GNET-QG: GRAPH NETWORK FOR MULTI-HOP QUESTION GENERATION

SAMIN JAMSHIDI
Bachelor of Computer Science, Amirkabir University of Technology, 2018

A thesis submitted
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

GNET-QG: GRAPH NETWORK FOR MULTI-HOP QUESTION GENERATION

SAMIN JAMSHIDI

Date of Defence: August 15, 2024

Dr. Y. Chali Thesis Supervisor	Professor	Ph.D.
Dr. W. Osborn Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. R. Benkoczi Thesis Examination Committee Member	Professor	Ph.D.
Dr. A. Fiori Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

This thesis is dedicated to those who have been told they're not good enough, smart enough, or worthy enough. It's for those who have faced countless obstacles and setbacks but refused to give up on their dreams.

Abstract

Question generation involves crafting questions based on a given input context and accompanying answers. While recent advancements in sequence-to-sequence models have proven successful in generating natural language questions, there is an increasing need for models capable of handling more intricate contexts to produce detailed questions. Multi-hop question generation, which is more challenging, aims to establish connections between multiple facts from diverse input contexts to formulate questions. In our research, we study the utilization of a Graph Attention Network (GAT) and a BART model for multi-hop question generation, Our proposed model, is GNET-QG (Graph Network for Multi-Hop Question Generation). GNET-QG efficacy is assessed on the HotpotQA dataset using metrics such as METEOR, BLEU, and ROUGE, showcasing an enhancement over previous methodologies.

Acknowledgments

I want to express my sincere thanks to my supervisor, Dr. Yllias Chali, for their valuable guidance, support, and encouragement throughout the duration of this project. Their expertise and mentorship have been instrumental in shaping the direction of my research.

I am deeply grateful to Dr. Wendy Osborn, a member of my supervisory committee, whose unwavering support has been a constant source of strength throughout this journey. Her guidance, especially during the challenging times, has been invaluable, shaping both my personal and academic growth.

I want to express my gratitude to Dr. Robert Benkoczi, a member of my supervisory committee. His recommendation letter was instrumental in securing a scholarship, and his guidance during my time as a Teaching Assistant was invaluable to my professional development.

Lastly,I am incredibly thankful to my husband, mother, and father for their unwavering support throughout my journey. Their belief in me, through the highs and lows, has been a constant source of strength and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Natural Language Processing (NLP) is an area within artificial intelligence (AI) and linguistics that centers on how computers interact with human language. The purpose of NLP is to allow machines to comprehend, analyze, and produce human language meaningfully and practically. One important task in NLP is Question Generation (QG), which involves automatically creating questions based on a given text.

In traditional single-hop question generation, a question is formulated using information from a single sentence or a specific part of the text. However, multi-hop question generation is more complex. It requires gathering and connecting facts from multiple parts of the text (or even different documents) to generate a meaningful question. This means the system needs to 'hop' between different pieces of information, understand the relationships between them, and formulate a question that requires deeper reasoning and synthesis of content. For example, a multi-hop question might ask for a conclusion based on details scattered across different sections of a document, making the task much more challenging for both the machine and the user.

Question generation can significantly advance several NLP domains, including question-answering systems. Using current data to automatically generate a variety of high-quality questions contributes to the development of training datasets that are more significant and resilient (Du and Cardie, 2017). Additionally, it can be integrated with question-answering models as two tasks, enhancing question-answering reasoning abilities

There are several real-world uses for multi-hop question generation, including:

- `Search Engine Query Suggestions`: Multi-hop question generation can be used to suggest follow-up queries based on a user's initial search, leading to a deeper exploration of a topic (Zamani et al., 2020).

- `Educational Assessment`:It can be used to formulate thorough reading comprehension questions that call for pupils to combine knowledge from many sources (Lindberg et al., 2013; Heilman and Smith, 2010; Yao et al., 2018).

- `Chat-bots`: In chatbots, multi-hop question generation is useful because it starts discussions and gives users detailed information from a variety of sources. This feature increases the usefulness of chatbots in posing intelligent queries (Yao et al., 2018).

  Early studies on question generation primarily focused on input contexts with a narrow scope, often limited to a single sentence or short paragraph (Zhao et al., 2018; Zhou et al., 2017). These studies also produced questions that required only basic reasoning to answer. In this thesis, we introduce a new model designed for multi-hop question generation. Our model incorporates GAT (Graph Attention Networks) combined with BART (Bidirectional and Auto-Regressive Transformers) to enhance question-generation capabilities. Notably, this is the first work to consider this specific integration of GAT and BART, which merges the generative abilities of BART with the graph-structured data-handling qualities of GAT.

1.2 Contributions

This thesis advances the area of multi-hop question generation through the following key contributions:

1. **Integration of GAT and BART Models**: This thesis is the first to explore the integration of the GAT (Graph Attention Network) model with the BART (Bidi-

rectional and Auto-Regressive Transformers) model. This approach leverages the generative capabilities of BART alongside the graph-structured data processing of GAT, resulting in an innovative framework for generating complex, multi-hop questions. This novel method offers a unique contribution by being the first to apply this specific combination for multi-hop question generation.

2. **Enhanced Multi-hop Question Generation**: By applying the merged GAT and BART model to the HotpotQA dataset (Yang et al., 2018), this research demonstrates significant improvements in generating coherent and contextually relevant multi-hop questions. This contribution addresses the challenge of synthesizing questions that need reasoning over different parts of data.

3. **Graph-based Contextual Understanding**: The use of GAT enhances the model's power to understand and utilize relationships between different entities within the input context by applying attention mechanisms to graph-structured data (Fei et al., 2022). GAT assigns varying levels of importance to different nodes (entities) and edges (relationships), allowing the model to focus on the most relevant connections. This graph-based approach leads to a deeper and more nuanced contextual understanding, which is vital for generating high-quality, coherent multi-hop questions.

4. **Benchmarking and Evaluation**: Comprehensive experiments and evaluations are conducted using the HotpotQA dataset to benchmark the performance of GNET-QG (Fei et al., 2022; Su et al., 2020). The results indicate a notable increase in the quality and relevancy of the generated questions compared to existing models, providing a valuable contribution to the field.

5. **Open-source Implementation**: The implementation of the integrated GAT and BART model is made available as an open-source resource. This contribution encourages further research and development in multi-hop question generation, facilitating the advancement of more sophisticated models and techniques.

GNET-QG presents an important improvement in the field of multi-hop question generation, providing a robust framework and valuable insights for future research.

1.3 Thesis Overview

The structure of this thesis is as follows:

- **Chapter 2** covers the foundational concepts and background details necessary to understand our proposed model's different technologies and methodologies.
- **Chapter 3** reviews the existing literature and prior research on question generation.
- **Chapter 4** describes GNET-QG in detail, including the integration of GAT (Graph Attention Network) and BART (Bidirectional and Auto-Regressive Transformers). This chapter covers the model architecture, design choices, and the underlying algorithms. It also explains how the model used GAT's strengths for handling graph-structured data and BART for generating high-quality, multi-hop questions.
- **Chapter 5** includes information on the dataset used in this study, the automated evaluation metrics applied, the details of the model implementation, and the presentation of the experimental results.
- **Chapter 6** outlines this thesis's conclusions and coming research directions for advancing multi-hop question generation tasks.

Chapter 2

Background

2.1 Introduction

This chapter focuses on the background required to understand the earlier research on question generation and the methodology introduced. We begin by exploring artificial neural networks, like recurrent neural networks, sequence-to-sequence learning, and attention processes. We then explain key concepts in natural language processing, such as entity extraction from text, tokenization, and word embeddings.

Next, we explore the cutting-edge developments in deep learning architectures including the GAT and BART models. Then we cover the definition of techniques that were used in this work to generate high-quality questions including transformer architecture, gated attention mechanism, and bidirectional autoregressive transformers. We examine the role of them in producing coherent and relevant texts.

In conclusion, we clarify important features of deep neural network training and inference phases. Additionally, we cover the application of the GAT and BART models for generating texts.

2.2 Artificial Neural Networks

ANNs (Artificial Neural Networks) are computational models inspired by the human brain (Bishop, 2006). They consist of connected nodes called neurons, organized into layers. Neurons process information data, apply mathematical operations, and produce output. ANNs learn from data by adjusting connection strengths (weights) between neurons, typically through a process called backpropagation. They are widely used in tasks like pattern recognition, classification, and prediction.

2.2.1 Recurrent Neural Networks

RNNs (Recurrent Neural Networks) are a type of neural network architecture designed to handle sequential data. Sequential data refers to data where the order of the elements is important, such as time series data or natural language text. Unlike classic neural networks that process individual data points independently, RNNs can capture dependencies and patterns across sequences.

RNNs excel in tasks involving sequence prediction, making them valuable for applications like language translation, speech recognition, and handwriting recognition. The core concepts of the RNNs are:

- **Memory:** RNNs possess an internal memory, unlike standard feedforward networks. This memory allows them to keep information about previous inputs, crucial for understanding sequences in which the meaning of one element frequently depends on the meaning of the one that came before it (Rumelhart and McClelland, 1987; Schmidt, 2019) .
- **Structure:** Like standard neural networks, RNNs consist of layers that process information. However, RNNs have a unique feedback mechanism, where connections loop back, allowing the hidden state from one time step to influence the processing of subsequent time steps. This recurrence is what enables RNNs to

retain information over time, creating a memory effect (Schmidt, 2019).

- **Training:** RNNs are trained using an adjusted version of backpropagation known as BPTT (backpropagation through time) (Keller, 2017). This algorithm helps the network learn how to effectively utilize its internal memory for sequence tasks (Schmidt, 2019).

2.2.2 Sequence-to-Sequence Learning

Seq2Seq (Sequence-to-Sequence Learning) is a deep learning technique focused on mapping sequences from one domain to another domain (Sutskever et al., 2014b). This makes it particularly valuable in tasks like machine translation, where you translate sentences from one language (e.g., French) to another (e.g., English) (Team, 2024). Here is a breakdown of the core idea behind Seq2Seq:

- **Encoder-Decoder Architecture:** Seq2Seq models typically depend on an encoder-decoder architecture (Sutskever et al., 2014a). The encoder processes the input sequence, capturing its meaning and context. This information is then compressed into a vector representation. The decoder, using this encoded representation, generates the result sequence one element at a time.
- **RNNs as Building Blocks:** RNNs are often the building blocks of Seq2Seq models because of their ability to effectively handle sequential data (Sutskever et al., 2014a). LSTMs (long short-term memory) (Hochreiter and Schmidhuber, 1997) and GRUs (gated recurrent units) (Cho et al., 2014) are popular RNN variants used in Seq2Seq models, designed to address the vanishing gradient problem that can hinder RNNs in learning long-term dependencies (Team, 2024).

2.2.3 Attention Mechanisms in Neural Networks

Attention mechanisms were introduced to address a traditional sequence-to-sequence model's limitations, particularly in handling long-range dependencies in data. The

seminal work by Bahdanau et al. (2014) presented the attention mechanism in machine translation. Their approach allowed the model to weigh different parts of the input sequence differently when producing each word in the output sequence.

Attention mechanisms in artificial neural networks draw inspiration from how humans selectively focus on specific details while processing complex information. In neural networks, attention allows the model to concentrate on relevant elements within the input data, leading to more accurate outputs (Vaswani et al., 2023). Attention assigns weights to different parts of the input data. These weights indicate the relative importance of each part for the current task. The network then focuses on the parts with higher weights, resulting in a more nuanced understanding of the input (Vaswani et al., 2023).

The evolution of attention mechanisms led to the development of the Transformer model by Vaswani et al. (2023). Transformers depend on self-attention mechanisms to global dependencies between input and output. The self-attention mechanism calculates a weighted average of all input elements for each element (Dixit, 2023), Allowing the model to capture relationships regardless of their distance in the sequence. This architecture outperformed previous models in translation tasks and has since become the basis for many cutting-edge models in NLP. Attention mechanism components are:

- **Query, Key, and Value (QKV):** In self-attention, each input element is transformed into three vectors: Query (Q), Key (K), and Value (V). The attention score is calculated as the dot product of all Keys with the Query. A softmax operation is applied to get the attention weights, which are then used to calculate a weighted sum of the values.
- **Multi-Head Attention:** This technique handles multiple self-attention operations simultaneously, each with various learned projections of Q, K, and V. The outputs are concatenated and linearly transformed to make the final output, al-

lowing the model to grab different aspects of the relationships in the data.

2.3 Natural Language Processing Concepts

We will begin with tokenization, a preprocessing step that divides text into smaller units, often words or subwords, facilitating language understanding for computational models. Next, we explore Named-Entity Recognition (NER), a task focused on identifying and categorizing key information, such as people, places, and dates, in text. Lastly, we delve into word embeddings, which convert words into dense vector representations, capturing semantic and syntactic relationships between them. Together, these concepts lay the groundwork for understanding and developing more sophisticated NLP models and methodologies.

2.3.1 Tokenization

Tokenization is a fundamental preprocessing step in NLP that involves breaking down text into smaller paces called tokens. These tokens can be words, subwords, or characters, relying on the task's specific requirements and the model's design. Tokenization is vital because it transforms raw text, which is unstructured and complex, into a structured format that machine learning models can easily process.

This step is essential for understanding the syntactic and semantic nuances of the text, facilitating more accurate and contextually relevant question generation.

Here there is an example of a tokenizer, consider the sentence: "Tokenization is the process of breaking text into tokens." (Baghaee, 2018) Tokenization might split this into individual words: ["Tokenization", "is", "the", "process", "of", "breaking", "text", "into", "token"]. On the other hand, subword tokenization, could break words into meaningful parts, handling out-of-vocabulary words more effectively: ["Token", "ization", "is", "the", "process", "of", "break", "ing", "text", "into", "token"].

To address these challenges, rule-based tokenizers are available through open-source libraries like NLTK (Natural Language Toolkit)[1] and spaCy[2], making them widely accessible for natural language processing and question generation research. These libraries offer robust tokenization tools that can handle a variety of languages and text formats.

Tokenization is particularly important in NLP because models like transformers require fixed-size inputs. Tokenizing text and subsequently padding or truncating sequences assures that all input data fits the model's expected input dimensions. This standardized input format is critical for the efficiency and accuracy of training and inference processes in NLP models.

2.3.2 Named-Entity Recognition

Named Entity Recognition (NER) is a subtask of natural language processing (NLP) that focuses on identifying and classifying named entities within text. These entities can belong to various categories, such as:

- **People:** Names of individuals (e.g., Barack Obama, Marie Curie)
- **Organizations:** Companies, institutions, government agencies (e.g., Apple Inc., Harvard University, NASA)
- **Locations:** Cities, countries, geographical features (e.g., New York, France, Mount Everest)
- **Dates and Times:** Specific points in time (e.g., July 4th, 2024, 3:12 PM)
- **Monetary Values:** Amounts of money (e.g., $100, 50)
- **Other Categories** Depending on the specific task, NER can also identify entities like percentages, vehicles, or creative works (books, movies).

[1] https://www.nltk.org/
[2] https://spacy.io/

NER has numerous applications in different domains. In information extraction tasks, NER can be used to identify relevant entities from unstructured text and enable the extraction of structured information for analysis. In question generation, NER can help identify key entities that can serve as subjects or objects in generated questions. Moreover, in sentiment analysis, NER can assist in identifying entities that play a significant role in expressing opinions or sentiments.

2.3.3 Word Embeddings

Word embedding is a technique in NLP where words or phrases are mapped to vectors of real numbers. This allows words with related meanings to have comparable representations, facilitating various NLP tasks such as question generation, machine translation, sentiment analysis, and information retrieval. The concept of word embedding has evolved significantly over time, starting from simple techniques like one-hot encoding to more advanced methods like Word2Vec and GloVe.

- **One-Hot Encoding**

  One-hot encoding is the most basic form of representing words in a numerical format. In this method, a one-hot vector is a one × N matrix (vector) employed to uniquely identify each word in a lexicon (Arnaud et al., 2021). This vector consists of 0s in all positions except for a single 1, which is used to represent the specific word. One-hot encoding is important because it prevents machine learning algorithms from assuming that certain numbers have higher importance. For example, while the number '8' is greater than '1', this difference in numerical value does not indicate more significance. Similarly, in the context of words, the term 'laughter' is not inherently more significant than 'laugh' just because it might appear 'larger' or more complex (Wikipedia contributors, 2023). Both 'laughter' and 'laugh' would have different representations in a one-hot encoding, as each word is treated as a unique token. However, the complexity or size

of the word does not impact the vector's length—each word still corresponds to a single position in the vector. The size of a one-hot vector depends on the total vocabulary size, not the word's length or complexity.

– **Word2vec**

Word2Vec draws inspiration from earlier work in deep learning, such as the paper from Bengio et al. (2009). While this paper mostly discusses deep representations for image recognition, it introduces the concept of learning distributed representations. These representations find the complex relationships between different features or entities in a high-dimensional space. This idea is a key baseline for the future progress in the field.

One of the pivotal papers in the field of word embedding is Mikolov et al. (2013). This paper introduces the Word2Vec model, which efficiently learns distributed representations of words from large text corpora. Word2Vec learns to predict words based on their context using either the continuous bag-of-words (CBOW) or skip-gram architectures. By training neural networks on large amounts of text data, Word2Vec embeds words in a consistent vector space, ensuring that words with similar meanings possess comparable vector representations. This approach revolutionized natural language processing tasks by enabling algorithms to capture semantic relationships between words and improve performance on tasks like language translation, sentiment analysis, and information retrieval.

– **Gloves**

Gloves have long served as a practical tool for protecting our hands. From keeping them warm in harsh winters to shielding them from abrasions and chemicals, gloves come in various materials and designs to suit our needs. But did you know the word "glove" can also refer to something entirely different in artificial intelligence?

Enter GloVe (global vectors), a model for distributed word representation. Just

like a well-fitting glove contours to your hand, GloVe maps words into a meaningful space where their relationships are captured. Words with similar meanings are positioned close together in this space, just as fingers on a hand are naturally grouped. This is achieved through unsupervised learning, a technique where the model learns from vast amounts of text data. By analyzing how often words co-occur with each other, GloVe creates a system where the distance between words reflects their semantic similarity (Pennington et al., 2014).

Designed as an open-source task at Stanford [3] in 2014 by Pennington et al. (2014), GloVe leverages the global structure of language, akin to a glove encompassing the entire hand, while also considering the local context in which words appear, similar to how fingers interact with specific objects (Mikolov et al., 2013).

The word "glove" embodies the concept of fitting things together. While a physical glove protects our hands, the GloVe model helps computers understand the nuances of human language. Both represent fascinating applications of form and function, each playing a vital role in their respective domains.

– **BERT Embedding:**

BERT (bidirectional encoder representations from transformers), introduced by Devlin et al. (2019a), represents a notable leap in NLP embeddings. Unlike traditional models like Word2Vec or GloVe that produce static word embeddings, BERT generates dynamic, context-sensitive embeddings. It utilizes a Transformer architecture, particularly its encoder stack, to capture bidirectional context, meaning it considers both left and right context in all layers. This allows BERT to analyze a word based on the entire sentence—taking into account the words that come before (left) and after (right)—so that it can adjust the meaning of the word depending on its surrounding context. This allows BERT to generate

[3] https://online.stanford.edu/

more nuanced and accurate word representations, significantly improving performance in various NLP tasks such as question answering, sentiment analysis, and named entity recognition (Devlin et al., 2019a).

Embeddings by BERT are created by training on large corpora using masked language modeling and next-sentence prediction tasks. In masked language modeling, some tokens in the input are randomly masked, and the model is trained to predict these masked tokens based on the context provided by the surrounding tokens. This training approach enables BERT to learn deep bidirectional representations, making it highly effective at capturing the complexities of natural language. Moreover, the next sentence prediction task helps BERT understand the relationship between sentences, enhancing its ability to handle tasks that require an understanding of sentence order and coherence (Devlin et al., 2019a).

The impact of BERT embeddings on NLP has been profound. They have set new benchmarks across numerous tasks and have become a cornerstone for many advanced NLP models and applications. By providing rich, context-aware word representations, BERT has paved the way for even more sophisticated language models, driving forward the capabilities of artificial intelligence in understanding and processing human language (Wolf et al., 2020).

Researchers have made significant strides in word embedding techniques. They are trying to make machines that can better understand and process human language.

2.4 Automated Evaluation Metrics

We begin by discussing Precision, Recall, and F-measure, which are fundamental metrics used to evaluate classification models and information retrieval systems. These metrics quantify the correctness of model outputs by balancing the trade-off between precision (accuracy) and recall (completeness). Next, we explore BLEU (Bilingual Evaluation Understudy), a popular metric for machine translation, which

measures the similarity between generated and reference texts based on overlapping n-grams. We then cover ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation), primarily used for text summarization tasks, focusing on capturing the longest common subsequences between the generated and reference summaries. Finally, we examine METEOR (Metric for Evaluation of Translation with Explicit ORdering), a metric designed to address some of BLEU's limitations by considering synonymy and stemming.

2.4.1 Precision, Recall, and F-measure

Precision measures the accuracy of positive predictions made by a model. It is defined as the ratio of correctly predicted positive instances to the total number of instances predicted as positive:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.1}$$

where:

- **True Positives (TP)**: The number of correct positive predictions.
- **False Positives (FP)**: The number of incorrect positive predictions.

Recall measures the model's ability to identify all relevant instances. It is defined as the ratio of correctly predicted positive instances to the total number of actual positive instances:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.2}$$

where:

- **False Negatives (FN)**: The number of actual positives that the model failed to identify.

F-measure (or F1-score) is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.3}$$

2.4.2 BLEU

BLEU (Bilingual Evaluation Understudy) is a metric to assess the quality of machine-generated translations. Initially introduced by Papineni et al. (2002), BLEU has become a standard in natural language processing and machine translation. It evaluates how closely a generated translation matches one or more reference translations.

The BLEU metric functions by comparing the output of an automated translation system with one or more reference translations and assigning a score based on their similarity. Scores range from zero to one, with a score of one indicating perfect similarity.

BLEU is based on the concept of precision, which measures how many of the words or phrases in the generated translation match those in the reference translations. BLEU considers n-grams, which are sequences of n consecutive words, typically up to four words long. BLEU-1 measures precision based on unigrams (single words), BLEU-2 measures precision based on bigrams (pairs of words), BLEU-3 measures precision based on trigrams (sequences of three words), and BLEU-4 measures precision based on four-grams (sequences of four words).

Each n-gram precision score is then combined using a weighted average to calculate the overall BLEU score. The weights for each n-gram precision score are typically uniform, although some variations of BLEU may use different weighting schemes.

Although BLEU is commonly used to evaluate machine translation systems, it is not without its limitations. For example, BLEU does not account for fluency, gram-

maticality, or semantic accuracy in translations. Additionally, BLEU may penalize translations that contain synonyms or paraphrases of the reference translations, even if they convey the same meaning effectively.

Despite its limitations, BLEU remains a valuable tool for researchers and developers working in the field of machine translation, providing a standardized way to measure and compare the quality of translation systems.

2.4.3 ROUGE-L

ROUGE-L is a metric used in NLP, specifically for evaluating text summarization and machine translation tasks. It belongs to the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) family of metrics (Lin, 2004), widely used to assess the quality of automatic summaries by comparing them to human-written references.

ROUGE-L focuses on the Longest Common Subsequence (LCS) (Lin and Och, 2004) between the generated and provided reference summaries. LCS considers the most extended sequence of words that appear in the same order in both summaries. For example, ”the quick brown fox” would be a longer LCS than ”brown fox the quick” even though they contain the same words.

ROUGE-L calculates precision, recall, and F-measure based on the LCS. Precision measures the proportion of words in the generated summary that are also in the reference summary. Recall measures the proportion of words in the reference summary that are also in the generated summary. F-measure is the harmonic mean of precision and recall, giving a single score that balances both.

Like other ROUGE metrics, ROUGE-L accounts for different summary lengths by normalizing the scores. ROUGE-L scores range between zero and one, with elevated scores indicating more substantial alignment between the generated and reference summaries.

ROUGE-L and other ROUGE metrics provide a quantitative measure of summary quality. However, it is important to note that ROUGE-L, like other automatic evaluation metrics, might not perfectly capture semantic similarity or fluency in summaries.

Example of ROUGE-L

Let's take an example where we have a generated summary G and a reference summary R:

$$G = \text{"The quick brown fox jumps over a lazy dog"}$$

$$R = \text{"A brown fox jumps over a lazy dog"}$$

To compute the ROUGE-L score, we first look for the LCS between G and R. In this case, the LCS is:

$$\text{LCS} = \text{"brown fox jumps over a lazy dog"}$$

Now, we can compute precision, recall, and F-measure using the LCS:

$$\text{Precision} = \frac{\text{length of LCS}}{\text{length of } G} = \frac{7}{9}$$

$$\text{Recall} = \frac{\text{length of LCS}}{\text{length of } R} = \frac{7}{8}$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \frac{7}{9} \times \frac{7}{8}}{\frac{7}{9} + \frac{7}{8}}$$

$$\text{F-measure} \approx 0.823$$

Therefore, the ROUGE-L score for this example is approximately 0.823.

2.4.4 METEOR

METEOR (Metric for Evaluation of Translation with Explicit ORdering) is a tool designed to assess the quality of machine-translation output. It was introduced by Lavie and Agarwal (2007) and aims to evaluate how effectively a machine translation system translates text.

How METEOR Works?

1. **Precision and Recall**
   - **Precision** is used to describe the percentage of pertinent items (accurate translations) that a system finds.
   - **Recall** It assesses how much relevant information the system discovers.
   - **METEOR** combines these two measures using the harmonic mean, but it gives more weight to recall. This is important because it ensures that the metric emphasizes the correct identification of all relevant items in the translation, which is crucial for assessing translation quality.
2. **Advanced Matching Features**
   - **Stemming**: METEOR reduces words to their root forms, allowing it to recognize various forms of similar words (e.g., “running” and “ran” would both match with “run”).
   - **Synonymy Matching**: This involves identifying words with similar meanings, which helps in evaluating translations that use synonyms rather than

exact word matches. For instance, “happy” and “joyful” would be considered a match.

These features allow METEOR to provide a more comprehensive evaluation of translation quality by considering not just exact word matches but also variations in word forms and meanings.

Why Use METEOR?

To solve some of BLEU’s limitations, METEOR was created, another widely used machine translation evaluation metric. BLEU focuses on the entire corpus and primarily considers exact word matches, which can sometimes lead to discrepancies with human judgment of translation grade. METEOR, conversely, aims to achieve a stronger correlation with human judgment, particularly at the sentence level. This makes METEOR a more nuanced and robust tool for evaluating translation quality, as it considers both the meaning and structure of the translated text.

2.5 Paragraph Selection

2.5.1 Definition

Paragraph selection is a process in NLP where specific paragraphs are identified and chosen from a larger text or document based on their relevance to a given task or query. This is specifically important in tasks such as summarization, question answering, and information retrieval, where it is necessary to prioritize the most pertinent sections of a document rather than analyzing the entire text.

2.5.2 Steps in Paragraph Selection

1. **Input Text**: The process starts with a large text or multiple documents that contain several paragraphs.

2. **Representation**: Each paragraph is represented in a way that captures its semantic content. This often involves converting the text into numerical vectors using various NLP models.
3. **Relevance Scoring**: Each paragraph is scored based on its relevance to a particular query or task. This is typically done using similarity measures, relevance models, or machine learning algorithms.
4. **Selection**: The paragraphs with the highest relevance scores are selected for further processing.

2.5.3 Example Use Cases

- **Question Answering**: Selecting paragraphs likely to include the answer to a user's question.
- **Summarization**: Identifying key paragraphs that encapsulate the main points of a document.
- **Information Retrieval**: Finding the most pertinent paragraphs in a document that matches a user's search query.

2.5.4 Using BERT for Paragraph Selection

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019b)is a robust NLP model that can be used for paragraph selection due to its ability to understand context and semantics effectively.

This means that in the system or application being described, BERT models are employed for the task of paragraph selection. BERT is typically used in this context:

1. **Encoding Paragraphs**: BERT takes each paragraph and converts it into a contextualized embedding, which is a numerical representation that captures the meaning of the paragraph in context.

2. **Relevance Scoring**: BERT then compares these embeddings against a query or task-specific criteria to generate a relevance score for each paragraph.
3. **Selecting Paragraphs**: Based on these scores, the most relevant paragraphs are selected.

By using BERT, the paragraph selection process becomes more accurate because BERT's contextual understanding helps in identifying paragraphs that are truly relevant, even when the language is complex or nuanced.

2.6 Graph Attention Network

A GAT (Graph Attention Network) constitutes a neural network architecture tailored for tasks centered on graph-structured data. They are particularly effective for problems where the relationships between data points (nodes) are as important as the data points themselves. This architecture, was introduced by Veličković et al. (2018). GAT blends attention mechanisms into GNNs (Graph Neural Networks) to weigh the importance of neighboring nodes.

2.6.1 Graph-Structured Data

A graph consists of nodes (vertices) and edges (links between nodes). Graph-structured data is prevalent in NLP, such as social networks, molecular chemistry, and recommendation systems (Veličković et al., 2018). In these contexts, the connections (edges) between entities (nodes) carry significant information.

2.6.2 Attention Mechanism

The attention mechanism is a component originally popularized by the Transformer architecture in NLP. It allows the model to focus on different parts of the input sequence dynamically, by assigning different weights to different parts. When applied

to graphs, the attention mechanism helps the model to assign varying levels of importance to neighboring nodes during the aggregation process.

2.7 Architecture of GAT

2.7.1 Input Features

Each vertex in the graph is linked to a feature vector. Feature vector is a numerical representation of the attributes of a node. Each feature vector is essentially an array of numbers, where each number corresponds to a specific characteristic or feature of the node. For example, in a social network, features could include age, interests, or number of friends. The input features are crucial as they represent the initial state of each node.

2.7.2 Attention Layers

GAT uses multiple attention layers where each layer performs the following steps:

- **Attention Coefficients Calculation:** For each node in a graph, attention coefficients are computed with respect to its neighboring nodes to determine the importance of each neighbor's features.

  Mathematically, for nodes i and j connected by an edge, the attention coefficient α_{ij} is calculated as follows:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]))}{\sum_{k\in\mathcal{N}(i)}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k]))}$$

  where:

  * $\mathbf{h}_i$ and $\mathbf{h}_j$ are the feature vectors of nodes i and j, respectively. These vectors represent the characteristics or embeddings of the nodes.
  * $\mathbf{a}$ is a learnable weight vector used to compute the attention score.

* $\mathbf{W}$ is a weight matrix applied to the input feature vectors $\mathbf{h}_i$ and $\mathbf{h}_j$.
* $\|$ denotes concatenation of the transformed feature vectors $\mathbf{Wh}_i$ and $\mathbf{Wh}_j$.
* $\mathcal{N}(i)$ represents the set of neighboring nodes of node i.

The function LeakyReLU is an activation function defined as:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

where α is a small positive constant (e.g., 0.01), LeakyReLU allows a small, non-zero gradient when the input is negative, helping to avoid the problem of "dying ReLUs" where neurons can become inactive and stop learning.

– **Feature Aggregation:** The node features are updated by aggregating the features of the neighboring nodes, weighted by the attention coefficients. The new feature representation of a node i is given by:

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$$

Where σ is a non-linear activation function, typically ReLU. ReLU (Rectified Linear Unit) is an activation function used in neural networks. It outputs the input directly if it's positive, and zero if it's negative. Mathematically, it's defined as:

$$f(x) = \max(0, x)$$

2.7.3 Multi-Head Attention

GAT employs multi-head attention to enhance learning stability and bolster the model's capability. This involves running multiple attention mechanisms (heads) simultaneously, concatenating their outputs to form the final node representations.

2.7.4 Advantages of GAT

- **Adaptive Weighting:** GAT can learn to focus on the most relevant parts of the graph, allowing for more nuanced and context-aware feature aggregation.
- **Versatility:** They can be applied to both homogeneous and heterogeneous graphs, making them useful in various applications.
- **Parallelizability:** The operations within each attention head are highly parallelizable, making GAT efficient to train on modern hardware.
- **Scalability:** GAT can handle large graphs more efficiently than models requiring global graph information by focusing on local neighborhoods.

2.7.5 Applications

GAT has been successfully applied in various fields:

- **Social Network Analysis:** Understanding user behavior, community detection, and link prediction.
- **Recommendation Systems:** using GAT in recommendation systems by incorporating knowledge graphs and geographic information into the model. This can improve the accuracy of recommendations by considering additional factors beyond user-item interactions (Yang et al., 2020).
- **Molecular Chemistry:** GAT show great promise in molecular chemistry, particularly for identifying molecular properties in drug discovery (Reiser et al., 2022).
- **Natural Language Processing:** Enhancing tasks like Question Answering (Kacupaj et al., 2021) and machine translation (Chen et al., 2021) by capturing relationships between words and phrases.

2.8 BART Model

BART (Bidirectional and Auto-Regressive Transformers) by Lewis et al. (2019) is a transformer-based neural network architecture developed by Facebook AI. It combines the strengths of both BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-training Transformer) models, making it effective for various NLP tasks, particularly those involving text generation and sequence-to-sequence tasks.

2.8.1 Architecture

BART is built upon the standard transformer architecture and follows a denoising autoencoder approach. It is composed of two primary parts: an encoder and a decoder.

Encoder: The encoder in BART is similar to BERT, employing a bidirectional attention mechanism. This mechanism allows the encoder to process the input text in both directions, by capturing rich contextual information from both preceding and following tokens. By doing so, it comprehensively understands the context of each word within the input sequence.

Decoder: The decoder in BART operates in an autoregressive manner, akin to models like GPT (Generative Pre-trained Transformer). It generates text token by token, conditioned on both the encoded representation from the encoder and the previously generated tokens. This autoregressive process ensures that the generated text is coherent and contextually relevant, as it attends to the full context of the input during generation.

Figure 2.1, is the images from the paper by Lewis et al. (2019). The left side of the picture shows a bidirectional encoder, which is used to encode a message. The message is represented by a series of letters, A through E. The right side of the image shows an autoregressive decoder, which is used to decode a message. The decoder

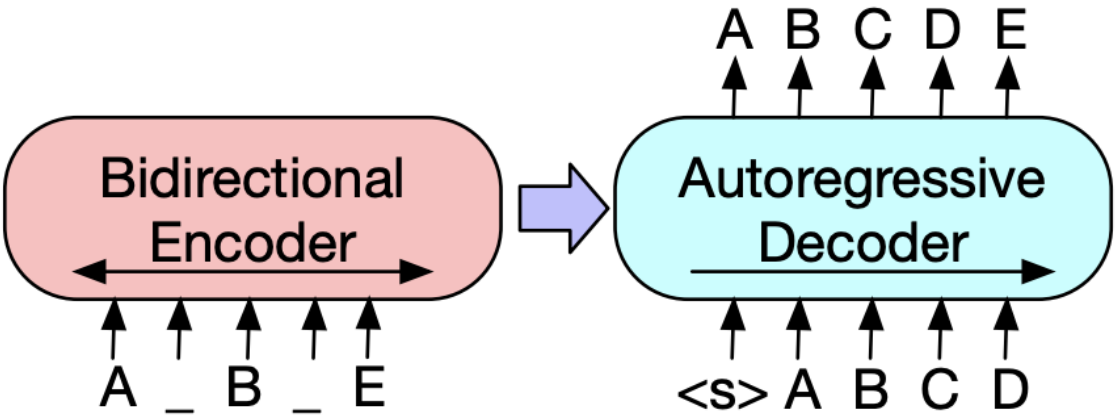

Figure 2.1: BART Architecture

starts with a special start token (<s>) and then generates a sequence of words, one at a time. In this example, the decoder generates the words "ABCDE".

2.8.2 Applications

Text Generation

BART excels in generating coherent and contextually relevant text. This makes it suitable for a range of tasks, such as:

- **Story Generation:** Creating narratives or fictional stories based on given prompts or contexts.
- **Dialogue Systems:** Generating responses in conversational agents or chatbots that are contextually appropriate.
- **Content Creation:** Automatically generating articles, blog posts, or other forms of content based on specified topics or themes.
- **Question Generation:** Crafting questions from given passages or documents is useful in educational settings or information retrieval systems.

Summarization

BART is effective in summarizing long documents into concise and informative summaries. It excels in both extractive summarization, which involves selecting the most important sentences from the original text, and abstractive summarization, which entails paraphrasing and generating new sentences to convey the main ideas. This dual capability is crucial for tasks such as:

- **News Summarization:** Creating brief summaries of news articles.
- **Document Summarization:** Generating synopses of research papers, legal documents, or reports.
- **Meeting Summarization:** Automatically summarizing meeting notes or transcripts.

Translation

BART can be used for machine translation tasks, translating text from one language to another while preserving the original meaning and context. This is valuable for:

- **Cross-Language Communication:** Enabling communication between speakers of different languages.
- **Multilingual Applications:** Supporting applications that need to operate in multiple languages simultaneously.
- **Localization:** Adapting content for different linguistic and cultural contexts.

2.8.3 Advantages

- **Versatility:** BART's combination of bidirectional encoding and autoregressive decoding makes it versatile for a wide range of NLP tasks.
- **Performance:** BART has achieved state-of-the-art results on several benchmarks, demonstrating its effectiveness in both text generation and understanding

tasks.

- **Robustness:** The denoising training objective helps BART learn robust representations, making it resilient to input noise and capable of handling various text corruptions.

2.9 T5

2.9.1 Introduction

T5, or the Text-To-Text Transfer Transformer, is a revolutionary model developed by researchers at Google Research. It was introduced by Raffel et al. (2023). T5 stands out in the field of NLP due to its unique approach of framing all tasks as text-to-text problems. This means that any NLP task, whether it involves translation, summarization, or question answering, is formulated as taking a piece of text as input and producing a piece of text as output.

2.9.2 Architecture

T5 is built upon the Transformer architecture, which is important in many state-of-the-art NLP models. The Transformer model utilizes self-attention mechanisms, enabling it to catch long-range text dependencies more effectively than traditional recurrent neural networks (RNNs). T5 extends this architecture with an encoder-decoder structure:

- **Encoder**: Processes input text and transforms it into a sequence of uninterrupted representations.
- **Decoder**: Takes these representations and generates the output text.

2.9.3 Unified Framework

A major innovation of T5 is its unified framework. By transforming every NLP task into a text-to-text format, T5 simplifies transfer learning across diverse tasks. For instance, a translation task is structured as ”translate English to French: [input text],” while a summarization task might be ”summarize: [input text].”

2.9.4 Variants

T5 is available in different sizes, from T5-Small to T5-XXL, each progressively increasing in the number of parameters. This scalability empowers users to select a model that balances computational efficiency with performance, based on their specific needs and resources.

2.9.5 Applications

The versatility of T5 makes it suitable for a wide range of NLP tasks, including translation, summarization, question answering, and text generation.

2.9.6 Performance

T5 has demonstrated impressive performance across multiple benchmarks, frequently achieving state-of-the-art results. Its effectiveness scales with larger model sizes and extensive pre-training data, showcasing the potency of the text-to-text framework.

2.10 Summary

Chapter 2 provided a comprehensive overview of foundational concepts in neural networks and natural language processing, including artificial neural networks, attention mechanisms, and automated evaluation metrics. Key technologies such as Graph

Attention Networks (GAT), BART, and T5 were explored, highlighting their architectures, advantages, and applications. This background lays the groundwork for the review of related work in the next chapter.

Chapter 3

Related Work

3.1 Introduction

In recent years, multi-hop question generation has advanced significantly, driven by the need for AI systems that can handle complex queries. This section reviews the key methods and techniques relevant to our research, tracing the evolution of question generation and highlighting the current state-of-the-art.

We start by discussing fundamental question generation methods, including syntax-based approaches, semantic-based methods, template-based techniques, and end-to-end sequence-to-sequence models. These methods have evolved to improve how questions are generated from input text.

Next, we focus on multi-hop question generation, the core of our study. We address the challenges of generating questions that require reasoning across multiple pieces of information, such as maintaining coherence, ensuring accuracy, and capturing logical relationships.

Graph-based approaches, particularly Graph Attention Networks (GATs), are examined for their ability to handle complex information structures in multi-hop question generation. We explore their architecture and effectiveness in this context.

We also review recent models, datasets, and evaluation metrics specifically designed for multi-hop question generation, analyzing how they build on earlier techniques and noting their strengths and limitations.

This review identifies trends, challenges, and gaps in current knowledge, providing a foundation for our research and highlighting the novel aspects of our approach to multi-hop question generation. It sets the stage for the subsequent chapters, where we will introduce and demonstrate our methodology.

3.2 Question generation

Over recent years, research into question generation (QG) has evolved substantially, propelled by its practical applications in education, question-answering systems, and information retrieval. This unit provides an overview of pivotal methodologies and advancements in QG techniques, structured around categories such as syntax-based, semantic-based, template-based, and end-to-end sequence-to-sequence learning models.

3.2.1 Syntax-Based Approaches

Syntax-based methods initially showed promise for short sentences but often struggled with grammatical correctness (Heilman and Smith, 2010; Yao, 2010; Wyse and Piwek, 2009). These approaches relied on syntactic structures to generate questions, highlighting their limitations in handling complex sentence structures effectively.

3.2.2 Semantic-Based Approaches

Semantic-based techniques employed semantic role labeling (Lindberg et al., 2013) and other target identification methods to enhance question relevance and accuracy. These approaches aimed to extract deeper meaning from the text to formulate contextually appropriate questions.

3.2.3 Template-Based Approaches

Template-based strategies (Chen and Aist, 2009) streamlined question generation by using predefined templates that could be applied across specific domains. While efficient in constrained environments, they required human intervention to create and maintain robust templates (Yao, 2010).

3.2.4 End-to-End Sequence-to-Sequence

The emergence of end-to-end sequence-to-sequence learning models (Yuan et al., 2017) marked a shift towards data-driven approaches that bypassed manual rule crafting. These models leveraged neural networks to map input sentences to corresponding questions directly, demonstrating proficiency in handling diverse linguistic structures and contexts.

3.3 Multi-hop Question Generation

Multi-hop Question Generation is a process that entails gathering related data dispersed over several documents and formulating questions that necessitate reasoning in multiple stages (Hwang et al., 2024).

3.3.1 Challenges in generating multi-hop questions

Multi-hop question generation represents a significant advancement over single-hop QC. Unlike single-hop QG, which typically relies on information contained within a single context, Multi-hop question generation demands a more sophisticated approach to information synthesis and question formulation. The first major challenge in Multi-hop question generation hinges on the model's need to comprehend and integrate information across multiple context documents. This requires not only the ability to extract relevant details from individual sources but also to understand the

intricate relationships between these pieces of information. Models must develop a holistic understanding of the subject matter, drawing connections that may not be explicitly stated in any single document. This demands advanced semantic processing capabilities and the ability to maintain and manage a much larger contextual scope than is typically required in single-hop QG. The second significant challenge in Multi-hop question generation is the requirement to establish complex linkages between entities mentioned across different contexts. This aspect is important in ensuring the depth and complexity of the generated questions, often referred to as ”deep questions” in the literature. Models must not only recognize and track entities across multiple documents but also understand their evolving roles and relationships in varying contexts. This necessitates a level of logical reasoning and relationship modeling that goes beyond simple entity recognition, often requiring multiple steps of inference to create meaningful connections. These challenges collectively contribute to the complexity of Multi-hop question generation, making it a task that demands more advanced natural language understanding and generation capabilities. The ability to perform complex reasoning tasks becomes paramount, as models must generate questions that truly require multiple hops to answer rather than superficially complex questions that can be resolved with a single piece of information. As such, Multi-hop question generation serves as a frontier in question generation research, pushing the boundaries of what is possible in automated comprehension and question formulation (Pan et al., 2020; Sachan et al., 2020; Fei et al., 2022).

3.4 Graph-based Approaches in Question Generation

While traditional question generation techniques concentrate on formulating questions from single passages or KG (knowledge graph) triples, multi-hop QG tackles the challenge of generating questions that necessitate reasoning across multiple hops within the KG. This paves the way for more intricate and insightful questions. Graph-

based approaches are particularly well-suited for this task. For instance, Su et al. (2020) proposes the Multi-hop Encoding Fusion Network (MulQG), which utilizes GCNs (Graph Convolutional Networks) on an answer-aware dynamic entity graph. This approach tackles the complexities of multi-hop QG by encoding context across multiple hops and leveraging GCNs to aggregate evidence related to the desired question. This exemplifies the effectiveness of graph-based methods in facilitating multi-hop reasoning for QG tasks.

3.4.1 Overview of graph based methods

Graph based methods have emerged as a vital tool for modeling relationships within data, making them appropriate for tasks that involve reasoning and information extraction. In the domain of question generation, several works have explored leveraging graphs to capture the semantic structure of text and generate high-quality questions. For instance, Wang (2022) employs GCNs to model the relationships between words and sentences within a document, enabling the generation of questions that target specific information.

3.4.2 Focus on Graph Attention Networks (GATs)

In addition to the mentioned works, He et al. (2023) utilizes gated GATs to focus on informative features of the document graph. This enables the generation of more relevant and diverse questions by attending to crucial relationships within the text. These studies highlight the effectiveness of graph-based approaches in capturing intricate relationships within text data and demonstrate their potential for question-generation tasks.

3.5 Multi-hop Question Generation Techniques

Multi-hop QG has gained important attention recently due to its potential to create complex, reasoning-based questions. As the field has evolved, researchers have proposed several approaches to address the unique challenges of multi-hop QG. These approaches can be broadly categorized based on their primary techniques, such as those leveraging knowledge graphs, those focusing on advanced encoding and fusion methods, and those adapting existing architectures for the multi-hop task. In the following sections, we will explore some of the most prominent and recent approaches in detail:

- **Multi-hop Encoding Fusion Network for Question Generation (MulQG)**:Su et al. (2020) presented MulQG, a model that employs a Graph Convolutional Network to conduct multi-hop context encoding and integrates this encoding through an Encoder Reasoning Gate. This approach tackles the problem of multi-hop reasoning across paragraphs without depending on sentence-level details (Su et al., 2020).
- **Multi-hop Question Generation without Supporting Fact Information**: Multi-hop question generation involves creating questions that link together various facts across different contexts, making it more complex than single-hop generation. In this approach, a transformer model is used without relying on sentence-level supporting facts. It incorporates techniques from single-hop question generation, such as placeholder tokens and a copy mechanism, to enhance the process (Emerson and Chali, 2023).
- **CQG**: Current state-of-the-art models can generate questions that match the given answers but often struggle with creating complex, multi-hop questions. As a result, they may produce simpler questions that don't require deep reasoning. To address this, we introduce the CQG framework, which effectively generates multi-hop questions by incorporating key entities from reasoning chains.

Additionally, we use a novel Transformer-based decoder to ensure these essential entities are included in the questions, enhancing both their complexity and quality (Fei et al., 2022).

3.6 Summary

This chapter provided an overview of recent advancements in multi-hop question generation, examining key models and their contributions to the field. We reviewed the evolution from traditional methods to modern transformer-based approaches and discussed integration strategies such as Graph Convolutional Networks (GCN). Despite significant progress, existing methods still face challenges, setting the stage for exploring new solutions.

Chapter 4

Multi-Hop Question Generation

4.1 Introduction

In natural language processing (NLP), generating coherent and contextually relevant questions is a critical aspect of intelligent systems. Multi-hop question generation entails crafting questions that necessitate reasoning across various pieces of information. This thesis is motivated by several key factors:

- **Complex Reasoning:**

  Multi-hop question generation tackles a challenging area of AI—reasoning across multiple pieces of information to ask relevant questions. This work will aid in developing more sophisticated artificial intelligence systems capable of critical thinking and complex problem-solving.

- **Unlocking Deeper Understanding:**

  Machines cannot ask complex questions, so this research will help to ask complex questions. This has the potential to revolutionize fields like information retrieval and question-answering. Effective multi-hop question generation can greatly improve information retrieval systems. By generating detailed questions, these systems can more accurately extract relevant information from large datasets. This has practical applications in different fields, including academia, law, and healthcare, where the precision and relevance of retrieved information are paramount.

- **Impact on Human-Technology Interaction:**

  This work has the potential to impact how humans interact with information and technology, making a real difference in the world. Improved AI models can provide more meaningful and contextually relevant interactions, improving user experience and accessibility. By addressing these areas, this thesis provides developments in multi-hop question generation and also has an impact on the way humans interact with information and technology.

4.2 Graph Creation

In this section, we explain how to create a graph. As illustrated in Figure 4.1, entities are first extracted and represented as nodes. Next, specific rules are applied to determine the edges between them. Further details on graph creation will be provided in the following section.

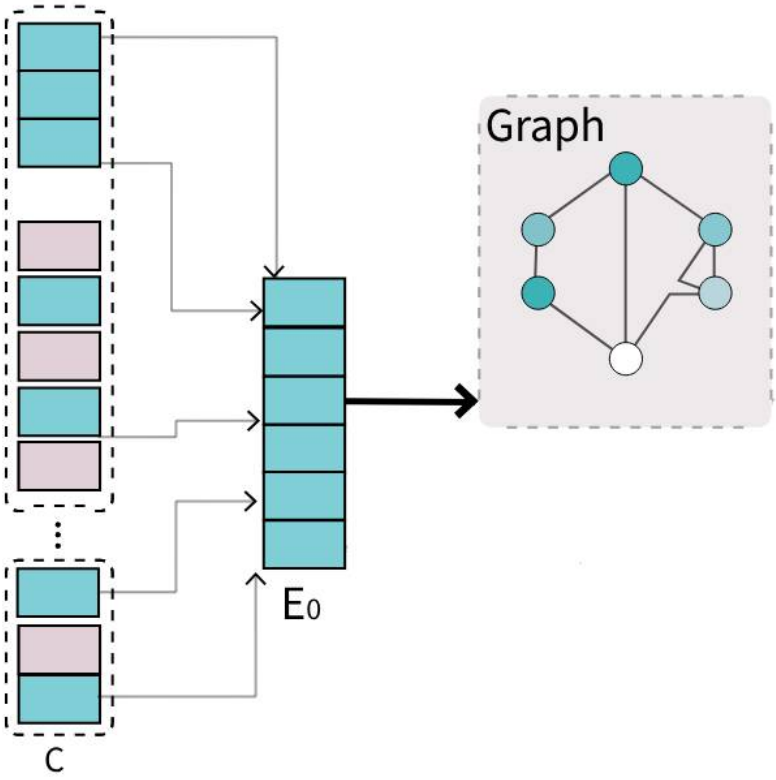

Figure 4.1: Graph creation

4.2.1 Graph Nodes

To create a graph for use in Graph Attention Networks (GAT), the first step involves extracting entities and setting them as nodes in the graph. We used a BERT-based Named Entity Recognition (NER) model to extract entities from the dataset, which is designed to identify named entities within a given context. Named entities can include names of people, organizations, locations, and other specific terms that are important for the analysis.

Once the named entities are identified by the NER model, they are represented as a set of entities $e = \{e_0, e_1, \ldots, e_n\}$. Each entity e_i in this set is associated with a node in the graph.

Example of Extracted Entities

The following sample shows how entities are extracted from a given context. Below is a sample data structure containing titles and contexts, followed by the extracted entities.

Listing 4.1 provides a sample data structure in the HotpotQA dataset (Yang et al., 2018). Each sample within this structure contains two paragraphs, from which we aim to extract entities.

Listing 4.1: Sample data structure

```
{
 'p1': {
        'title': 'Oberoi family',
        'context': 'The Oberoi family is an Indian family
           that
        is famous for its involvement in hotels, namely
           through The Oberoi Group.'
    },
 'p2': {
        'title': 'The Oberoi Group' ,
        'context': 'The Oberoi Group is a hotel company with
           its head office in Delhi. Founded in 1934, the
           company owns and/or operates 30+ luxury hotels and
            two river cruise ships in six countries,
           primarily under its Oberoi Hotels & Resorts and
           Trident Hotels brands.'
    }
}
```

The following entities were extracted from the above test by BERT as nodes:

1. Oberoi
2. Indian
3. Oberoi Group
4. Oberoi Group
5. Delhi
6. 1934
7. Oberoi Hotels & Resorts
8. Trident Hotels

4.2.2 Graph Edges

The graph is constructed by creating edges between nodes based on the following criteria similar to Su et al. (2020):

- Nodes are linked if they occur within the same sentence.
- If the title of a paragraph is also an entity in the context of the same paragraph, it has a relation with other entities in that context.
- Entities appearing in different paragraphs but being the same entity will also have an edge between them.

Let M be a matrix where $M_{j,i} = 1$ indicates a link from node j to node i. The matrix M can be represented as:

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

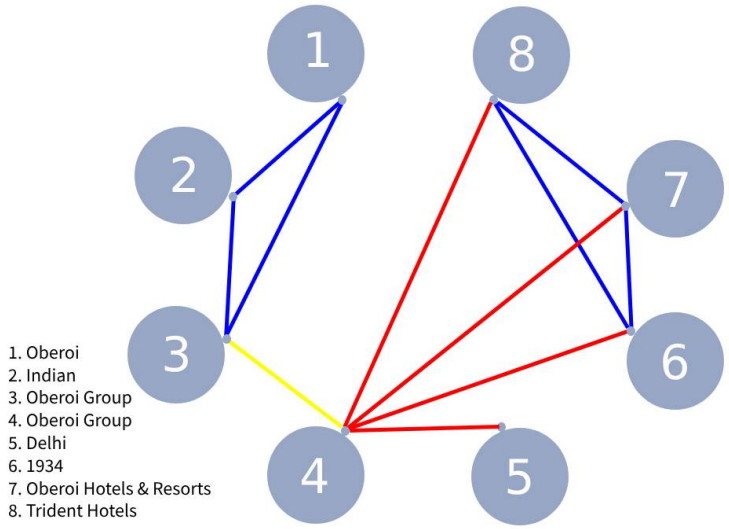

Figure 4.2: Visual illustration of the graph for Listing 4.1 graph

Figure 4.2 illustrates a graphical representation of the graph based on the extracted entities as nodes and the defined edges in this graph:

- The blue edges represent the connection between two entities that are in the same sentence. For example, ”Oberoi” is connected to ”Indian”, ”Indian” is connected to ”Oberoi Group”, and ”Oberoi Group” is connected to ”Oberoi”.
- The yellow edges represent the connection between two entities that are in different paragraphs but similar to each other. For example, ”Oberoi Group” in paragraph 1 is connected to ”Oberoi Group” in paragraph 2.
- The red edges represent connections where an entity similars the title of its paragraph. In such cases, the entity is linked to all other entities in the text. For example, ”Oberoi Group” is connected to ”Delhi”, ”1934”, ”Oberoi Hotels & Resorts”, and ”Trident Hotels”.

4.2.3 Entity Mask

The entity mask is a binary vector showing whether each entity exists in the answer from the dataset. If an entity is present in the answer, the related entry in the mask is

set to one; otherwise, it is set to zero, similar to Qiu et al. (2019). This mask helps in focusing the attention of the model on relevant entities during training and inference.

Example

Consider the following example:

- Answer: ”The Oberoi Group is headquartered in Delhi.”
- Entities: ”Oberoi”, ”Indian”, ”Oberoi Group”, ”Delhi”, ”1934”, ”Oberoi Hotels & Resorts”, ”Trident Hotels”

For this example, the entity mask would be:

$$\text{Entity Mask} = [1, 0, 1, 1, 0, 0, 0]$$

Here, ”Oberoi Group” and ”Delhi” exist in the answer, so their corresponding entries in the mask are set to 1. The rest of the entities do not appear in the answer, so their entries are set to 0.

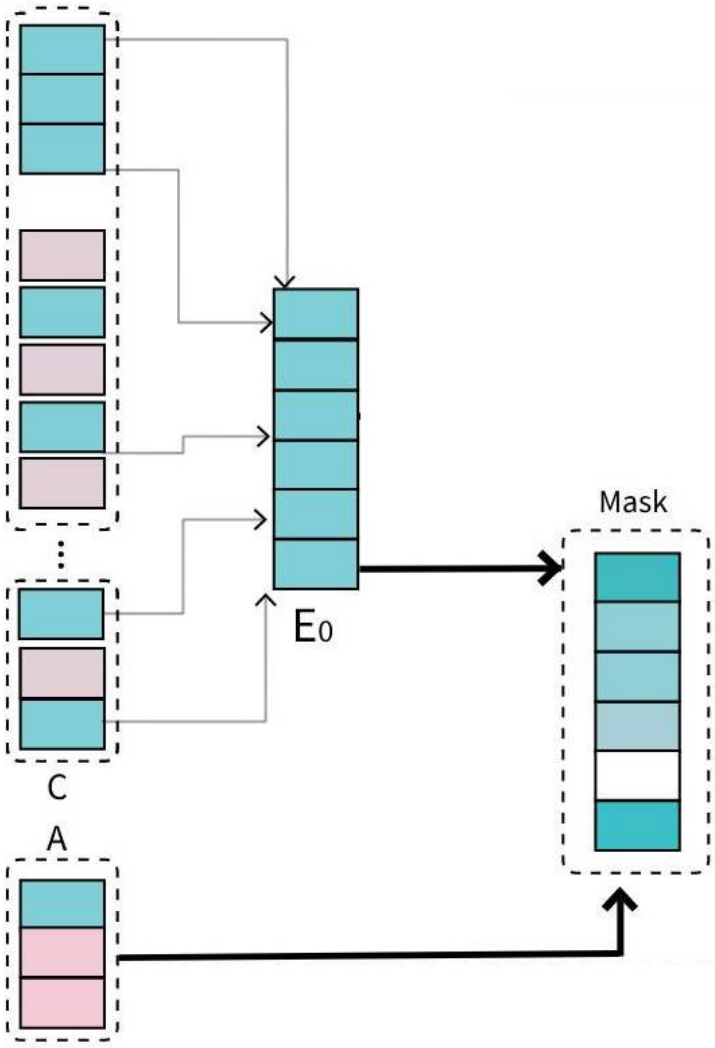

Figure 4.3: Mask of entities

In Figure 4.3, the answer is used to mask the entities.

4.3 Encoder

4.3.1 GAT

The GAT (Graph Attention Network) is used to process and update node features in a graph by considering the importance of each node's neighbors. Here's an explanation of its components and operations:

1. **Input**:
   - **Graph (Nodes and Edges)**: The input includes of a set of nodes $e = \{e_0, e_1, \ldots, e_n\}$ and edges $M_{i,j}$ representing the relationships between nodes.
   - **Entity Masks**: Masks are applied to focus the attention mechanism on relevant parts of the graph.
2. **Attention Mechanism**:
   - **Attention Coefficients**: For each pair of connected nodes (e_i, e_j), the attention coefficient α_{ij} is calculated to determine the importance of node e_j to node e_i:

$$\alpha_{ij} = \text{softmax}_j\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j\right]\right)\right)$$

     where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the feature vectors of nodes e_i and e_j, $\mathbf{W}$ is a weight matrix, and $\mathbf{a}$ is an attention vector.
3. **Updating Node Features**:
   - **Feature Update**: The new feature vector for each node e_i is calculated as a weighted sum of its neighbors' transformed features:

$$\mathbf{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}\mathbf{W}\mathbf{h}_j\right)$$

     where $\mathcal{N}(i)$ denotes the set of neighbors of node i and σ is a non-linear activation function.
4. **Node Representation**:
   - **New Node Features**: After passing the node representations through the GAT, The GAT has a new set of node features.

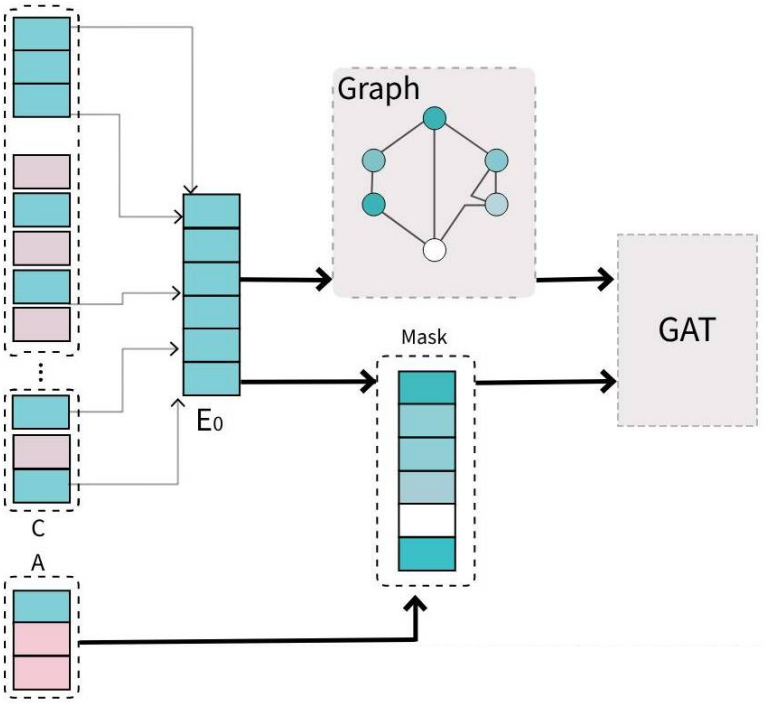

Figure 4.4: Visualization of GAT Input and GAT Network

Figure 4.4 illustrates how the graph and the masked entities are sent as input to the GAT, which generates a new representation of the entities.

4.3.2 Flattening

This step involves taking the updated node features, which are likely multi-dimensional (e.g., a matrix or tensor), and converting them into a single-dimensional vector. Flattening simplifies the representation by collapsing all dimensions into one. For example, if your node features are represented in a matrix of size (80, 1024), flattening them results in a vector of size 80*1024 = 81920. This step prepares the data for the next operation, which is linear transformation, by ensuring all the data is in a uniform, one-dimensional format.

Given a matrix $X \in \mathbb{R}^{m \times n}$, where m is the number of rows and n is the number of columns (in your example, $m = 80$ and $n = 1024$):

$$\text{Flatten}(X) = \text{Vec}(X) \in \mathbb{R}^{m \cdot n}$$

In your specific case:

$$\text{Flatten}(X) = \text{Vec}(X) \in \mathbb{R}^{80 \cdot 1024} = \mathbb{R}^{81920}$$

4.3.3 Linear Transformation

After flattening, a linear layer is applied to reduce the size of the features. The linear layer is a fully connected (dense) layer that maps the large flattened vector (81920 elements) into a more compact representation, reducing it to 80 elements. This is important because it helps to reduce the complexity of the model, making the learning process more efficient and reducing the risk of overfitting by condensing the information in a meaningful way.

4.3.4 Sigmoid Activation function

Finally, a Sigmoid activation function is applied to the transformed node features, producing a probability score for each node. The Sigmoid function outputs values between zero and one, which represent the likelihood of each node being relevant to the given question. Nodes with a probability score above 0.5 are considered relevant and selected as part of the subset of entities.

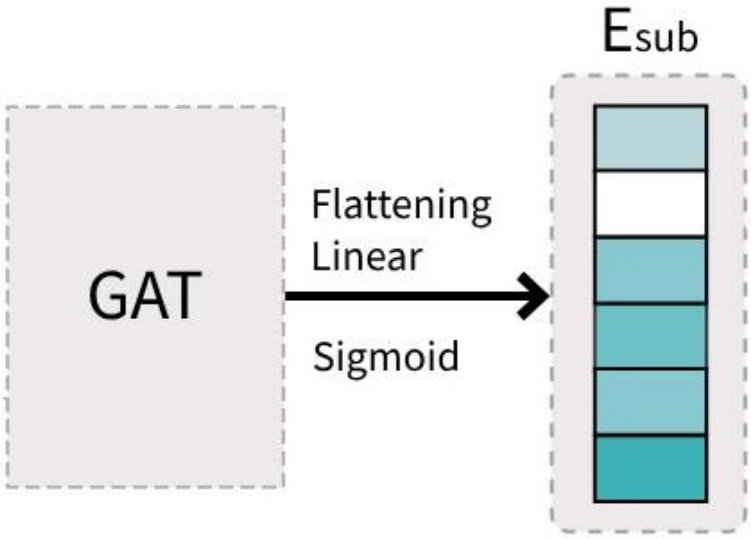

Figure 4.5: Extract a subset of entities

As shown in Figure 4.5, after applying flattening, linear transformation, and the Sigmoid function, entities with a score greater than 0.5 are chosen and grouped into a subset of relevant entities.

4.3.5 Enriched Input Context

To effectively utilize BART, we first construct an ”enriched input context.” This context is derived by selecting pertinent nodes from the graph structure and concatenating them with the original context. This process ensures that the input context contains the most relevant entities and information, thus enriching it with important details that can guide the question-generation process.

This is the “Enriched Input Context” of the above example:

– **Context:** The Oberoi family is an Indian family known for its involvement in the hotel industry, particularly through The Oberoi Group. **Question entity:** Oberoi family.

4.3.6 Tokenize Enriched Input Context

In this section, we tokenize the Enriched Input Context using the BART tokenizer. BART was selected for its:

- **Subword Tokenization:** BART utilizes a subword tokenization technique, often based on Byte-Pair Encoding (BPE). This method, initially described by Gage (1994), effectively handles both rare and common words by breaking them down into smaller subword units.
- **Compatibility with BART Architecture:** BART's tokenizer is designed to seamlessly integrate with the model's Transformer architecture. This ensures optimal performance and compatibility throughout the tokenization, encoding, decoding, and attention mechanisms of the model.

Example: If the enriched input context is "What is the capital of France?", the BART tokenizer might produce tokens such as [101, 2054, 2003, 1996, 3007, 1997, 2605, 102]. Each number corresponds to a token ID. Tokens 101 and 102 specifically represent special tokens such as [CLS] (used for sequence start) and [SEP] (used for sequence separation), which the tokenizer employs to delineate the beginning and end of input sequences.

4.4 Encoder

Following the processing by GAT, the output (enriched input context) after tokenization is fed into the BART model's encoder component. BART's encoder improves the representation of its input (enriched input context as explained in 4.3.2) through the transformer architecture, which includes multi-head self-attention mechanisms and feed-forward neural networks. This process led us to contextualized embeddings containing the semantic meaning and relationships within the input context.

Figure 4.6 illustrates the encoder component of GNET-QG with a BART backbone. Initially, entities (nodes) from the contexts (C) are identified and labeled as E0. Next, a graph is created, and the answer (A) is used to compute a mask. This mask, together with the entity graph (as described in the Nodes and Edges section), is passed to the GAT. After applying flattening, linear transformation, and sigmoid activation, we derive Esub, which is then concatenated with the contexts and answer to form the enriched input context. This enriched input is then ready to be fed into the BART encoder.

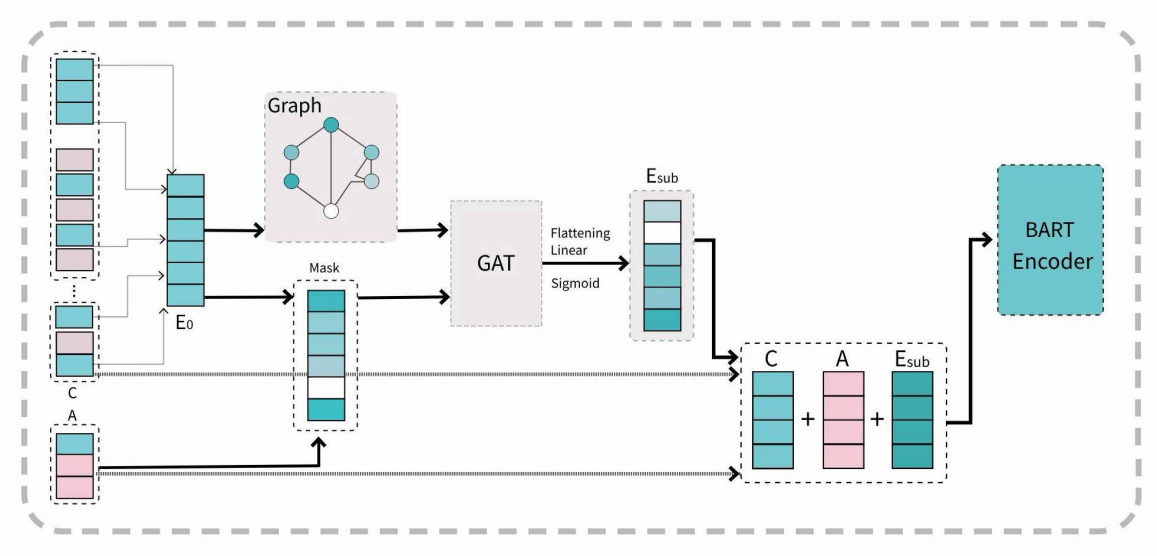

Figure 4.6: Encoder part of GNET-QG with BART backbone

4.5 Decoder

The BART decoder's role is to generate the desired output based on the defined task. The plan is to generate questions with the latent encoded embeddings that the encoder part provides. The decoder operates autoregressively, predicting each token in the output sequence based on the encoder's output embeddings. The decoder applies masked self-attention to guarantee that each token only regards previously generated tokens and itself, regarding the autoregressive nature of sequence generation.

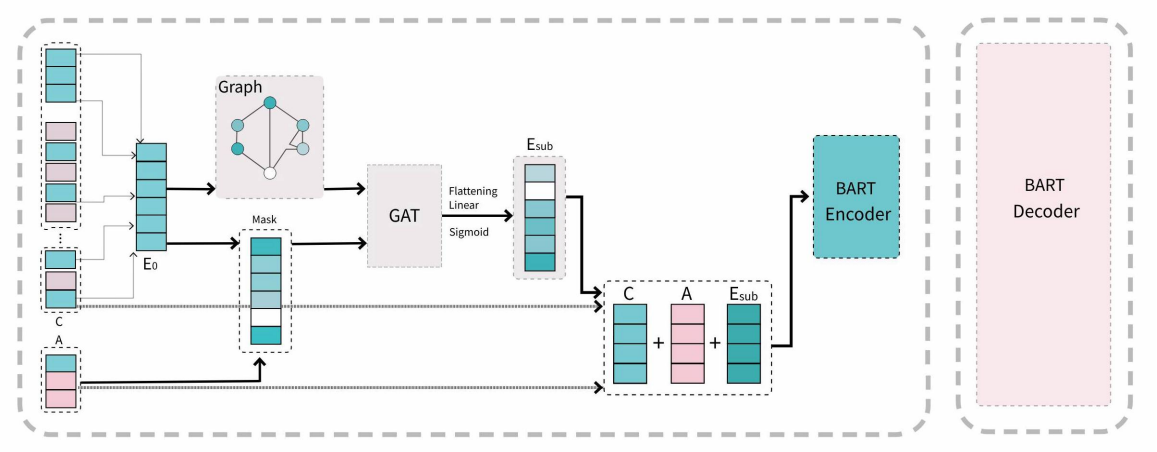

Figure 4.7: Full architecture of GNET-QG with BART backbone

Figure 4.7 represents the GNET-QG model, which includes both the encoder and decoder parts. It shows how the encoder's output is sent to the decoder.

4.6 Using T5 as A Backbone

In addition to BART, as a backbone for GNET-QG, the T5 encoder-decoder model was also considered for the multi-hop question generation task. T5 is recognized for its flexibility and effectiveness across various text-generation approaches because of its unified text-to-text framework. Although T5 was not the primary focus of this research, its inclusion provided a valuable benchmark. The introduced architecture of GNET-QG has shown promise in substantially enhancing the efficacy of large language models in question generation tasks. By grasping the strengths of graph attention networks (GATs) to capture relational information and large language model's (LLM) generative capabilities, our approach offers a robust solution for multi-hop question generation. This technique not only emphasizes the versatility of our architecture but also suggests its more general applicability in improving the performance of other LLMs, including T5.

4.7 Details of Implementation

Our proposed model, GNET-QG (Graph Network for Multi-Hop Question Generation), was developed in Python using PyTorch by Paszke et al. (2019). PyTorch is one of the popular open-source libraries in deep learning. This provides a flexible and efficient platform for building and training artificial neural networks. Based on PyTorch features and its workflow, researchers have flexibility in implementing their algorithms, architectures, and/or strategies. Therefore, it is a suitable framework for research and development purposes in machine learning. PyTorch supports GPU acceleration, which is critical for handling the computational demands of deep learning models that need to be trained.

For the task of multi-hop question generation, We implemented GNET-QG by the fusion of the GAT from this GitHub repository[4] with an encoder-decoder model as the main backbone. We connected the non-trained version of the mentioned GAT architecture with the pre-trained versions of both BART and T5 models to generate the multi-hop questions. Our candidate pre-trained models for the proposed architecture were the p208p2002/bart-squad-qg-hl [5]version of BART model and the mrm8488/t5-base-finetuned-question-generation-ap [6] version of T5 model. These models were selected due to their proven efficacy in question generation tasks and their availability for fine-tuning on domain-specific datasets.

4.7.1 BART

The p208p2002/bart-squad-qg-hl model is a fine-tuned version of BART designed for QG in the SQuAD dataset. BART combines the properties of BERT and GPT, making it suitable for both understanding and generating text. This particular fine-tuned version leverages BART's robust architecture to effectively generate coherent

[4]hhttps://github.com/HLTCHKUST/MulQG
[5]https://github.com/p208p2002/Transformer-QG-on-SQuAD
[6]https://github.com/patil-suraj/question_generation

and contextually relevant questions from given passages.

4.7.2 T5

The mrm8488/t5-base-finetuned-question-generation-ap model is a fine-tuned version of T5 and is explicitly fine-tuned for question-generation tasks. T5 converts all NLP tasks into a text-to-text format, making it highly versatile. This fine-tuned version is adept at generating questions by understanding the context of the input text and producing appropriate question outputs.

4.7.3 Fine-tuning

In this stage, we initialized the GAT with random weights and connected the pre-trained candidate model, as discussed earlier. Both combinations of GAT with BART and GAT with T5 were fine-tuned together via the designed architecture in Figure 4.2. meaning the weights of the GAT and the candidate model were updated simultaneously. This joint fine-tuning was performed using the HotpotQA dataset. The fine-tuned model generated the questions, fulfilling multi-hop reasoning tasks over distributed information among the input data. The implementation includes setting the training procedure and model optimization methods to obtain the complex multi-hop questions.

By choosing these models and fine-tuning them on our dataset, we were able to leverage their strengths and adapt them to the specific needs of our multi-hop QG task. This method enhanced the quality of the generated questions and provided a robust framework for comparing the performance of different large language models.

The backbone of our approach involved leveraging encoder-decoder models like BART and their tokenizers to extract meaningful embeddings from the input. The candidate model should demonstrate effective handling of sequence-to-sequence tasks. We

mainly considered BART as the backbone of the investigations performed in this literature. BART has a vocabulary of 50,266 tokens and embeds each token into a vector space of 768 dimensions.

To optimize the training process, the AdamW optimizer (Loshchilov and Hutter, 2019) was selected, which is well-suited for transformer-based models like BART. AdamW extends the original Adam optimizer by decoupling weight decay from the optimization steps, often leading to improved generalization and training stability.A learning rate of 1×10^{-5} was carefully selected to balance convergence with the optimal solution and training speed.

The computational power for training and inference was provided by a V100 GPU, equipped with 32GB of RAM, sourced from Compute Canada. This setup provided sufficient resources to handle the computational demands of fine-tuning tasks, which is particularly beneficial for processing large datasets and complex model architectures efficiently.

Chapter 5

Experiments And Results

In this chapter, we present the experimental setup and the results obtained from evaluating our proposed multi-hop question generation strategy. We begin by providing an overview of the HotpotQA dataset, which serves as the foundation for our experiments.

Following the dataset description, we detail the evaluation metrics used to measure the performance of our strategy. These metrics are important for understanding how effectively our approach generates multi-hop questions compared to existing methods. We then present the results of our experiments.

5.1 Dataset

In this thesis, our proposed multi-hop question generation strategy is evaluated, utilizing the HotpotQA dataset (Yang et al., 2018). This comprehensive dataset was created to train question-answering (QA) systems for complex reasoning and explanation generation. It contains 113k question-answer pairs based on Wikipedia. Each sample includes the following components:

- Ten paragraphs, which include titles.
- An answer.
- The type of question (bridge or comparison).
- A question.

- A unique ID number.

The HotpotQA dataset mainly features two types of question-answer pairs for multi-hop reasoning tasks: bridge and comparison. These types of questions are formed from information in multiple paragraphs, making the dataset ideal for training and evaluating systems capable of sophisticated question generation.

To generate questions related to information from different paragraphs and improve their quality, we employed a paragraph selection technique using BERT. This involves selecting the two most relevant paragraphs out of the ten provided, as a result reducing noise and focusing the system's attention on the most relevant information. By doing this, we aimed to improve the system's accuracy and coherence in generating questions through effective multi-hop reasoning.

In Figures 5.1 and 5.2, two instances of bridges and comparisons from the HotpotQA dataset are presented.

Context1:
The Oberoi family is an Indian family that is famous for its involvement in hotels, namely through The Oberoi Group.

Context2:
The Oberoi Group is a hotel company with its head office in Delhi. Founded in 1934, the company owns and/or operates 30+ luxury hotels and two river cruise ships in six countries, primarily under its Oberoi Hotels & Resorts and Trident Hotels brands.

Answer: Delhi

Question: Oberoi family is part of a hotel company that has a head office in what city?

Figure 5.1: Bridge-style question derived from the HotpotQA dataset (Yang et al., 2018).

In Figure 5.1, the question is classified as a Bridge-type question because its generation requires connecting information from multiple contexts described below:

- **Context 1** provides information about the Oberoi family and their involvement in the hotel industry through the Oberoi Group.
- **Context 2** provides detailed information about the Oberoi Group, including the fact that its head office is in Delhi.

To generate the question ”Oberoi family is part of a hotel company that has a head office in what city?” one must bridge the information from both contexts. The reasoning here involves:

- Identifying the relationship between the Oberoi family and The Oberoi Group from Context 1.
- Connecting the relationship to the detail about the head office location from Context 2.

The generated question asks for the head office location of the hotel company associated with the Oberoi family. This requires understanding the linkage between the Oberoi family (from Context 1) and The Oberoi Group’s head office in Delhi (from Context 2). Thus, the reasoning process spans across both contexts and is a classic example of the Bridge-type question.

Bridge-type questions are characterized by the necessity to integrate pieces of information from different parts of the provided context. In this case, the integration of the family’s involvement (Context 1) with the company’s headquarters (Context 2) exemplifies this multi-hop reasoning, where the answer can only be determined by combining the information from both contexts.

In Figure 5.2, the question is classified as a Comparison-type question because it requires comparing information from multiple contexts to generate the proposed question. As you can see here:

- **Context 1** provides information about Henry Roth, stating he was an American novelist and short story writer.

Context1:
Henry Roth (February 8, 1906 – October 13, 1995) was an American novelist and short story writer.

Context2:
Robert Erskine Childers DSC (25 June 1870 – 24 November 1922), universally known as Erskine Childers, was a British writer, whose works included the influential novel "The Riddle of the Sands", and a Fenian revolutionary who smuggled guns to Ireland in his sailing yacht "Asgard". He was executed by the authorities of the nascent Irish Free State during the Irish Civil War. He was the son of British Orientalist scholar Robert Caesar Childers, the cousin of Hugh Childers and Robert Barton, and the father of the fourth President of Ireland, Erskine Hamilton Childers.

Answer: Robert Erskine Childers DSC

Question: Which writer was from England, Henry Roth or Robert Erskine Childers ?

Figure 5.2: Comparison-style question derived from the HotpotQA dataset (Yang et al., 2018).

- **Context 2** provides information about Robert Erskine Childers, stating he was a British writer.

To generate the question "Which writer was from England, Henry Roth or Robert Erskine Childers?" one must compare the nationalities of the two writers mentioned in the contexts. The reasoning involves:

- Identifying the nationality of Henry Roth from Context 1 (American).
- Identifying the nationality of Robert Erskine Childers from Context 2 (British).

Eventually, The Comparison-type generated question asks to compare the two writers' nationalities to determine which one was from England. This requires understanding the distinctions between the provided nationalities in each context and comparing them to identify the correct answer.

Comparison-type questions are the ones that compare different facts among the ex-

isting contexts. for the above example, the comparison of the writers' nationalities (American from Context 1 and British from Context 2) illustrates this type of reasoning. The answer is determined by evaluating and contrasting the information extracted from mentioned contexts.

5.1.1 Data Preprocessing

As highlighted by the work of Su et al. (2020), it is essential to filter out all yes/no data examples from the HotpotQA dataset to attain multi-hop capability. Yes/no questions are typically simpler and require less reasoning, often involving single-hop reasoning where the answer can be found directly from a single piece of information like a part of context. By removing these simpler questions, we ensure that the dataset eventually contains multi-hop questions, allowing for a clearer evaluation of the model's capability to perform multi-step reasoning without the results being influenced by simpler yes/no questions.

After filtering our dataset consists of 79k samples, obtained from a similar approach taken in Su et al. (2020). We used their dataset which was partitioned into three parts:

- Training dataset: For adjusting the model's trainable parameters.
- Validation dataset: To tune the hyperparameters.
- Testing dataset: For evaluating the trained model after hyperparameters had been finalized.

Particularly, the training part encompassed 80% of the dataset, whereas the validation and testing parts each comprised 10% of it. Before splitting, we shuffled the dataset to have balanced parts based on the sample types and question difficulties.

Table 5.1 displays the overall number of examples in each dataset after excluding yes/no samples.

Table 5.1: The processed dataset's statistics

	Train	**Validation**	**Test**
# pairs	63,120	7,900	7,900

5.2 Evaluation Results and Discussion

To evaluate GNET-QG's effectiveness, we employed automated evaluation metrics to assess its predictions on samples from the test dataset, comparing the generated questions from the model with the reference questions from the dataset. The metrics used include BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Lavie and Agarwal, 2007), chosen for their extensive adoption in the question-generation research field. This evaluation allows us to directly compare GNET-QG's performance with previous studies on multi-hop question generation.

Table 5.2: GNET-QG vs. Existing Approaches

Model	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**	**ROUGE-L**	**METEOR**
BART	41.41	30.90	24.39	19.75	36.13	25.20
MulQG	40.08	26.58	19.61	15.11	35.35	20.24
TRANSFORMER-BASED	42.13	30.44	23.84	19.42	39.26	22.78
CQG	49.71	37.04	29.93	25.09	**41**.83	27.45
GNET-QG (BART backbone)	**49.72**	**38.95**	**32.88**	**27.93**	40.25	**49.87**
GNET-QG (T5 backbone)	42.10	31.92	26.42	22.05	34.38	42.51

In Table 5.2, we showed comparative results of GNET-QG with BART and T5 backbones against other models including BART by Lewis et al. (2019), MulQG by Su et al. (2020), TRANSFORMER-BASED by Emerson and Chali (2023) and CQG by Fei et al. (2022). The motivation behind selecting BART and MulQG models for comparison is twofold. Firstly, we included a comparison with BART since BART also was used as the backbone of the GNET-QG architecture. By comparing GNET-QG with BART backbone to singly BART in multi-hop question generation, we demonstrated the improvements and advantages introduced by our designed architecture. Furthermore, the inclusion of T5 provides an additional perspective on the

performance of different large language models (LLMs) under similar conditions. Despite not being the primary focus, T5's results offer valuable insights into how different architectures can handle multi-hop question generation via our introduced architecture.

Moreover, MulQG benefited graph construction by enhancing reasoning in multi-hop question generation, a technique that closely aligns with the methodology used in GNET-QG architecture. This similarity makes MulQG an appropriate point of comparison.

Through these comparisons, we tried to highlight the significant improvements in the performance and significance of GNET-QG in generating multi-hop questions, showcasing its advancements over the baseline models. One of the foremost motivations behind the work presented in this thesis was to enhance the quality of generated questions. As evidenced by the improved results across various evaluation metrics, GNET-QG has successfully achieved this goal. Despite the increased complexity of GNET-QG with BART backbone, which consists of 150 million parameters, it significantly outperforms other models in terms of question quality based on the evaluation results.

For instance, the MulQG model, which also focuses on multi-hop question generation using graph construction, contains approximately 57 million parameters. Although GNET-QG has a more complex architecture and can be larger based on the backbone we select for it, the notable performance improvement justifies this sophistication. The results emphasize the efficacy of GNET-QG's approach in generating high-quality questions, thereby demonstrating the advantages of the additional parameters and more complex architecture.

Additionally, we compared GNET-QG with the TRANSFORMER-BASED and CQG models, both of which are significant benchmarks in the field. Across all metrics, GNET-QG outperformed the TRANSFORMER-BASED model, and in all metrics

except ROUGE-L (where the difference is less than one percent), GNET-QG also surpassed CQG.

Chapter 6

Conclusion

6.1 Summary

We address the challenge of multi-hop question generation, which involves creating questions that need reasoning over different parts of information. We utilized the same dataset as previous works to ensure comparability and focused on enhancing the grade of generated questions through rigorous evaluation metrics. Our model leverages a GAT to capture complex relationships within the data and the BART model for its robust natural language generation capabilities. The integration of GAT and BART has significantly improved the quality and coherence of the generated questions, as evidenced by our evaluation results. This work contributes to advancing the field of automatic question generation by demonstrating effective techniques for generating high-quality multi-hop questions. At the end of this thesis, I have included an appendix with examples of predicted questions alongside their corresponding reference questions, which were randomly selected. Additionally, I have provided a comparison between the predicted questions and their reference counterparts.

6.2 Future Work

Coming work for the multi-hop QG can focus on several key areas to improve and expand upon existing methodologies:

- Evaluation Metrics: Develop and standardize evaluation metrics that better capture the nuances of multi-hop question generation. This includes measures for logical consistency, relevance, and diversity of the generated questions. For example:

  > Text: ”The theory of relativity, proposed by Albert Einstein, revolutionized our understanding of space, time, and gravity.”
  > Generated Question: ”Who proposed the theory of relativity, and what areas did it revolutionize?”

  Challenges include ensuring evaluation metrics go beyond grammaticality and overlap with reference questions to assess the model's ability in reasoning across multiple hops and maintaining logical consistency.

- Domain Adaptation: Extend multi-hop question generation to various domains and specialized fields (e.g., medicine, law, science) by fine-tuning models for optimal performance.

- Human-in-the-Loop Approaches: Incorporate human feedback during the training process to enhance the model's question generation capabilities iteratively, aiming for more relevant and higher-quality questions.

- Cross-lingual and Multilingual Capabilities: Develop models capable of generating multi-hop questions in multiple languages, facilitating use in non-English speaking regions and tasks requiring understanding across different linguistic contexts.

- Handling Ambiguity and Uncertainty: Create methods to mitigate ambiguity and uncertainty in the information used for question generation, thereby improving the precision and clarity of generated questions.

- Real-world Applications: Explore practical applications of multi-hop question generation in educational tools, virtual assistants, and other interactive AI systems. Conduct pilot studies and real-world deployments to refine and validate the technology.

Bibliography

Émilien Arnaud, Mahmoud Elbattah, Maxime Gignon, and Gilles Dequen. Nlp-Based Prediction of Medical Specialties at Hospital Admission Using Triage Notes. In *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*, pages 548–553, 2021. doi: 10.1109/ICHI52183.2021.00103.

Tina Baghaee. *Automatic Neural Question Generation Using Community-Based Question Answering Systems*. University of Lethbridge (Canada), 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*, 2014.

Yoshua Bengio et al. Learning Deep Architectures for Ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer, New York, 2006. ISBN 978-0-387-31073-2.

Linjie Chen, Jianzong Wang, Zhangcheng Huang, and Jing Xiao. An Approach for Neural Machine Translation with Graph Attention Network. In *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition*, ICCPR '20, page 472–478, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450387835. doi: 10.1145/3436369.3437429. URL https://doi.org/10.1145/3436369.3437429.

Wei Chen and Gregory Aist. Generating Questions Automatically from Informational Text. 2009. URL https://api.semanticscholar.org/CorpusID:15246592.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014. URL https://arxiv.org/abs/1406.1078.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019a.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019b. URL https://arxiv.org/abs/1810.04805.

Hina Dixit. Introduction to Transformers in Machine Learning. https://www.hinadixit.com/post/introduction-to-transformers-in-machine-learning, 2023. Accessed: 2 July 2024.

Xinya Du and Claire Cardie. Identifying Where to Focus in Reading Comprehension for Neural Question Generation. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1219. URL https://aclanthology.org/D17-1219.

John Emerson and Yllias Chali. Multi-hop Question Generation without Supporting Fact Information. *The International FLAIRS Conference Proceedings*, 36(1), May 2023. doi: 10.32473/flairs.36.133320. URL https://journals.flvc.org/FLAIRS/article/view/133320.

Zichu Fei, Qi Zhang, Tao Gui, Di Liang, Sirui Wang, Wei Wu, and Xuanjing Huang. CQG: A Simple and Effective Controlled Generation Framework for Multi-hop Question Generation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6896–6906, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.475. URL https://aclanthology.org/2022.acl-long.475.

Philip Gage. A New Algorithm for Data Compression. *C Users J.*, 12(2):23–38, feb 1994. ISSN 0898-9788.

Yunjie He, Philip John Gorinski, Ieva Staliunaite, and Pontus Stenetorp. Graph Attention with Hierarchies for Multi-hop Question Answering. *arXiv preprint arXiv:2301.11792*, 2023.

Michael Heilman and Noah A. Smith. Good Question! Statistical Ranking for Question Generation. In Ron Kaplan, Jill Burstein, Mary Harper, and Gerald Penn, editors, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California, June 2010. Association for Computational Linguistics. URL https://aclanthology.org/N10-1086.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Seonjeong Hwang, Yunsu Kim, and Gary Geunbae Lee. Explainable Multi-hop Question Generation: An End-to-End Approach without Intermediate Question Labeling. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint*

International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 6855–6866, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.599.

Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. Conversational Question Answering over Knowledge Graphs with Transformer and Graph Attention Networks, 2021.

Thomas Anderson Keller. *Comparison and Fine-grained Analysis of Sequence Encoders for Natural Language Processing*. University of California, San Diego, 2017.

Alon Lavie and Abhaya Agarwal. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In Chris Callison-Burch, Philipp Koehn, Cameron Shaw Fordyce, and Christof Monz, editors, *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/W07-0734.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. 2019. URL https://arxiv.org/abs/1910.13461.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Chin-Yew Lin and Franz Josef Och. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain, July 2004. doi: 10.3115/1218955.1219032. URL https://aclanthology.org/P04-1077.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. Generating Natural Language Questions to Support Learning On-Line. In Albert Gatt and Horacio Saggion, editors, *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-2114.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality, 2013.

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. Semantic Graphs for Generating Deep Questions. *arXiv preprint arXiv:2004.12704*, 2020.

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 on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically Fused Graph Network for Multi-hop Reasoning. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1617. URL https://aclanthology.org/P19-1617.

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, 2023. URL https://arxiv.org/abs/1910.10683.

Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph Neural Networks for Materials Science and Chemistry, 2022.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

Devendra Singh Sachan, Lingfei Wu, Mrinmaya Sachan, and William Hamilton. Stronger Transformers for Neural Multi-hop Question Generation. *arXiv preprint arXiv:2010.11374*, 2020.

Robin M. Schmidt. Recurrent Neural Networks (rnns): A Gentle Introduction and Overview, 2019.

Dan Su, Yan Xu, Wenliang Dai, Ziwei Ji, Tiezheng Yu, and Pascale Fung. Multi-hop Question Generation with Graph Convolutional Network. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4636–4647, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.416. URL https://aclanthology.org/2020.findings-emnlp.416.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence Learning with Neural Networks, 2014a.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks, 2014b. URL https://arxiv.org/abs/1409.3215.

Keras Team. A Ten-minute Introduction to Sequence-to-Sequence Learning in Keras, 2024. URL https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2023.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks, 2018.

Zimu Wang. Generating Complex Questions from Knowledge Graphs with Query Graphs. In *2022 IEEE 10th International Conference on Information, Communication and Networks (ICICN)*, pages 606–613, 2022. doi: 10.1109/ICICN56848.2022.10006514.

Wikipedia contributors. One-hot Encoding — wikipedia, The Free Encyclopedia, 2023. URL https://en.wikipedia.org/wiki/One-hot#cite_note-5. [Online; accessed 20-May-2024].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's Transformers: State-of-the-art Natural Language Processing, 2020.

Brendan Wyse and Paul Piwek. Generating Questions from Openlearn Study Units. January 2009.

Susen Yang, Yong Liu, Yonghui Xu, Chunyan Miao, Min Wu, and Juyong Zhang. Contextualized Graph Attention Network for Recommendation with Item Knowledge Graph, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.

Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. Teaching Machines to Ask Questions. pages 4546–4552, July 2018. doi: 10.24963/ijcai.2018/632.

Xuchen Yao. Question Generation with Minimal Recursion Semantics. 2010. URL https://api.semanticscholar.org/CorpusID:18796690.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. Machine Comprehension by Text-to-Text Neural Question Generation. In Phil Blunsom, Antoine Bordes, Kyunghyun Cho, Shay Cohen, Chris Dyer, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston, and Scott Yih, editors, *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2603. URL https://aclanthology.org/W17-2603.

Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. Generating Clarifying Questions for Information Retrieval. In *Proceedings of The Web Conference 2020*, WWW '20, pages 418–428, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380126. URL https://doi.org/10.1145/3366423.3380126.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1424. URL https://aclanthology.org/D18-1424.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural Question Generation from Text: A Preliminary Study, 2017. URL https://arxiv.org/abs/1704.01792.

Appendix A

Evaluation of Model Predictions

Example 1

Prediction: *the hork-bajir chronicles is the second companion book to what science fantasy series of young adult books written by katherine applegate and her husband michael grant?*
Reference questions: *what science fantasy young adult series, told in first person, has a set of companion books narrating the stories of enslaved worlds and alien species?*
Comparison: The prediction is more straightforward and directly asks about the science fantasy series related to ”The Hork-Bajir Chronicles.” The reference questions adds more descriptive elements about the narrative style and content.

Example 2

Prediction: *where is the director of the 2014 american drama romantic comedy film ”big stone gap” based?*
Reference questions: *the director of the romantic comedy ”big stone gap” is based in what new york city?*
Comparison: Both questions effectively seek the same information. The prediction is slightly clearer by specifying ”the director of the 2014 American drama romantic comedy film ’Big Stone Gap.’”

Example 3

Prediction: *2014 s/s is the debut album of a south korean boy group formed by what company?*
Reference questions: *2014 s/s is the debut album of a south korean boy group that was formed by who?*
Comparison: The prediction and reference questions are almost identical in content, with the prediction being more grammatically correct by using ”what company” instead of ”who.”

Example 4

Prediction: *what is the capacity of the arena at which the lewiston maineiacs played its home games?*
Reference questions: *the arena where the lewiston maineiacs played their home games can seat how many people?*
Comparison: Both questions are asking for the seating capacity of the arena. The reference questions is slightly clearer in asking ”can seat how many people.”

Example 5

Prediction: *what photographer has photographed annie morton and has shot advertising campaigns for marc jacobs?*
Reference questions: *who is older, annie morton or terry richardson?*
Comparison: The prediction and reference questions are asking different questions. The prediction is relevant to the context provided, while the reference questions is not related to the given context.

Example 6

Prediction: *what is the name of the fight song at the university located in lawrence, kansas?*
Reference questions: *what is the name of the fight song of the university whose main campus is in lawrence, kansas and whose branch campuses are in the kansas city metropolitan area?*
Comparison: Both questions seek the name of the fight song. The prediction is more concise, while the reference questions provides additional detail about branch campuses.

Example 7

Prediction: *the family man is a 2000 american romantic comedy-drama film directed by brett ratner, written by david diamond and which screenwriter and director, his film credits include ”evolution” (2001), and ”when in*
Reference questions: *what screenwriter with credits for ”evolution” co-wrote a film starring nicolas cage and téa leoni?*
Comparison: The prediction is incomplete, making the reference questions question clearer and more precise.

Example 8

Prediction: *”oh my god” is a song by guns n’roses released in 1999 on the soundtrack to the film ”end of days”, an american fantasy action horror thriller film directed by peter hyams and starring arnold*
Reference questions: *what year did guns n roses perform a promo for a movie starring arnold schwarzenegger as a former new york police detective?*

Comparison: The prediction is incomplete, making the reference questions question clearer and more precise.

Overall Assessment

Clarity: In some cases, the predictions are more concise and clear.
Completeness: The reference questions questions are generally more detailed and specific.