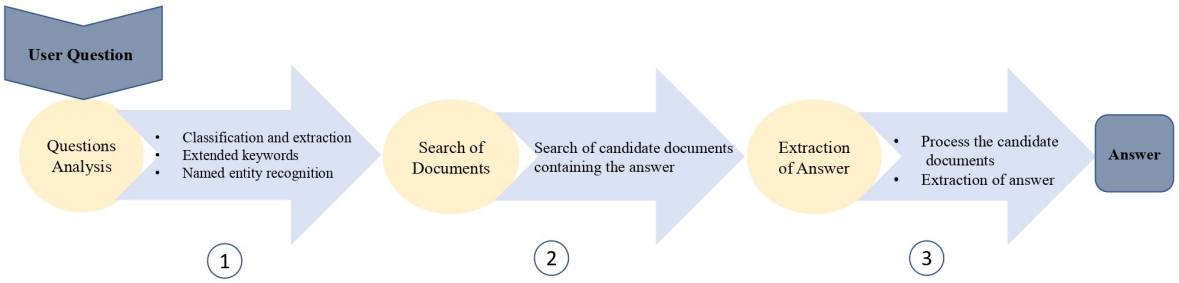

Figure 2.14: Framework of a QA system [8]

to detect the keywords, which are the output of KIGN. The soft switch controls whether to generate the target from the target's vocabulary or copy it from the original text.

2.5 Question Answering (QA) Systems

Question Answering (QA) aims to provide answers to questions formulated in natural language. Question Answering Systems provide an automated method for acquiring answers to questions posed in natural language. Numerous QA studies have categorised Question Answering systems based on many criteria, including user-entered questions, data base characteristics, the type of generated replies, and question answering methodologies and strategies. Question Answering systems typically adhere to a pipeline architecture with three key modules: Question Analysis, Passage Retrieval, and Answer Extraction [8]. Flow of the QA framework is depicted in Fig 2.14.

Question Analysis Module This module's activities consist of parsing, question classification, and query reformulation [224, 225]. This means that the query is analysed for representation of the main information required to answer the user's query; the question is classified according to the keyword or taxonomy used in the query, which leads to the expected answer type; additionally, the query is reformulated for improving question phrasing and the query is transformed into semantically equivalent ones, which aid in the information retrieval process.

Paragraph Retrieval Module A conventional search engine can be used to extract a set of significant candidate passages or sentences from a knowledge base while doing passage re-

trieval. This step utilises the queries generated by the question analysis module and searches information sources for appropriate answers to the submitted questions. Candidate answers from dynamic sources such as the Internet and online databases can also be included. Text retrieval structures divide the retrieval process into three stages: retrieval, processing, and ranking [226].

Answer Extraction Module Answer extraction is a crucial component of a Question Answering system. It generates the answer from the retrieved passages. It accomplishes this by first generating a collection of candidate answers from the produced passages, followed by a ranking of those answers using scoring functions.

2.5.1 Question Analysis

The Question Analysis phase has dual objectives. On the one hand, it tries to simplify the retrieval of question-relevant documents, for which a Query Formulation module is frequently used to build search queries. On the other hand, it is anticipated that using a Question Classification module to forecast the type of the provided question, which results in a collection of expected answer types, would improve the performance of the Answer Extraction step. Query Formulation often use linguistic techniques such as POS tagging, stemming, parsing, and stop word removal, to extract retrieval keywords. However, the phrases used in questions are frequently different from those in the documents containing the correct answers. This problem, known as "term mismatch", is a long-standing and crucial issue in Information Retrieval (IR) [9]. A simple illustration of this stage is given in the leftmost grey box of Fig 2.15.

Question/Query Expansion

To address the "term mismatch" problem, query expansion [227] and paraphrasing techniques [228, 229, 230, 231], are often designed to produce additional search words or phrases so as to retrieve more relevant documents. This subsection explains recent question expansion (reformulation) approaches proposed for QA systems. According to GOLDEN Retriever [232], the query reformulation task can be recast as an MRC task because they both take a question and some context documents as inputs and aim to generate natural language strings as outputs. The query expansion module in GAR [233] is built using a pre-trained Seq2Seq model BART [92] to take the original query as input and generate new queries. The model is trained with various generation targets: the answer, the sentence containing the

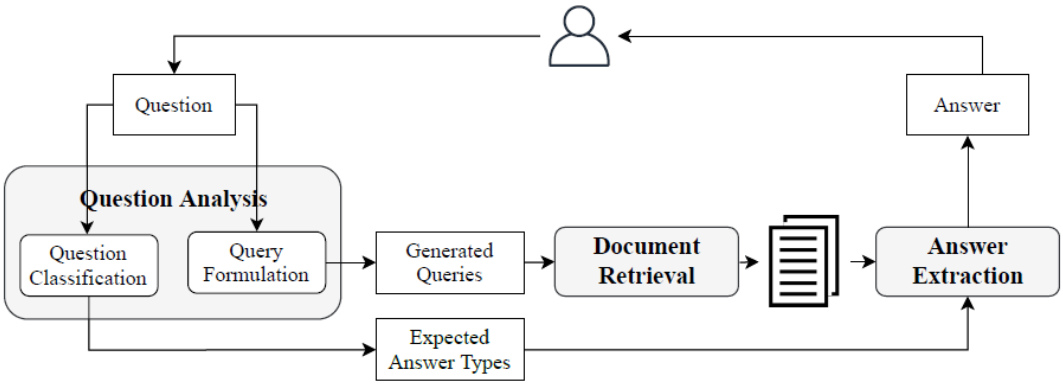

Figure 2.15: An illustration of traditional architecture of a QA system [9].

answer, and the passage title. Some other works generate dense representations to be used for searching in a latent space. For example, Multi-step Reasoner [234] employed a GRU [60], taking token-level hidden representations from MRC and the question as input to generate a new query vector. The new query vector is then trained using Reinforcement Learning (RL) by comparing the extracted answer to the ground-truth. Xiong et al. [235] uses a pre-trained masked language model (such as RoBERTa [89]) as its encoder, which concatenates all previous passages and the question representation to encode a dense query.

2.5.2 Machine Reading Comprehension (MRC)

Machine reading comprehension (MRC) is a fundamental task of textual question answering (QA), in which each question is given a related context from which to infer the answer [236] (step 3 in Fig. 2.14). The purpose of MRC is to extract the correct response from a given context or construct a more comprehensive answer based on the environment. MRC provides the potential to bridge the gap between human and computer interpretation of natural language.

The breakthrough of the sentence encoder, from the basic LSTM to the pre-trained transformer-based model [237], which has significantly improved the performance of all MRC models, is essential to the advancement of MRC research. Moreover, the attention mechanisms between the context and the question might result in improved performance for neural network-based models [238]. Moreover, approaches such as answer verification [239], multi-hop reasoning [240], and synthetic data augmentation might be useful.

Traditional reading comprehension question answering systems rely on a pipeline of NLP

models that heavily employ language annotation, structured world knowledge, semantic parsing, and other NLP pipeline outputs [241]. The availability of large-scale benchmark datasets and the capacity to train large-scale end-to-end neural network models have contributed significantly to the recent advancements in machine reading comprehension. Children's Book Test [242] and CNN/Daily Mail [241] are the first large-scale datasets for reading comprehension tasks. These datasets, however, are cloze-style, where the objective is to anticipate the missing word (typically a named entity) in a text. In addition, Chen et al. demonstrated that these cloze-style datasets need less reasoning than was previously believed [243]. The SQuAD benchmark dataset is more difficult than previous benchmark datasets, with the goal to extract an arbitrary answer span from the original text.

The key to the MRC task is incorporating the question context into the paragraph, in which attention mechanism is most widely used. Despite a variety of model structures and attention types [244, 245, 238, 246, 247], a typical attention-based neural network model for MRC encodes the symbolic representation of the question and passage in embedding space, then identifies answers with specific attention functions in that space. Regarding question and passage attention or matching approach, these attention-based models are classified into two main groups: one-way and two-way attention. Hermann et al. are the first to use attention-based neural network approaches to the MRC problem and to introduce an attentive reader and an impatient reader by utilizing a two-layer LSTM network [241]. Chen et al. [243] construct a bilinear attention function based on the attentive reader that demonstrates higher performance on the CNN/Daily Mail dataset. However, a portion of the information may be lost while summarising the question, and it is more appropriate to focus on the question and passage terms with more precision.

The two-way attention model therefore decomposes both the question and the passage into their respective word embeddings and computes the attention in a two-dimensional matrix. The majority of top-ranked SQuAD approaches are based on this attention mechanism [246, 248, 249]. Cui et al. [244] and Xiong et al. [249] introduce the co-attention method to improve the coupling between the question and document representations. Seo et al. propose a bi-directional attention flow network to capture relevance at various granularity levels [238]. Further, Wang et al. [246] provide the self-attention method to modify the representation by comparing the passage against itself in order to better capture the global passage information. Huang et al. provide a fully-aware attention mechanism with a unique idea of word history [248]. Wang et al.[250] propose a hierarchical attention network that leverages

both co-attention and self-attention mechanisms in different layers to capture the relevance between the question and passage at various granularities. In contrast to the aforementioned techniques, they developed a fusion function that combines the aligned representation with the original representation from the preceding layer inside each attention.

With the use of BERT Reader, Dense Passage Retrieval [251] estimates the likelihood that a passage contains the answer and the probability that the token is the beginning and end of an answer span. It then selects the most probable answer based on what it calculates. Readers are often developed as graph-based systems to extract answer spans from passages [252, 253]. For example, in Graph Reader [253], the graph is used as input, and Graph Convolution Networks [254] are primarily used to learn the passage representation before pulling the answers from the most probable span. In DrQA [255], various features, such as POS, named entities (NE), and term frequencies (TF), are extracted from the context. The multilayer Bi-LSTM then predicts the span of the answer based upon the inputs, the question, and the paragraphs. As part of this process, argmax is applied across all answer spans to get a final average of answer scores across paragraphs using an un-normalized exponential function. BERTserini [146] provides a reader that works on BERT by removing the softmax layer, which allows for comparison and aggregation across different paragraphs. A Shared Normalization mechanism modifies the objective function and normalizes the start and end scores across all paragraphs to achieve consistent performance gains [256]. This mechanism eliminates the problem of unnormalized scores (e.g. exponential scores or logit scores) for all answer spans.

2.5.3 Open-domain QA vs Closed-domain QA

The QA system can be categorized based on the domain of knowledge that they have, the Question types that they answer, or the input knowledge sourced type Fig 2.16. In this section the QA systems are studied based on their domain of knowledge (open-domain and closed-domain). In closed-domain QA, the focus is on a particular domain of interest where the goal is to retrieve answers to questions within that domain. Instead the term Open-domain refers to systems whose purpose is to answer questions about “anything” [257].

History of Open-domain Textual QA START became the first knowledge-based question-answering system on the Web in 1993 [258], and since then responded to millions of questions from Web users across the world. In 1999, the QA track was added to the 8th TREC

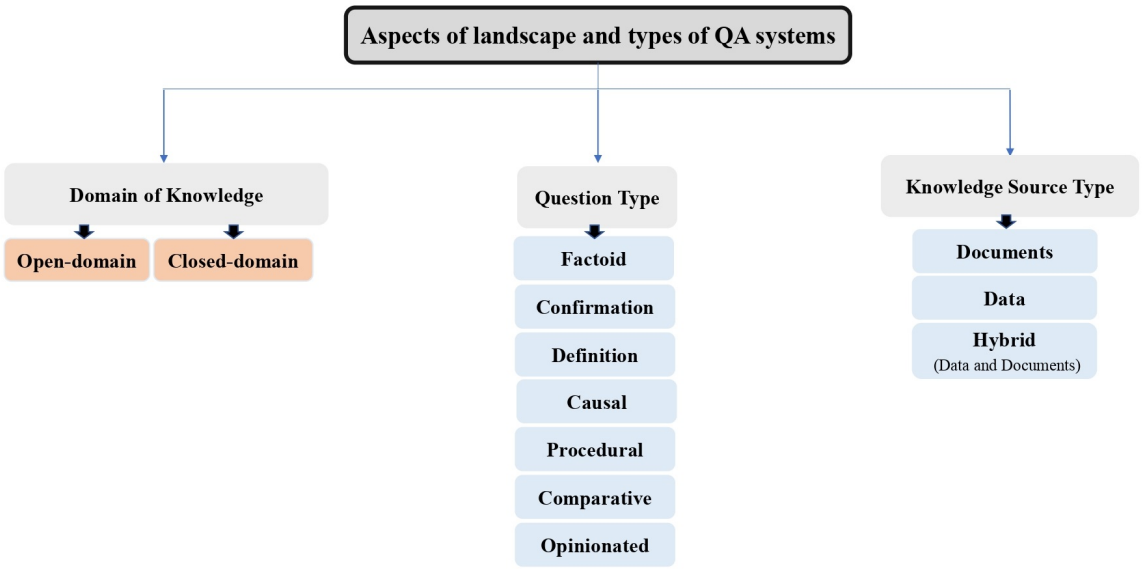

Figure 2.16: Aspects of the landscape and types of the QA systems: (1) Domain of Knowledge (2) Type of Question (3) Type of Knowledge Source

events [259]. At the 38th ACL conference, the following year, a specific discussion subject entitled "Open-domain Question Answering" was introduced. Since then, the open-domain QA system has become a popular topic of discussion in the research world. With the introduction of structured KBs such as Freebase [260], several works, such as WebQuestions [261] and SimpleQuestions [262], have advocated constructing QA systems with KBs. The breadth of these techniques is restricted to the ontology of the KBs; nonetheless, they often achieve high accuracy and almost complete the task for simple questions [263]. In addition, there are pipelined QA strategies that utilise a multitude of data resources, including as unstructured text collections and structured KBs. ASKMSR [264], DEEPQA [265], and YODAQA [266] are the landmark methods. The triumph of IBM Watson [265], who won the Jeopardy! game show in 2011, represents a milestone in this field. This complex system utilised a hybrid design that included technology from IR, NLP, and KB. Recent years have seen the emergence of NLP-based QA systems that can directly perform end-to-end processing of unstructured text sequences at the semantic level using a neural network model [267]. This is made possible by the advent of deep learning. DrQA [255] was the first neural network-based model for open-domain textual QA. On the basis of this paradigm, many end-to-end textual QA models, such as R3 [268], DS-QA [256], DocumentQA [247], and RE3QA [269], have been presented.

Why Deep Learning for Open-domain Textual QA Understanding the motivation behind these techniques for open-domain textual QA is advantageous. Why must we employ deep learning techniques to develop open-domain textual QA systems? What are the benefits of architectures based on neural networks? In this part, the aforementioned questions will addressed in order to demonstrate the following strengths of deep learning-based QA models:

- **Automatically learn complex representation:** There are two benefits to using neural networks to learn representations: (1) It decreases the amount of labor required to hand-craft feature designs. Deep learning provides autonomous feature learning from uncontrolled or supervised raw data [270]; while, feature engineering is a labor-intensive process. (2) Unlike linear models, neural networks can describe non-linearity in data using activation functions like Relu, Sigmoid, Tanh, etc. This characteristic enables the capturing of complicated and sophisticated user-item interaction patterns [270].

- **End-to-end processing:** Many QA systems in the early years depended significantly on the question and answer templates, which were largely manually produced and time-consuming. Later, the majority of QA research embraced a pipeline of conventional NLP approaches, such as semantic parsing, part-of-speech tagging, and co-reference resolution. This might result in the error propagating across the entire process. Neural networks, on the other hand, offer the benefit that several building blocks may be combined into a single (huge) differentiable function and trained end-to-end. In addition, models at various phases can share learned representations and benefit from multitask learning [271].

- **Data-driven paradigm:** Deep learning is fundamentally a statistical discipline, and one of its inherent properties is that it follows a data-driven paradigm. In other words, neural networks can learn statistical distributions of features from vast amounts of data, and the model's performance may be continuously enhanced as additional data are employed [272]. This is crucial for open-domain textual QA, as it often covers a wide variety of domains and a large text corpus.

Open-domain Deep Learning Approaches Lin et al. [256] developed a distantly supervised open-domain QA model that utilises an information retrieval-based paragraph selector to filter out noisy paragraphs and a paragraph reader to extract the correct answer using a multi-layer long short-term memory network. Yang et al. [146] demonstrated an end-to-end

question answering system that integrates a BERT-based reader with the open-source Anserini information retrieval (IR) toolkit to identify answers from a large corpus of Wikipedia articles in an end-to-end fashion. Karpukhin et al. [251] focused on establishing the optimal training procedure utilising a sparse set of question and passage pairs. They designed retrieval solely through dense representations, with embeddings learnt from a modest number of questions and passages using a simple dual-encoder system. Seo et al. [273] introduced Dense-Sparse Phrase Index (DENSPI), an indexable query-agnostic phrase representation model for real-time open-domain QA on SQuAD. In their model, phrase representation combines dense and sparse vectors based on BERT and term-frequency-based encoding, respectively. Qu et al. [274] proposed an open-retrieval conversational QA (ORConvQA) containing a retriever, reranker, and a reader that are all based on fine-tuned BERT and ALBERT based encoders and decoders. They evaluated their model on the OR-QuAC dataset they created for conversational QA. Soni et al. [275] evaluated the performance of various Transformer language models when pre-trained and fine-tuned on different combinations of open-domain, biomedical, and clinical corpora on two clinical QA datasets. They conducted experiments and found that an initial fine-tuning clinical BERT on an open-domain dataset, SQuAD, improves the clinical QA performance.

Closed-domain Approaches Fu et al.[276] introduced a QA system for music using the database ontology knowledge and proposed two approaches to retrieve an answer, the FAQ module, and ontology knowledge. If the answer for the question could not be found in the FAQ module, the system will scan the ontology knowledge base for the appropriate answer by the following steps: question classification, question analysis, and answer extraction. Bhoir et al. [277], proposed a closed-domain QA system for the "Tourism" domain. After pre-processing, sentences containing any number followed by "km", "miles" will be considered as ace sentences and if the question is related to the distance then an accurate answer will be given. Lende and Raghuwanshi [278] proposed a system for closed-domain QA for user queries related to education. An index term dictionary was created for the keywords extracted from a corpus created for the education domain. To obtain the relevant answer, they apply Part-of-speech (POS) tagging to all the filtered documents to find the suitable answer, which contains the same sense as the query. Sarkar et al. [279] developed a knowledge-based QA system, which only understands predefined insurance-related queries. In the first step, the Apache OpenNLP tool is used to detect the query's subject-to-predicate triplets, and then relevant content was retrieved and ranked using matching criteria (query sentence similarity,

sentence length, relative word importance, etc.). Badugu and Manivannan [280] created a closed-domain question answering framework for "Hyderabad Tourism" based on rule-based classification and similarity measures. The corpus is preprocessed, divided into sentences, and then grouped into various inquiry types such as *What*, *Where*, *Who*, and *When*. Sentence retrieval is conducted, based on the question category, and their vectors are generated based on the term frequency and the inverse document frequency of the term. The Jaccard similarity score determines the final answer for each question. A BERT-based clinical question answering system was proposed by Rawat et al. [281], using fine-tuned BERT on medical corpora. Entity-level clinical concepts were integrated into the BERT architecture using the Enhanced Language Representation with Informative Entities (ERNIE) framework. ERNIE extracts contextualized token embeddings using BERT and generates entity embeddings using a multi-head attention model. Godavarthi and Sowjanya [282] built a closed-domain QA system that answers queries from the COVID-19 open research data set (CORD-19). They fine-tuned a BERT model for self-supervised learning of language representations (ALBERT) [90] for retrieving all COVID relevant information to the query. Cai et al. [283] proposed an integrated framework for answering Chinese questions in restricted domains by modeling the question pair, comparing the input question to the existing question, and then identifying the answer output.

2.5.4 Multi-hop QA

Multi-hop question answering requires models to gather information from different parts of a text to answer a question. Most current approaches learn to address this task in an end-to-end way with neural networks, without maintaining an explicit representation of the reasoning process [284]. Rapid progress has been made on multi-hop QA systems with regard to standard evaluation metrics, including EM and F1 [285]. The multi-hop recent works can be divided into two categories, the approaches based on graph Neural Network and hierarchical Coarse-to-Fine modeling approaches.

Graph Neural Network Recent research on multi-hop QA builds graphs based on entities and uses graph neural networks to reason over the generated graph [286, 287]. MHQA-GRN [288] and Coref-GRN [289] built an entity graph via co-reference resolution or sliding windows. Entity-GCN [290] considered three distinct types of edges that connect entity graph entities. HDE-Graph [291] extended the entity graph with document nodes and interactions

between documents, entities, and answer candidates by adding document nodes. Cognitive Graph QA [292] used an MRC model to predict answer spans and likely next-hop spans before organising them in a cognitive graph. DFGN [293] built a dynamic entity graph in which irrelevant entities are softly masked away after each reasoning step and a fusion module is intended to improve the interaction between the entity graph and documents. SAE [294] proposed three types of edges in the sentence graph depending on the named entities and noun phrases present in the query and sentences. C2F Reader [295] employed graph attention or self-attention on an entity graph and claimed that this graph is not always required for multi-hop reasoning. Asai et al. [252] presented a new information retrieval-focused graph-based recurrent technique for identifying evidence documents as reasoning paths. Fang et al. [296] presented a method that generates a hierarchical network, probing interactions on multiple granularities and utilizing distinct nodes to conduct distinct tasks.

Hierarchical Coarse-to-Fine Modeling Prior work on hierarchical modelling for question answering relied mostly on a coarse-to-fine structure. Choi et al. [297] suggested using reinforcement learning to first identify relevant sentences and then generate answers based on these sentences. Min et al. [298] explored the minimal context necessary to answer a question and find that the vast majority of questions can be answered with a small number of sentences. Swayamdipta et al. [299] built lightweight models and merged them into a cascade structure to derive the answer. Zhong et al. [300] suggested utilizing hierarchies of co-attention and self-attention to aggregate information from evidence across multiple documents.

2.6 Summary

The NLP and deep learning concepts related to the problem scenario in this thesis have been presented in this chapter. The approach and related works in text generation using GAN and VAN, different automatic text summarisation approaches, and various QA systems have been described in detail. In text generation, a key barrier is a language's inherent characteristics, such as syntax, grammar, and semantic aspects. The model must learn the correct connection between words and characters to generate a viable text, commonly accomplished through various memories and situations (prior knowledge). Such issues can be addressed in a more robust pre-learning step, in which pre-trained embedding models BERT [237], A lite bert for self-supervised learning of language representations (ALBERT) [90], ELECTRA [91], or

GPT-2 are combined with transformer-based seq2seq architectures to be capable of generating plausible “natural” language text. Transformer-based GANs incorporating contextualized pre-trained language models and stepwise evaluation are blank spots that still need to be appropriately addressed for text generation, which has been presented in this thesis. In this chapter, recent query-based text summarisation approaches have been studied to examine their ability to be used for question-driven text summarisation. The main goal in query-based text summarisation approaches is to summarise the retrieved relevant information to the query, but in the question-driven text summarisation, answer detection and explaining that answer in a summarised form is desired. Furthermore, the query-based text summarisation approaches are not adaptable for the question-driven summarisation problem.

Chapter 3

A Hybrid Extractive-Abstractive Question-driven Summariser Model

3.1 Introduction

An overview of the proposed hybrid text summarisation approach for generating question-driven extractive-abstractive summaries is presented in this chapter. The inputs are a text document and a question as depicted in Fig. 3.1. A high-level introduction to the proposed model is presented in this chapter and more details are provided in chapter 4 and chapter 5.

An open-domain multi-hop QA system is developed to select the answer sentence and extract the supporting sentences for the answer sentence and generates the question-driven extractive summary. To generate high-quality summaries for human consumption, a novel paraphrase generation and sentence fusion model is proposed to rewrite the sentences of the extractive summary and construct a question-driven abstractive summary. In a nutshell, in the proposed novel framework, the advantages of both extractive and abstractive models are exploited. The novel extractive model automatically selects the most appropriate sentences from the document that conveys non-redundant and important information to the question. Subsequently, the novel paraphrasing model based on GAN and transformers followed by a fusion method generates high-quality abstractive summaries so that the resulting summaries are coherent and readable.

As mentioned above, a key advantage of the model is that the extractive phase helps remove redundant information which not only helps improve the quality of the summary generated by the abstractive summariser but also makes it efficient because the abstractive

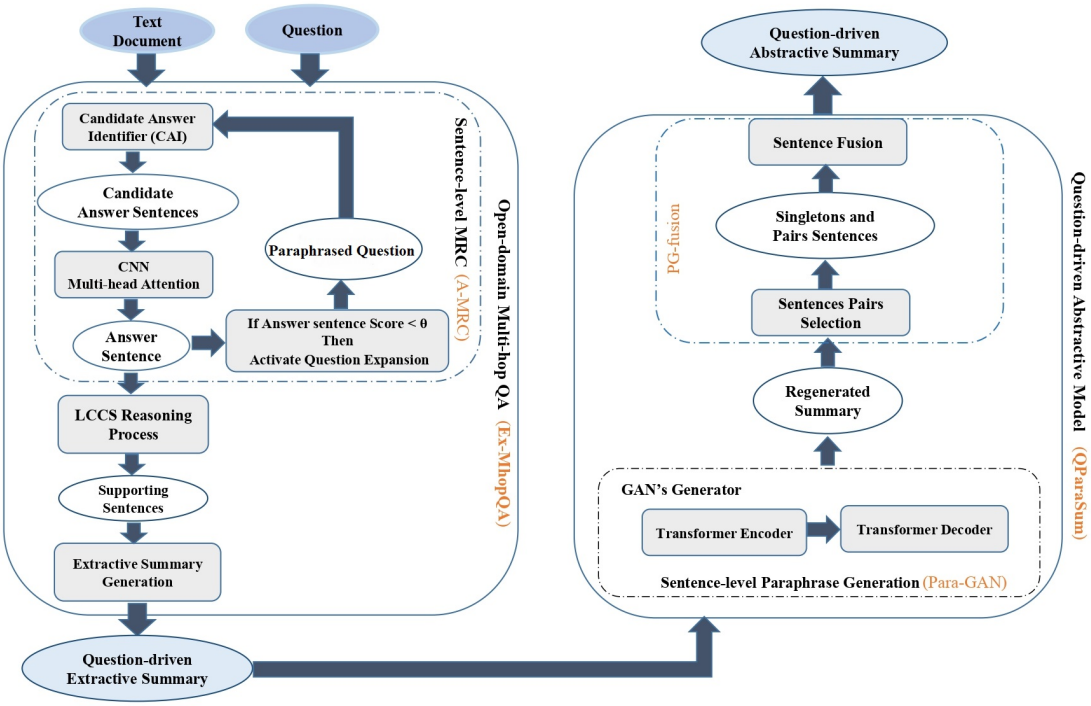

Figure 3.1: The proposed hybrid question-driven text summarisation framework.

phase does not have to deal with a large amount of data. In the subsections below, the overall proposed models for extractive and abstractive question-driven text summarisation are described.

3.1.1 Question-driven Extractive Model (Ex-MhopQA)

A question-driven extractive summariser has been proposed based on an open-domain multi-hop QA system with the name Ex-MhopQA comprising a sentence-level MRC method and reasoning process. The MRC and reasoning components are named A-MRC and LCSS which have been explained below. Candidate Answer Identifier, CNN and multi-head attention-based answer selector and Question Expansion module are MRC components in the multi-hop QA. The Candidate Answer Identifier module is introduced with six functions based on linguistic and syntactic features and patterns for reducing the document to sentences (candidate answer sentences) that could answer the given question. A joint CNN and multi-head attention neural network is designed to analyse and assign a score to each candidate answer sentence based on its relevance to the question. The CNN-attention layer calculates the relevance score based on the correlation of the semantic features extracted from the question and

the candidate answer sentence. If the selected answer sentence score calculated by MRC component is less than Θ, the question expansion module generates paraphrased questions until a candidate answer achieves a score greater than Θ. A lightweight hybrid question expansion is designed based on contextualized embedding and lexical resources (WordNet) that replaces some question keywords with domain-related synonyms to generate paraphrased questions. After selecting the answer sentence, an unsupervised reasoning process (*LCCS* reasoning process) based on ***L***exical ***C***overage and ***C***ontextualized ***S***imilarity is proposed for selecting supporting sentences (justification sentences). All the sentences in the document are considered as the candidate justifications sentences, and those candidates that are closest to the question, answer sentence, and selected justification sentences in the embedding space are selected. A pre-trained BERT and cosine similarity are utilized for measuring the semantic similarity of the candidate justification sentences, the answer sentence, and the question. Also, the lexical coverage of each candidate justification sentence with the answer and the question is calculated and included in the overall score for finding the correct justification sentences. The LCSS reasoning process helps to select appropriate, relevant sentences explaining the answer sentence and then constructs the final extractive summary. The overall framework of designed extractive Ex-MhopQA model is shown in Fig. 3.1. The proposed model is designed to require minimum supervision with the capability of being fine-tuned utilising a small training dataset and being adaptable to different domains. The training procedure for A-MRC model and Para-GAN is designed in two stages. The model will be pre-trained using a large dataset once and be ready to be fine-tuned with small and specialised domain datasets in the second stage.

The justification sentences and answer sentence are rearranged according to their original indexes in the given document to bring coherence in the selected sequence of sentences and generate the question-driven extractive summary. The Ex-MhopQA question-driven extractive model is different with extractive query-based summarisaton models described in the chapter 2. The main goal in query-based text summarisation approaches is to summarise the retrieved relevant information to the query, but in the question-driven text summarisation, answer detection and explaining that answer in a summarised form is desired which the Ex-MhopQA is designed based on this strategy. Furthermore, the query-based text summarisation approaches are not adaptable for the question-driven summarisation problem.

3.1.2 Question-driven Abstractive Model (QParaSum)

The proposed question-driven abstractive model based on paraphrasing is named QParaSum. A paraphrase and sentence fusion framework are proposed to transform the generated extractive summary to an abstractive summary. The proposed paraphraser and sentence fusion framework are named Para-GAN and PG-fusion respectively. The input to this novel model is the extractive summary that is obtained in the previous stage. The details of this framework are presented in the chapter 5. The trained Para-GAN model is used for regenerating the extractive summary and then the paraphrased sentences are analysed to detect the singletons and pairs of sentences to be used in the fusion step.

For rewriting the extractive summary and generating summaries close to human-generated ones, it has been found that GANs are suitable to handle this task because of their ability to generate new samples. The Para-GAN model is based on GAN, transformers, and Q-leaning which evaluates the generated sub-sequence in every step by calculating a score. After regenerating the extractive summary and producing the paraphrases, the next sentence prediction task is utilized to detect the sentence pairs and singleton sentences to find out which sentences could be merged. The PG-fusion fusion model based on Pointer Generator networks [10] consumes the sentence pairs and merges these sentences to improve the generated summary while singleton sentences remained in their paraphrased form. Pointer generator networks are applied to solve various combinatorial optimization and combinatorial search problems. Pointer networks can be said to be derived from the attention mechanism and its potential has been utilized for sentence fusion in the model. In the last step, a question-driven abstractive summary comprising the compressed paraphrased sentences originating from the generated extractive summary is produced. As a result of designing a hybrid question-driven text summarisation model, both question-driven extractive and abstractive summaries are generated as outputs.

The existent abstractive approaches described in chapter 2 are mostly based on RNN and LSTM are not efficient due to using recurrent units and those approaches based on a beam search may not result in an optimal goal and may not even reach a goal at all. The abstractive approaches based on policy gradient have a major disadvantage. A lot of the time, they converge on a local maximum rather than on the global optimum. In comparison to the deep Q-learning, which always tries to reach the maximum, policy gradients converge slower, step by step and they can take longer to train. Regarding the mentioned reasons and drawbacks in the existing approaches, a transformer-based GAN with Q-stepwise evaluation for the ab-

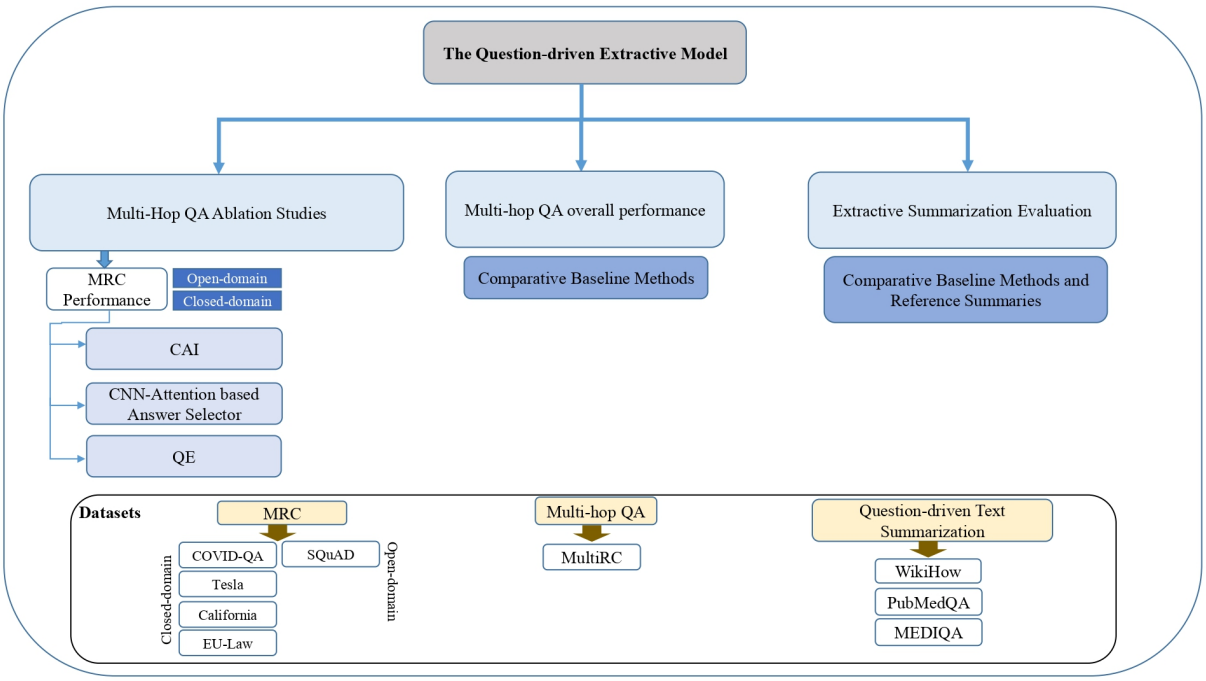

Figure 3.2: The experiments for proposed question-driven extractive model

stractive part is designed, which regenerates and rewrites the generated extractive summaries and produces reliable abstractive summaries. Using transformers architectures with GAN and applying stepwise evaluation for generating text is an unexplored architecture which has been studied in this thesis.

3.2 Experiment Procedure

In this section the detailed experiment procedure designed for extractive and abstractive stages is described. The generated extractive and abstractive summaries are evaluated based on comparison with reference summaries (human-generated summaries) and results of the baseline methods. Ablation studies are designed to investigate the performance of the Ex-MhopQA and QParaSum models by removing certain components to understand the contribution of the component to the overall system. For the extractive Ex-MhopQA model, not only the overall model performance for the generated summaries are evaluated but also the generated answers by the multi-hop QA are compared with strong comparative models, shown in Fig. 3.2 . The A-MRC method is evaluated separately with and without certain components for both open-domain and closed-domain sentence-level QA. The A-MRC and Candidate

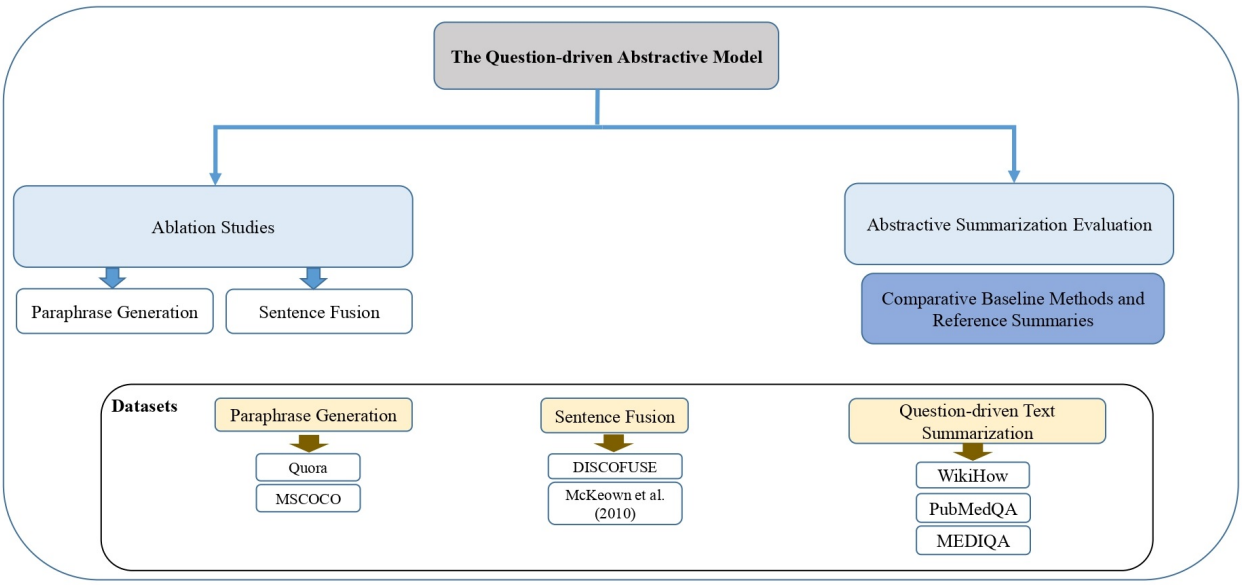

Figure 3.3: The experiments for proposed question-driven abstractive model

Answer Identifier performance for different question categories are examined and reported in chapter 4.

The abstractive QParaSum model is evaluated by comparing the generated abstractive summaries to the reference summaries and different strong baseline methods, depicted in Fig. 3.3. The paraphraser Para-GAN and PG-fusion fusion model are evaluated separately to validate their performance and effectiveness in the framework as well as the unified question-driven abstractive QParaSum summariser.

3.3 Evaluation Metrics

In this section, the evaluation metrics used for evaluating the model and its components are described.

MRC evaluation metrics

Two metrics are adopted including Exact Match (EM) and F1 scores to evaluate the MRC model. The EM score determines the percentage of predictions that perfectly match the ground truth answer, and the F1 score demonstrates the average overlap between the prediction and the ground truth answer.

Multi-hop evaluation metrics

$F1_m$, $F1_a$, and EM evaluation metrics introduced in [301] are used for evaluating the open-domain multi-hop QA.

Paraphrase generation evaluation metrics

Some automatic metrics are used to evaluate the paraphraser model (Para-GAN) framework and compare it with other methods.

- BLEU4 [109] is the most widely used evaluation metric in paraphrase generation. This approach works by counting matching n-grams in the generated sentence and the reference sentence.
- METEOR [302] metric is based on the harmonic mean of unigram precision and recall, with recall taking precedence over precision. Along with the basic precise word matching, it also offers other features which are not found in other measures, such as stemming and synonym matching. METEOR was designed to address some of the flaws in the BLEU metric while also producing a high level of correlation with human judgement at the segment or sentence level.

Sentence fusion evaluation metrics

For sentence fusion evaluation, the BLEU4, METEOR (described above), are used and additionally, the mean Compression Ratio (CR) of the generated sentence fusions is also reported. It is a measure of how concise the generated sentence fusions are with respect to the input sentences. For a given set of input sentences $\{S_1, S_2, ..., S_n\}$, let S be a generated sentence fusion, then $CR(S)$ is defined:

$$CR(S) = \frac{Count(deletedtokens)}{Count(inputtokens)} \tag{3.1}$$

Question-driven text summarisation evaluation metrics

The proposed extractive framework is evaluated by ROUGE metric [122], it compares an automatically generated summary with a set of human-produced summaries. ROUGE-N measures unigrams, bigrams, trigrams, and higher-order n-grams overlap. ROUGE-L utilizes the Longest Common Subsequence method (LCS) to determine the longest matching sequence of

words. A predefined n-gram length is not required since it automatically contains the longest in-sequence common n-grams.

3.4 Summary

In this chapter, an overview of the proposed extractive (Ex-MhopQA) and abstractive (QParaSum) summarisers, experiment procedures, datasets, and evaluation metrics are described. The extractive Ex-MhopQA summariser is a multi-hop QA that comprises an MRC (A-MRC) model and a reasoning process (LCSS). The A-MRC model contains three components, the Candidate Answer Identifier, answer sentence selector, and Question Expansion component. The QParaSum summariser contains a paraphraser (Para-GAN) and a fusion model (PG-fusion). Compared to the existing hybrid text summarisation methods, the proposed model is significantly different in several ways. It consists of two main components, an extractive summariser and an abstractive summariser. First, an extractive summary is generated using the proposed Ex-MhopQA by filtering the irrelevant information and feeding the pruned information to the abstractive component. Then, the Para-GAN paraphraser and PG-fusion fusion model are designed to transform the extractive summary into an abstractive. A transformer-based GAN with Q-stepwise evaluation is designed for the abstractive part, which regenerates and rewrites the generated extractive summaries and produces reliable abstractive summaries. The performance of the components is evaluated by conducting ablation studies to understand the component's contribution to the overall system and their performance independently.

Chapter 4

Question-driven Extractive Model

4.1 Introduction

A question-driven summary must satisfy three goals: answerability, understandability, and persuasiveness. For question-driven summarisation, answer detection and the reasoning on the detected answer are needed. The extractive model selects the answer sentence and its supporting sentences, which provide details or explanations for the answer sentence.

A novel open-domain multi-hop QA model based on a CNN and multi-head attention mechanism is designed to comprehend the document and question for constructing the question-driven extractive summary. The multi-hop QA comprises a sentence-level Machine reading comprehension (MRC) method and reasoning process. MRC is the core task for textual QA, which aims to infer the answer to a question given the related context [303]. The answers could be sentences or paragraphs, or even n-grams. In practice, sentences are a good size to present a user with a detailed answer. For instance, given the question "Why is the Pfizer vaccine better than Sinovac?", one would expect the answer in one or two sentences rather than a single phrase. The task is more challenging compared to others in information retrieval (IR) [304], where the goal is to retrieve a ranked list of relevant documents. Candidate Answer Identifier (CAI), CNN and multi-head attention-based answer selector, and Question Expansion (QE) module are MRC components in the multi-hop QA. The answer selector module measures the semantic dependencies between the local features extracted from the document's sentences and question to select the answer sentence.

The goal of proposing the sentence-level MRC method is to have an MRC style adaptable QA model that could be used for both open-domain and closed-domain QA. The closed-domain QA is as important as open-domain QA since many problems could be addressed by

building domain-specific QA systems.

For example, technology companies building systems for their call agents to answer user queries would benefit from a system that uses their internal call records so their call agents could efficiently get an answer to the questions of their clients. In a further example, students studying a particular subject would benefit from closed-domain QA systems to help them answer questions surrounding their syllabus, rather than using a general open-domain QA system that might retrieve irrelevant answers due to the diversity of topics covered.

Designing an adaptable QA system is challenging because there are a variety of domains, each with its vocabulary, language syntax, and semantics. Ideally, the same computational model would be applied in different domains with minimal human supervision to avoid needing tailor-made models for every domain, which would be time-consuming and expensive.

There are systems in both open and closed-domain QA that have used popular pre-trained neural contextual language encoders such Bidirectional Encoder Representations from Transformers (BERT) [237] and other variants [238, 305]. The language models have achieved near-human, or even better performance, on popular open-domain QA tasks such as SQuAD [306]. Despite this progress in open-domain QA, existing models for closed-domain QA [281, 280, 278, 279, 307] are comparatively less effective and open-domain QA models do not perform as expected for domain-specific questions. The goal in this chapter is to develop a Multi-hop QA system that can be easily adapted to open domain and different closed domains.

Closed-domain QA does not typically have large-scale datasets that could help develop a statistical model and, as a result, many strong open-domain QA models will struggle in closed domains. Applying statistical learning models on small datasets also introduces the problem of reliable generalization thus, the fine-tuning process is divided into two steps: 1) transfer to the task (open-domain training); and 2) adapt to the target domain (closed-domain training). The first fine-tuning step only needs to be done once, but the second step is required each time adapting the model to a new domain. To enable the model to focus on question-relevant sentences, an unsupervised filtering technique is applied to remove those sentences which do not contain an answer to the question. The model also attempts to rewrite some questions (determined by a tuned parameter) to make them less ambiguous.

A novel reasoning approach is proposed for analysing the document regarding the detected answer sentence and searching for relevant supporting sentences based on lexical coverage and contextual semantic similarity.

4.2 Proposed Multi-hop QA Approach

The Fig. 4.1 shows the overall framework of the multi-hop QA which generates the question-driven Extractive summaries.

4.2.1 Adaptable Machine Reading Comprehension Method

In this section, the novel reading comprehension model for sentence-level QA is presented that can be utilized for open-domain and tuned with a small training dataset for various closed domains. CAI module is based on syntactic and linguistic rules has been introduced to reduce the context to the sentences that could contain the answer for the given question (candidate answer sentences). A neural network based on CNN and multi-head attention mechanism is designed to analyse and score the candidate answer sentences. The novelty lies in obtaining different levels of contextual understanding of context sentences and the question by extracting important semantic features and their correlations. The CNN-attention layer assigns a relevance score for each candidate answer sentence selected by the CAI. Also, a QE module is introduced for rewriting ambiguous questions shown in Fig. 4.4, which in spirit, is close to the query expansion technique in Information Retrieval. The key advantage of this module is that it rewrites the question and produces paraphrased versions to help the system select the answer sentence with more confidence. The proposed adaptable MRC method based on CNN and multi-head attention mechanism is called A-MRC.

CAI

Unlike previous approaches, the irrelevant content from context P is filtered out to improve the results. The co-reference resolution is not applied prior to filtering in this component, the linguistic features for each sentence are analysed to determine its capability for answering different question categories (*When*, *Where*, *Who*, *What*, *Why*, *How*). A strategy is developed to classify the context sentences into question categories to facilitate the answer selection. To this end, a popular tool called Giveme5W1H [308] is used, an open-source system that uses syntactic rules to automatically extract the relevant phrases from English news articles for answering the 5W1H questions. The advantage of this tool is that it can be customized towards one's needs. Since the main goal is identifying candidate answers for each question category, different components in Giveme5W1H functions have been customized. New methods and rules are designed to improve and adapt the Giveme5W1H for candidate answer

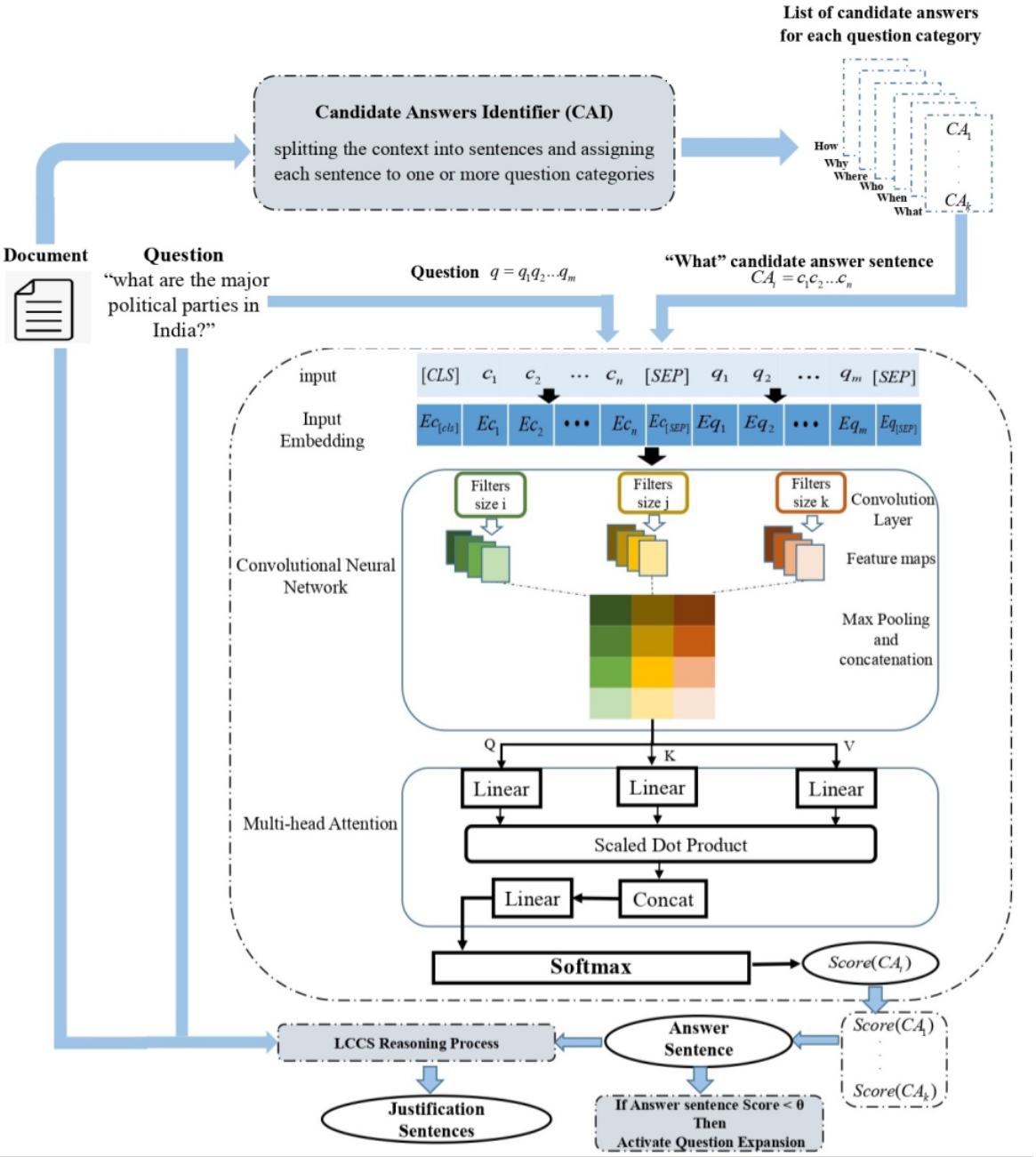

Figure 4.1: The overall framework of the proposed multi-hop QA system for an example "What" question.

selection since the Giveme5W1H does not cover all the syntactic rules for “Why”, “What”, and “Where”. The methods and rules indicated in this section are based on the Giveme5W1H methods which have been evolved by the cognitive process of answering wh-questions studied from different sources [309, 310, 311, 312, 313, 314]. A new parser function is used for finding all types of date-time named entities (NEs) for “When”. Additional methods are added to Giveme5W1H to support all types of “How” questions such as “How many”, “How much” and “How”. Six independent identification functions are performed to retrieve the candidate answers for the six (5W1H) categories. The candidate answer identifier module uses the Giveme5W1H preprocessing steps, gets the context as input, and splits it into sentences to process them separately. After checking all the rules and methods for each sentence, it will be added to correspondent categories, and in the end, a list of candidate answers for each question category will be prepared. More details on different rules and how they have been incorporated into the candidate answer identifier module are mentioned below and depicted in Fig. 4.2.

- When: For detecting all types of temporal NEs including all formats of DateTime, duration, etc the dateparser[1] python package have been added, as well as using SUTime[315] which is used in Giveme5W1H.

- Where: Giveme5w1H only looks for sentences containing tokens classified as NEs of the type Location. A new method searching for movement verbs like “go”, “move”, “run”, “jump”, “bolt”, and others, following by a preposition (“to”, “toward”, etc.) is used to point to a location.

- Who: In Giveme5W1H, the sentences that have the subject are considered as “who” candidate answers. The first noun phrase (NP) that is a direct child to the sentence in the parse tree and has a verb phrase (VP) as its next right sibling is the sentence subject. The sentences containing Person or Organization NEs are considered as “who” candidate answers which are missed in the original Giveme5w1H.

- What: In Giveme5W1H, the “who” candidates that a VP is the next right sibling in their parse tree are considered as “what” candidates. This function has been extended because the original function does not reliably work on many “what” candidate answers. An extra function is added to select sentences as the candidate answers for “what”;

[1] https://github.com/scrapinghub/dateparser

however, the order of tags is not important. The pattern that needs to be looking for is $(Noun^{+}Verb^{*}Preposition^{*}Adjective^{*})$.

- Why: In Giveme5W1H, sentences containing causal conjunctions ("due to", "result of", "because" and "effect of"), causative adverbs ("therefore", "hence", and "thus"), causative verbs ("activate", "implicate", "make to", etc.) are considered as "why" candidates. Two syntactical rules are added for covering all "why" candidate answers. The sentences containing the following sequence(s) are "why" candidate answers. The patterns that should be looking for are:[2]
  $(\text{“}to + VB + IN^{*} + NN/NNS/NNPS/NNP/PRP\text{”})$
  $(\text{“}for + VBG + IN^{*} + NN/NNS/NNPS/NNP/PRP\text{”}).$

- How: Giveme5W1H proposed a combined method consisting of two subtasks, one analysing copulative conjunctions, the other looking for adjectives and adverbs of manner for the "How" category. An extra method is added to search for Money NEs, Percent NEs, or numbers as the candidate answers for "How many" and "How much".

The candidate answer sentences list for questions that do not belong to the 5W1H categories contains all the context sentences.

CNN-Attention based Answer Selector

Given the question q, represented as a sentence, there are K possible candidate answers $CA_1, CA_2, \cdots, CA_k$ which are present in the accompanying context P associated with the q. Question q with m tokens ($q = q_1, q_2, \cdots, q_m$) and candidate answer sentence CA_i with n tokens ($CA_i = c_1, c_2, \cdots, c_n$) are combined together into a single sequence, separated by a special token $[SEP]$ as the input of the CNN attention layer. The output of BERT is taken only for the first token $[CLS]$, which is used as the aggregate representation of the sequence. The semantic representation of q and CA_i is derived by using a pre-trained contextual language model such as BERT or ALBERT for the embedding layer. The advantage is that high-quality representations are derived, which cannot be obtained using methods such as static word embeddings [72, 316]. The goal is to obtain a reliable or most plausible answer CA_j to the question q in P. BERT uses a multi-layer bidirectional transformer [67] network to encode contextualized language representations. Similar to BERT, the ALBERT model introduces two parameter-reduction techniques to lower memory consumption and increase the training

[2]VB, VBG, NN, NNS, NNP, NNPS, PRP, and IN stand for base form verb, present participle verb, singular noun, plural noun, singular proper noun, plural proper noun, personal pronoun, and Preposition or subordinating conjunction respectively.

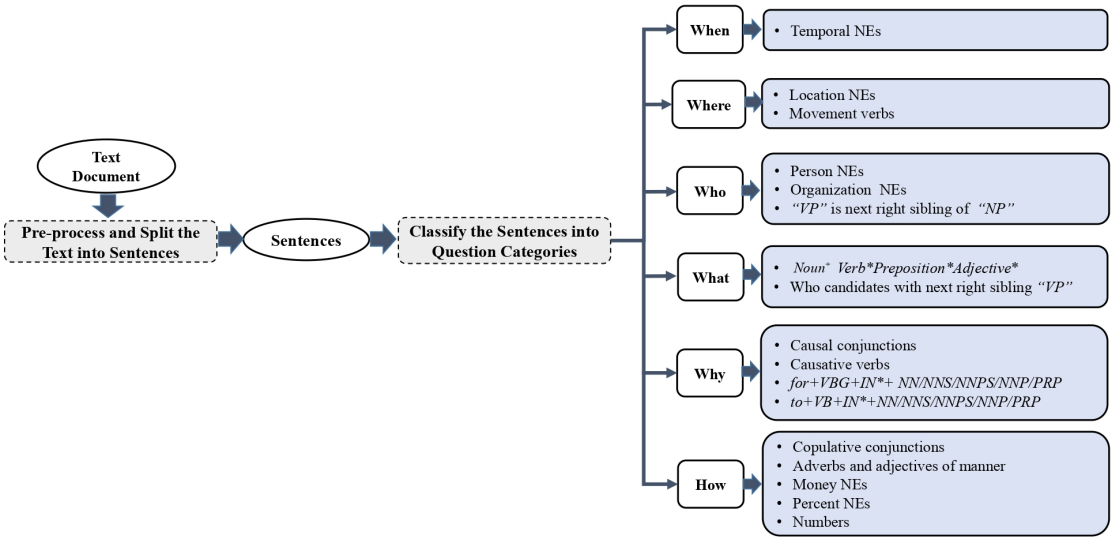

Figure 4.2: Processing the context and identifying the candidate answers for each question category based on linguistic and syntactic patterns and features.

speed of BERT. To calculate the scores for candidate answer sentences, the BERT and ALBERT pre-trained model are fine-tuned with untrained layers of CNN, pooling, and attention. The CNN and multi-head attention mechanism focus the model on the most important features and their correlations when constructing the question and sentence representation.

Convolutional Neural Network

The CNN extracts salient n-gram features from the input sentence to create an informative latent semantic representation of the sentence for downstream tasks [317]. By using the convolution-max pooling operation, local contextual information at the word n-gram level is modeled first. Then, salient local features in a word sequence are combined to form a global feature vector. Finally, the high-level semantic information of the word sequence is extracted to form a global vector representation. Finally, the high-level semantic information of the word sequence is extracted to form a global vector representation.

For each sentence, let $e_i \in R^d$ represent the word embedding for the i^{th} word in the sentence, where d is the dimension of the word embedding, and the given sentence has n words. Convolution is then performed on this input embedding layer. It produces a new feature by applying a filter $K \in R^{hd}$ of size h on a window of h words. For example, a feature

c_i is generated using the window of words $e_{i:i+h-1}$ by (4.1).

$$c_i = f(e_{i:i+h-1}.K^T + b) \tag{4.1}$$

Here, f is a non-linear activation function, for example, the hyperbolic tangent, and $b \in R$ is the bias term. The filter (also called kernel) K is applied to all possible windows (slide over the entire sentence embedding matrix) using the same weights to create the feature map. The sentence with length n is divided into $\{e_{1:h}, e_{2:h+1}, \cdots, e_{i:i+h-1}, \cdots, e_{n-h+1:n}\}$ and perform the filter on each component. The feature map obtained by filter is shown in (4.2).

$$c = [c_1, c_2, ..., c_i, ..., c_{n-h+1}] \tag{4.2}$$

A convolution layer is usually followed by a pooling strategy on each filter to provide a fixed-length output and reduce the output's dimension while retaining the most salient features. The maximum pooling method on each feature map is applied, which gives low dimensions dominant features, as shown in (4.3).

$$\hat{c} = max\{c\} \tag{4.3}$$

The $\hat{c}$ is obtained by one convolution filter along with maximum pooling layer, and a feature sequence obtained with t convolution filters is shown in (4.4).

$$\hat{C} = [\hat{c_1}, \hat{c_2}, \cdots, \hat{c_t}] \tag{4.4}$$

In this stage, important n-gram features of the candidate answer sentence and question are extracted by CNN, and the generated feature vectors should be concatenated to form the new global feature vector matrix Y as the input to the attention layer.

Multi-head Attention Layer

The self-attention mechanism primarily focuses on the internal dependence of input [318]. In the A-MRC model, the attention layer calculates the semantic association between the extracted features from the question and candidate answer sentence to determine the candidate answer's relevance score. In each self-attention mechanism, there is a query matrix (Q), a key matrix (K) and a value matrix (V). The output of the CNN layer, matrix Y, is the the initial value of query matrix (Q), key matrix (K) and value matrix (V), as shown in (4.5).

$$Q = K = V = Y \tag{4.5}$$

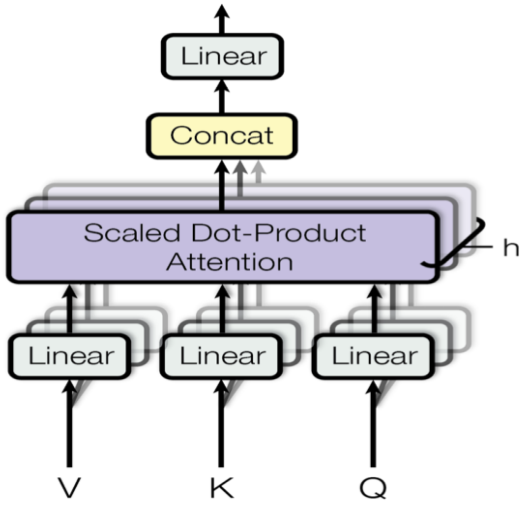

Figure 4.3: Multi-head attention structure

Scaled Dot-product Attention (SDA) is the main concept of the self-attention mechanism. It first computes the similarity by solving the dot product of Q and K, then divides by $\sqrt{d_k}$ (d_k is the dimension of matrix K) to avoid the dot product result from being too large. The result is then normalized using the Softmax function before being multiplied by the matrix V to obtain the expression of attention. SDA operation is depicted in (4.6).

$$SDA(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4.6}$$

The conventional attention mechanism is confined to acquiring attention information from a single level. Multiple linear transformations are performed to the input feature matrix in the multi-head attention mechanism to learn the attention representation of the text for obtaining more comprehensive semantic information [67]. A multi-head attention comprising multiple self-attention mechanism is employed (shown in Fig. 4.3) to assess the semantic connection between the key features of the question and candidate answer sentence for determining the relevance score. Using different parameters ${W_i}^Q$, ${W_i}^K$, ${W_i}^V$ to perform linear transformation is the core idea of the multi-head attention mechanism. Applying the SDA on the linear transformation results is demonstrated by $head_i$, as shown in (4.9).

$$head_i = SDA(Q{W_i}^Q, K{W_i}^K, V{W_i}^V) \tag{4.7}$$

Concatenating the computed results $head_1$ to $head_h$ creates a matrix that is multiplied by the parameter W to complete the final linear transformation. H is the attention value of

the entire sentence, depicted in (4.10), where h is the number of heads in the multi-head attention mechanism.

$$H = MultiHead(Q, K, V) = Concat(head_1, \cdots, head_h)W \tag{4.8}$$

An average pooling is performed on the output matrix of the multi-head attention layer to obtain the feature vector f for integrated CA_i and q. The f is the input through the fully connected layer to the final softmax layer. In the answer selection task, there are two classifier labels (similar = 1, dissimilar = 0). The final layer is modified to get the predicted $Score(CA_i)$ for the similar label, as shown in (4.9) and (4.10).

$$Score(CA_i) = P(C = 1|CA_i, q) \tag{4.9}$$

$$P(C|CA_i, q) = softmax(w_c f + b_c) \tag{4.10}$$

where w_c is the weight matrix, b_c is the bias and C is label. All the candidate answer sentences are ranked based on the obtained scores, and the candidate answer sentence with the highest score is selected as the answer sentence for the question q. Having more than one sentence with label 1 is prevented with this method.

Question Expansion

Some questions are more ambiguous or convey less domain-related information than others [319]. Inspired by research in Information Retrieval, where query terms are expanded with relevant keywords from the vocabulary, a strategy is developed to use more appropriate terms if the question does not convey much information to the model. A parameter θ is introduced where $0 < \theta < 1$, which is automatically tuned from the data and helps to assess whether question expansion is needed. If the selected answer sentence score is less than θ, the question expansion module generates paraphrased versions of the question until a candidate answer achieves a score greater than θ. A lightweight hybrid question expansion is designed based on contextualized embedding and lexical resources (WordNet) that replaces some question keywords with domain-related synonyms. The WordNet is utilised for extracting the expansion terms to minimize the supervision in this component (question expansion) and generate paraphrased versions of the question with less computational cost. The question keywords are extracted by POS tagging the question and removing the symbols, stopwords, and NEs to keep the words most important to the question.

After selecting the keywords, expansion terms are extracted from WordNet considering the keyword's role in the question (for example, if the keyword is an adjective, adjective synonyms are selected accordingly). Thereafter ranking and filtering functions are applied to choose the most appropriate expansion terms for each keyword.

$$Question - Keywords = [K_1, K_2, \cdots, K_m] \tag{4.11}$$

$$\begin{aligned} expansion - list = [&(K_1 : et_1, \cdots, et_w) \\ &, \cdots, \\ &(K_m : et_1, \cdots, et_z)] \end{aligned} \tag{4.12}$$

The domain vocabulary is generated based on the available corpora for the domain and the expansion terms that do not exist in the domain vocabulary are eliminated from the list, and the remaining ones are considered for calculating their relevance to the question. The pre-trained BERT model is trained using the domain-specific corpora to generate domain-specific embedding vectors for expansion terms and question. The semantic similarity between question and expansion terms embedding vectors is calculated to keep those terms that are semantically more related to the question. The expansion terms are ranked regarding their relatedness to the whole question, and those more semantically related to the question are retained.

After finalizing the expansion list, each expansion term is transformed to the appropriate form to get the same POS tag as the keyword (for example, if the keyword is plural Noun(NNS), its expansion term should be the same). Then, each keyword is replaced with one of the expansion terms to form a paraphrased version of the question that conveys the same context. The generated paraphrased versions have the same structure as the original question since only some keywords are replaced with their synonyms. As a result, there is no need to do grammar checking for the generated versions.

For example, *"What are main steps for mitigating the COVID -19 transmission during transport of suspected and confirmed patients?"* is a question from the COVID-QA dataset that needs expansion because its answer sentence score is less than θ. The first step is keyword selection, {*"main", "steps", "mitigating", "transmission", "transport", "suspected", "confirmed", "patients"*} are the question keywords and their expansion terms are extracted by using WordNet (synonyms with the same role as the keyword are selected for each keyword).

After the first step of domain vocabulary filtering, the list of question keywords and their domain-related expansion terms are { *main(adj): major, primary, principal – transmis-*

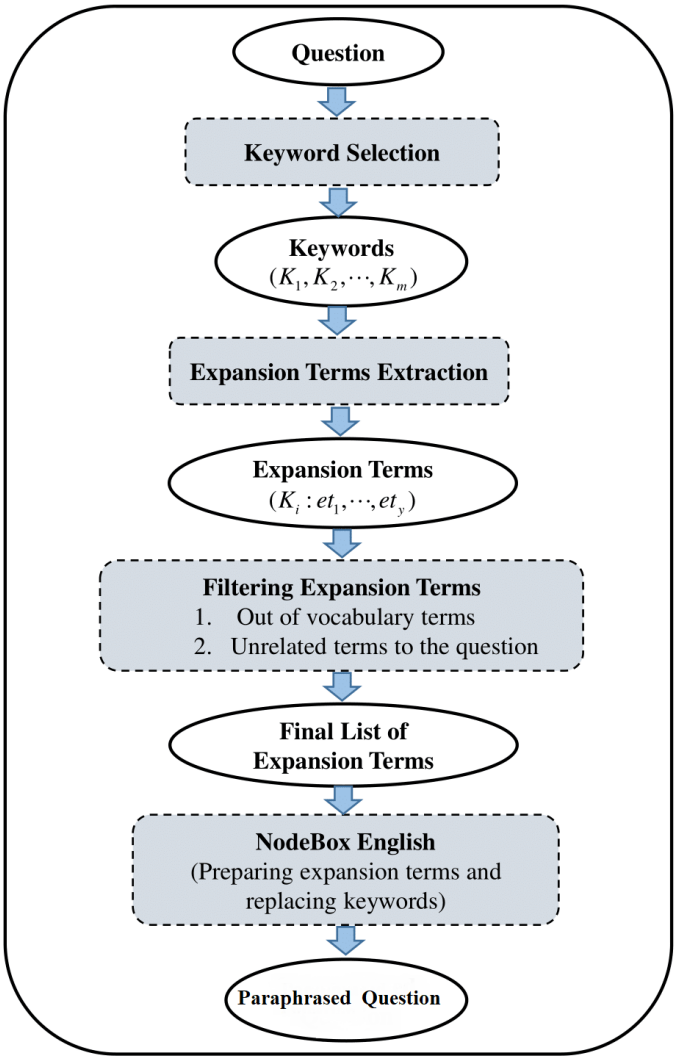

Figure 4.4: The overall steps of the QE module for rewriting ambiguous questions.

sion(noun): infection, contagion – transport(noun): transfer – confirmed(verb): corroborate, affirm, substantiate}.

The second step of the filtering is to measure the expansion terms' semantic relevance to the question. The terms with lower relevance (lower semantic similarity) to the question are filtered for keywords with more than one synonym. The average of the semantic relevance to the question is calculated for all the keyword synonyms and those obtaining the semantic relevance more than the average value (α) will remain for the keyword. After this step, the final list of expansion terms with higher semantic relevance to the question remains {*main: major, primary - transmission: infection - transport: transfer - confirmed: corroborated, affirmed* }. The synonyms are automatically transformed to their appropriate form to get the same POS tag as the keyword, "confirmed" has the "VBN: present participle" POS tag so its synonyms are converted to present participle form.

The paraphrased versions for the question are generated by replacing the keywords with their synonyms. One of the expanded versions of the example question is *"What are major steps for mitigating the COVID -19 infection during transfer of suspected and corroborated patients?"*.

Replacing the keywords (one adjective, two nouns, and one verb for this example) with domain-relevant and question-related synonyms generates other versions of the question with the same meaning. The candidate answers are analysed for the generated paraphrased version of the question to find the answer sentence more accurately. The final selected answer sentence is *"HCWs who handle the transport of COVID-19 patients must consider the following principles: firstly, early recognition of the deteriorating patient; secondly, HCW safety; thirdly, bystander safety; fourthly, contingency plans for medical emergencies during transport; fifthly, post-transport decontamination."* with the score 0.72. If the scores for the selected answers by the paraphrased versions are lower than θ, the answer sentence with the highest score (among all the selected answers) will be chosen. The question expansion module described in Algorithm 1 takes a question as input and generates the paraphrased version of the question in four steps: 1) keyword detection; 2) expansion terms (synonyms) extraction; 3) filtering inappropriate synonyms; and 4) preparing expansion terms and replacing keywords with their corresponded synonyms to generate various synonyms of the question.

Algorithm 1: Question Expansion

Input: Question (q)
Output: Paraphrased versions of Question
question-keywords=Removing stopwords, symbols and NEs(q)
for keyword **in** question-keywords **do**
  expansion-list.Add(find-synonyms(keyword, WordNet(keyword)))
end for
for term **in** expansion-list **do**
  if term **is not in** Vocabulary **then**
    expansion-list.Remove(term)
  end if
end for
for keyword **in** question-keywords **do**
  α=AVG(CosineSimilarity(Emb(q),Emb(keyword.expansion-term(i))) {i in range Size(keyword.expansion-list)}
  {Emb(x) stands for Embedding vector for x}
  {With the α, the synonyms which are more semantically related to the question will be selected for each keyword.}
  for term **in** keyword.expansion-list **do**
    if CosineSimilarity(Emb(q), Emb(term)) $< \alpha$ **then**
      expansion-list.Remove(term)
    end if
  end for
end for
for expansion-term **in** expansion-list **do**
  {Preparing the expansion terms} expansion-term=NodeBox English(expansion-term, keyword.POStag)
end for
for expansion-term **in** expansion-list **do**
  paraphrased-version=Question.Replace(expansion-term, correspondent-keyword)
  Question-paraphrased-list.Add(paraphrased-version)
end for
return Question-paraphrased-list

4.2.2 Reasoning Process

To tackle question-driven extractive summarisation, the content selection process should not be limited to selecting the answer sentence to the given question. It also necessitates human-like reasoning for considering the content interrelationships thoroughly and meticulously across the whole document text. In other words, if the focus is only on the answer sentence for the given question, the resulting summary will likely miss vital information. A reasoning process based on ***L***exical ***C***overage and ***C***ontextualized ***S***imilarity is proposed for selecting justification sentences (LCCS reasoning process). All the sentences in the document (D) are considered as the candidate justifications sentences(JC_i), and those candidates that are closest to the question (q), answer sentence (AS), and selected justification sentences (JS_i) in the embedding space are selected. Pre-trained BERT is utilized for generating the contextualized embedding for the candidate sentences, question, and AS, then the cosine similarity is calculated to generate a contextualized similarity score. Also, the lexical coverage of the candidates with the q, AS, and JS_i keywords (unique terms) in 4.13 ($X = q$,$X = AS$,$X = JS_i$) is measured.

$$C(X, JC_i) = \frac{\mid t(X) \cap t(JC_i) \mid}{max(\mid t(X) \mid, \mid t(JC_i) \mid} \tag{4.13}$$

$\mid t(X) \cap t(JC_i) \mid$ is the size of common terms in X and JC_i and $\mid t(X) \mid$, $\mid t(JC_i) \mid$ are the size of unique terms of X and JC_i.

4.3 Experiments

4.3.1 Experimental Dataset

MRC datasets

The MRC method is evaluated for both open-domain and closed-domain QA. The popular SQuAD [306] dataset is used for open-domain QA experiments. Four closed-domain datasets are used to verify the performance of the proposed model. Three datasets were derived from SQuAD collection due to the limited number of closed-domain QA datasets that are publicly available. The datasets are from three domains with different concepts and different sizes: Tesla (person); California (region); and European-Union-law (system) referred to as EU-law in the results. COVID-QA [320], a SQuAD style Question Answering dataset, was added as the fourth closed-domain dataset for the experiments. The datasets consist of Context-Answer-Question triples. The Tesla, California, EU-law, and COVID-QA consist of 565, 746, 315, and

Algorithm 2: Reasoning Process

Input: Question (q), Document (D), Answer Sentence (AS), size of justification set (J-num)

Output: Set of justification sentences (JS-list) with size J-num

k=1

while ($k <=$J-num) **do**

  for sentence(JC) **in** D **do**

    ASq-score=C(AS,JC)+C(q,JC)+CosSimilarity(AS,JC)+CosSimilarity(q,JC)

    if ($k > 1$) **then**

      JS-score=$\sum_{i=1}^{|JS-list|} C(JS_i, JC) + CosSimilarity(JS_i, JC)$

    else

      JS-score=0

    end if

    Score(JC)=ASq-score+JS-score

  end for

  return JS=(JC with highest score)

  JS-list.Add(JS)

end while

return JS-list

2019 questions, respectively, along with annotated answers and context (see Table 4.6).

Multi-hop QA datasets

MultiRC dataset is used for evaluating the proposed open-domain multi-hop QA model. Multi-sentence reading comprehension (MultiRC) is a reading comprehension dataset administered via a multiple-choice QA task [301]. Each question is based on a paragraph that comprises the question's gold justification sentences.

Question-driven text summarisation datasets

The proposed extractive model is evaluated on three large-scale summarisation datasets, WikiHow [47], PubMedQA [48], and MEDIQA dataset [49].

- **WikiHow** is an abstractive summarisation dataset accumulated from the WikiHow community-based QA website, with each sample consisting of a lengthy article, a non-factoid question, and the associated summary as the answer to the question.
- **PubMedQA** is a biomedical QA dataset derived from PubMed2 abstracts. Each sample includes a question, an article, and an abstractive answer which summarises the context corresponding to the question.
- **MEDIQA** is a dataset comprising 156 consumer-submitted health questions, corresponding articles to these questions, and expert-written summaries of the answers.

4.3.2 Evaluation Metrics

Two metrics including Exact Match (EM) and F1 scores are adopted to evaluate the MRC model. $F1_m$, $F1_a$, and EM evaluation metrics are used for evaluating the multi-hop QA. ROUGE-N and ROUGE-L are utilized for evaluating the proposed extractive framework.

4.3.3 Data Pre-processing and Experimental Settings

Since the CAI module is based on Giveme5W1H, pre-processing steps are implemented the same as Giveme5W1H. The Stanford CoreNLP is used [321] for sentence splitting, tokenization, full parsing, POS-tagging, preprocessing, and preparing the context in the CAI module for candidate answers selection. A two-step training is used for the contextualized CNN-attention answer selector model: 1) transfer to the task (open-domain training); and 2)

adaptation to the target domain. Performing a single fine-tuning for closed-domain requires a large dataset, which is impractical due to the difficulty and cost of collecting training data specific to that domain. Thus, the first step transfers the model to the target task and prepares the model for open-domain QA, and the second step can adapt the model to the target closed-domain with a small training dataset. The Natural Questions (NQ) dataset is utilized [322] consisting of 300,000 naturally occurring questions, along with human-annotated answers from Wikipedia pages, to be used for the first step of training. This dataset provides a whole Wikipedia page for each question which is significantly longer compared to MRC datasets (e.g. SQuAD). Following Liu et al. [323], multiple document spans are generated by splitting the Wikipedia page using a sliding window with the size and stride 512 and 192 tokens respectively to generate the negative (i.e. no answer) and positive (i.e. has answers) spans. Then, the positive spans are only preserved (the span containing the annotated short answer) as the context, and the negative ones were discarded. For both the first and second steps of training, the question sentence pairs were generated by CAI. For the open-domain QA, only the first step of training is applicable. After generating the candidate answer sentences for question categories with CAI, the candidate answer sentence and question pairs were generated for training the CNN attention-based answer selector. The candidate answer sentence which contains the annotated answer gets the label 1, and other candidate sentences get label 0. The first fine-tuning step is done only once, and the second step is performed each time adapting the model to a new domain. The pre-trained BERT base and ALBERT base model are used for token embeddings, consisting of 12 Transformer blocks with 12 self-attention heads and the hidden size of 768. There is no analytical formula to calculate an appropriate value of the hyperparameters to obtain the optimal model parameter. Therefore, tools are used to automatically tune the model hyperparameters. Hyperparameter optimization is performed using Ray Tune Python library[3] with Hyperopt algorithm [324]. Filter size, number of filters, learning rate, batch size, and theta (QE threshold) hyperparameters were optimized for each domain shown in Table 4.1. The search spaces are (0,1), {2, 3, 4, 5}, {10, 20, 30, 50 , 100}, {1e-5 , 2e-5, 1e-6, 1e-7, 2e-7, 1e-8, 2e-8, 5e-8}, {4, 8, 16, 32, 64} for θ, filter size, number of filters, learning rate, and batch size respectively. The optimal combination of hyperparameters values that maximize the model performance is discovered by the Hyperopt algorithm for each time tuning the model for a new domain. The Hyperopt algorithm utilizes a form of Bayesian optimization and requires the search space, the loss function, the

[3] https://docs.ray.io/en/latest/tune/index.html

Table 4.1: Optimal hyperparameters for closed-domain datasets (Tesla, California, EU-law, and COVID-QA) and open-domain datasets (SQuAD). The search spaces are $(0, 1)$, $\{2, 3, 4, 5\}$, $\{10, 20, 30, 50, 100\}$ $\{1e-5, 2e-5, 1e-6, 1e-7, 2e-7, 1e-8, 2e-8, 5e-8\}$, $\{4, 8, 16, 32, 64\}$ for θ, filter size, number of filters, learning rate, and batch size respectively.

Hyperparameters	Tesla	California	EU-law	COVID-QA
θ	0.76	0.74	0.71	0.67
learning rate	1e-8	2e-8	1e-8	5e-8
filter size	2,3,4	2,3,4	2,3,4	2,3,4
filter number	20	20	20	30
batch size	8	8	4	16

optimization algorithm, and a database for recording hyperparameter tuning history (score, configuration). The maximum sequence length is set to 128 tokens for BERT and ALBERT. The Adam optimization algorithm [325] is utilized for the parameter update. The cross entropy loss function is used to calculate the loss. The optimal values for filter size, number of filters, learning rate, and batch size for the first step of training are calculated as follows: {2, 3, 4}, 100, 2e-5, 64. Early stopping is applied on the development set for both training stages on the loss value. The max number of epochs is set to 9 and 3 for transfer and adapt steps, respectively. For the closed-domain QE, domain-specific corpora is used (concatenation of contexts for one domain) for tuning the pre-trained BERT for generating domain-specific embeddings. The domain-specific corpus is prepared automatically for "masked Language Model" and "next sentence prediction" to generate the data for pre-training on each domain. For the open-domain QE, the pre-trained BERT is utilized for the embedding generation. The NodeBox English library is utilized, which has been succeeded by the Pattern Python library[4], for analysing the keyword's role and expansion term transformation. The pre-trained BERT basic model have been utilized for generating the sentence embedding for calculating the cosine similarity in the reasoning process.

[4]https://github.com/clips/pattern

4.3.4 Sentence-level MRC model

Closed-domain Comparative Methods

To demonstrate the effectiveness of the proposed MRC model for closed-domain QA, a comparison against several other comparative approaches is conducted. The approaches with publicly available codes are selected to be tuned for sentence-level MRC:

- KPOS-QA [278] is a closed-domain QA system (their dataset is not publicly available). Their approach is simulated for sentence-level QA regarding the details provided in their paper (ranking and selecting the answer based on extracted keywords and POS tags for query and context).

- R-TFIDF [280] is another closed-domain QA system (their dataset is also not publicly available). Their approach is simulated for sentence-level QA regarding the details provided in their paper (a rule-based sentence classification and measuring cosine similarity on TF-IDF vectors for question and sentences).

- AttReader [326] presented BiLSTM networks based on an attention mechanism and the GLoVe language model for reading comprehension in QA.

- QANET [327], is an MRC model for open-domain QA based on convolutions, global self-attention, and the GLoVe language model.

- cdQA is an end-to-end closed domain QA system built on top of the pre-trained BERT [5].

- Retro-reader [328] is an "open-domain" MRC model and ranks 5th in the SQuAD2.0 leaderboard.[6] An approach with two reading modules (sketchy reading module and intensive reading module) is proposed to find answer span and detect unanswerable questions. In the intensive reading module, two question-aware matching mechanisms based on the transformer and multi-head attention are introduced for predicting the answer.

- ZCovid-QA [329], employed RoBERTa fine-tuned on the SQuAD and QuAC datasets for zero-shot evaluation on the COVID-QA dataset for Covid-19 QA.

[5] `https://github.com/cdqa-suite/cdQA`

[6] I did not find openly available source codes of other top-ranking models even after contacting their authors. As a result, I compare my method with the model whose code I could obtain.

- EtoE-Covid-QA [330] fine-tuned RoBERTa-large on SQuAD2.0, NQ, and proposed both language modeling on the CORD-19 collection and example generation model for the MRC training for Covid-19 QA.
- OCovid-QA [331] utilized a variant of BioBERT fine-tuned on the SQuAD2.0 and COVID-QA datasets for Covid-19 QA.

Closed-domain Results and Discussion

The results obtained by the A-MRC model and others are presented on the development set in Table 4.2. The results are presented for two variants of the proposed model (A-MRC): 1) pre-trained BERT; and 2) pre-trained ALBERT. For AttReader, QANET, cdQA, Retro-Albert, their public code is used to apply their model to the datasets and generate results for closed-domain sentence-level QA. The same pre-trained language models are used as reported in their respective papers and fine-tuned the models with two stages of training as mentioned in 4.3.3. ZCovid-QA, EtoE-Covid-QA, and OCovid-QA baselines are only designed for Covid-19 QA and the results for these models are reported in their respective papers. The comparative models are categorized into two groups: 1) based on conventional language models (KPOS-QA, R-TFIDF, AttReader, QANET); and 2) contextualized language models (cdQA, RetroReader, ZCovid-QA, EtoE-Covid-QA, OCovid-QA). The results show that KPOS-QA, which is based on context and question keyword extraction using POS tags, achieves the worst results. Hence, the results display that there is a strong need for high-quality vectors representing context and question. In R-TDIDF, an improvement of 3%-11% of the F1 score is obtained by applying traditional TF-IDF vectors and sentence classification. The QANET and AttReader outperformed the KPOS and R-TFIDF, whereas the pre-trained GLoVE language model encodes context and question. The QANET outperformed AttReader because it's not relying on the recurrent structure, unlike the AttReader, which is based on BiLSTM. cdQA outperformed QANET and AttReader and improved the EM due to its reader architecture based on BERT. RetroReader outperformed the other baseline methods for all datasets since it employed a pre-trained transformer-based language model and attention mechanism for reading comprehension. Evaluating RetroReader for closed-domain reading comprehension shows its performance has degraded slightly (its performance in open-domain QA is 91.3 for F1 score and 88.8 for EM). The A-MRC model outperforms all baseline models for all datasets because the association between the extracted features from the question and candidate answer sentences is explored by applying CNN and multi-head attention on the joint representation of question

and candidate answer sentences. Also, the CAI and QE module's effect on selecting appropriate sentences from context and rewriting the vague questions should not be disregarded. The performance of all baseline methods is worse on the COVID-QA dataset since Biomedical QA (BQA) is more challenging than other domains, and more reasoning is needed for the question and biomedical text compared to other domains. Another challenge is clinical term ambiguity due to the variation of clinical terminology and the frequent use of abbreviations and esoteric medical terminology. BQA evaluation is also challenging because most evaluation metrics do not consider the rich biomedical synonym relationships. Since biomedicine is a highly specialized domain, understanding complex biomedical knowledge is required, and using contextualized language models pre-trained on open-domain corpora is inefficient. The A-MRC approach utilizing the pre-trained BERT on the biomedical domain is also evaluated as shown in Table 4.2. SciBERT [332] is trained on a large corpus of scientific text, including text from the biomedical domain, and BioBERT [333] is the first domain-specific BERT-based model pre-trained on biomedical corpora. DeepSet[7] has made available a BERT-base model pre-trained on CORD-19 [334], and it is evident that pre-training BERT with CORD-19 corpus improves the A-MRC model performance since this model is certainly more "in-domain" than BioBERT-base or SciBERT-base for COVID-19 QA. OCovid-QA outperformed EtoE-Covid-QA and ZCovid-QA since it is based on BioBERT, which is more appropriate for Covid-19 QA than RoBERTa used in EtoE-Covid-QA and ZCovid-QA.

Open-domain Comparative Methods

To demonstrate the effectiveness of the A-MRC model for open-domain QA, it has been compared against several other comparative approaches (their publicly available codes are used to tune them for sentence-level MRC):

- AttReader [326] presented BiLSTM networks based on an attention mechanism and the GLoVe language model for reading comprehension in QA.
- QANET [327], is an MRC model for open-domain QA based on convolutions, global self-attention, and the GLoVe language model.
- cdQA is an end-to-end closed domain QA system built on top of the pre-trained BERT [8].

[7] https://huggingface.co/deepset
[8] https://github.com/cdqa-suite/cdQA

Table 4.2: Performance comparison of A-MRC models for closed-domain QA against other baselines.

	Tesla	California	EU-law	COVID-QA
Model	EM / F	EM / F	EM / F	EM / F
KPOS-QA	53.2 / 61.1	52.0 / 60.3	48.4 / 57.2	0 / 9.1
R-TFIDF	63.1 / 70.8	63.4 / 70.9	60.1 / 68.8	0 / 12.3
AttReader	71.3 / 79.5	70.8 / 79.8	68.9 / 78.7	11.2 / 41.4
QANET	75.2 / 83.3	75.3 / 83.1	73.8 / 82.2	12.4 / 43.6
cdQA	80.0 / 84.2	80.3 / 84.8	78.2 / 83.0	33.5 / 65.9
RetroReader	87.4 / 90.1	86.9 / 90.3	86.5 / 89.9	52.4 / 74.0
ZCovid-QA [329]				25.9 / 59.5
EtoE-Covid-QA [330]				38.6 / 62.8
OCovid-QA [331]				39.1 / 72.0
A-MRC (BERT)	**89.8 / 93.2**	**89.5 / 92.8**	**88.0 / 92.2**	**55.6 / 76.4**
A-MRC (ALBERT)	**92.6 / 95.5**	**92.3 / 95.3**	**91.2 / 95.4**	**57.5 / 78.8**
A-MRC (SciBERT)				**68.8 / 80.6**
A-MRC (BioBERT)				**73.2 / 83.8**
A-MRC (CORD-19)				**80.6 / 87.9**

- Retro-reader [328] is an "open-domain" MRC model and ranks 5th in the SQuAD2.0 leaderboard.[9] An approach with two reading modules (sketchy reading module and intensive reading module) is proposed to find answer span and detect unanswerable questions. In the intensive reading module, two question-aware matching mechanisms based on the transformer and multi-head attention are introduced for predicting the answer.

- SPARC-QA [335] considered the open-domain QA as a phrase retrieval problem and proposed a technique for learning a Contextualized Sparse Representation (SPARC) for each phrase, demonstrated the efficiency of SPARC for encoding phrases with rich lexical information in open-domain QA.

- End-to-End-QA [336] proposed a system which different retrievers can be plugged in directly and a neural reader outputs the answer to the question. The reader assigns a passage selection score to top-k retrieved passages. They indicated that the retrieval can be implemented using dense representations alone while utilising a simple dual-encoder architecture to learn the embeddings from a limited number of questions and passages.

- Lev-Gen-QA [337] presented a two-step method that first fetches supporting passages using sparse or dense representations. The answer is then generated using a seq2seq model that takes the retrieved passages as well as the question as input.

Open-domain Results and Discussion

The results obtained by the A-MRC model and others on the development set are presented in Table 4.3. The results are presented for two variants of the A-MRC model: 1) pre-trained BERT; and 2) pre-trained ALBERT. The QANET and AttReader have the worst due to using the pre-trained GLoVE language model that encodes context and question. Same as the closed-domain results, the QANET outperformed AttReader because it's not relying on the recurrent structure, unlike the AttReader, which is based on BiLSTM. RetroReader outperformed the AttReader and QANET since it employed a pre-trained transformer-based language model and attention mechanism for reading comprehension. It is evident that SPARC-QA and Lev-Gen-QA gained higher F1 and EM scores in comparison to other baseline methods because of

[9] I did not find openly available source codes of other top-ranking models even after contacting their authors. As a result, I compare my method with the model whose code I could obtain.

Table 4.3: Performance comparison of A-MRC models against other baselines for open-domain MRC style QA.

Model	SQuAD
	EM / F
AttReader	71.4 / 80.1
QANET	76.2 / 84.6
RetroReader	88.8 / 91.3
End-to-End-QA	86.7 / 90.3
SPARC-QA	87.3 / 91.3
Lev-Gen-QA	88.4 / 91.4
A-MRC (BERT)	**89.1 / 92.8**
A-MRC(ABLERT)	**91.5 / 94.2**

utilizing contextualized representations. End-to-End-QA outperformed RetroReader because of the learned dense representations by a dual-encoder framework. The A-MRC outperforms all baseline methods, F1 and EM are improved by approximately 2% and 1% for SQuAD. The performance of the open-domain version of A-MRC (as an sentence-level MRC style QA system) is elevated due to the dense contextualized representations generated by the CNN and multi-head attention for the question and candidate answer sentences which are combined with complementary components (candidate answer identifier and question expansion module).

Ablation Study

The effect of the question expansion component and CNN Attention layer are investigated individually to understand the overall role they play in the A-MRC model. In Table 4.4, the results are presented without the CNN attention module in the A-MRC model. Fine-tuning the QA pipeline without the CNN-Attention layer with pre-trained BERT and ALBERT reduced the performance significantly. Utilizing the CNN-Attention layer captures the semantic connections between the sentence and question features which boosts the model performance 7%-11% for EM and and F1 score. The quantitative results are depicted in Table 4.5, where the model's performance with and without the QE component is shown. Additionally, the BART and T5 pre-trained question paraphrasers are examined for rewriting the vague questions. T5 caused a slight performance degradation compared to the "A-MRC(ALBERT) without

Table 4.4: The effect of CNN and Attention mechanism on the proposed models on all datasets.

Model	Closed-domain Datasets				Open-domain Dataset
	Tesla	California	EU-Law	COVID-QA	SQuAD
	EM / F	EM / F	EM / F	EM / F	EM / F
A-MRC(BERT) without CNN-Attention	82.3 / 85.0	81.1 / 84.7	80.0 / 84.2	45.0 / 66.4	81.2 / 84.5
A-MRC(BERT)	**89.8 / 93.2**	**89.5 / 92.8**	**88.0 / 92.2**	**55.6 / 76.4**	**89.1 / 92.8**
A-MRC(ALBERT) without CNN-Attention	84.0 / 87.5	83.2 / 86.9	83.0 / 86.7	46.8 / 67.9	83.7 / 88.4
A-MRC(ALBERT)	**92.6 / 95.5**	**92.3 / 95.3**	**91.2 / 95.4**	**57.5 / 78.8**	**91.5 / 94.2**

QE" since it is not tuned with any domain-specific training data for question paraphrasing. BART paraphraser outperformed T5, although it couldn't elevate the model performance significantly. T5 and BART need fine-tuning on a domain-specific paraphrase dataset for a better performance which is not feasible for every domain.

It can be concluded that rewriting the question without considering the domain terminology in the closed-domain QA misleads the model by generating domain irrelevant questions. Therefore, for generating in-domain question paraphrases (paraphrased questions) without the need for training data for every domain, the proposed QE module operated well, and it improved the EM and F1 score by 1%-2% for all closed-domains. Also, selecting the most appropriate expansion terms by considering their semantic correlation to the question is the key for outperforming in open-domain.

Additional experiments are conducted to study the role played by each question type, i.e. " What", "Where", "When", "Why", "Who", "How". Besides that, the goal is also to portray that the new customizations that have been made to Giveme5W1H are useful to the framework. This will help for understanding the role that each question type plays in the study. The F1 score is calculated for each question type individually for analysing the A-MRC model performance on different question categories (see Table 4.7). The number of instances in each question category on four datasets is reported in Table 4.6. One observation is that the performance is not impacted by the number of instances in the category because the proposed framework does not heavily rely on statistical information, which makes it reliable even under low-resource situations. The number of questions that do not belong to the 5W1H categories is 15, 76, 59, 200, 6770 for Tesla, California, EU-law, COVID-QA, and SQuAD datasets.

Table 4.5: The effect of QE component on the proposed models on all datasets.

Model	Closed-domain Datasets				Open-domain Dataset
	Tesla	California	EU-law	COVID-QA	SQuAD
	EM / F	EM / F	EM / F	EM / F	EM / F
A-MRC(BERT) without QE	88.4 / 91.3	88.2 / 90.4	88.1 / 91.0	54.0 / 75.5	88.2 / 90.4
A-MRC(BERT)	89.8 / 93.2	89.5 / 92.8	88.0 / 92.2	55.6 / 76.4	89.0 / 92.3
A-MRC(ALBERT) without QE	90.9 / 93.5	91.1 / 93.0	90.8 / 94.6	57.2 / 77.6	88.7 / 92.2
A-MRC(ALBERT) + T5	90.5 / 93.3	90.8 / 92.8	90.3 / 93.8	57.0 / 77.2	89.3 / 92.4
A-MRC(ALBERT) + BART	91.2 / 94.0	91.1 / 93.6	91.0 / 94.8	57.2 / 77.7	90.5 / 93.0
A-MRC(ALBERT)	**92.6 / 95.5**	**92.3 / 95.3**	**91.2 / 95.4**	**57.5 / 78.8**	**91.3 / 94.0**

Table 4.6: The count of each question category for closed-domain datasets.

Dataset	What	Where	When	Why	Who	How
Tesla	268	76	54	25	71	56
California	323	96	80	52	59	60
EU-law	136	17	23	23	39	18
COVID-QA	1335	53	53	80	22	272
SQuAD	41598	5558	7938	2295	8535	14126

Table 4.7: Performance comparison (F1) of the A-MRC variants for each question category with the proposed candidate answer identifier (customized Giveme5W1H) and Giveme5W1H candidate answer identifier.

		Proposed CAI (customized Giveme5W1H)						Giveme5W1H Candidate Answer Identifier					
Model	Dataset	What	Where	When	Why	Who	How	What	Where	When	Why	Who	How
A-MRC(BERT)	Tesla	93.5	93.3	93.0	92.5	93.6	93.5	87.8	91.6	89.1	88.5	91.5	89.6
	California	92.7	92.1	93.0	91.5	93.5	94.2	87.9	89.2	88.7	87.4	91.4	90.0
	EU-law	92.8	92.1	92.5	91.2	92.8	92.0	89.0	91.3	88.2	88.4	90.0	90.7
	COVID-QA	75.3	76.7	76.9	74.9	76.6	77.7	72.3	74.5	71.8	70.9	74.2	75.5
	SQuAD	93.6	92.4	93.7	92.0	92.3	92.8	89.3	90.1	88.5	88.8	90.2	88.7
A-MRC(ALBERT)	Tesla	96.5	96.4	96.6	94.5	94.3	94.5	92.0	94.6	92.8	90.6	92.2	90.4
	California	96.6	95.9	95.0	94.3	94.2	95.6	92.1	92.8	90.6	90.2	92.8	91.3
	EU-law	95.9	95.7	95.6	94.4	95.8	95.0	92.4	93.3	91.7	92.2	93.1	93.8
	COVID-QA	78.9	78.6	79.7	77.2	78.5	79.9	75.0	75.7	74.6	73.1	76.3	77.0
	SQuAD	94.9	95.1	94.0	93.3	93.6	94.3	90.6	93.6	90.1	89.8	91.5	90.2

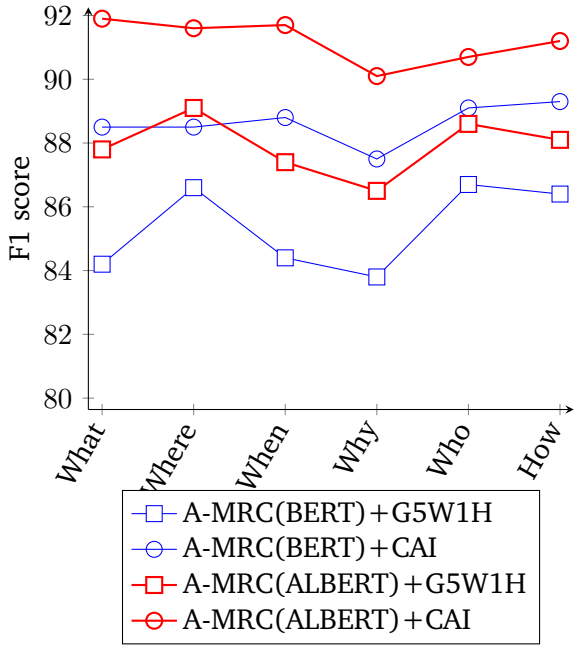

Figure 4.5: The average F1 score across all datasets for the model, A-MRC (ALBERT), with the original Giveme5W1H and the proposed CAI on each question category.

Furthermore, the candidate answer identifier helps automatically select the appropriate sentences in each question category based on the linguistic rules in 4.2.1. An advantage that A-MRC model gets by the CAI component is reducing the number of candidate answers, which significantly impacts the model's effectiveness for long contexts by excluding the question-irrelevant sentences. Fig. 4.5 displays the average F1 score across all datasets for the model with the original Giveme5W1H and the proposed CAI on each question category. The model performance for each question category is improved by adding linguistic rules and functions to Giveme5W1H, and the "What", "Why", and "When" categories improved the most.

4.3.5 Open-domain multi-hop QA

Four baseline methods are considered for the multi-hop QA evaluation stage which are described below.

WAIR [338] utilizes the alignment IR approach [339] to retrieve justification sentences and a RoBERTa binary classifier for answer selection. The WAIR technique, in two iterations, reduces the weights of question terms that have already been addressed by previously retrieved sentences and increases the weights of reformulated question terms that have not yet been covered. The second iteration reranks the clusters of evidence sentences using a regres-

sion task, with each sentence cluster allocated an F1 score generated from the gold annotated evidence sentences.

AIR [340] discovers justification sentences by an unsupervised strategy based on GloVe embeddings and an alignment model. To choose answers, a RoBERTa binary classifier is utilized. The question and candidate answer text are used to initiate the query. AIR adjusts its query after each repetition to focus on the missing information in the current set of justifications. The alignment approach computes the cosine similarity between each token's word embeddings in the query and the provided text sentence, resulting in a matrix of cosine similarity scores.

ROCC [341] presented an unsupervised technique for maximising the relevance of selected sentences, minimising the overlap between selected facts, and maximising both question and answer coverage. The relevance, coverage, and overlap scores of candidate justification sets are calculated. They used BERT as a binary classifier to choose answers.

Multee [342] presented models of entailment for multi-hop QA composing a relevance module and multi-layer aggregation module. Both modules make use of ESIM [343], a recently developed sentence-level entailment model that has been trained on the SNLI and MultiNLI datasets.

The results are reported in Table 4.8 for the baseline methods from their paper. It is evident that WAIR outperformed other baselines since it introduced several attention and embedding-based analyses. It demonstrates that by combining retrieval and reranking techniques, it is possible to acquire the compositional knowledge necessary for multi-hop reasoning. AIR is the second-best baseline and outperformed ROCC and Multee due to the iterative method used to reformulate queries and focus on words not covered by existing justifications. AIR is an unsupervised alignment technique that uses only GloVe embeddings to soft-align questions and answers with justification sentences. ROCC outperformed Multee because it is an unsupervised strategy that utilizes a BERT answer classifier and three scoring functions to rank candidate reason sets. In compared to Multee's entailment technique, the ranking functions improve ROCC performance by increasing the relevance of the selected sentences and decreasing lexical overlap between the selected facts. The Ex-MhopQA model outperformed all baselines since it investigates the the semantic correlations between the features extracted from the question and relevant sentences in document. The semantic correlations between features are obtained by applying CNN and multi-head attention to the combined representation of the question and candidate answer sentences. Also, the reasoning process

Table 4.8: $F1_m$, $F1_a$, and EM score for the proposed method (Ex-MhopQA) and open-domain Multi-hop QA baseline methods on MultiRC dataset.

Model	MultiRC dataset		
	$F1_m$	$F1_a$	EM
WAIR [338]	79.5	76.5	35.4
AIR [340]	79.0	76.4	36.3
ROCC [341]	73.8	70.6	26.1
Multee [342]	71.7	68.3	-
Ex-MhopQA	82.2	79.8	40.3
Ex-MhopQA (with self-attention)	80.7	77.9	37.6

based on lexical coverage and BERT embedding is a complement to answer selector module for selecting justification sentences. The results of the proposed model with self-attention are compared to the performance with multi-head attention. As it is shown, the multi-head attention variant performs better due to the less number of layers and training stability in comparison to self-attention variant. Fig. 4.6 shows the Ex-MhopQA performance on each question category for MultiRC dataset. The multi-head attention variant outperformed the self-attention variant on all question categories since multi-level attention information is acquired.

4.3.6 Question-driven Extractive Text Summarisation

In the third stage, the generated extractive summaries are evaluated by comparing them to the reference summaries and results of the baseline methods, which have been described below.

HSCM [31] presented an approach for extractive answer summarisation consisting of three components. In the first and second components (Word-level and Sentence-level Compare-Aggregate), an attention operation is used to align the word-level and sentence-level information between the answer sentence and question. In the third component, Question-aware Sequential Extractor, a RNN decoder is designed to label each sentence consecutively and construct the answer summary for the target question.

MSG [29] proposed Multi-hop Selective Generator (MSG), a question-driven abstractive summarisation approach that integrates multi-hop reasoning to identify the key content for assisting the answer generation. In addition to the multi-view pointer network, they in-

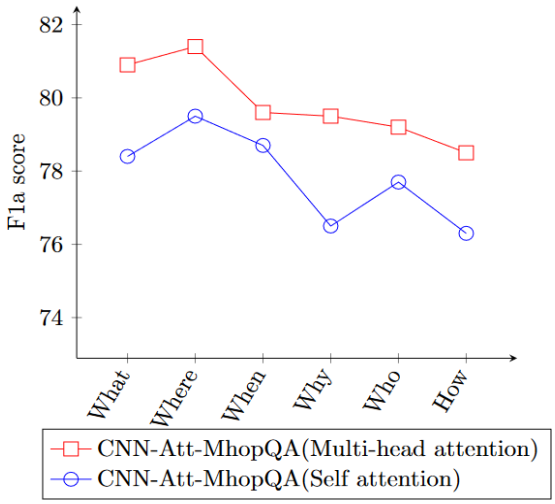

Figure 4.6: Ex-MhopQA multi-head attention and self-attention variants' performance on each question category for the MultiRC dataset.

troduced a multi-view coverage technique to overcome the duplication issue and generate informative and precise answers.

QPGN [32] presented a question-driven pointer-generator network that utilizes the correlation information between question-answer pairs to add substantial information when generating abstractive answer summaries. Their framework consists of four components: Bi-LSTM Encoder, seq2seq Model with Question-aware Attention, Question-Answer Alignment with Summary Representations, Question-driven Pointer-generator Network.

Trans [33] has studied the capability of three state-of-the-art transformers for question-driven text summarisation: BART, T5, and PEGASUS in both zero-shot and few-shot learning settings for question-driven abstractive text summarisation on MEDIQA dataset. T5 outperformed the others thus it has been considered as a baseline method.

Table 4.9 shows the experimental results for one extractive (HSCM) and three abstractive (MSG, QPGN, Tran(T5)) question-driven summarisation approaches on WikiHow, PubMedQA, MEDIQA datasets. All of the results for the baseline methods are reported from their paper. ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE (RL) are considered to evaluate the quality of the extractive summaries. HSCM generated extractive answer summaries and as it is shown in Table 4.9 the R1, R2, and RL for the generated extractive summaries and other abstractive baseline methods are superior to HSCM. Employing Glove language model and relying on a recurrent structure decoder for generating extractive answer summaries caused this inef-

Table 4.9: Results on WikiHow, PubMedQA, and MEDIQA

Model	WikiHow			PubMedQA			MEDIQA		
	R1	R2	RL	R1	R2	RL	R1	R2	RL
HSCM	27.84	7.75	25.85	32.34	10.07	25.98	-	-	-
MSG	30.5	10.5	29.3	37.2	14.8	30.2	-	-	-
QPGN	28.8	9.7	27.7	34.2	12.8	28.7	-	-	-
Trans(T5)	-	-	-	-	-	-	38.56	18.52	26.00
Ex-MhopQA Extractive	**31.71**	**11.23**	**30.09**	**38.89**	**14.81**	**30.76**	**40.94**	**20.11**	**27.46**

ficiency and poor performance in HSCM. MSG achieves relatively better performance than QPGN because of incorporating multi-hop reasoning for abstractive summarisation. MSG and QPGN have used the pre-trained Glove model [72] which is not a very efficient model because of the co-occurrence matrix of words that consumes a considerable amount of memory. Besides using an inefficient language model, having multi-stages of training is another problem of these baseline methods. In Trans each language generation model (BART, T5, PEGASUS) is pre-trained with different strategies which is unclear whether these strategies are the optimal ones.

Favorably the Ex-MhopQA model obtains the state-of-the-art results for all three datasets with the generated extractive summaries. The results indicate that the generated extractive summary covers the essential information for satisfying answerability, understandability, and persuasiveness measures by finding the AS and its supporting sentences using the proposed MRC model and the reasoning process.

4.4 Summary

The proposed multi-hop QA, Ex-MhopQA, contains an adaptable MRC model that improves upon state-of-the-art models across different closed-domain datasets: Tesla (person); California (region); EU-law (system); COVID-QA (biomedical) datasets and open-domain dataset (SQuAD). A novel approach is presented by exploiting CNN and the multi-head attention mechanism to solve the generalization problem by training on small datasets. The A-MRC model calculates the semantic association between the extracted local features from context sentences and the question by employing CNN and the multi-head attention mechanism. Furthermore, components such as the candidate answers identifier and question expansion assist the model by limiting the choice to relevant sentences for each question category and remov-

ing ambiguity in questions by replacing some keywords. Experimental results and ablation studies illustrate that the proposed model outperforms different models on closed-domains and open-domain without any knowledge base. The reasoning process based on contextualized semantic similarity and lexical coverage is a compliment to the MRC model that explores the document for finding the answer's supporting sentences. The Ex-MhopQA model is evaluated in three stages to examine the effectiveness of the different components. The A-MRC and Ex-MhopQA are sentence level MRC model and multi-hop QA system which are adaptable to both open-domain and closed-domains. The comparative baseline methods and closed-domains are used to evaluate the adaptability and flexibility of the proposed model. The proposed mutli-hop QA generates the question-driven extractive summary which covers all the required information showing by results compared to other question-driven baseline methods.

In this chapter, the MRC and Multi-hop QA are designed in a way that could be capable of overcoming some of the problems associated with QA systems. Ojokoh et al. [8] noted many problems associated with QA systems. The proposed model attempted to provide solutions for Question Processing, Question Classes, Answer Extraction, Answer Formulation, and Advanced reasoning for QA.

Chapter 5

Question-driven Abstractive Model

5.1 Introduction

In the previous chapter, the extractive summariser discovers appropriate non-redundant sentences as plausible answers to the question using a multi-hop QA system containing an adaptable MRC and a reasoning process. Separating the two tasks of content selection (extractive summarisation) and abstractive summary generation allows to closely examine every component (paraphrase, compressing, and fusing mechanisms) of an abstractive summariser.

Most of the recent abstractive summarisation models are based on sequence-to-sequence (seq2seq) neural networks [39, 40, 41, 42, 43, 44]. They are made up of encoders to comprehend input sequence and decoders to generate output sequence. However, there are four key problems with using seq2seq neural networks to generate reasonable text: (1) out-of-vocabulary (OOV) problem (2) generating a particular word or phrase repeatedly which brings in redundancies, (3) test-time exposure bias, and (4) non-optimized learning for evaluation metrics used by models in fields such as text summarisation and machine translation. As a result, they cannot generate appropriate abstractive summaries since they cannot convey the semantics of the document [45, 46]. To recreate the key content in a fresh way, abstractive summarisation requires advanced natural language techniques for reading and understanding the text. As described in Chapter 2 GANs are algorithmic architectures that use two neural networks, pitting one against the other (thus the "adversarial") in order to generate new, synthetic instances of data that can pass for real data. Transformers leverage the concept of self-attention to develop simpler models. A transformer is based solely on the concept of attention and it eliminates the need for recurrent connections. Integrating GAN and transformers is an architecture which is not yet been explored for text generation and in

this chapter, an abstractive summarisation model based on this novel architecture has been proposed and presented.

A novel paraphrasing GAN model based on transformers and Q-learning is proposed to rewrite the extracted sentences in the abstractive setup. In addition, a fusing mechanism is proposed for compressing the sentence pairs in the paraphrased summary selected by a BERT next sentence prediction model. The Pointer Generator network [10] has been utilized for fusing and compression, which has been pre-trained and fine-tuned for being used in the abstractive stage. Pointer generator networks solve various combinatorial optimization and combinatorial search problems. Pointer networks are derived from the attention mechanism by [344] comprising an attention-based model seq-to-seq model, pointer generator network, and coverage mechanism.

The experiments show that this setup results in a more reliable abstractive summary than competing methods. Two fine-tuning setups have been considered for the fusing mechanisms, and both outperformed the extractive summaries generated by the multi-hop QA.

5.2 The Question-driven Abstractive Summariser

The Fig. 5.1 shows the overall framework of the abstractive summarisation model (QPara-Sum) which generates the question-driven abstractive summaries.

5.2.1 Paraphrase Generation Model

It can be started by defining two sequences of tokens $X_{1:n} = \{x_1, ..., x_n\}$ and $Y_{1:T} = \{y_1, ..., y_T\}$, where the sequence X represents an input sequence and Y represents a paraphrase. A GAN model is designed, depicted in Fig. 5.2, for generating paraphrases. To this end, there are G_θ and D_ϕ to be a θ parameterized generator and a ϕ parameterized discriminator. The G_θ is trained to generate a sequence of tokens $\hat{Y}_{1:T} = \{\hat{y_1}, ..., \hat{y_t}\}$ that is similar to Y for the given X. The D_ϕ is trained to discriminate between Y and $\hat{Y}$ for input X. In the following sections, X, Y, and $\hat{Y}$ are called as input sentence, target sentence, and generated sentence, respectively.

Generator: Generator is an encoder-decoder model based on transformers. It consists of an encoder and a decoder that are both stacks of residual attention blocks. The transformer-based encoder-decoder models process the input sequence $X_{1:n}$ of variable length n with residual attention blocks without performing a recurrent structure, which is their main ad-

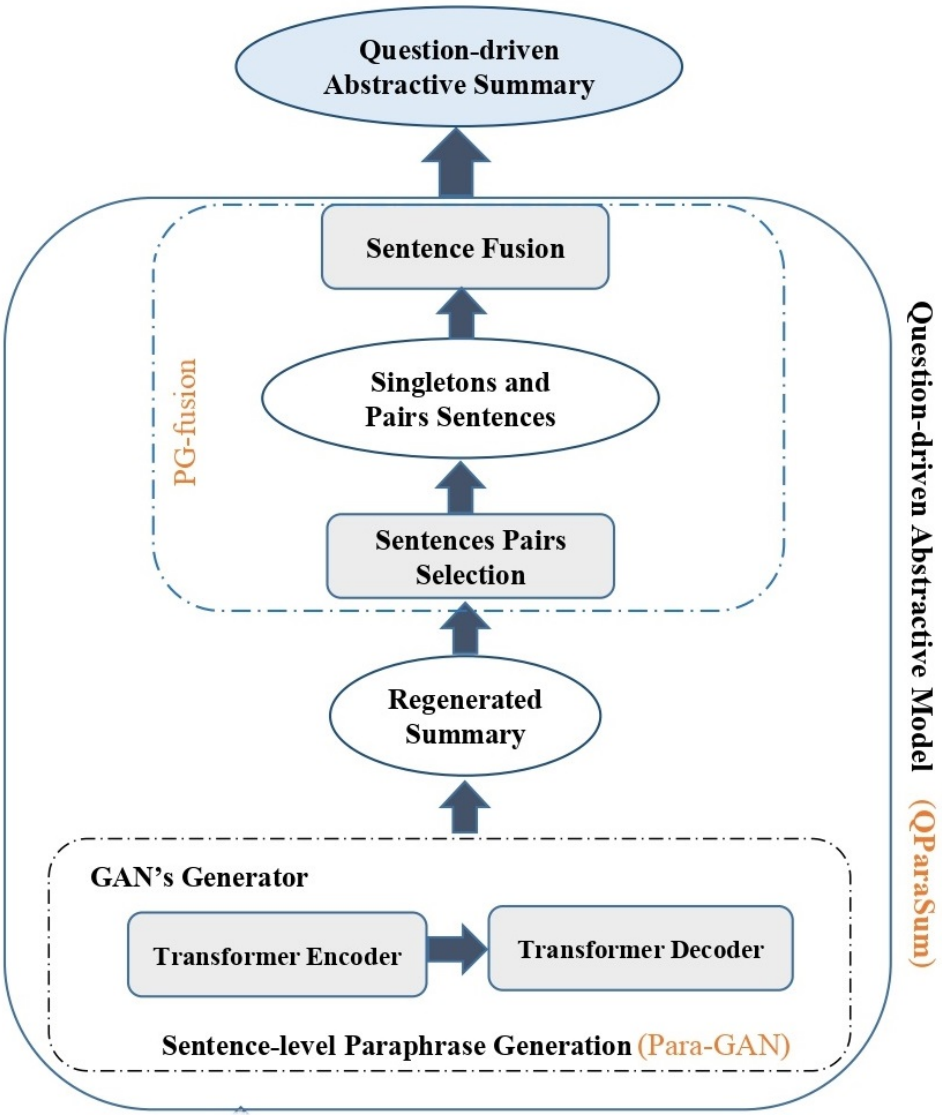

Figure 5.1: The overall framework of the abstractive summarisation model (QParaSum).

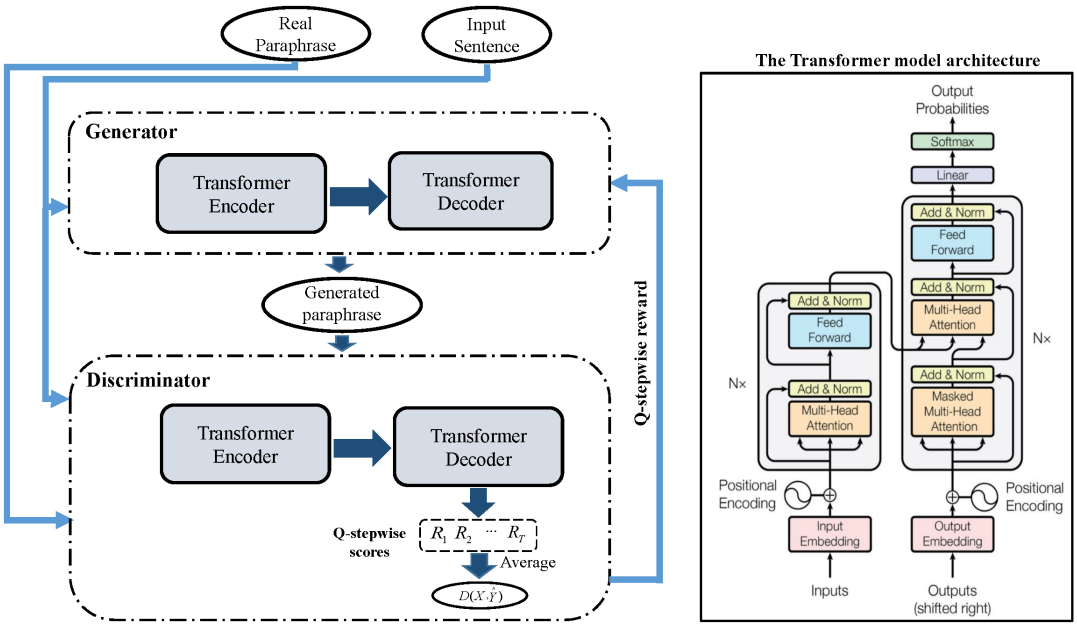

Figure 5.2: The illustration of the proposed GAN for paraphrasing

vantage and innovation. Transformer-based encoder-decoders are extremely parallelizable since they don't depend on a recurrent structure, which makes them more computationally efficient on modern hardware compared to RNN-based encoder-decoder models. The transformer-based encoder encodes the input sequence $X_{1:n}$ to a sequence of hidden states and the transformer-based decoder models the conditional probability distribution of the $\hat{Y}$ sequence given the sequence of encoded hidden states from the encoder.

Discriminator: The architecture of discriminator is similar to the generator, a transformer-based encoder-decoder model that accepts X as encoder inputs, and Y (either $\hat{Y}$ or Y) as decoder input. Rather of computing a scalar as the ultimate discriminator score $D(X, \hat{Y})$, a stepwise evaluation [149] is employed. After reading the input sentence X and a portion of the output sequence $\hat{Y}_{1:t}$, the discriminator creates a scalar R_t. The ultimate discriminator score for the entire created sentence is the sum of all the scalars $R_{1:T}$ throughout the length T of the generated sequence.

$$D(X, \hat{Y}) = \frac{1}{T} \sum_{t=1}^{T} R_t \tag{5.1}$$

Training At each generation step, the discriminator is customized to automatically allocate scores measuring the quality of each subsequence. Stepwise evaluation has substantially lower computational costs than MCTS, and the discriminator estimates instantaneous rewards by leveraging the idea of Q-learning and calculating state-action values without conducting tree search.

$$Q(s_t, \hat{y}_t) = \underset{z \sim P_G(.|x, \hat{y}_{1:t})}{E}[D(x, \hat{y}_{1:t}, z)] \tag{5.2}$$

$$R_t = Q(s_t, \hat{y}_t) \tag{5.3}$$

The current generator creates word sequence z with input X and generated prefix $\hat{Y}_{1:t}$.Thus, the anticipated return value of all the responses with the same prefix $\hat{y}_{1:t}$ is the state-action value $Q(s_t, \hat{y}_t)$. $s_t = (X, y_{1:t-1})$ and y_t are discrete tokens which are the inputs of the Q-function. A Kronecker delta function (or a sharp distribution) can be used for P_G in which all probabilities are zero except for the chosen sample. By this stepwise method a step dependent value, R_t, is calculated for each generation step which is called Q-stepwise reward.

Algorithm 3: Training the paraphrasing model

Result: Trained G_θ
Pre-train G_θ
Generate samples using G_θ
Pre-train D_ϕ with fake and real pairs
for n rounds **do**
  for $i = 1$ to G-iteration **do**
    Sample X from real data
    Generate a sequence $\hat{Y}$ using G_θ
    Calculate R for each sequence step
    Update G_θ using equation 5.5
  end for
  for $j = 1$ to D-iteration **do**
    Sample (X, Y) from real data
    Sample $(X, \hat{Y})$ using G_θ
    Update D_ϕ using equation 5.4
  end for
end for

An approach for estimating R_t value is designed for generator while training the discriminator. For predicting the expected value $V(s_t)$, a value network V with the same structure as discriminator is trained. The value network is trained to approximate the predicted R_t for every previous states s_t. As a result, the discriminator D_ϕ receives a pair of sentences and generates a score for each step. D_ϕ acquires knowledge using the following function:

$$J(\phi) = -logD_\phi(X, Y) - log(1 - D_\phi(X, Y)) \tag{5.4}$$

The G is trained with a stepwise evaluation technique, the objective function $J(G_\theta)$ of G_θ is:

$$J(\theta) = \sum_{t=1}^{T} R_t \nabla logP_G(y_t \mid x, y_{1:t-1}) \tag{5.5}$$

As the first step, real data is used to pre-train G_θ using the maximum likelihood. Also, supervised learning is applied to pre-train D_ϕ using pairs composed of real and created data. Then several rounds of adversarial training are begun. First, real samples are used to train G_θ using (5.5). G_θ is used to output a generated sample for each input sentence once the settings are updated. As a result, D_ϕ is fed a well-balanced set of real and fake (created) pairs. Finally, 5.4 is used to train D_ϕ.

5.2.2 Singletons and Sentence Pairs Selection

The Next Sentence Prediction (NSP) is utilized for detecting the sentences pairs and singletones (Fig. 5.3). A sequence can be a single sentence or singleton (A) or pair of sentences $(A + B)$. Devlin et al. [345] proposed the Bidirectional Encoder Representation from Transformers (BERT), which is designed to pre-train a deep bidirectional representation by jointly conditioning on both left and right contexts. BERT is trained using two novel unsupervised prediction tasks: Masked Language Modeling and Next Sentence Prediction (NSP). The NSP task has been formulated as a binary classification task: the model is trained to distinguish the original following sentence from a randomly chosen sentence from the corpus, and it showed great helps in multiple NLP tasks. NSP consists of giving BERT two sentences, sentence A and sentence B. First, the two sentences are merged with a $[SEP]$ token, a separator token, in between both sentences. Finally, the loss is calculated by processing the inputs and labels (NotNextSentence, IsNextSentence) through the model. All the sentences in the paraphrased summary will be analysed for singletons and pair sentences selection (any of two sentences detected as a pair will not be paired with any other sentences).

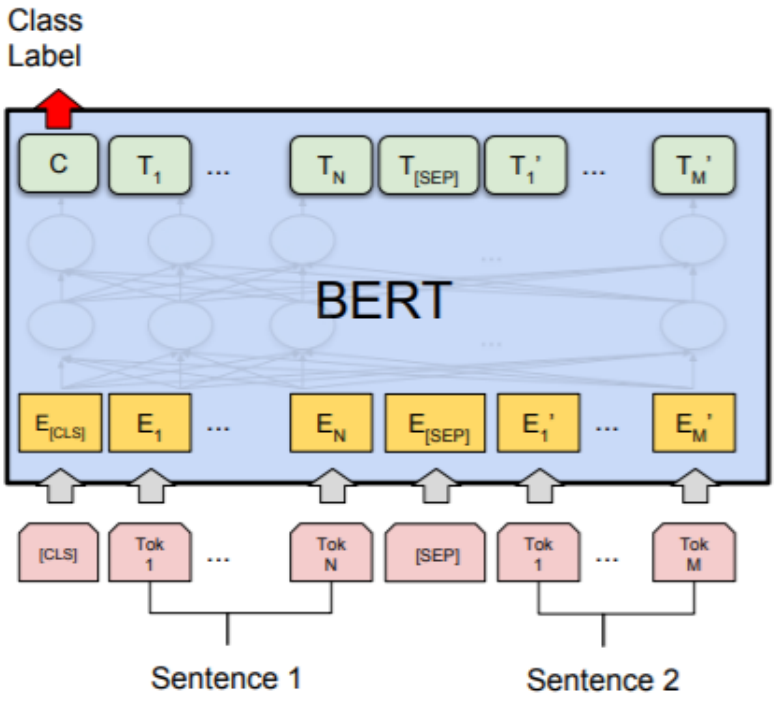

Figure 5.3: The illustration of the Next Sentence Prediction (NSP) for selecting sentences pairs [237].

5.2.3 Sentence Fusion

To generate shorter high-quality abstractive summaries, a sentence fusion component is designed to consume the paraphrased summary sentence-by-sentence produced in the previous step 5.2.1. For sentence pairs, the representations are expected to further encode sentential semantic compatibility. In this stage, the pairs are merged (i.e. fused), and singletons are remained unchanged.

The pointer generator (PG) network [10] is used to fuse the pair sentences into one summary sentence. PG is a sequence-to-sequence model that has achieved state-of-the-art performance in abstractive summarisation by copying tokens from the document and generating new tokens from the vocabulary (Fig. 5.4). PG addresses three issues: incorrect reproduction of factual facts, failure to handle out-of-vocabulary (OOV) words, and repetition. The PG network is used to summarise the sentence pairs while avoiding OOV and repetition. The sequence-to-sequence attentional model, the Pointer-generator Network, and the coverage mechanism are the three components of the PG network.

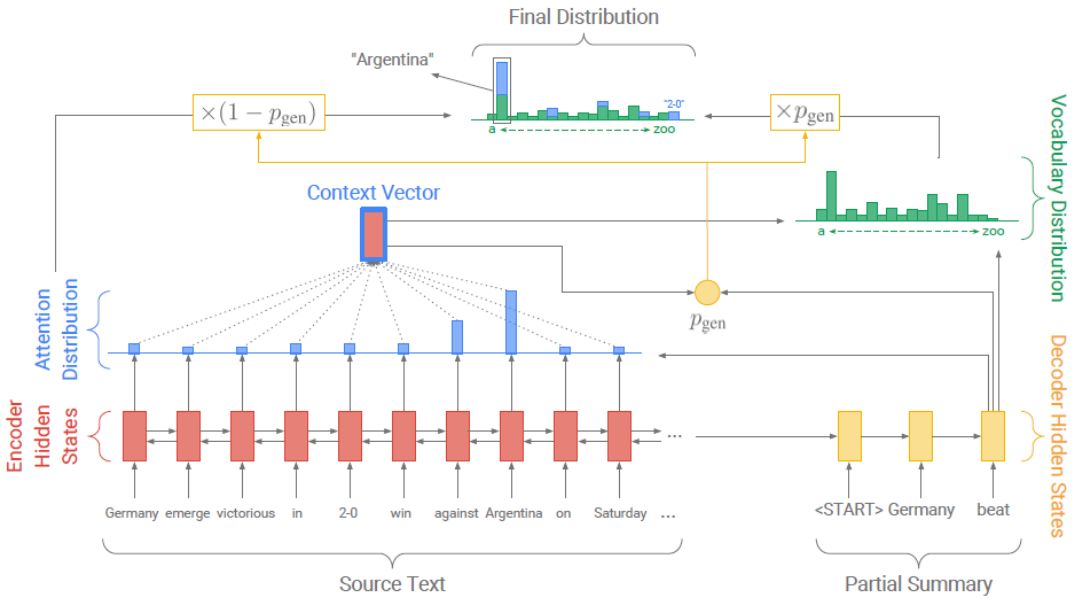

Figure 5.4: The Pointer-generator model [10].

Sequence-to-sequence attentional model

The tokens of the sentence pair w_i are input sequentially into the encoder (a single-layer bidirectional LSTM), which produces a series of encoder hidden states h_i. On each step t, the decoder (a single-layer unidirectional LSTM) gets the word embedding of the previous word (during training, this is the previous word in the reference summary; during testing, this is the previous word output by the decoder) and has state s_t. Calculating the attention distribution a_t:

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + b_{attn}) \tag{5.6}$$

$$a^t = softmax(e^t) \tag{5.7}$$

where v, W_h, W_s and b_{attn} are parameters that can be learned. Attention distribution may be considered as a probability distribution across the source words that guides the decoder where to search for the next word. Next, the attention distribution is utilised to generate the context vector h_t^*:

$$h_t^* = \sum_i a_i^t h_i \tag{5.8}$$

The context vector, which can be considered as a fixed-size representation of what has been read from the source for this step, is concatenated with the decoder state s_t and passed to two linear layers to generate the vocabulary distribution P_{vocab}:

$$P_{vocab} = softmax(V^{'}(V[s_t, h_t^*] + b) + b^{'}) \tag{5.9}$$

where V, $V^{'}$, b and $b^{'}$ are learnable parameters. P_{vocab} is a probability distribution over all words in the vocabulary, and provides the final distribution from which to predict words w:

$$P(w) = P_{vocab}(w) \tag{5.10}$$

During training, the loss of time step t is the negative log-likelihood of the target word w_t^t for that time step:

$$loss_t = -logP(w_t^*) \tag{5.11}$$

and the overall loss for the whole sequence is:

$$loss = \frac{1}{T} \sum_{t=1}^{T} loss_t \tag{5.12}$$

Pointer-generator network

PG network allows both copying words via pointing, and generating words from a fixed vocabulary. In the pointer-generator model (depicted in Fig. 5.4) the attention distribution a_t and context vector h_t^* are calculated as in 5.7, 5.8. In addition, the generation probability $p_{gen} \in [0, 1]$ for time step t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t:

$$p_{gen} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}) \tag{5.13}$$

where vectors w_{h^*} , w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Next, p_{gen} is employed as a soft switch to select between generating a word from the vocabulary by sampling from P_{vocab} and copying a word from the input sequence by sampling from the attention distribution a^t. The extended vocabulary for each document represent the union of the vocabulary and all terms appearing in the source document. The following probability distribution is obtained for the extended vocabulary:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \tag{5.14}$$

Note that if w is an out-of-vocabulary (OOV) word, $P_{vocab}(w)$ is zero; similarly, $\sum_{i:w_i=w} a_i^t$ is also 0 if w does not occur in the source document. One of the key benefits of pointer-generator models is their capacity to construct OOV terms; in contrast, the baseline models are limited to their predefined vocabulary. The loss function is as defined by equations 5.11 and 5.12, but with regard to the modified probability distribution $P(w)$ as specified by equation 5.14.

Coverage mechanism

In the coverage model, a coverage vector c_t is maintained, which is the sum of attention distributions over all previous decoder time steps:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{5.15}$$

Intuitively, c^t is a (non-normalized) distribution over the source document words that indicates the degree of coverage that those words have received from the attention mechanism thus far. Note that c^0 is a zero vector since none of the source document has been covered during the initial timestep. The coverage vector is utilised as an additional input to the attention mechanism, modifying the equation 5.6 to:

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{attn}) \tag{5.16}$$

where w_c is a learnable parameter vector of same length as v. This ensures that the attention mechanism's current decision (choosing where to attend next) is informed by a reminder of its previous decisions (summarised in c^t). This should make it simpler for the attention mechanism to avoid constantly attending to the same locations, hence preventing repetitive text from being generated. A coverage loss is defined in order to punish repeated attending to the same locations:

$$covloss_t = \sum_i min(a_i^t, c_i^t) \tag{5.17}$$

Note that the coverage loss is bounded; in particular $covloss_t \leq \sum_i a_i^t = 1$. Finally, the coverage loss, reweighted by some hyperparameter λ, is added to the primary loss function to yield a new composite loss function:

$$loss_t = -logP(w_t^*) + \lambda \sum_i min(a_i^t, c_i^t) \tag{5.18}$$

Training

The PG network is pre-trained on the DISCOFUSE and then fine-tuned on a smaller dataset from a different distribution. A transfer learning setting in which model performance improves when pre-trained with DISCOFUSE because of existing fusion datasets are small. McKeown et al. [346] introduced a human-generated corpus of 3,000 examples. Elsner et al. [347] extracted around 300 fusion examples from pre- and post editing news articles. Thadani et al. [348] constructed 1,858 examples from summarisation tasks. Such datasets are too small to train modern data-hungry neural models.

For each domain, it is required to link each summary sentence s_n with a subset of the document sentences $\hat{D} \subset D$, i.e. the sentences that are merged to produce s_n. The technique selects many sentences that work together to capture the most overlap with summary sentence s_n, as follows.

The average ROUGE-1, -2, and -L scores are utilized [122] to reflect sentence similarity. The source sentence most similar to s_n is selected; referred to it as $\hat{d_1}$. Then, all shared words are deleted from s_n to generate $s_n^{'}$, effectively removing all information already captured by $\hat{d_1}$. A second source sentence $\hat{d_2}$ most similar to the remaining summary sentence $s_n^{'}$ is picked, and shared terms are again deleted from $s_n^{'}$ to form $s_n^{''}$.

This process of sentence selection and overlap removal is repeated until no remaining sentences have at least two overlapping content words (words that are non-stopwords or punctuation) with s_n. The result is referred to as a ground-truth set $(s_n; \hat{D})$ where $\hat{D}$ =$\{\hat{d_1}$,$\hat{d_2}$,...,$\hat{d_j}\}$,

$j = |\hat{D}|$. To train the models, $\hat{D}$ is limited to one or two sentences because it captures the large majority of cases. All empty ground-truth sets are eliminated, and for all ground-truth sets containing more than two sentences, just the first two sentences are selected. In a limited number of summary sentences, the ground-truth sets are empty.

5.3 Experiments

5.3.1 Experimental Datasets

Paraphrase generation datasets

The two most widely used datasets, Quora[1] and MSCOCO [349] are chosen for paraphrase generation experiments.

- **Quora** dataset consists of over 400K candidate question paraphrase pairs with manually annotated labels. Two questions are paraphrasing each other only when the question pair's label is Two different training sizes (100K and 150K) from Quora are used to have the same setting with baseline methods and show how the size of the dataset can affect the results of paraphrase generation.
- **MSCOCO** is a benchmark for the task of image captioning which contains over 82K training and 42K validation images, and at most five human-labeled caption are provided for each image. Similar to the previous works on paraphrase generation, different captions of the same image are considered as paraphrases.

Sentence fusion datasets

To assess the efficacy of the proposed sentence fusion model the datasets below are utilized for pre-training and fine tuning.

- **DISCOFUSE** a dataset of 60 million sentence fusion examples from two different corpora.
- **McKeown** contains 300 pairs of sentences retrieved from newswire[2] articles. Each pair of sentences in the dataset is accompanied by five manually written sentence fusions. The fusions are created by using workers through Amazon's Mechanical Turk service [3].

[1]https://www.quora.com/share/First-Quora-Dataset-Release-Question-Pairs
[2]http://www.cs.Columbia.edu/nlp/newsblaster.
[3]https://www.mturk.com.

Question-driven text summarisation datasets

The proposed abstractive QParaSum model is evaluated on three large-scale summarisation datasets, WikiHow [47], PubMedQA [48], and MEDIQA dataset [49].

- **WikiHow** is an abstractive summarisation dataset accumulated from the WikiHow community-based QA website, with each sample consisting of a lengthy article, a non-factoid question, and the associated summary as the answer to the question.
- **PubMedQA** is a biomedical QA dataset derived from PubMed2 abstracts. Each sample includes a question, an article, and an abstractive answer which summarises the context corresponding to the question.
- **MEDIQA** is a dataset comprising 156 consumer-submitted health questions, corresponding articles to these questions, and expert-written summaries of the answers.

5.3.2 Evaluation Metrics

BLEU4 and METEOR are used for evaluating the Para-GAN and PG-fusion model plus the mean Compression Ratio (CR) which is used as an additional metric for PG-fusion fusion model. ROUGE-N and ROUGE-L are utilized for evaluating the generated abstractive summaries.

5.3.3 Data Pre-processing and Experimental Settings

The input representation for the paraphrase model is the pre-trained wordpiece embeddings from ALBERT. For training the parapharse model, the model is trained for 10 epochs by Q-stepwise evaluation method after pre-training the generator by MLE. The discriminator is pre-trained on the generated samples from the pre-trained generator and real data. Adam optimization algorithm is used to pre-train the generator and train the discriminator. The optimal learning rates for G_θ, D_ϕ are $2e{-}6$, $5e{-}6$ calculated by Hyperopt algorithm. Hyperopt determines the optimal batch size 32 and 64 for Quora (100K, 150K) and MSCOCO [349] datasets to feed the generator and discriminator, and 20 rounds of adversarial training is performed.

The Pointer-Generator model is re-implemented as described by [10]. To have a comparable number of parameters to previous work, an encoder with 256 hidden states are used for both directions in the one-layer LSTM, and 512 for the one-layer decoder. The embedding

size is set to 128. The model is trained with the same Adagrad configuration as the content selector. Additionally, the learning rate halves after each epoch once the validation perplexity does not decrease after an epoch. Dropout is not utilized and gradient-clipping with a maximum norm of 2 has been used. All inference parameters are tuned on a 200 example subset of the validation set. Length penalty parameter α and copy mask ϵ differ across models, with α ranging from 0.6 to 1.4, and ϵ ranging from 0.1 to 0.2. The coverage penalty parameter β is set to 10, and the copy attention normalization parameter λ to 2 for both approaches.

5.3.4 Paraphrase Generation Results

At the first stage of the experiments, the paraphrase generation model (Para-GAN) is evaluated independently to assess its capability for paraphrasing extractive summaries.

Four recent paraphrase generation approaches based on GANs are employed as baseline methods, which are described in detail below.

EndtoEnd-GAN [146] regarded the generator (two stacked LSTMs encoder and decoder) as the stochastic policy and the output of discriminator (one LSTM) as its reward. In this way, they propagated the gradients from the discriminator to both the generator models and encoder models.

Div-GAN [147] proposed a conditional GAN-based framework consisting a GRU-based generator and a CNN-based discriminator. They adopted the policy gradient and early feedback techniques described in [105] for training.

Pen-GAN [148] utilized a Convolutional seq2seq model for both generator and discriminator. They engage the discriminator output as penalization rather than using policy gradients, and they avoid the Monte-Carlo search by proposing a global discriminator.

SE-GAN [149] proposed the stepwise evaluation for chit-chat dialogue generation using GRU encoder decoder for both generator and discriminator and estimated state-action values for each generation step by modifying the architecture of the discriminator.

Table 5.1 summarises the experimental results for paraphrasing on Quora (with 2 training sizes, 100K and 150K) and MSCOCO datasets. The results are reported for EndtoEnd-GAN, Div-GAN, and Pen-GAN from their paper. SE-GAN outperformed on all datasets compared to other baseline methods due to employing stepwise evaluation. Div-GAN has the worst performance on Quora-150K and MSCOCO datasets because of using policy gradient. EndtoEnd-GAN and Pen-GAN are in the second and third places respectively regarding their BLEU scores; however, Pen-GAN has a better METEOR score on Quora-100K dataset. EndtoEnd-GAN out-

Table 5.1: Experimental results of paraphrase generation on Quora (with 100K and 150K training set size) and MSCOCO datasets. The results for EndtoEnd-GAN, Div-GAN, and Pen-GAN are reported from their paper.

Method	Quora-100K		Quora-150K		MSCOCO	
	BLEU4	METEOR	BLEU4	METEOR	BLEU4	METEOR
EndtoEnd-GAN[146]	41.33	28.46	43.31	28.25	42.53	32.77
Div-GAN[147]	-	-	28.49	-	20.63	-
Pen-GAN[148]	29.07	31.27	-	-	-	-
SE-GAN[149]	41.96	30.36	43.62	31.04	42.70	32.89
Para-GAN	**43.79**	**32.41**	**44.71**	**33.23**	**45.03**	**33.97**

perfromed Pen-GAN because of proposing a generator based on stacked LSTMs and applying stochastic policy. The Para-GAN model improved the BLEU and METEOR scores compared to all these baseline methods because of using transformers and Q-stepwise rewarding jointly in the discriminator. In detail, stacks of residual attention blocks in transformer, not relying on a recurrent structure, and reward calculation based on Q-learning for each generation step are the reasons for better performance in Para-GAN.

5.3.5 Sentences Fusion Results

At the second stage of the experiments, the sentence fusion model (PG-fusion) is implemented and evaluated independently to assess its capability for compressing the sentence pairs. Four sentence fusion approaches are employed as baseline methods, which are described in detail below.

ILP [123] used the word graphs along with the Integer Linear Programming(ILP) to create the multi-document abstractive summaries. They first identified the most important documents among the documents to be summarised with the help of LexRank, cosine similarity score and overall document collection similarity score. Then, they created clusters of similar sentences among the important documents. Shortest paths are obtained by creating the word-graphs and ILP model is applied to find the sentences with maximum information and readability. ILP helps minimize the redundancy in the summary.

KeyRank [350] presents an N-best reranking method based on keyphrase extraction. Compression candidates generated by a word graph-based Multi-Sentence Compression (MSC)

Table 5.2: Results of the proposed sentence fusion (PG-fusion) model pre-trained on DISCOFUSE and fine-tuned on McKeown dataset. The results for baseline methods are reported from their papers.

Model	METEOR	BLEU	CR
ILP [123]	34.3	42.30	44.90
KeyRank [350]	35.12	44.64	37.95
Paraphrastic-Fusion [190]	43.7	42.5	41.95
Align-Fuse [351]	36.06	62.54	57.15
PG-fusion	**37.01**	**63.32**	**55.12**

approach are reranked according to the number and relevance of keyphrases they contain.

Paraphrastic-Fusion [190] designed a paraphrastic sentence fusion model which jointly performs sentence fusion and paraphrasing using skip-gram word embedding model at the sentence level. Their model improves the information coverage and at the same time abstractiveness of the generated sentences.

Align-Fuse [351] proposed a technique for sentence fusion using Word Graph based representation of an input text. A Word Graph is a directed graph comprising of a set of vertices representing the words along with a set of directed edges depicting the adjacency between corresponding words. The main advantage of their approach is using a graph representation Which generates new sentences by traversing the graph between a pair of fixed nodes.

Table 5.2 summarises the experimental results for sentence fusion on McKeown dataset. Align-Fuse outperformed (regarding BLEU metric) all the baseline methods due using BERT language model and word graph representation. The KeyRank model outperformed Paraphrastic-Fusion because of re-ranking based on the extracted keyphrases and proposing a word-graph compression method. The Paraphrastic-Fusion designed a paraphrasing and compression model based on skip-gram word embedding which outperformed ILP which is based on integer linear programming. The PG-fusion model outperformed all baseline methods due to using PG network which solves the repetition, OOV, and low coverage problems. In addition, pre-training the PG network by DISCOFUSE dataset and fine-tuning on McKeown dataset empower the PG network ability for sentence fusion.

5.3.6 Question-driven Abstractive Text Summarisation Results

At the final phase of the experiments, four recent question-driven text summarisation and three query-based[4] baseline methods are considered for evaluating the proposed model (QPara-Sum).

HSCM [31] presented an approach for extractive answer summarisation consisting of three components. In the first and second components (Word-level and Sentence-level Compare-Aggregate), an attention operation is used to align the word-level and sentence-level information between the answer sentence and question. In the third component, Question-aware Sequential Extractor, a RNN decoder is designed to label each sentence consecutively and construct the answer summary for the target question.

MSG [29] proposed Multi-hop Selective Generator (MSG), a question-driven abstractive summarisation approach that integrates multi-hop reasoning to identify the key content for assisting the answer generation. In addition to the multi-view pointer network, they introduced a multi-view coverage technique to overcome the duplication issue and generate informative and precise answers.

QPGN [32] presented a question-driven pointer-generator network that utilizes the correlation information between question-answer pairs to add substantial information when generating abstractive answer summaries. Their framework consists of four components: Bi-LSTM Encoder, seq2seq Model joint with question-aware attention, question-answer alignment with summary representations, question-driven Pointer-generator Network.

Trans [33] has studied the capability of three state-of-the-art transformers for question-driven text summarisation: BART, T5, and PEGASUS in both zero-shot and few-shot learning settings for question-driven abstractive text summarisation on MEDIQA dataset. T5 outperformed the others thus it has been considered as a baseline method.

Div-qsum [37] introduced a typical encode-attend-decode model (based on LSTM) for query-based abstractive summarisation, which first computes a vectorial representation for the document and the query, and then the decoder produces a contextual summary one word at a time.

PGRU-qsum [352] is a pointer-generator model based on GRU encoder-decoder with attention and a pointer mechanism, for generating query-based summaries.

SummerTime [353] is a comprehensive text summarising toolkit that interfaces with libraries built for NLP researchers and provides simple-to-use APIs to users. For query-based

[4]Their public code is used to apply their model to the datasets and generate question-driven summaries

summarisation, the top-k query-relevant phrases are retrieved using TF-IDF and BM25.

Table 5.3 shows the experimental results for one extractive (HSCM), three abstractive question-driven summarisation approaches (MSG, QPGN, Tran(T5)), and three query-based summarisation approaches (Div-qsum, PGRU-qsum, SummerTime) on WikiHow, PubMedQA, MEDIQA datasets. The results for HSCM, MSG, QPGN, Tran(T5) are reported from their papers, and the results for query-based baselines are generated by using their public code for the datasets. The evaluation for the question-driven extractive summary is also included to assess the impact of the paraphrasing and fusing process for generating abstractive summaries. HSCM generated extractive answer summaries and as it is shown in Table 5.3 the R1, R2, and RL for the generated extractive summaries and other abstractive baseline methods are superior to HSCM. Employing the GloVe language model and relying on a recurrent structure decoder for generating extractive answer summaries caused this inefficiency and poor performance in HSCM. MSG achieves relatively better performance than QPGN because of incorporating multi-hop reasoning for abstractive summarisation. MSG and QPGN have used the pre-trained GloVe model [72] which is not a very efficient model because of the co-occurrence matrix of words that consumes a considerable amount of memory. Besides using an inefficient language model, having multi-stages of training is another problem of these baseline methods. In Trans, each language generation model (BART, T5, PEGASUS) is pre-trained with different strategies which are unclear whether these strategies are the optimal ones. The query-based baselines have poor performances compared to question-driven baselines since answer selection and reasoning are not considered in query-based summarisation. PGRU-qsum outperformed Div-qsum and SummerTime because of utilizing attention and pointer mechanism. Div-qsum outperformed SummerTime due to using an attention mechanism for encoding documents and queries.

Favorably QParaSum model obtains the state-of-the-art results for all three datasets with the generated extractive and abstractive summaries. The results indicate that the generated extractive summary covers the essential information for satisfying answerability, understandability, and persuasiveness measures by finding the AS and its supporting sentences using the proposed multi-hop QA system. Also, the extractive stage prunes the text (the input) for the abstractive stage by removing the irrelevant and redundant information regarding the question. It was shown that the idea of exploiting an appropriate paraphrasing and fusing model for transforming the extractive summaries to abstractive is feasible since for all the three datasets the abstractive summaries obtain higher R1, R2, and RL. The paraphrase and

Table 5.3: Results on WikiHow, PubMedQA, and MEDIQA. The results for HSCM, QPGN, MSG, and Trans(T5) are reported from their paper.

	WikiHow			PubMedQA			MEDIQA		
Model	R1	R2	RL	R1	R2	RL	R1	R2	RL
HSCM	27.84	7.75	25.85	32.34	10.07	25.98	-	-	-
QPGN	28.8	9.7	27.7	34.2	12.8	28.7	-	-	-
MSG	30.5	10.5	29.3	37.2	14.8	30.2	-	-	-
Trans(T5)	-	-	-	-	-	-	38.56	18.52	26.00
PGRU-qsum	19.82	6.41	17.96	24.58	8.13	17.67	25.32	8.98	18.42
Div-qsum	18.56	5.10	15.81	22.67	7.55	16.80	23.56	7.08	17.67
SuumerTime	15.43	4.67	13.78	19.67	5.98	14.60	20.42	6.31	15.66
QParaSum-Extractive	**31.71**	**11.23**	**30.09**	**38.89**	**14.81**	**30.76**	**40.94**	**20.11**	**27.46**
QParaSum-Abstractive(pre-trained fusion)	**33.69**	**12.05**	**31.79**	**41.00**	**16.42**	**32.93**	**44.32**	**23.02**	**29.81**
QParaSum-Abstractive(fine-tuned fusion)	**33.88**	**12.43**	**31.92**	**41.30**	**16.87**	**33.21**	**44.89**	**23.61**	**30.11**

fusing models make the hybrid summariser capable to generate high-quality abstractive summaries that are closer to the human-generated ones. The QParaSum-Abstractive(pre-trained fusion) is the model with pre-trained fusion(PG) on DISCOFUSE dataset and QparaSum-Abstractive(fine-tuned fusion) comprises the the PG which is pre-trained in the first step and then fine-tuned with small set of training instances in the second step. The methodology for preparing the fusion fine-tuning dataset is explained in 5.2.3. The results show that fine-tuning the sentence fusion model on each dataset can empower the abtractive summariser. However, the QParaSum-Abstractive(pre-trained fusion) with pre-trained fusion model is capable of generating quality abstractive summaries when fusion fine-tuning is not possible for small datasets.

In Table 5.4, a practical example of the framework outputs has been shown for an instance from the MEDIQA dataset. At first stage, the AS is selected by the proposed answer selector module based on CNN and multi-head attention and then IS1 is detected by the proposed LCCS reasoning method as the most semantically relevant sentence to AS. IS2 and IS3 are selected as the next supporting sentences for IS1 and AS. Since the IS1 contains general and important information about the AS (it contains the symptoms, medication and surgery as the treatment for hernia), finding sentences explaining and extending this information is needed. The extractive summary is constructed by concatenating AS, IS1, IS2, and IS3 with the same order that they have in the article. At the second stage, the trained paraphrase model (the generator) is used for rewriting the sentences of the extractive summary to improve it and make it more similar to the human-generated summary. The fusing mechanism based

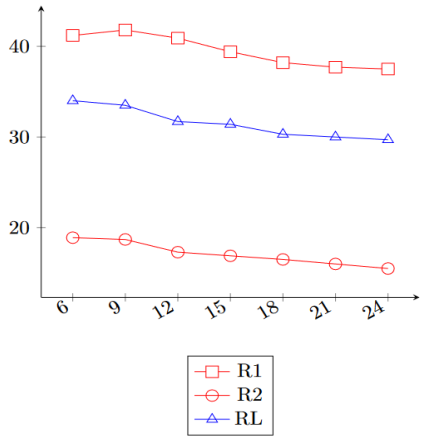

Figure 5.5: QParaSum-Abstractive performance (average R1, R2, RL across all datasets) with different question lengths.

on PG networks is applied to the paraphrased sentences and generated the final question-driven abstractive summary. IS1 and IS3 are selected as a pair of sentences and merged together while AS and IS2 are singleton sentences and their paraphrased form are used in the abstractive summary.

After generating the extractive summary, it is evident that the model tried to simulate the human action for text summarisation, regenerating and rewriting the sentences using a paraphraser and fusing mechanism. To evaluate the generated abstractive summary and compare its quality to the extractive summary, the human generated summary (gold summary) and R1, R2, RL metrics are used to demonstrate whether the generated abstractive summary is similar to the gold summary. The generated abstractive summary obtained higher R1, R2, and RL, and it shows that the generated abstractive summary has more in common sequences of words with the gold summary in comparison to the extractive and paraphrased summary. The paraphrasing and sentence fusion in QParaSum-Abstractive transform the extractive summary to abstractive which shows the proposed model could simulate the human skill for question-driven abstractive summarisation.

Fig. 5.5 shows the average of R1, R2, and RL scores for QParaSum-Abstractive(pre-trained fusion) model across all datasets with different question lengths. It is evident that the model performance is not impacted by the length of the input question since an insignificant performance degradation (R1, R2, RL) is observed when the question length increases.

Table 5.4: An example from MEDIQA dataset for extractive and abstractive summaries generated by the framework that are evaluated by the gold summary.

Question: I have an hernia I would love to take care of it ASAP I was wondering if you guys could help and tell me what should I do?
Article: Hiatal hernia (Treatment): The majority of patients who have a hiatal hernia will exhibit no signs or symptoms and will not require treatment. If you have persistent heartburn or acid reflux, you may require medication or surgery. If you suffer from heartburn or acid reflux, your doctor may prescribe the following medications:- Antacids that act as a buffer for stomach acid. Anti-acid medications such as Mylanta, Rolaids, and Tums may give immediate relief. Certain antacids may have adverse effects such as diarrhoea or kidney problems if used in excess.-Medications that inhibit acid production. Cimetidine (Tagamet), famotidine (Pepcid), nizatidine (Axid), and ranitidine are all H-2 receptor antagonists (Zantac). Prescriptions are required for stronger versions. - Anti-acid medications that aid in the healing of the esophagus. Proton pump inhibitors are more effective acid blockers than H-2 receptor antagonists, and they allow a longer time for injured esophageal tissue to repair. Lansoprazole (Prevacid 24HR) and omeprazole are two proton pump inhibitors available over-the-counter (Prilosec, Zegerid). Surgery is normally reserved for those who are unable to control their heartburn or acid reflux with medicines or who have problems such as significant inflammation or esophageal constriction. Surgery to repair a hiatal ...
Question-driven Extractive Summary: (Answer Sentence) Hiatal hernia (Treatment): The majority of patients who have a hiatal hernia will exhibit no signs or symptoms and will not require treatment. (IS1) If you have persistent heartburn or acid reflux, you may require medication or surgery. (IS2) If you suffer from heartburn or acid reflux, your doctor may prescribe the following medications:- Antacids that act as a buffer for stomach acid. (IS3) Surgery is normally reserved for those who are unable to control their heartburn or acid reflux with medicines or who have problems such as significant inflammation or esophageal constriction.
Paraphrased Summary: Hiatal hernia (Treatment): Most individuals with a hiatal hernia don't have any signs or symptoms and will not require treatment. If you have signs like repetitive acid reflux and heartburn, you may require medication or surgery. Your doctor may recommend Antiacids to neutralize stomach acid if you experience acid reflux and heartburn. Surgery is recommended if the medications do not help the individual to soothe acid reflux and heartburn, or have complexities like serious inflammation or narrowing of the esophagus.
Question-driven Abstractive Summary:Hiatal hernia (Treatment): Most individuals with a hiatal hernia don't have any signs or symptoms and will not require treatment. Your doctor may recommend Antiacids to neutralize stomach acid if you experience acid reflux and heartburn. If you have signs like repetitive acid reflux and heartburn, you may require medication or surgery, if have complexities like serious inflammation or narrowing of the esophagus.
Gold Summary: If a hiatal hernia does not have any symptoms, it won't require treatment. If the hernia causes heartburn and acid reflux, your doctor may recommend antacids. If the medications do not help or hiatal hernia causes inflammation, narrowing of the esophagus or continuous heartburn or acid reflux, your doctor might recommend surgery.
Extractive Summary Evaluation:{R1:0.367, R2: 0.147, RL: 0.220} **Paraphrased Summary Evaluation**:{R1: 0.530, R2: 0.275, RL: 0.346} **Abstractive Summary Evaluation**:{R1: 0.536, R2: 0.276, RL: 0.348}

5.4 Summary

This chapter presented the abstractive framework from the hybrid text summarisation model. The question-driven extractive summaries that are generated by the extractive framework (previous chapter) are consumed by the abstractive framework. A key barrier for generating new text is a language's inherent characteristics, such as syntax, grammar, and semantic aspects. The model must learn the correct connection between words and characters to generate a viable text, commonly accomplished through various memories and situations (prior knowledge). Such issues can be addressed in a more robust pre-learning step, in which pre-trained embedding models BERT [237], A lite bert for self-supervised learning of language representations (ALBERT) [90], ELECTRA [91], or GPT-2 are combined with transformer-based seq2seq architectures to be capable of generating plausible "natural" language text. Transformer-based GANs incorporating contextualized pre-trained language models and stepwise evaluation are blank spots that still need to be appropriately addressed for text generation, which we have presented in this chapter.

The sentences in the extractive summary are paraphrased by the paraphrase model based on GANs and transformers with Q-learning stepwise evaluation. After paraphrasing the extractive summary sentence-by-sentence, a sentence fusion mechanism based on the Pointer Generator network is designed for compressing the sentences which are labeled as pair sentences in the Next Sentence Prediction (NSP) task. The results show that the generated abstractive summary is closer to the gold summary compared to the generated extractive summary. The reason for this quality improvement in the abstractive summary is empowering the GAN with transformers and Q-learning stepwise evaluation for paraphrase generation which is complemented with a sentence fusion model.

The previous question-driven text summarisation approaches' shortcomings have been explored and considered while proposing the model. HSCM [31], MSG [29], and QPGN [32] have used the pre-trained Glove model which is not efficient because of the co-occurrence matrix of words that takes a lot of memory for storage. Besides, having multiple steps of training is another problem in these approaches. In Trans [33] each language generation model (BART, T5, PEGASUS) is pre-trained with different strategies which are unclear whether these strategies are the optimal ones. Recent query-based text summarisation approaches have bee studied in this chapter to examine their ability to be used for question-driven text summarisation. The main goal in query-based text summarisation approaches is to summarise the retrieved relevant information to the query, but in the question-driven text summarisation,

answer detection and explaining that answer in a summarised form is desired. Therefore, the query-based text summarisation approaches are not adaptable for the question-driven summarisation problem.

The results are evaluated on WikiHow, PubMedQA, and MEDIQA datasets which are appropriate for the question-driven summarisation problems. The results are compared to several baseline methods. It can be concluded that the abstractive QParaSum framework improved the quality of the summaries, and the generated abstractive summaries are more similar to the reference summaries (human-generated summaries).

Chapter 6

An Industrial Case Study

6.1 Introduction

In this chapter an industrial case is studied and the proposed model described in chapter 4 and 5 is used for processing real industry data. The industrial case and its challenges are introduced first and the dataset, experiments, and results have been demonstrated subsequently.

Openreach [1] Limited runs the UK's digital network. It's a wholly owned subsidiary of BT Group and its customers are the 688 communications providers who sell phone, broadband and Ethernet services to homes and businesses. Openreach have 37,000 people who tackle complicated engineering problems – from coordinating works with councils, highways agencies, energy suppliers and landowners, to installing and maintaining the complex kit that provides fibre broadband services.

Large volumes of information about orders are captured by and stored in different internal systems at Openreach, which are being used by teams handling the orders (for example, planners/surveyors, field engineers, customer centre agents, etc.) Having an automated system for processing this information and generating summaries based on three main questions "When is the next action on the order, What is the current stage of the order, Why is it in this stage" is crucial for having a clear picture of the latest status of the order and find out about the progress of the order easily.

Openreach orders are structured in such a way that there is an update over time. The orders' information is in structured and free text format. Every time that an action has been done for an order an update (note) will be created by an engineer or a system. These updates

[1] https://www.openreach.com/

are required to be issued to the summariser as data over time, which means that summaries at time t are required to contain an update to the previous summary at time $t-1$.

6.2 The Model for Openreach Data

Therefore, an online demo was designed for tracking the orders' progress at Openreach, generating progress summaries, and answering the user's (customer desk agent, planners, etc.) questions regarding the recent notes about the order. The online demo was based on the proposed adaptable MRC method based on CNN and multi-head attention mechanism, A-MRC Figure 6.1 that has been introduced in chapter 4. The demo contains two parts, QA and question-driven extractive summariser. The first part is a sentence-level QA model and the second part is an extractive summariser which generates progress summaries based on the three fixed questions mentioned above. The A-MRC model has been fine-tuned and experiments conducted to evaluate the model performance for Openreach orders' information. In the chapter 4 and 5 I have introduced a reasoning process and the abstractive stage which are not applied for the telecom domain due to some special characteristics of orders' notes. The reasoning process is not utilized since the extractive summary is based on three questions and their answers form the ideal summary for tracking the order's progress and also notes are quite short. In this example of telecom domain, the abstractive stage is not utilized since it has been decided to only have extractive summaries and use the exact pieces from the notes in the summary without any abstraction. The datasets, fine-tuning settings, experiments, baseline models, results, and demo examples are described in this chapter.

6.3 Dataset

The Openreach dataset contains the 4696 completed orders with start date from one-year period. Each order has 34 numerical, categorical, and unstructured text features and the most important features are mentioned below which I consider for sampling training and test datasets.

- Received-date (the date of receiving the order)
- Comp-date (the date of the order completion)
- ORDER-TYPE

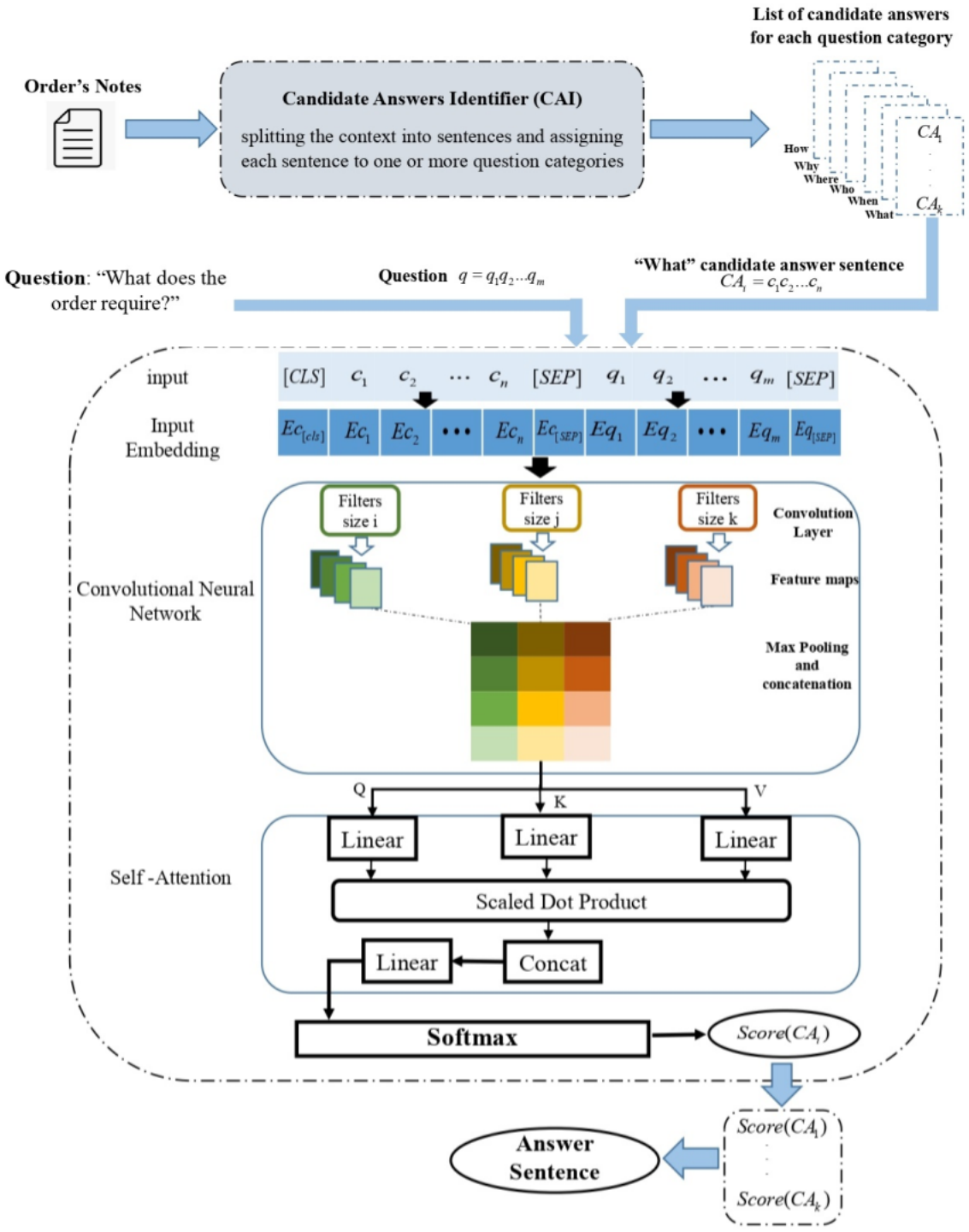

Figure 6.1: The online demo is based on the MRC model, A-MRC that has been introduced in chapter 4.

- Geographical region (such as East Anglia)
- CSP-Name (Name of Communication Service Provider)

There are 19 categories for notes which could be generated by the systems or people (engineers, planners, etc.) working on the orders,e.g. warning, notes about potential hazards, engineering notes, and etc.

6.3.1 Data Characteristics

The engineers' notes are selected as the most informative notes for processing and generating the progress summaries. Engineers' notes contain the notes generated by engineers and the systems during the order completion period. Following is an example of three consecutive engineers' notes generated for an order.

Table 6.1: An example of three consecutive engineers' notes generated for an order

16/10/2019 11:43:00 - <Name A> Associated A55 with this job. `<link>` Please follow this link to access A55 details. I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has been submitted. The A55 reference is <reference-id>.
28/10/2019 09:27:00 - <Name B> Photographic evidence available I cannot complete this task because the end customer or their representative was not present at the premises. Jumper work is completed at the cabinet. Please see additional information New DLI lead in needs full fit.
29/10/2019 18:14:00 - <Name C> I cannot complete this task because I have run out of time within my scheduled hours. The end customer cannot use the service. The work outstanding is Install completed. Tested dis 20m from bt66.

As it is evident the notes contain technical terms, links, reference numbers and etc. For understanding the meaning of notes knowing the meaning of the technical terms is essential(e.g. A55 is Excavation Drawing, Information required by contractors before they can start work and DLI is an equipment).

6.3.2 Training Dataset

I have prepared a training dataset for fine-tuning the A-MRC to Openreach dataset and to verify the performance of the proposed model. A SQuAD style Question Answering training dataset is prepared by using engineering notes and 5 different questions:

- Why is the order not complete?
- Where is the order passed to?

Table 6.2: The summary of orders' features selected for test dataset

Order-ID	RECEIVED-DATE	COMP-DATE	ORDER-TYPE	Geographical region	CSP-NAME
000000001	27-Mar-2019	06-Dec-2019	TYPE1	Northern England	CSP1
000000002	15-Aug-2019	28-Nov-2019	TYPE1	London and South East	CSP1
000000003	04-Nov-2019	04-Dec-2019	TYPE1	Wales and Midlands	CSP3
000000004	22-Nov-2019	25-Nov-2019	TYPE1	Wessex	CSP1
000000005	25-Sep-2019	25-Nov-2019	TYPE1	Wales and Midlands	CSP3
000000006	08-Oct-2019	26-Nov-2019	TYPE1	Wales and Midlands	CSP1
000000007	02-Sep-2019	26-Nov-2019	TYPE1	East Anglia	CSP3
000000008	28-May-2019	26-Nov-2019	TYPE2	Northern England	CSP2
000000009	11-Jul-2019	28-Nov-2019	TYPE2	Wales and Midlands	CSP4
000000010	12-Nov-2019	28-Nov-2019	TYPE2	Wales and Midlands	CSP2

- What has been done for the order so far?
- What does the order require?
- What is the next step for the order?

The datasets consist of 400 Note-Answer-Question triples. I sampled the notes from different orders considering the distribution of different orders in the main dataset to prepare a training dataset with diverse types of orders and notes.

6.3.3 Test Dataset

The test dataset contains 10 orders selected with different features, the main distribution of the Openreach dataset is considered while selecting the sample data. For example, if the 66% of the dataset is orders with "ORDER-TYPE TYPE1", same portion has been considered for sample selection. Due to the manual anonymization, data preparation, etc. only a few indicated examples are used since it's not possible to self-evaluate the answers and it needs to be done with support of knowledge domain experts, e.g. potential users. The summary of the selected orders for testing the fine-tuned model is provided in Table 6.2.

6.3.4 Data Anonymisation

Due to nature of the dataset it might contain personally identifiable information (PII) or confidential information like people's names, address parts or location references, phone numbers, unique order references, etc. The model and dataset is not intended for releasing extern-

ally or publishing and it was not required to do anonymisation while training. However, any examples of notes in this document were anonymised for privacy purposes. The name of the engineers, reference numbers, links, addresses, and numbers are anonymised by replacing them with a token with named entity type, such as <Name X>, <reference-id>, <link>, <Address/street name>, <number>.

6.4 Experiments

6.4.1 Fine-tuning Process

I have utilized the same setting explained in chapter 4, the Natural Questions(NQ) dataset is used for pre-training the A-MRC. For both the pre-training and fine-tuning, the question sentence pairs were generated by CAI. After generating the candidate answer sentences for question categories with CAI, the candidate answer sentence and question pairs were generated for training the A-MRC model. The candidate answer sentence which contains the answer gets the label 1, and other candidate sentences get the label 0.

I used the pre-trained ALBERT base model for token embeddings, consisting of 12 Transformer blocks with 12 self-attention heads and the hidden size of 768. There is no analytical formula to calculate an appropriate value of the hyperparameters to obtain the optimal model parameter. I have utilized Ray Tune Python library with Hyperopt algorithm as explained in chapter 4. Filter size, number of filters, learning rate, and batch size hyperparameters were optimized for telecom domain. The search spaces are {2, 3, 4, 5}, {10, 20, 30}, {1e-7, 2e-7,1e-8, 2e-8, 5e-8}, {4, 8, 16} for filter size, number of filters, learning rate, and batch size respectively. The optimal combination of hyperparameters values that maximize the model performance are {2,3,4}, 10, 2e-8, and 4 for filter size, number of filters, learning rate, and batch size respectively. The optimal values for filter size, number of filters, learning rate, and batch size for the pre-training are calculated as follows: {2, 3, 4}, 100, 2e-5, 64. I applied early stopping on the development set for both training stages on the loss value. I set the max number of epochs to 9 and 3 for transfer and adapt steps, respectively. I set the maximum sequence length for BERT and ALBERT to 128 tokens. I utilized the Adam optimization algorithm for the parameter update. The cross-entropy loss function is used to calculate the loss. Figure 6.2 demonstrates the online demo based on the fine-tuned model. In window 1, up to 3 notes with Date-time Name of Engineer content format should be entered. The input for window 2 is optional (it can be blank), you can enter a list of questions separated with ‘,’.

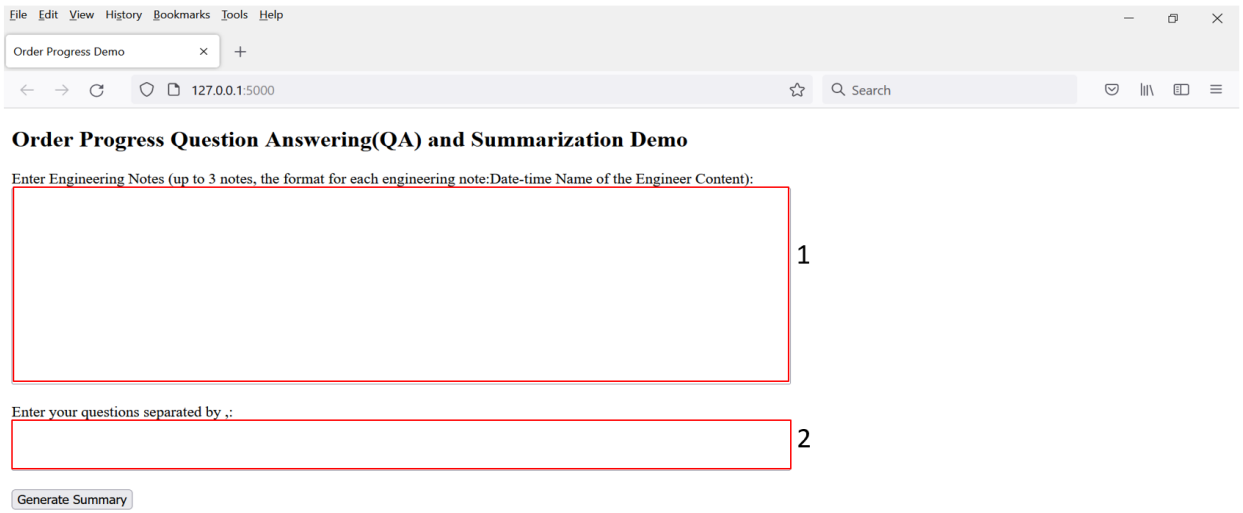

Figure 6.2: The online demo designed for tracking progress of the orders at Openreach.

Then by entering “Generate Summary” button, you will see results after a few seconds.

The demo processes three engineering notes at a time and the model starts finding the answers to the questions in the most recent notes (the notes are sorted based on their Date-Time attribute). The demo will go to the older notes if the answer to the question couldn’t be found in the most recent note. If the user enters the question(s), the QA system will extract the answer sentences along with their DateTime attribute and scores calculated by the A-MRC model.

The summarisation part will generate extractive summaries based on three fixed questions “Why is the order not complete?, What does the order require?, Where is the order passed to?”. The fine-tuned model extracts the answer sentences for the above questions and the answers form the question-driven summary for the order to show the progress of the order and the reason for any delays. The description for technical words and abbreviations will be provided at the end of the summary. The redundant sentences will be removed since maybe one sentence could contain the answer to two questions. If any answer is not from the most recent note the DateTime of the answer will be printed showing that the sentence has been selected from an older note. Section 6.4.4 illustrates some example inputs and the generated outputs for the both QA and summarisation part.

6.4.2 Baseline Methods

To demonstrate the effectiveness of the proposed model after fine-tuning, I compare against several ready-to-use QA and extractive summarisation tools:

- Facebook's BART summariser: This Transformer-based model is very complex and takes time to return on a CPU, even for small pieces of text. It is compatible with many languages thanks to the multi-language add-on.

- T5 summariser: T5 is an extremely large new neural network model that is trained on a mixture of unlabeled text and labeled data from popular natural language processing tasks. The most obvious new idea behind T5 is that it is a text-to-text model and during the training, the model is asked to produce new text as an output.

- Roberta's QA: The Roberta Transformer-based model is accurate and fairly fast. Roberta's QA within context is about sending a piece of text, and asking a question that the model answers thanks to your piece of text.

6.4.3 Results and Discussion

The results are generated for the test dataset by baseline methods and the proposed demo which are evaluated by humans since reference summaries (gold summaries) are not available for Openreach orders. A 5-point scale (very good "5", good "4", moderate "3", bad "2", very bad "1") is considered for the performance evaluation [354]. For each question either in the QA or the summarisation part if the extracted answer is correct and selected from the right note the rate will be very good(5), but if the answer is correct but selected from an old note while the recent note contains an answer the rate will be moderate (3). If the extracted answer is not correct the rate will be very bad (1).

For the summarisation section, the overall quality of the generated summaries is also rated based on the coherency and readability which are the results of the pre-processing step and removing the redundancy in the methods. The mean rates of the three fixed questions and the overall quality score is the final rating for the generated summary.

The baseline methods get the same input as the online demo (up to 3 notes). The baseline models are designed for generic summarisation and the DateTime attribute is not defined and recognized by these models. The results show that BART summariser outperformed the T5 summariser. The proposed model outperformed the baseline methods for both QA and

Table 6.3: The mean scores for ready-to-use summarisation and QA tools and the proposed demo on the test dataset orders' notes.

Model	summarisation score	QA score
Facebook's BART summariser	2.6	___
T5 summariser	1.4	___
Roberta's QA	___	2.8
Online demo	**3.8**	**4.0**

summarisation task as shown in Table 6.3. The result for each model is the mean of the summaries' and answers' scores generated for the orders' notes in the test dataset.

If the sequence of the notes changes (not the notes' content) the baseline models generate different summaries since the position of the sentences is important in the input text not the DateTime attribute(example1 in Figure. 6.3). The example2 in the Figure. 6.4 shows that by changing the date and time of the notes, the BART generates the same summary because the generic summarisation tools are not designed for processing time-stamped text documents. The designed demo recognized this change, sort the notes based on their DateTime and starts processing the most recent note first.

The baseline tool for the QA part, Roberta QA, cannot detect the most recent note and the first answer found in the document even from an older note is selected (all the notes together are considered as a single text document, examples are shown in 6.4.4).

6.4.4 Demo Examples

In this section, some examples of the notes in the test dataset are shown to compare the baselines' outputs and the demo output. In Table 6.4 T5 and BART summariser generated generic summaries for the notes regardless of their DateTime attribute. BART outperformed T5 due to using both BERT (bidirectional encoder) and GPT (left to the right decoder) architecture with seq2seq translation and achieves the state of the art results in the summarisation task.

The summaries generated by the proposed demo contains the name of the engineer who has been involved lately. Every update in the summary comes with its DateTime attribute to show the progress of the order regarding the three fixed questions at different times. A definition for the technical terms and abbreviations in the summary is provided at the end to make it more understandable for non technical users. Also, the equipment's name used in the

Example 1:	
13/10/2019 10:02:00 - <Name A> Associated A55 with this job. Please follow this link to access A55 details. <link> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has been submitted. The A55 reference is NA. Please see additional information Joint meet required to locate cable. Current dig has not found cable. 27/08/2019 11:34:00 - <Name B> Attempted to track the cable But it comes up in serveral places and no clear path. Ltok when together best track was on top of the CSP cable. But schematics show it along the path but can't fine it on thencat and genny here at all. I have sent Eu a message stating that dp will be in contact to make an appointment so we can look inside again. Seems likely that the lead in has been cut or pushed back behind the CSP socket when they installed. As next door number <number> have a new lead in that goes back to The CSP equipment. 12/10/2019 15:11:00 - <Name C> I cannot complete this task because I do not have the skills required to complete the task and could not obtain assistance on the day. I am passing to an engineer with track and locate equipment to complete the order and no appointment required. Please see additional information Task is going to need a track and locate doing. Dig has been done and new lead in and duct put in but there is no joint to join it to. They have exposed a cable but it isn't our cable so I have nothing to link the armoured cable to .	27/08/2019 11:34:00 - <Name B> Attempted to track the cable But it comes up in serveral places and no clear path. Ltok when together best track was on top of the CSP cable. But schematics show it along the path but can't fine it on thencat and genny here at all. I have sent Eu a message stating that dp will be in contact to make an appointment so we can look inside again. Seems likely that the lead in has been cut or pushed back behind the CSP socket when they installed. As next door number <number> have a new lead in that goes back to The CSP equipment. 12/10/2019 15:11:00 - <Name C> I cannot complete this task because I do not have the skills required to complete the task and could not obtain assistance on the day. I am passing to an engineer with track and locate equipment to complete the order and no appointment required. Please see additional information Task is going to need a track and locate doing. Dig has been done and new lead in and duct put in but there is no joint to join it to. They have exposed a cable but it isn't our cable so I have nothing to link the armoured cable to . 13/10/2019 10:02:00 - <Name A> Associated A55 with this job. Please follow this link to access A55 details. <link> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has been submitted. The A55 reference is NA. Please see additional information Joint meet required to locate cable. Current dig has not found cable.
BART Summary: Joint meet required to locate cable. Current dig has not found cable. Seems likely that the lead in has been cut or pushed back behind the CSP socket when they installed. As next door number <number> have a new lead in that goes back to the CSP equipment. They have exposed a cable but it isn't our cable so I have nothing to link the armoured cable to.	**BART Summary:** A55 has been submitted for the job. Current dig has not found cable. I cannot complete this task because I do not have the skills required to complete the task and could not obtain assistance on the day. I am passing to an engineer with track and locate equipment to finish the order and no appointment required.
Demo Summary: The update that our engineer <Name A> has reported on 13/10/2019 10:02:00 about your order: *** [link] I can not complete this task because further work is required by the Dig and Aux control team for a maintenance dig , An A55 has been submitted . Please follow this link to access A55 details . *** Please Continue to review fault tracker for further updates. List of technical terms: A55: Excavation Drawing - Information required by contractors before they can start work.	**Demo Summary:** The update that our engineer <Name A> has reported on 13/10/2019 10:02:00 about your order: *** [link] I can not complete this task because further work is required by the Dig and Aux control team for a maintenance dig , An A55 has been submitted . Please follow this link to access A55 details . *** Please Continue to review fault tracker for further updates. List of technical terms: A55: Excavation Drawing - Information required by contractors before they can start work.

Figure 6.3: An example of the baseline's performance for processing order's notes.

Example 2:	
31/10/2019 14:51:00 - <Name A> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has not been submitted because Associated/common jobs with this task. second stage onsite and dealing. needs putting into hold delay. associated job numbers are <number> and <number> faults at <number>, <number> and <number> <street name>.(No manually entered closure notes) 01/11/2019 08:59:00 - <Name B> Task complete. Still outstanding work to be done by tails advised to come off by <Name C> at CER to stop it keep coming out 09/11/2019 13:05:00 - <Name D> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has not been submitted because Needs track and locate . Two digs been done on this job and they have now exposed a joint and the line is good to that joint. The has been an armoured cable put in from the customer to another dig and it's just left in the dig with no cable to joint it to. They exposed a power cable that is not ours so we have no cable in the same dig point that we can joint it to. We need to expose our cable and if we can't we need to but a duct in to the next dig down so we can link it to the joint exposed in the other dig location.	01/11/2019 08:59:00 - <Name A> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has not been submitted because Associated/common jobs with this task. second stage onsite and dealing. needs putting into hold delay. associated job numbers are <number> and <number> faults at <number>, <number> and <number> <street name>.(No manually entered closure notes) 09/11/2019 13:05:00 - <Name B> Task complete. Still outstanding work to be done by tails advised to come off by <Name C> at CER to stop it keep coming out 31/10/2019 14:51:00 - <Name D> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has not been submitted because Needs track and locate . Two digs been done on this job and they have now exposed a joint and the line is good to that joint. The has been an armoured cable put in from the customer to another dig and it's just left in the dig with no cable to joint it to. They exposed a power cable that is not ours so we have no cable in the same dig point that we can joint it to. We need to expose our cable and if we can't we need to but a duct in to the next dig down so we can link it to the joint exposed in the other dig location.
BART Summary: An A55 has not been submitted because Associated/common jobs with this task. second stage onsite and dealing. needs putting into hold delay. associated job numbers are <number> and <number> faults at <number>, <number> and <number> <street name>.(No manually entered closure notes)	**BART Summary:** An A55 has not been submitted because Associated/common jobs with this task. second stage onsite and dealing. needs putting into hold delay. associated job numbers are <number> and <number> faults at <number>, <number> and <number> <street name>.(No manually entered closure notes)
Demo Summary: The update that our engineer <Name D> has reported on 09/11/2019 13:05:00 about your order: *** can not complete this task because further work is required by the Dig and Aux control team for a maintenance dig , An A55 has not been submitted because Needs track and locate . Task complete . [01/11/2019 08:59:00] We need to expose our cable and if we can't we need to but a duct in to the next dig down so we can link it to the joint exposed in the other dig location . *** Please Continue to review fault tracker for further updates. List of technical terms: A55: Excavation Drawing - Information required by contractors before they can start work. ca: Customer Apparatus duct: The pipe holding cables joint: Where two wires are connected together, usually in a Joint Box	**Demo Summary:** The update that our engineer <Name B> has reported on 09/11/2019 13:05:00 about your order: *** can not complete this task because further work is required by the Dig and Aux control team for a maintenance dig , An A55 has not been submitted because Associated/common jobs with this task . [01/11/2019 08:59:00] Task complete. needs putting into hold delay . [01/11/2019 08:59:00] *** Please Continue to review fault tracker for further updates. List of technical terms: A55: Excavation Drawing - Information required by contractors before they can start work.

Figure 6.4: An example of the baseline's performance for processing order's notes.

Table 6.4: An example for summarisation outputs of the T5, BART and the proposed demo.

16/10/2019 11:43:00 - <Name A> Associated A55 with this job. Please follow this link to access A55 details. <link> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has been submitted. The A55 reference is <reference-id>.(No manually entered closure notes) 28/10/2019 09:27:00 - <Name B> Photographic evidence available I cannot complete this task because the end customer or their representative was not present at the premises. Jumper work is completed at the cabinet.Please see additional information New DLI lead in needs full fit . 29/10/2019 18:14:00 - <Name C> I cannot complete this task because I have run out of time within my scheduled hours. The end customer cannot use the service. The work outstanding is Install completed. Tested dis 20m from bt66.(No manually entered closure notes)
BART Summary: I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has been submitted. The A55 reference is <reference-id>. The work outstanding is Install completed. The end customer cannot use the service. New DLI lead in needs full fit.
T5 Summary: the end customer or their representative was not present at the premises. the work outstanding is Install completed. Tested dis 20m from bt66.(No manually entered closure notes)
Demo Summary: The update that our engineer <Name C> has reported on 29/10/2019 18:14:00 about your order: *** I can not complete this task because I have run out of time within my scheduled hours. The work outstanding is Install completed. Please follow this link to access A55 details. <link> I can not complete this task because further work is required by the Dig and Aux control team for a maintenance dig , An A55 has been submitted. [16/10/2019 11:43:00] *** Please Continue to review fault tracker for further updates. List of technical terms: A55: Excavation Drawing - Information required by contractors before they can start work.

summary are clarified by adding “(the equipment)” after the corresponding word (Example in Table 6.5).

The Table 6.6 shows the outputs of the Roberta QA and the proposed demo for the QA part. The Roberta outputs the first piece of the text that answers the question regardless of it DateTime attribute. So, for getting the most up-to-date answer, the last note should be placed at the beginning of the input text. The proposed demo sorts the notes based on their DateTime attribute and starts searching for the answer from the most recent note, if it couldn’t be found the second recent note will be processed. Also, the answers from the demo are in sentence-level which provide enough information to the user. The name of the engineer

Table 6.5: An example for summarisation outputs of the T5, BART and the proposed demo.

11/08/2019 12:09:00 - <Name A> I cannot complete this task because I was unable to access Openreach line plant located within Openreach network. Work not commenced.Please see additional information Still waiting new duct . 26/08/2019 19:54:00 - <Name B> I cannot complete this task because I have run out of time within my scheduled hours. The end customer cannot use the service. The work outstanding is Need to chase up maint dig on this job but triage are shut raitining for tomorrow advised Eu I will let him know what's happening as it's been going on since <date>.(No manually entered closure notes) 27/08/2019 11:27:00 - <Name C> Photographic evidence available I cannot complete this task because the end customer or their representative was not present at the premises. Line ok at jf4 o/s <Address> nte5 detected to the customer and cable tracks towards house access required .(No manually entered closure notes)
BART Summary: I cannot complete this task because I have run out of time within my scheduled hours. The end customer cannot use the service. The work outstanding is Need to chase up maint dig on this job but triage are shut raitining for tomorrow. Still waiting new duct.
T5 Summary: the work outstanding is Need to chase up maint dig but triage are shut raitining for tomorrow advised Eu I will let him know what's happening as it's been going on since <date>.
Demo Summary: The update that our engineer <Name C> has reported on 27/08/2019 11:27:00 about your order: *** evidence available I can not complete this task because the end customer or their representative was not present at the premises . Line ok at jf4(the equipment) o/s <Address> nte detected to the customer and cable tracks towards house access required . *** Please Continue to review fault tracker for further updates. List of technical terms: nte: Network Terminating Equipment

and the DateTime of the answer sentence is provided to show when and who was involved in that particular stage of the order. The "Model confident score" shows the score of the selected answer explained in 4.2.1 and the model's confidence about that answer. These are certain features of the proposed demo for providing comprehensive answers and summaries for the orders.

Table 6.6: An example for Question Answering outputs of the Roberta and the proposed demo.

31/10/2019 14:51:00 - <Name A> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has not been submitted because Associated/common jobs with this task. second stage onsite and dealing. needs putting into hold delay. associated job numbers are <number> and <number> faults at <number>, <number> and <number> <street name>.(No manually entered closure notes) 01/11/2019 08:59:00 - <Name B> Task complete. Still outstanding work to be done by tails advised to to com off by <Name C> at CER to stop it keep coming out 09/11/2019 13:05:00 - <Name D> I cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig, An A55 has not been submitted because Needs track and locate . Two digs been done on this job and they have now exposed a joint and the line is good to that joint. The has been an armoured cable put in from the customer to another dig and it's just left in the dig with no cable to joint it to. They exposed a power cable that is not ours so we have no cable in the same dig point that we can joint it to. We need to expose our cable and if we can't we need to but a duct in to the next dig down so we can link it to the joint exposed in the other dig location.
Roberta QA: Why is the order not complete? Associated/common jobs with this task What does the order require? further work What has been done for this order so far? Two digs
Demo Answers: Why is the order not complete? Model confident score: 0.9733 09/11/2019 13:05:00 <Name D> cannot complete this task because further work is required by the Dig and Aux control team for a maintenance dig An A55 has not been submitted because Needs track and locate . What does the order require? Model confident score: 0.8456 09/11/2019 13:05:00 <Name D> We need to expose our cable and if we can't we need to but a duct in to the next dig down so we can link it to the joint exposed in the other dig location. What has been done for this order so far? Model confident score: 0.9140 09/11/2019 13:05:00 <Name D> Two digs been done on this job and they have now exposed a joint and the line is good to that joint.

6.5 Summary

In this chapter, I presented the online demo designed for telecom domain based on the proposed MRC model explained in chapter 4. I have utilized the A-MRC model for generating

question-driven extractive summaries and answering questions to show the orders' progress at Openreach. First I fine-tuned the A-MRC model to the telecom domain by using the small training dataset based on the past completed orders. The Openreach dataset and its features are described in 6.3 which shows special settings are required for fine-tuning the proposed model and designing the demo. Then, a test dataset is designed for evaluating ready-to-use QA and summarisation tools and the proposed demo. A human evaluation is designed since there are no gold summaries available for this telecom domain, which has been described in 6.4.3. There are other telecom datasets available, some of them might have gold summaries but for this specific dataset (Openreach dataset) from telecom domain, a human evaluation process is required. The results show the proposed demo outperformed both QA and summarisation tools due to the fine-tuning process and its design for processing timestamped text. Although, the performance of the adaptable A-MRC model has been evaluated for various domains in chapter 4, adding some features for understanding the structure of orders' notes and DateTime attribute complement its capability for processing the Openreach data. Finding the answer sentences for the three main questions "Why is the order not complete?, What does the order require?, Where is the order passed to?" helps to generate an extractive summary showing the progress of the order which will help teams (for example, planners/surveyors, field engineers, customer centre agents, etc.) at Openreach having a clear picture of the latest status of the order. Furthermore, the A-MRC is utilized for a sentence-level QA system which help Openreach users to ask any questions about the orders and get the answers instantly along with the corresponding DateTime and engineer's name who was involved with that particular stage. The answer provided for each question has a score that shows how confident the model is for the extracted answer. Different examples for QA and summarisation parts in the demo are provided in 6.4.4.

Chapter 7

Conclusion and Future Work

7.1 A Hybrid Question-driven Text Summarisation Model

A hybrid adaptable approach has been proposed for generating extractive and abstractive question-driven summaries, which can be easily adapted to different domains with specialized vocabulary by utilizing a small training dataset. The proposed hybrid approach takes the benefits of both extractive and abstractive techniques and generates an extractive summary based on the question in the first stage. Then, the generated extractive summary is rewritten and shortened to produce the abstractive summary.

The question-driven extractive stage is based on an open-domain multi-hop QA system comprising a sentence-level MRC method and reasoning process. CAI, CNN and multi-head attention-based answer selector, and QE module are MRC components in the multi-hop QA. The CAI module with six functions based on linguistic and syntactic features and patterns has been introduced for reducing the document to sentences (candidate answer sentences) that could answer the given question. A joint CNN and multi-head attention neural network has been designed to analyse and assign a score to each candidate answer sentence based on its relevance to the question. The CNN-attention layer calculates the relevance score based on the correlation of the semantic features extracted from the question and the candidate answer sentence. If the selected answer sentence score calculated by MRC component is less than Θ, the question expansion module generates paraphrased questions until a candidate answer achieves a score greater than Θ. A lightweight hybrid question expansion based on contextualized embedding and lexical resources (WordNet) is designed that replaces some question keywords with domain-related synonyms to generate paraphrased questions. After selecting the answer sentence, an unsupervised reasoning process (it is called *LCCS* reasoning

process) based on ***L***exical ***C***overage and ***C***ontextualized ***S***imilarity is proposed for selecting supporting sentences (justification sentences). All the sentences in the document are considered as the candidate justifications sentences, and those candidates that are closest to the question, answer sentence, and selected justification sentences in the embedding space are selected. A pre-trained BERT and cosine similarity are utilized for measuring the semantic similarity combined with the lexical coverage for calculating the justification score of each sentence in the document.

The reasoning process helps us select appropriate, relevant sentences explaining the answer sentence and then constructs the final extractive summary. The justification sentences and answer sentence are rearranged according to their original indexes in the given document to bring coherence in the selected sequence of sentences and generate the question-driven extractive summary. Experiments are conducted to evaluate the proposed MRC model and its components to make sure they perform efficiently. The results show the MRC model outperforms all the baseline models for MRC-style sentence-level QA in closed-domain and open-domain. Then, the proposed multi-hop QA model results are evaluated for both multi-hop Question Answering and question-driven extractive summarisation (chapter 4). The results show the model performs well in comparison to the multi-hop QA and question-driven text summarisation baseline methods.

An abstractive stage based on a paraphrase framework and sentence fusion mechanism has been designed. The paraphrase generation model is designed with GANs containing transformers encoders and decoders and Q-stepwise evaluation for regenerating the extractive summaries produced in stage one. The regenerated summary is then consumed with the proposed fusion mechanism based on the next sentence prediction and the PG network to produce shorter high-quality abstractive summaries. The next sentence prediction is utilized for detecting the sentence pairs and singletons to choose the relevant sentences from the paraphrased summary generated by the GAN model. The sentence pairs are provided to the sentence fusion mechanism, which aims to create shorter abstractive sentences. The PG network gets a sentence pair and generates a fused sentence. After fusing the sentence pairs, the singletons and fused sentences construct the question-driven abstractive summary. Several experiments are implemented to evaluate the proposed paraphrase generation model, and fusion mechanism independently, and then the whole abstractive framework is evaluated on the question-driven datasets.

The experimental results for stages one and two, extractive and abstractive question-

driven text summarisation, show that the hybrid approach generates quality question-driven abstractive summaries in different domains. Selecting the question-relevant information (answer sentence and its supporting sentences) reduces the input document and helps the abstractive summariser to process and abstract the question-relevant information only. The results are compared to baseline methods which are available for question-driven text summarisation, query-based text summarisation, paraphrase generation, sentence fusion, MRC models, and multi-hop QA. The model is capable to be adapted to different domains by the training strategies described in section 4.3.3 and 5.3.3.

The proposed model in chapter 4 and 5 simulated the humans steps for generating a question-driven abstractive summary. Selecting pieces of information answering the target question, rewriting and shortening them are steps which are simulated by the proposed model. This simulation is performed by employing an designing an extractive and an abstractive methods described above. The presented hybrid question-driven text summariser generates efficient question-driven extractive and abstractive summaries. The generated summaries satisfied the answerability, understandability, and persuasiveness, as they not only provide the answer sentence, they also provide the complementary information to the answer sentence.

7.2 Applications

We are now observing an exponential surge of data originating from many sources, such as various forms of company records. The processing of this sparse, noisy, and domain-specific data is extremely difficult. BT, a UK-based technology corporation and the Ph.D. project's sponsor, has a huge number of field engineers, desk-based agents, and customer support services who produce, collect, and handle daily vast volumes of temporally organized unstructured and semi-structured data. The challenge they encounter is effectively and efficiently managing the order's progress and identifying the causes of delays.

The objective of this Ph.D. project is to provide computational models that can effectively and efficiently extract useful information for technical and non-technical users at BT from diverse technical order record documents that are extremely noisy and lack a structural pattern. To this purpose, it would be advantageous to automatically summarise and derive relevant information in the form of answers to questions from this huge collection of distributed occurring data.

BT's order information is collected and stored in structured and unstructured text formats

by many internal systems. These are utilized by order processing teams. This amount of text about orders is an excellent source of information that must be properly summarised in a manner that includes the most pertinent information for:

- why is the order not complete?
- What does the order require?
- where has the order been passed to?

Consequently, the purpose of this study is to summarise the source text in relation to a specific question(s). A question-driven summary must attain three objectives: answerability, readability, and persuasion. In recent years, there have been several attempts to create algorithms for question-driven automated text summarisation. A summary based on a question should include a credible answer as well as information that support and explain that answer. The query-based document summarisation methods are not ideal for this problem scenario since they try to provide a fluent summary of a given document that answers or is related to the search query. The public question-driven datasets contain a question, text document, and an abstractive human-generated summary based on the question. In contrast, the ideal summary for a BT order should cover information about the three questions mentioned above while the input text has a Datetime attribute described in chapter 6.

The question-driven summary will enable the desk agents to have a clear picture of the latest status of the order journey at time t, rather of having to review order information from several systems and databases when a client contacts to inquire about the status of their order. In other words, the goal is to create summaries at time t that assist humans in comprehending the text material rapidly and effectively. The structure of BT orders is such that they are updated at regular intervals. These changes must be supplied as data over time into the summariser. It indicates that summaries at time t must include an update to the prior summary at time $t-1$. The summary S will contain the most recent and vital information regarding an order, depending on the WH questions indicated above for reporting the order's status. The BT orders' data characteristics require a different fine-tuning procedure compared to general question-driven text summarisation.

The online demo designed for the BT domain is presented in chapter 6 based on the proposed MRC model for generating question-driven extractive summaries and answering questions to show the orders' progress at BT. First, the A-MRC model is fine-tuned to the BT domain by using the small training dataset based on the past completed orders. The BT

dataset and its features are described in 6.3 which shows special settings are required for fine-tuning the proposed model and designing the demo. Then, a test dataset is designed for evaluating ready-to-use QA and summarisation tools and the proposed demo. A human evaluation is designed since there are no gold summaries available for the BT domain. The results show the demo outperformed both QA and summarisation tools due to the fine-tuning process and its design for processing timestamped text. Although, the performance of the adaptable A-MRC model has been evaluated for various domains in 4, adding some features for understanding the structure of orders' notes and the DateTime attribute complement its capability for processing the BT data.

7.3 Future Work

The proposed paraphrase framework for rewriting the extractive sentences is based on GAN containing transformer encoder and decoder and Q-learning for evaluating the generated subsequences. VAE and GAN learn by different loss functions. VAE uses the probabilistic graph model and learns by finding good posterior $p(z \mid x)$ and likelihood $p(x \mid z)$. To generate new instances, VAE first chooses a prior distribution $p(z)$ according to the expected x, and then samples a hidden state from $p(z)$ and feeds it into the decoder [355]. GAN directly tries to find a suitable generator by the "min-max two-player game." Utilizing VAEs based on transformers for converting extractive summaries into abstractive could be a variant of the proposed model which will be evaluated in the future.

In the proposed abstractive summariser, the singletons have remained unchanged and only sentence pairs are merged. Designing and proposing a sentence compression technique complements the sentence fusion mechanism for compressing the singletons in the abstractive summary, which is listed as one of the future works. Regarding the publicly available training datasets for sentence compression, proposing a sequence-to-sequence model could be beneficial for generating shorter abstractive summaries.

The proposed reasoning process is based on contextualized similarity and lexical coverage, which shows promising performance in joint with the CNN-attention-based MRC model. A graph-based model for selecting the supporting sentences and reasoning on the text document is the potential future work for the extractive approach. A self-attention on the sentences' graphs depending on the named entities, verb phrases, and noun phrases is the future idea for the supporting sentence selection.

The CAI component designed for filtering irrelevant content to the question from the context is based on linguistic and syntactic features and it could be modified by adding more methods for applying co-reference resolution. The QE module designed for paraphrasing the questions could be evolved by back-translation for rewriting ambiguous questions.

Also, the summaries generated by the extractive summariser in stage one comprise four sentences, the answer sentence, and three supporting sentences. Calculating the optimum length of the summary is the future work for question-driven text summarisation. The number of supporting sentences required for each answer sentence and the optimal size of the summary could be calculated based on the text document, question, and the selected answer sentence.

7.4 Future Text Summarisation Research Directions

The future text summarisation research directions are briefly explained below:

- Contributing to complex issues in multi-document text summarisation like redundancy, temporal dimension, co-reference, and sentence reordering [154].
- Proposing ATS systems in which the input can be in the form of meetings, videos, sounds, etc., and the output in a format other than text. For example, the input might be in the form of text, and the output can be represented as tables, statistics, graphics, visual rating scales, etc. ATS systems that allow visualization of the summaries will help users to get the required content in less time [154].
- Designing ATS systems that achieve high accuracy and efficiency for both long and short text summarisation [7].
- Proposing multi-lingual ATS systems since most ATS systems focus on the English language content. The quality of the current ATS systems needs to be improved for many other languages. Developing and improving NLP tools to generate summaries for non-English languages is required [356].
- Building ATS systems using a small amount of training data through traditional NLP techniques such as syntactic analysis, grammar analysis, semantic analysis, etc [7].
- Proposing new metrics and approaches for the automatic evaluation of the computer-generated summaries. It is very hard to find out what an ideal (or even correct) sum-

mary is because the ATS systems can generate good summaries that are different from human-generated summaries [170]. It is very subjective to identify a good summary. Therefore, manual evaluations may not be suitable for all types of summaries [357].

Bibliography

[1] S Syed. Abstractive summarization of social media posts: A case study using deep learning. *Weimar: Bauhaus University*, 2017.

[2] Yuanhang Su and C-C Jay Kuo. On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, 356:151–161, 2019.

[3] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[4] Haifeng Wang, Hua Wu, Zhongjun He, Liang Huang, and Kenneth Ward Church. Progress in machine translation. *Engineering*, 2021.

[5] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 2020.

[6] Mohammad Shehab, Ahamad Tajudin Khader, and Mohammad A. Alia. Enhancing cuckoo search algorithm by using reinforcement learning for constrained engineering optimization problems. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 812–816, 2019.

[7] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679, 2021.

[8] Bolanle Ojokoh and Emmanuel Adebisi. A review of question answering systems. *Journal of Web Engineering*, 17(8):717–758, 2018.

[9] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*, 2021.

[10] Abigail See, Peter J Liu, and Christopher 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: Long Papers)*, pages 1073–1083, 2017.

[11] Horacio Saggion and Thierry Poibeau. Automatic text summarization: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 3–21. Springer, 2013.

[12] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[13] Gerald DeJong. An overview of the frump system. *Strategies for natural language processing*, 113:149–176, 1982.

[14] Karen Sparck Jones and Brigitte Endres-Niggemeyer. Automatic summarizing, 1995.

[15] Paul Over, Hoa Dang, and Donna Harman. Duc in context. *Information Processing & Management*, 43(6):1506–1520, 2007.

[16] Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. Overview of the tac 2010 knowledge base population track. In *Third text analysis conference (TAC 2010)*, volume 3, pages 3–3, 2010.

[17] Karen Sparck Jones. What might be in a summary? *Information retrieval*, 93(1):9–26, 1993.

[18] Inderjeet Mani. *Automatic summarization*, volume 3. John Benjamins Publishing, 2001.

[19] Karen Sparck Jones. Automatic summarizing: factors immarizing: factors and directions. *Advances in automatic text summarization*, page 1, 1999.

[20] Karen Spärck Jones. Automatic summarising: The state of the art. *Information Processing & Management*, 43(6):1449–1481, 2007.

[21] Elena Lloret and Manuel Palomar. Text summarisation in progress: a literature review. *Artificial Intelligence Review*, 37(1):1–41, 2012.

[22] Christopher C Yang and Fu Lee Wang. Hierarchical summarization of large documents. *Journal of the American Society for Information Science and Technology*, 59(6):887–902, 2008.

[23] Sanda Harabagiu and Finley Lacatusu. Using topic themes for multi-document summarization. *ACM Transactions on Information Systems (TOIS)*, 28(3):1–47, 2010.

[24] Lei Huang, Yanxiang He, Furu Wei, and Wenjie Li. Modeling document summarization as multi-objective optimization. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 382–386. IEEE, 2010.

[25] Ani Nenkova, Sameer Maskey, and Yang Liu. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*, HLT '11, USA, 2011. Association for Computational Linguistics.

[26] Wen-tau Yih, Joshua T Goodman, Lucretia H Vanderwende, and Hisami Suzuki. Document summarization by maximizing informative content words, April 20 2010. US Patent 7,702,680.

[27] Konstantinos Koumpis and Steve Renals. Automatic summarization of voicemail messages using lexical and prosodic features. *ACM Transactions on Speech and Language Processing (TSLP)*, 2(1):1–es, 2005.

[28] Mohammad Reza Fani Sani, Ahmad A. Kardan, and Arman Cohan. A supporting tool in online learning forums based on multi-documents summarization. In *4th International Conference on e-Learning and e-Teaching (ICELET 2013)*, pages 30–35, 2013.

[29] Yang Deng, Wenxuan Zhang, and Wai Lam. Multi-hop inference for question-driven summarization. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6734–6744, 2020.

[30] Hongya Song, Zhaochun Ren, Shangsong Liang, Piji Li, Jun Ma, and Maarten de Rijke. Summarizing answers in non-factoid community question-answering. Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pages 405–414, 2017.

[31] Yang Deng, Wenxuan Zhang, Yaliang Li, Min Yang, Wai Lam, and Ying Shen. Bridging hierarchical and sequential context modeling for question-driven extractive answer

summarization. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 1693–1696, 2020.

[32] Yang Deng, Wai Lam, Yuexiang Xie, Daoyuan Chen, Yaliang Li, Min Yang, and Ying Shen. Joint learning of answer selection and answer summary generation in community question answering. Proceedings of the AAAI Conference on Artificial Intelligence, pages 7651–7658, 2020.

[33] Travis R Goodwin, Max E Savery, and Dina Demner-Fushman. Flight of the pegasus? comparing transformers on few-shot and zero-shot multi-document abstractive summarization. Proceedings of COLING. International Conference on Computational Linguistics, page 5640. NIH Public Access, 2020.

[34] Nazreena Rahman and Bhogeswar Borah. A survey on existing extractive techniques for query-based text summarization. In *2015 International Symposium on Advanced Computing and Communication (ISACC)*, pages 98–102. IEEE, 2015.

[35] Mahsa Afsharizadeh, Hossein Ebrahimpour-Komleh, and Ayoub Bagheri. Query-oriented text summarization using sentence extraction technique. 2018 4th international conference on web research (ICWR), pages 128–132. IEEE, 2018.

[36] Hadrien Van Lierde and Tommy WS Chow. Query-oriented text summarization based on hypergraph transversals. *Information Processing & Management*, 56(4):1317–1338, 2019.

[37] Preksha Nema, Mitesh M Khapra, Anirban Laha, and Balaraman Ravindran. Diversity driven attention model for query-based abstractive summarization. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1063–1072, 2017.

[38] Tatsuya Ishigaki, Hen-Hsen Huang, Hiroya Takamura, Hsin-Hsi Chen, and Manabu Okumura. Neural query-biased abstractive summarization using copying mechanism. *Advances in Information Retrieval*, 12036:174, 2020.

[39] DMITRII AKSENOV. Abstractive text summarization with neural sequence-to-sequence models. Master's thesis, 2020.

[40] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. International Conference on Machine Learning, pages 11328–11339. PMLR, 2020.

[41] Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 4453–4460, 2018.

[42] Kaiqiang Song, Bingqing Wang, Zhe Feng, Ren Liu, and Fei Liu. Controlling the amount of verbatim copying in abstractive summarization. Proceedings of the AAAI Conference on Artificial Intelligence, pages 8902–8909, 2020.

[43] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. Neural abstractive text summarization with sequence-to-sequence models. *ACM Transactions on Data Science*, 2(1):1–37, 2021.

[44] Panagiotis Kouris, Georgios Alexandridis, and Andreas Stafylopatis. Abstractive text summarization: enhancing sequence to sequence models using word sense disambiguation and semantic content generalization. *Computational Linguistics*, pages 1–41, 2021.

[45] Ziqiang Cao, Wenjie Li, Furu Wei, Sujian Li, et al. Retrieve, rerank and rewrite: Soft template based neural summarization. Association for Computational Linguistics (ACL), 2018.

[46] Min Yang, Qiang Qu, Wenting Tu, Ying Shen, Zhou Zhao, and Xiaojun Chen. Exploring human-like reading strategy for abstractive text summarization. Proceedings of the AAAI Conference on Artificial Intelligence, pages 7362–7369, 2019.

[47] Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.

[48] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2567–2577, 2019.

[49] Max Savery, Asma Ben Abacha, Soumya Gayen, and Dina Demner-Fushman. Question-driven summarization of answers to consumer health questions. *Scientific Data*, 7(1):1–9, 2020.

[50] AD Dongare, RR Kharde, Amit D Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012.

[51] Konstantin Lopyrev. Generating news headlines with recurrent neural networks. *arXiv preprint arXiv:1512.01712*, 2015.

[52] Shengli Song, Haitao Huang, and Tongxiao Ruan. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, 78(1):857–875, 2019.

[53] C Lee Giles, Gary M Kuhn, and Ronald J Williams. Dynamic recurrent neural networks: Theory and applications. *IEEE Transactions on Neural Networks*, 5(2):153–156, 1994.

[54] Anthony J Robinson. An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*, 5(2):298–305, 1994.

[55] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[56] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[57] Pattreeya Tanisaro and Gunther Heidemann. An empirical study on bidirectional recurrent neural networks for human motion recognition. In *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[58] Kamal Al-Sabahi, Zhang Zuping, and Yang Kang. Bidirectional attentional encoder-decoder model and bidirectional beam search for abstractive summarization. *arXiv preprint arXiv:1809.06662*, 2018.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[60] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[61] Elliott Jobson and Abiel Gutiérrez. Abstractive text summarization using attentive sequence-to-sequence rnns, 2016.

[62] Qicai Wang, Peiyu Liu, Zhenfang Zhu, Hongxia Yin, Qiuyue Zhang, and Lindong Zhang. A text abstraction summary model based on bert word embedding and reinforcement learning. *Applied Sciences*, 9(21):4701, 2019.

[63] Elozino Egonmwan and Yllias Chali. Transformer-based model for single documents neural summarization. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 70–79, 2019.

[64] Ye Zhang and Byron C Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.

[65] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015.

[66] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[68] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[69] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[70] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394, 2010.

[71] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[72] Jeffrey Pennington, Richard Socher, and Christopher 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.

[73] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[74] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[75] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding.

[76] Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, 2019.

[77] John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, 2019.

[78] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

[79] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.

[80] Prajit Ramachandran, Peter J Liu, and Quoc Le. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, 2017.

[81] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[82] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[83] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[84] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[85] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[86] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8968–8975, 2020.

[87] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[88] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*, 2019.

[89] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[90] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[91] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[92] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[93] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[94] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[95] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[96] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[97] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, 2017.

[98] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation.

[99] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in neural information processing systems*, pages 4601–4609, 2016.

[100] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1171–1179, 2015.

[101] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, 2018.

[102] Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. Toward diverse text generation with inverse reinforcement learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4361–4367, 2018.

[103] Philip Bachman and Doina Precup. Data generation as sequential decision making. *Advances in Neural Information Processing Systems*, 28:3249–3257, 2015.

[104] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction.

[105] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[106] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

[107] Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. *Advances In Neural Information Processing Systems*, 29, 2016.

[108] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. *Advances in neural information processing systems*, 30, 2017.

[109] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[110] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *International Conference on Machine Learning*, pages 4006–4015. PMLR, 2017.

[111] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[112] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[113] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[114] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[115] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[116] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[117] Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21, pages 21–32, 2016.

[118] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637, 2017.

[119] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. A generative model for category text generation. *Information Sciences*, 450:301–315, 2018.

[120] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[121] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[122] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. Text summarization branches out, pages 74–81, 2004.

[123] Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[124] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[125] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer, 2016.

[126] Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. On accurate evaluation of gans for language generation. *arXiv preprint arXiv:1806.04936*, 2018.

[127] Guy Tevet, Gavriel Habib, Vered Shwartz, and Jonathan Berant. Evaluating text gans as language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2241–2247, 2019.

[128] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

[129] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[130] James Lucas, George Tucker, Roger Grosse, and Mohammad Norouzi. Understanding posterior collapse in generative latent variable models. 2019.

[131] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space.

[132] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International conference on machine learning*, pages 3881–3890. PMLR, 2017.

[133] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[134] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687. PMLR, 2018.

[135] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[136] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[137] Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450, 2018.

[138] Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405. PMLR, 2019.

[139] Suguru Yasutomi and Toshihisa Tanaka. Parameter estimation for von mises-fisher mixture model via gaussian distribution. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–5. IEEE, 2014.

[140] Jiacheng Xu and Greg Durrett. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513, 2018.

[141] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[142] Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. Decomposable neural paraphrase generation. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3403–3414, 2019.

[143] Yao Fu, Yansong Feng, and John P Cunningham. Paraphrase generation with latent bag of words. *Advances in Neural Information Processing Systems*, 32:13645–13656, 2019.

[144] AB Siddique, Samet Oymak, and Vagelis Hristidis. Unsupervised paraphrasing via deep reinforcement learning. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1800–1809, 2020.

[145] Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. Unsupervised paraphrasing by simulated annealing. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 302–312, 2020.

[146] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, 2019.

[147] Yue Cao and Xiaojun Wan. Divgan: Towards diverse paraphrase generation via diversified generative adversarial network. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 2411–2421, 2020.

[148] Gerson Vizcarra and Jose Ochoa-Luna. Paraphrase generation via adversarial penalizations. Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), pages 249–259, 2020.

[149] Yi-Lin Tuan and Hung-Yi Lee. Improving conditional sequence generative adversarial networks by stepwise evaluation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):788–798, 2019.

[150] Monika Joshi, Hui Wang, and Sally McClean. Dense semantic graph and its application in single document summarisation. In *Emerging Ideas on Information Filtering and Retrieval*, pages 55–67. Springer, 2018.

[151] Virendra Kumar Gupta and Tanveer J Siddiqui. Multi-document summarization using sentence clustering. In *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, pages 1–5. IEEE, 2012.

[152] Deepak Sahoo, Ashutosh Bhoi, and Rakesh Chandra Balabantaray. Hybrid approach to abstractive summarization. *Procedia computer science*, 132:1228–1237, 2018.

[153] M Jishma Mohan, C Sunitha, Amal Ganesh, and A Jaya. A study on ontology based abstractive summarization. *Procedia Computer Science*, 87:32–37, 2016.

[154] Parul Agarwal and Shikha Mehta. Empirical analysis of five nature-inspired algorithms on real parameter optimization problems. *Artificial Intelligence Review*, 50(3):383–439, 2018.

[155] Mudasir Mohd, Rafiya Jan, and Muzaffar Shah. Text document summarization using word embedding. *Expert Systems with Applications*, 143:112958, 2020.

[156] Iram Khurshid Bhat, Mudasir Mohd, and Rana Hashmy. Sumitup: A hybrid single-document text summarizer. Soft computing: Theories and applications, pages 619–634. Springer, 2018.

[157] Kazuhiro Takeuchi. A study on operations used in text summarization. 2002.

[158] Vishal Gupta and Gurpreet Singh Lehal. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268, 2010.

[159] Franck Dernoncourt, Mohammad Ghassemi, and Walter Chang. A repository of corpora for summarization. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[160] Kristian Woodsend and Mirella Lapata. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, 2010.

[161] Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd*

Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 829–833, 2015.

[162] Misha Denil, Alban Demiraj, and Nando De Freitas. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*, 2014.

[163] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, 2016.

[164] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[165] Yuxiang Wu and Baotian Hu. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[166] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, 2018.

[167] Parth Mehta, Gaurav Arora, and Prasenjit Majumder. Attention based sentence extraction from scientific articles using pseudo-labeled data. *arXiv preprint arXiv:1802.04675*, 2018.

[168] Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. Summarization based on embedding distributions. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1984–1989, 2015.

[169] Laifu Chen and Minh Le Nguyen. Sentence selective neural extractive summarization with reinforcement learning. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–5. IEEE, 2019.

[170] N Moratanch and S Chitrakala. A survey on extractive text summarization. In *2017 international conference on computer, communication and signal processing (ICCCSP)*, pages 1–6. IEEE, 2017.

[171] Elena Baralis, Luca Cagliero, Saima Jabeen, Alessandro Fiori, and Sajid Shah. Multi-document summarization based on the yago ontology. *Expert Systems with Applications*, 40(17):6976–6984, 2013.

[172] Jianwei Niu, Huan Chen, Qingjuan Zhao, Limin Su, and Mohammed Atiquzzaman. Multi-document abstractive summarization using chunk-graph and recurrent neural network. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[173] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing 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.

[174] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 93–98, 2016.

[175] Gaetano Rossiello, Pierpaolo Basile, Giovanni Semeraro, MD Ciano, and Gaetano Grasso. Improving neural abstractive text summarization with prior knowledge. *URANIA*, 16, 2016.

[176] Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100, 2017.

[177] Angela Fan, David Grangier, and Michael Auli. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, 2018.

[178] Pashutan Modaresi and Stefan Conrad. Simurg: An extendable multilingual corpus for abstractive single document summarization. In *Proceedings of the 8th annual meeting of the Forum on Information Retrieval Evaluation*, pages 24–27, 2016.

[179] Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computa-*

tional Linguistics (Volume 2: Short Papers), pages 163–169, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[180] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[181] Patrick Doetsch, Albert Zeyer, and Hermann Ney. Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 361–366. IEEE, 2016.

[182] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[183] Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. Generative adversarial network for abstractive text summarization. Thirty-second AAAI conference on artificial intelligence, 2018.

[184] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Discriminative adversarial search for abstractive summarization. International Conference on Machine Learning, pages 8555–8564. PMLR, 2020.

[185] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 13063–13075, 2019.

[186] Banafsheh Rekabdar, Christos Mousas, and Bidyut Gupta. Generative adversarial network with policy gradient for text summarization. 2019 IEEE 13th international conference on semantic computing (ICSC), pages 204–207. IEEE, 2019.

[187] Nobel Dang, Ashish Khanna, and Viswanatha Reddy Allugunti. Ts-gan with policy gradient for text summarization. Data Analytics and Management, pages 843–851. Springer, 2021.

[188] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, 2002.

[189] Regina Barzilay and Kathleen R McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005.

[190] Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. Abstractive unsupervised multi-document summarization using paraphrastic sentence fusion. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1191–1204, 2018.

[191] Jackie Chi Kit Cheung and Gerald Penn. Unsupervised sentence enhancement for automatic summarization. In *EMNLP*, pages 775–786, 2014.

[192] Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond Ng, and Bita Nejat. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1602–1613, 2014.

[193] Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond Ng. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 136–146, 2013.

[194] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah 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*, pages 1077–1086, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[195] Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. Discofuse: A large-scale dataset for discourse-based sentence fusion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3443–3455, 2019.

[196] Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1430–1441, 2018.

[197] Kaiqiang Song, Lin Zhao, and Fei Liu. Structure-infused copy mechanisms for abstract-

ive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1717–1729, 2018.

[198] Tobias Falke, Leonardo FR Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, 2019.

[199] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact aware neural abstractive summarization. In *thirty-second AAAI conference on artificial intelligence*, 2018.

[200] Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[201] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, 2016.

[202] Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. Improving sentence compression by learning to predict gaze. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1528–1533, San Diego, California, June 2016. Association for Computational Linguistics.

[203] Yishu Miao and Phil 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*, pages 319–328, 2016.

[204] Thibault Févry and Jason Phang. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, 2018.

[205] Chanakya Malireddy, Tirth Maniar, and Manish Shrivastava. Scar: sentence compression using autoencoders for reconstruction. In *Proceedings of the 58th Annual Meeting*

of the Association for Computational Linguistics: Student Research Workshop, pages 88–94, 2020.

[206] Liangguo Wang, Jing Jiang, and Lejian Liao. Sentence compression with reinforcement learning. In *International Conference on Knowledge Science, Engineering and Management*, pages 3–15. Springer, 2018.

[207] Yang Zhao, Zhiyuan Luo, and Akiko Aizawa. A language model based evaluator for sentence compression. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 170–175, 2018.

[208] Xiaojun Wan and Jianguo Xiao. Graph-based multi-modality learning for topic-focused multi-document summarization. Twenty-First International Joint Conference on Artificial Intelligence, 2009.

[209] You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47(2):227–237, 2011.

[210] Nazreena Rahman and Bhogeswar Borah. A method for semantic relatedness based query focused text summarization. International Conference on Pattern Recognition and Machine Intelligence, pages 387–393. Springer, 2017.

[211] Hajime Morita, Tetsuya Sakai, and Manabu Okumura. Query snowball: a co-occurrence-based approach to multi-document summarization for question answering. *Information and Media Technologies*, 7(3):1124–1129, 2012.

[212] Weikang Li, Xingxing Zhang, Yunfang Wu, Furu Wei, and Ming Zhou. Document-based question answering improves query-focused multi-document summarization. CCF International Conference on Natural Language Processing and Chinese Computing, pages 41–52. Springer, 2019.

[213] Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei, and Yanran Li. Attsum: Joint learning of focusing and summarization with neural attention. *arXiv preprint arXiv:1604.00125*, 2016.

[214] Mingjun Zhao, Shengli Yan, Bang Liu, Xinwang Zhong, Qian Hao, Haolan Chen, Di Niu, Bowei Long, and Weidong Guo. Qbsum: A large-scale query-based document

summarization dataset from real-world applications. *Computer Speech & Language*, 66:101166, 2021.

[215] Sheng-hua Zhong, Yan Liu, Bin Li, and Jing Long. Query-oriented unsupervised multi-document summarization via deep learning model. *Expert systems with applications*, 42(21):8146–8155, 2015.

[216] Mahmood Yousefi-Azar and Len Hamey. Text summarization using unsupervised deep learning. *Expert Systems with Applications*, 68:93–105, 2017.

[217] Tal Baumel, Matan Eyal, and Michael Elhadad. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704*, 2018.

[218] Shuai Wang, Xiang Zhao, Bo Li, Bin Ge, and Daquan Tang. Integrating extractive and abstractive models for long text summarization. 2017 IEEE International Congress on Big Data (BigData Congress), pages 305–312. IEEE, 2017.

[219] Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. *arXiv preprint arXiv:1909.03186*, 2019.

[220] Yangbin Chen, Yun Ma, Xudong Mao, and Qing Li. Multi-task learning for abstractive and extractive summarization. *Data Science and Engineering*, 4(1):14–23, 2019.

[221] Hanqi Jin, Tianming Wang, and Xiaojun Wan. Multi-granularity interaction network for extractive and abstractive multi-document summarization. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6244–6254, 2020.

[222] Chenliang Li, Weiran Xu, Si Li, and Sheng 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 (Short Papers)*, pages 55–60, 2018.

[223] Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tie-Yan Liu. Decoding with value networks for neural machine translation. *Advances in Neural Information Processing Systems*, 30, 2017.

[224] Sanjay K Dwivedi and Vaishali Singh. Research and reviews in question answering system. *Procedia Technology*, 10:417–424, 2013.

[225] Anjali Saini and PK Yadav. A survey on question answering system. *International Journal of Engineering and Computer Science*, 6(3), 2017.

[226] Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Survey on challenges of question answering in the semantic web. *Semantic Web*, 8(6):895–920, 2017.

[227] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)*, 44(1):1–50, 2012.

[228] Chris Quirk, Chris Brockett, and Bill Dolan. Monolingual machine translation for paraphrase generation. 2004.

[229] Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, 2005.

[230] Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. Combining multiple resources to improve smt-based paraphrasing model. In *Proceedings of acl-08: Hlt*, pages 1021–1029, 2008.

[231] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, 2010.

[232] Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D Manning. Answering complex open-domain questions through iterative query generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2590–2602, 2019.

[233] Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100, 2021.

[234] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*, 2018.

[235] Wenhan Xiong, Xiang Lorraine Li, Srinivasan Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. Answering complex open-domain questions with multi-hop dense retrieval. *International Conference on Learning Representations*, 2021.

[236] Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. Neural machine reading comprehension: Methods and trends. *Applied Sciences*, 9(18):3698, 2019.

[237] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[238] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[239] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6529–6537, 2019.

[240] Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. *arXiv preprint arXiv:1905.06933*, 2019.

[241] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

[242] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[243] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, 2016.

[244] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602, 2017.

[245] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[246] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.

[247] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, 2018.

[248] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations*, 2018.

[249] Caiming Xiong, Victor Zhong, and Richard Socher. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*, 2017.

[250] Wei Wang, Ming Yan, and Chen Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1705–1714, 2018.

[251] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[252] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*, 2019.

[253] Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*, 2019.

[254] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[255] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, 2017.

[256] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745, 2018.

[257] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. A survey on question answering systems over linked data and documents. *Journal of intelligent information systems*, 55(2):233–259, 2020.

[258] Boris Katz, Sue Felshin, Jimmy Lin, and Gregory Marton. Viewing the web as a virtual database for question answering. In *New Directions in Question Answering*, pages 215–226, 2004.

[259] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999.

[260] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[261] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[262] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[263] Michael Petrochuk and Luke Zettlemoyer. Simplequestions nearly solved: A new upperbound and baseline approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 554–558, 2018.

[264] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the askmsr question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 257–264, 2002.

[265] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[266] Petr Baudiš. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165, 2015.

[267] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, 2016.

[268] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[269] Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2285–2295, 2019.

[270] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[271] Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. Meta multi-task learning for sequence modeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[272] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[273] Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. *arXiv preprint arXiv:1906.05807*, 2019.

[274] Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W Bruce Croft, and Mohit Iyyer. Open-retrieval conversational question answering. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 539–548, 2020.

[275] Sarvesh Soni and Kirk Roberts. Evaluation of dataset selection for pre-training and fine-tuning transformer language models for clinical question answering. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5532–5538, 2020.

[276] Jibin Fu, Jinzhong Xu, and Keliang Jia. Domain ontology based automatic question answering. In *2009 international conference on computer engineering and technology*, volume 2, pages 346–349. IEEE, 2009.

[277] Varsha Bhoir and MA Potey. Question answering system: A heuristic approach. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 165–170. IEEE, 2014.

[278] Sweta P Lende and MM Raghuwanshi. Question answering system on education acts using nlp techniques. In *2016 world conference on futuristic trends in research and innovation for social welfare (Startup Conclave)*, pages 1–6. IEEE, 2016.

[279] Sanglap Sarkar, Venkateshwar Rao, SM Baala Mithra, and Subrahmanya VRK Rao. Nlp algorithm based question and answering system. In *Seventh International Conference on Computational Intelligence, Modeling and Simulation*, 2015.

[280] Srinivasu Badugu and R Manivannan. A study on different closed domain question answering approaches. *International Journal of Speech Technology*, pages 1–11, 2020.

[281] Bhanu Pratap Singh Rawat, Wei-Hung Weng, Preethi Raghavan, and Peter Szolovits. Entity-enriched neural models for clinical question answering. *arXiv preprint arXiv:2005.06587*, 2020.

[282] Deepthi Godavarthi and A. M. Sowjanya. Queries related to covid-19: a more effective retrieval through finetuned albert with bm25l question answering system. *World Journal of Engineering*, 2021.

[283] Lin-Qin Cai, Min Wei, Si-Tong Zhou, and Xun Yan. Intelligent question answering in restricted domains using deep learning and question pair matching. *Ieee Access*, 8:32922–32934, 2020.

[284] Jifan Chen, Shih-ting Lin, and Greg Durrett. Multi-hop question answering via reasoning chains. *arXiv preprint arXiv:1910.02610*, 2019.

[285] Yixuan Tang, Hwee Tou Ng, and Anthony Tung. Do multi-hop question answering systems know how to answer the single-hop sub-questions? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3244–3249, 2021.

[286] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017. *ArXiv abs/1609.02907*, 2017.

[287] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017.

[288] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*, 2018.

[289] Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 42–48, 2018.

[290] Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of NAACL-HLT*, pages 2306–2317, 2019.

[291] Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2704–2713, 2019.

[292] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, 2019.

[293] Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150, 2019.

[294] Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9073–9080, 2020.

[295] Nan Shao, Yiming Cui, Ting Liu, Shijin Wang, and Guoping Hu. Is graph structure necessary for multi-hop question answering? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7187–7192, 2020.

[296] Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8823–8838, 2020.

[297] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–220, 2017.

[298] Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. Efficient and robust question answering from minimal context over documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1725–1735, 2018.

[299] Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. Multi-mention learning for reading comprehension with neural cascades. *arXiv preprint arXiv:1711.00894*, 2017.

[300] Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. Coarse-grain fine-grain coattention network for multi-evidence question answering. *arXiv preprint arXiv:1901.00603*, 2019.

[301] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 252–262, 2018.

[302] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. Proceedings of the second workshop on statistical machine translation, pages 228–231, 2007.

[303] Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. Neural machine reading comprehension: Methods and trends. *Applied Sciences*, 9(18), 2019.

[304] G Salton. The SMART system. *Retrieval Results and Future Plans*, 1971.

[305] Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. Sg-net: Syntax-guided machine reading comprehension. In *AAAI*, pages 9636–9643, 2020.

[306] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[307] Jafar A Alzubi, Rachna Jain, Anubhav Singh, Pritee Parwekar, and Meenu Gupta. Cobert: Covid-19 question answering system using bert. *Arabian Journal for Science and Engineering*, pages 1–11, 2021.

[308] Felix Hamborg, Corinna Breitinger, and Bela Gipp. Giveme5w1h: A universal system for extracting main events from news articles. *arXiv preprint arXiv:1909.02766*, 2019.

[309] Murray Singer. Answering wh-questions about sentences and text. *Journal of Memory and Language*, 25(2):238–254, 1986.

[310] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008.

[311] Suzan Verberne. Developing an approach for why-question answering. In *Student Research Workshop*, pages 39–46, 2006.

[312] Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Dan Jurafsky. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 217–220, 2005.

[313] Daniel Gildea and Julia Hockenmaier. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 57–64, 2003.

[314] Frederik Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska De Jong. An overview of event extraction from text. *DeRiVE@ ISWC*, pages 48–57, 2011.

[315] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740, 2012.

[316] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[317] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[318] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4291–4308, 2020.

[319] Leon Derczynski, Jun Wang, Robert Gaizauskas, and Mark A Greenwood. A data driven approach to query expansion in question answering. *arXiv preprint arXiv:1203.5084*, 2012.

[320] Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. Covid-qa: A question answering dataset for covid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, 2020.

[321] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David 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.

[322] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[323] Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. Rikinet: Reading wikipedia pages for natural question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6762–6771, 2020.

[324] James Bergstra, Dan Yamins, David D Cox, et al. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. Citeseer, 2013.

[325] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[326] Linlong Xiao, Nanzhi Wang, and Guocai Yang. A reading comprehension style question answering model based on attention mechanism. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–4. IEEE, 2018.

[327] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[328] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. *arXiv preprint arXiv:2001.09694*, 2020.

[329] Arantxa Otegi, Jon Ander Campos, Gorka Azkune, Aitor Soroa, and Eneko Agirre. Automatic evaluation vs. user preference in neural textual questionanswering over

covid-19 scientific literature. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, 2020.

[330] Revanth Gangi Reddy, Bhavani Iyer, Md Arafat Sultan, Rong Zhang, Avi Sil, Vittorio Castelli, Radu Florian, and Salim Roukos. End-to-end qa on covid-19: domain adaptation with synthetic training. *arXiv preprint arXiv:2012.01414*, 2020.

[331] Sharon Levy, Kevin Mo, Wenhan Xiong, and William Yang Wang. Open-domain question-answering for covid-19 and other emergent domains. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 259–266, 2021.

[332] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, 2019.

[333] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[334] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, et al. Cord-19: The covid-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, 2020.

[335] Jinhyuk Lee, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang. Contextualized sparse representations for real-time open-domain question answering. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 912–919, 2020.

[336] Devendra Singh Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. End-to-end training of neural retrievers for open-domain question answering. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 6648–6662. Association for Computational Linguistics, 2021.

[337] Gautier Izacard and Édouard Grave. Leveraging passage retrieval with generative models for open domain question answering. Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, 2021.

[338] Vikas Yadav, Steven Bethard, and Mihai Surdeanu. If you want to go far go together: Unsupervised joint candidate evidence retrieval for multi-hop question answering. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4571–4581, 2021.

[339] Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Alignment over heterogeneous embeddings for question answering. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2681–2691, 2019.

[340] Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Unsupervised alignment-based iterative evidence retrieval for multi-hop question answering. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4514–4525, 2020.

[341] Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2578–2589, 2019.

[342] Harsh Trivedi, Heeyoung Kwon, Tushar Khot, Ashish Sabharwal, and Niranjan Balasubramanian. Repurposing entailment for multi-hop question answering tasks. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2948–2958, 2019.

[343] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1657–1668, 2017.

[344] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28, 2015.

[345] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[346] Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–320, 2010.

[347] Micha Elsner and Deepak Santhanam. Learning to fuse disparate sentences. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 54–63, 2011.

[348] Kapil Thadani and Kathleen McKeown. Supervised sentence fusion with single-stage inference. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1410–1418, 2013.

[349] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. European conference on computer vision, pages 740–755. Springer, 2014.

[350] Florian Boudin and Emmanuel Morin. Keyphrase extraction for n-best reranking in multi-sentence compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 298–305, 2013.

[351] Raksha Agarwal and Niladri Chatterjee. Improvements in multi-document abstractive summarization using multi sentence compression with word graph and node alignment. *Expert Systems with Applications*, 190:116154, 2022.

[352] Johan Hasselqvist and Niklas Helmertz. Query-based abstractive summarization using neural networks. Master's thesis, 2017.

[353] Ansong Ni, Zhangir Azerbayev, Mutethia Mutuma, Troy Feng, Yusen Zhang, Tao Yu, Ahmed Hassan, and Dragomir Radev. Summertime: Text summarization toolkit for

non-experts. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 329–338, 2021.

[354] Neslihan Iskender, Tim Polzehl, and Sebastian Möller. Reliability of human evaluation for text summarization: Lessons learned and challenges ahead. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 86–96, 2021.

[355] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans): Challenges, solutions, and future directions. *ACM Comput. Surv.*, 54(3), may 2021.

[356] Riadh Belkebir and Ahmed Guessoum. Talaa-atsf: a global operation-based arabic text summarization framework. In *Intelligent Natural Language Processing: Trends and Applications*, pages 435–459. Springer, 2018.

[357] Elena Lloret, Laura Plaza, and Ahmet Aker. The challenging task of summary evaluation: an overview. *Language Resources and Evaluation*, 52(1):101–148, 2018.