TÉCNICO LISBOA

Question Generation using Deep Neural Networks

Exploring deep learning methods to generate questions

Ângela Margarida Gonçalves Ferreira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

Prof. Bruno Emanuel Da Graça Martins

Examination Committee

Chairperson: Prof. Francisco António Chaves Saraiva de Melo

Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

Member of the Committee: Prof. David Manuel Martins de Matos

June 2019

Abstract

Automatic question generation is the task of producing questions from a given text passage, with neural approaches currently achieving state-of-the-art results. In this work, the techniques applied to this challenge were studied, from the neural architectures to the different available datasets and evaluation metrics.

Following the steps of previous related work, we developed an attention based, recurrent neural model that generates questions based on sentences. In addition, we also explored the Transformer, a novel neural network architecture whose implementation in Question Generation has not been reported to date. We experimented these models both with and without pre-trained word embeddings.

The assessment of the models was performed by automatic and human evaluation. Automatic evaluation shows that our models come very close to the state of the art, specially when using Recurrent Neural Networks. Human evaluators considered our questions correct, but not very relevant when considering the input sentences.

Keywords: Question generation, recurrent neural networks, attention, Transformer

Resumo

A geração automática de perguntas é a tarefa de produzir perguntas a partir de uma passagem de texto. Os resultados do estado da arte para esta tarefa são obtidos por modelos de redes neuronais. Neste trabalho foram estudadas as técnicas previamente aplicadas nesta tarefa, desde arquiteturas de redes neuronais, a corpora e métricas de avaliação.

Seguindo procedimentos existentes na literatura, foi desenvolvido um modelo neuronal recorrente com base em atenção que gera perguntas a partir de frases. Foi também explorado o Transformer, uma arquitetura de redes neuronais recente cuja aplicação em geração de perguntas ainda não foi reportada. Estes modelos foram aplicados com e sem recurso a Word Embeddings pré-treinados.

A avaliação dos modelos foi realizada através de avaliação automática e humana. Os resultados da avaliação automática mostram que os modelos desenvolvidos se aproximam do estado da arte, principalmente usando redes recorrentes. Os avaliadores humanos consideraram a maioria das perguntas geradas corretas mas pouco relevantes para a passagem de texto dada.

Palavras chave: Geração de perguntas, redes neuronais recorrentes, atenção, Transformer

Contents

List of Figures

List of Tables

List of Acronyms

NLP Natural Language Processing

NLG Natural Language Generation

QG Question Generation

QA Question Answering

RC Reading Comprehension

NN Neural Network

ReLU Rectifier Linear Unit

SGD Stochastic Gradient Descent

Adam Adaptive Momentum Estimation

FFN Feed Forward Network

RNN Recurrent Neural Network

BI-RNN Bidirectional Recurrent Neural Network

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

Seq2Seq Sequence-to-Sequence

SQuAD Stanford Question-Answering Dataset

MARCO Microsoft Machine Reading Comprehension

POS Part-of-Speech

NER Named Entity Recognition

1

Introduction

1.1 Motivation

Humans start asking questions almost as soon as they learn to talk. Questions can be used to request information with the intent of better understanding our surroundings and improving our knowledge. They can also serve as a way to check proficiency and test the comprehension of a given subject.

Even though asking questions seems like an elementary component of our communication as human beings, it can become a very complex topic to explore, especially if the intention is to apply it to machines. To pose a question, it is important to produce a correct natural language sequence that is in the interrogative form, which by itself is a task that is not trivial to achieve automatically. The difficulty of the task increases when we expect substantial questions that are related to a given topic, usually written in an unstructured piece of text. Whichever system is generating these questions, it needs to extract relevant information from that text and use it in the question. If we also expect our questions to be more challenging to answer, we need to go beyond the transformation of declarative to interrogative form and actually enhance the question with new words or multiple pieces of information.

Considering all those previous constraints, Question Generation (QG) becomes a very abstractive procedure, as mentioned by Yuan et al. [50], since it requires the creation of text that might not be present in the source. Table 1.1 presents a statement and three columns with questions that are related to it. Even though the statement has less than 20 words, the table shows fifteen questions related to the information conveyed in the sentence. Aside from those, there are many other questions that could be asked about the text. Even the questions in the table could also be morphed into similar ones, but with different words or phrasing. For instance, instead of asking "Who wrote The Hobbit?", one could ask "Who is the author of The Hobbit?" and expect the same answer in both; instead of asking the nationality

Table 1.1: Example of a sentence and questions related to it (generated manually).

"The Hobbit is a children's fantasy novel by English author J. R. R. Tolkien and it was published in 1937."		
What is The Hobbit?	What is a Hobbit?	When is J.?
Who wrote The Hobbit?	What is the first name of the English author Tolkien?	What is the name of the children?
When was the book published?	Who is the main character in The Hobbit?	When did the Tolkien take place?
What is the nationality of the author?	Do you like The Hobbit?	Which novel published the fantasy?
What kind of book is The Hobbit?	What is a fantasy novel?	How is it a novel?

of the author, one could ask "Where is J. R. R. Tolkien from?".

The green and the blue questions in the table are the ones that can be considered correct and relevant. More specifically, the green ones can even be answered by the text: the answer to "Who wrote The Hobbit?" is "J. R. R. Tolkien". On the other hand, the blue questions need more information in order to be answered, even though they seem related to the topic: there is nothing in the text that can tell us who the main character is. Depending on the final application of these questions, they can be considered useful or not. Finally, the red questions might seem related to the content of the text, but do not mean anything (at least given this context).

With the previous example, it is made clear that automatic QG is a challenging task. However, having machines generating questions could benefit multiple areas of interest and specific tasks. For instance, the technique could be applied to interactive virtual agents with educational or entertainment goals, such as museum guides or virtual school tutors [5, 2, 22]. Conversational agents could also improve their credibility by asking correct questions related to previous conversation topics. On the other hand, with automatic QG, Frequently Asked Questions (FAQs) documentation could be created with more ease and less human input. Finally, QG can support the task of Machine Comprehension, which is the field of research focused in giving machines the ability to answer questions about some given unstructured text or even images. By synthetically developing extensive datasets with a considerable number of Sentence-Question pairs, they could be used as training data in Machine Comprehension models [40]. Alternatively, learning to ask good questions might be able to improve the quality of question answering models [48].

After the work of Rus et al. [38], which defined the question generation task, it has been tackled by many angles and it has achieved state of the art results with Deep Learning. Many researchers have been applying the neural techniques explored in Machine Translation to obtain an interrogative sequence from text segments. This has been possible thanks to the increasingly amount of Machine Comprehension data available with Sentence-Question pairs.

In spite of the attention given to this topic, it is still very hard to appraise how good QG solutions are. The work executed in the area has not found a reliable evaluation method to check the quality and relevance of their resultant questions. In most of the literature, the results are reported with automatic metrics based on n-gram overlaps, originally created to evaluate Machine Translation. Taking Table 1.1 into consideration, specifically the question "When was the book published?", the word "novel" present in the original sentence was changed to "book": in this case the change happens without altering the meaning. However, it is a change that might be penalized when using automatic metrics. Human evaluation is sometimes used, since it captures these semantic similarities instinctively, but its subjective nature can also influence the results.

1.2 Thesis Proposal

Our main objective with this work is to develop a QG model that meets the following requirements:

1. Given a set of natural language text sequences, it returns correct natural language questions;
2. The generated questions relate to the source sequences;
3. The QG system produces results that are close to or even surpass the state of the art;
4. The questions are positively evaluated by humans.

To reach these goals, we will delve into neural networks models. We start by exploring previous similar methods, which are mostly based on Recurrent Neural Networks (RNNs). We will also try a novel

neural network architecture, the Transformer [46], which is known to achieve state of the art results in Machine Translation and that, as far as our knowledge goes, has not been applied to the QG task yet.

The resultant questions will be later inspected so that the performance of the final solutions can be judged and compared against each other. We rely on automatic and human evaluation to achieve this comparison.

1.3 Document Organization

This document is organized as follows: Chapter 2 exposes other works that relate to the present research. It starts with the description of the fundamental concepts in the neural universe, introducing the essential background needed to understand Deep Learning. Then, it presents a comparison between datasets that could be used for QG. This chapter ends by exposing the most recent work done in the field of Automatic QG, giving a special focus to neural models.

Chapter 3 presents the experiments with Neural Networks (NNs) that were implemented to generate questions. The results are discussed in Chapter 4.

Finally, Chapter 5 summarizes our findings with this project, and suggests future work that could be executed to better improve our solutions.

2

Related Work

In this chapter, we review important work that has been executed towards the automatic generation of questions and others that could support that task. Section 2.1 gives some theoretical background to better understand the Deep Learning techniques that are explored in Section 2.4, which focuses on previous Question Generation (QG) work that has been executed specifically with Neural Networks (NNs) and that inspired our own research. Section 2.2 describes multiple datasets that possess question-answer pairs that could be used to train QG models, while displaying examples taken directly from the data.

2.1 Fundamental Concepts

In this section there is an introduction to the fundamental concepts of NNs, such as composition and training, giving the necessary theory to understand the work that is later explored in this report.

2.1.1 Neural Networks

A perceptron could be seen as the smallest unit in the neural networks universe. It takes multiple inputs and produces a single binary output of either 0 or 1. Each input can have an associated weight representing its importance. Figure 2.1 consists of a simple NN with three input nodes x_i, with respective weights W_i, one hidden layer and a single output y. The perceptron in the hidden layer outputs the weighted sum of the inputs, transformed by an activation function f, which is usually a non linear function. The output sometimes considers a constant term: the perceptron's bias, also shown in the figure as b.

Given the complexity of certain problems, the decision-making process benefits from using extended networks of perceptrons. These networks possess multiple levels called layers, and each layer allows more intricate decisions than the one before. Networks have an input and an output layer, that are separated by the middle layers, typically called hidden layers. The information flows from the input layer to the output, and that is why these are sometimes called Feed Forward NNs. It can be quite tricky choosing the right amount of hidden layers and their respective number of perceptrons. Different tasks and different datasets work with different network configurations, and this choice can be based on specific job-oriented heuristics.

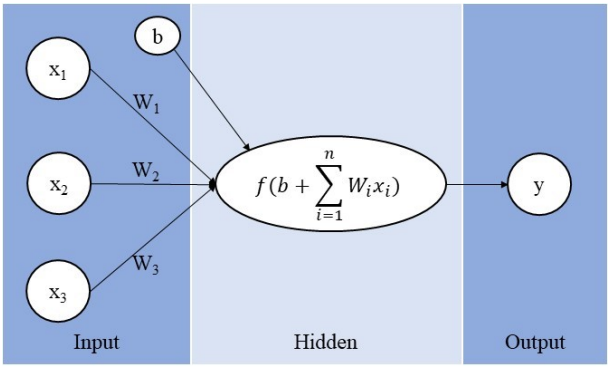

Figure 2.1: Example of a simple NN.

While connected to others, perceptrons are challenging to tune and adapt: flipping a value from 0 to 1 (and vice-versa) could lead to drastic changes in the learning process. That is why there is a transformation with a non-linear activation function, such as Sigmoid, presented in Equation 2.1, Hyperbolic Tangent in Equation 2.2 or Rectifier Linear Unit (ReLU) in Equation 2.3, among others. The Softmax activation function (Equation 2.4) is commonly used at the output layer of the network. The non-linearity is able to restrict the obtained values to a certain interval, allowing the production of softer results, making them easier to control and less susceptible to small changes, leading to more accurate results.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.2}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.3}$$

$$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_j^K e^{x_j}} \tag{2.4}$$

When training a NN, we need to use a dataset called the training set, which has the correct classification associated with the analyzed data. Then, we have the test or validation datasets, that are used to evaluate the learning performance of the network. Training a network consists in producing the optimized parameters, such as weights, that lead to a minimized loss function, thus achieving an acceptable level of a certain chosen metric, such as accuracy. The loss function $L(y', y)$ is the error between an expected output y and the one obtained y'. An example of a commonly used effective loss function is the Negative Log-Likelihood, which presents a probabilistic interpretation of the obtained error.

Loss function optimization techniques use gradients to find the parameters θ that optimize the model over the training set, namely the Stochastic Gradient Descent (SGD) algorithm [24]. It repeatedly computes the gradient to find the global minimum of the loss function, without requiring extensive and complex computations. One variation of the SGD is the mini-batch SGD, demonstrated in Equation 2.5, which uses a small sample of inputs and averages these gradients to get quicker results and a good estimate of the true gradient. When one full cycle covers the whole training set, it is called an epoch.

$$\theta = \theta - \eta \cdot \nabla f(\theta, x^{i:i+n}, y^{i:i+n}) \tag{2.5}$$

Each update is calibrated by a learning rate η: if its value is too low, convergence might take too long; if it is too high, the network might not converge at all. However, this value does not have to be fixed throughout the execution. There are many optimizations for the SGD algorithm, like adding the concept

of Momentum or Nesterov Accelerated Momentum to the calculations, allowing previous gradients to influence the current update with the use of a decaying parameter μ and the momentum vector. Other techniques consist of adaptive rate algorithms that manage the selection of the learning rate per batch, such as the Adaptive Momentum Estimation (Adam) method [19]. This algorithm computes the decaying average of both past gradients and squared gradients to adapt the values of η when updating parameters. It also includes bias correction, since there might be some instability if the values begin with 0. Even though Adam has helped the overcoming of certain issues that are known to exist in the SGD algorithm, there are other similar algorithms that are also worth mentioning, like the AMSGrad [36], which recently and achieved state-of-the-art results.

An efficient method to compute the gradients is the Backpropagation Algorithm. It transmits intermediate results of obtained errors backwards in the network. This algorithm often works extremely well, thus optimizing the learning capabilities of networks.

One of the major issues with neural models is that, while working with large amounts of data, it is likely to overfit. Regularization is a technique that can be applied to avoid, or at least attenuate, the effects of overfitting, while increasing the ability to generalize better to data previously unseen. The simplest is the Early Stopping, that is applied when the validation errors start increasing, so it stops updating weights. Other regularization method, that is known to be very effective in Natural Language Processing (NLP) tasks, is called Dropout [42]: it was developed to randomly drop to 0 the weights of a certain amount of perceptrons in the network in each training instance.

2.1.2 Recurrent Neural Networks

Deep Neural Networks is the name given to networks that have multiple hidden layers. These additional layers can generate long networks that manage to model complex non-linear relationships. There are different types of deep architectures and each of them can either be outstanding in certain tasks or fall short in others.

One Deep Learning technique that is known to be very fruitful in NLP and Natural Language Generation (NLG) is the Recurrent Neural Network (RNN) architecture. While in traditional NNs information is expected to move forward, in this paradigm information persists as a result of loops in the network. They repeatedly perform the same task over the same element of the input sequence, which will result in an output based on previous computations.

RNNs operate on sequences of ordered vectors, making them more flexible than the NNs described in Section 2.1.1. As shown in Figure 2.2, each step combines the input x_i with a state vector s_{i-1},

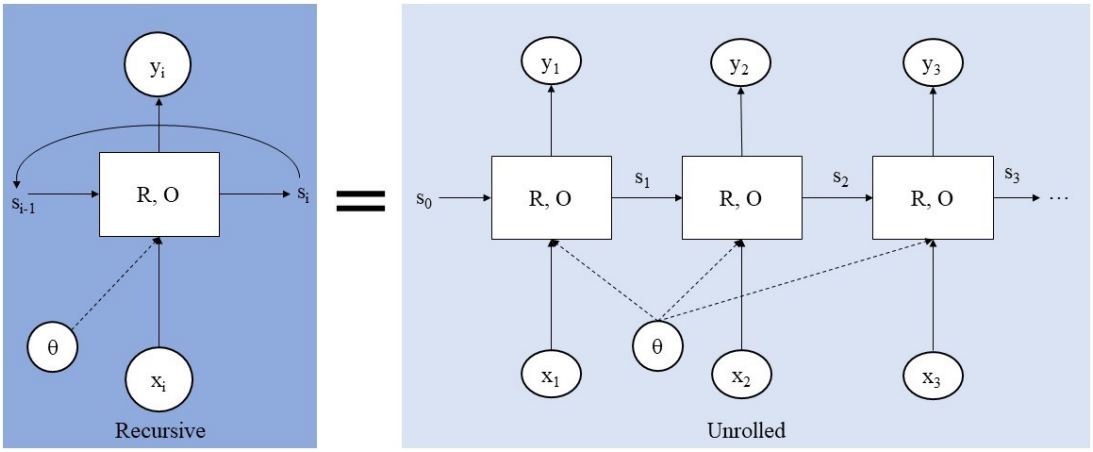

Figure 2.2: Graphical illustration of an RNN, based on the representation by Goldberg [10].

generating an output y_i and a new state s_i. On the left side, the recursive representation, and on the right how the calculations unroll. Note how the parameters and the functions are shared through computations. In Equation 2.6 it is made clear that, to obtain these values, the network counts on a recursively defined function R to generate the newer states and another function O to map the current state into an output.

$$\begin{aligned} \mathrm{RNN}(s_{i-1}, x_i) &= s_i, y_i \\ s_i &= \mathrm{R}(s_{i-1}, x_i) \\ y_i &= \mathrm{O}(s_i) \end{aligned} \tag{2.6}$$

The main objective while training the network is to tune the parameters in order to produce states that transmit useful information. These parameters are all shared between steps with different inputs. The RNN training method is similar to the one used in feed forward NNs described before, using the SGD and Backpropagation Algorithm, but in the RNN context it is called Backpropagation Through Time [49], since parameters are shared across iterations, and even the gradients are related to previous steps.

One variation of RNNs is the Bidirectional Recurrent Neural Network (BI-RNN) [39], which applies the concept of RNNs looking back, but it also looks ahead and uses words that come after the one that it is currently trying to predict. It keeps a forward and a backward state vectors, generated by two different RNNs. Then, the output will be a combination of the results of both RNNs. Even though the networks are independent from one another, they share the error gradients. Sometimes, the input sequence is fed to the RNN in a backward order, where the vectors are read backwards.

It should be mentioned that these RNN architectures still have some issues with far-apart dependencies, leading to either vanishing or exploding gradients. This means that the obtained error gradients can either be so small that the network cannot learn any longer, or that its updates are so big that it becomes unstable. Some more sophisticated RNN architectures have been developed to deal with these problems. One simple technique that deals with exploding gradients is called the Gradient Clipping, consisting of limiting the magnitude of the gradient between two values.

2.1.2.1 Long Short Term Memory

A well-known improvement to RNNs is the Long Short-Term Memory (LSTM) [15]. They do not differ much from the standard RNNs, but they introduce a small, repeating module to deal with long-term dependencies. Introducing the concept of gates, they control what information can flow into the next state. The gate is, essentially, one layer composed by perceptrons: values closer to 0 will not get through, and closer to 1 are kept for the next computations. Figure 2.3 shows how this module works. The top line represents the transformation of the cell state from s_{t-1} to s_t. Below, the visualization divides each step in different shades of blue to better understand the purpose of each gate.

In the following examples, z_t is considered the combination, such as the concatenation or the sum, of the input x_t and the output of the previous computation h_{t-1}. The Forget Gate Layer, f_t functions as a whole NN whose objective is to use z_t and output values in the interval of 0 and 1. The values closer to 1 are the ones we should keep, while closer to 0 means that they should be forgotten.

$$f_t = \sigma(W_f \cdot z_t + b_f) \tag{2.7}$$

In the next couple of steps, it is decided what to store in the cell state. In the Input Gate, z_t is used to choose the values to be updated, while the third layer creates a vector, s'_t, of new candidate values to be added to the new state.

$$i_t = \sigma(W_i \cdot z_t + b_i) \tag{2.8}$$

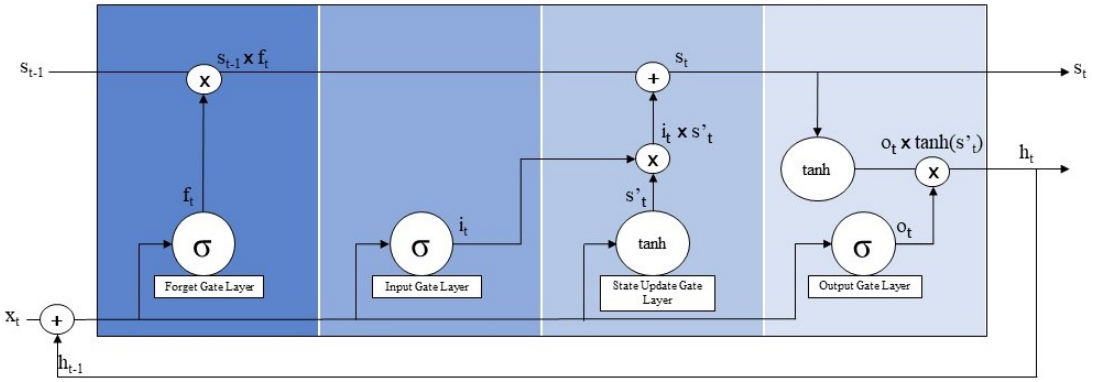

Figure 2.3: Graphical representation of the LSTM module, with the gate system that manages the long-term dependencies.

$$s'_t = \tanh(W_s \cdot z_t + b_s) \tag{2.9}$$

With the previous results, it is easy to obtain the new cell state s_t, by multiplying the old state with the elements to be forgotten, f_t, with the addition of the new candidates calibrated with the values that were kept to update:

$$s_t = f_t \cdot s_{t-1} + i_t \cdot s'_t \tag{2.10}$$

The final step is to decide the output h_t, which is built with the help of an Output Gate Layer and resultant cell state s_t, after being transformed by a $\tanh$ layer that will set the values between -1 and 1.

2.1.2.2 Gated Recurrent Unit

Another RNN architecture that is worth mentioning is the Gated Recurrent Unit (GRU) [4], which is also based on a gating mechanism. However, GRUs use fewer gates and do not rely on a memory unit. There are essentially two gates: an Update Gate z and a Reset Gate r, computed by the following equations:

$$z = \sigma(W^{xz} x_j + h_{j-1} W^{hz}) \tag{2.11}$$

$$r = \sigma(W^{xr} x_j + h_{j-1} W^{hr}) \tag{2.12}$$

The current state of the network, or activation, s_j, can be seen as the output y_j. It is based on the linear interpolation between the previous state and a candidate activation h_j, using the Update Gate to decide how much the activation should be updated.

$$s_j = (1 - z) s_{j-1} + z \cdot h \tag{2.13}$$

The candidate is controlled by the Reset Gate. This gate is responsible for ignoring the previous states, when set to values close to 0, simulating the beginning of an input sequence.

$$h = \tanh(W^{xh} x_j + (r \cdot h_{j-1}) W^{hg}) \tag{2.14}$$

There are many similarities between LSTMs and GRUs, such as the updates from one state to the next, keeping past information and adding new content, allowing not only the remembrance of important features, but most importantly the back propagation of the gradients without fast vanishing. Both architectures are very effective, and they are known to perform similarly in machine translation tasks.

2.1.3 Word Embeddings

In the context of NNs applied to NLP, words cannot be passed in its original form to the network. When a certain piece of text is going through a network, the first step for that to happen is to change those words to a representation that can be manipulated by the equations mentioned previously in this section. This can be achieved by changing the words into real-valued vectors of a certain constant dimensional space. The vector representation of words is called Word Embeddings.

When training a NN, one can initialize these word embeddings randomly, while having its weights updated alongside the other trainable parameters of the network. However, it might take some time until the embeddings start representing meaningful relationships between words: for instance, similar words, or words that go together, often have similar values.

What is usually done in this area, which eases the process of capturing meaning in the words of the data, is to initialize the values already trained with a large corpus. The most common resources are based on the following techniques, trained with a considerable amount of data:

Word2vec [28] uses statistical methods that learn word embeddings from a given text corpus. It can be based on the Continuous Bag of Words (CBOW), as to obtain context based on the words present in a certain window. On the other hand, it can also be based on the Skip-Gram Model, in this case to predict the surroundings of a certain word. There are pre-trained word embeddings already trained with this method, based on the Google News corpus, that has over 100 billion words. These words are represented with vectors with 300 features, or some other values.

The Global Vectors for Word Representation (GloVe) [32] is an algorithm that can efficiently learn word vectors. It does not use windows for context, like the previous method. Instead, it constructs a matrix that represents word co-occurrences probabilities with Latent Semantic Analysis, where the rows of the matrix are the words, and the columns represent the documents in the corpus. GloVe offers many options that were already pre-trained, with multiple vector-dimensionality and number of words.

Pre-trained word embeddings are widely used in NLP tasks, even in QG, as it can be seen in Section 2.4. They are known to improve the results. When this does not happen, it could be because the chosen pre-trained embeddings are trained with a small number of words in common with the text that is being processed.

2.2 Corpora

The QG task is supported by data that must contain a large number of questions and sentences. Most of the available datasets that could contribute to this task were meant to support similar research. For instance, Reading Comprehension (RC) can be described as the process of reading a text and acquiring enough information to be able to answer questions about it. Several specific datasets have been emerging to support this task, and since they possess a large number of questions, they can be used for QG. Some corpora are briefly described in Table 2.1. The first column has the name of the corpus and the second has the total number of questions in the data. The third column reveals the data source, while the fourth exposes the type of questions that compose the data.

There are many datasets not mentioned in this report that have been used in RC research [13, 3, 14, 1], however, it was decided to give focus to those that had natural question-answer pairs ready to use. Cloze datasets were not considered as they remove a word or entity from a given sentence, working like "fill-in-the-blanks" exercises. Others were omitted because their small size that does not allow a proper training for neural models.

The first free and publicly available RC dataset was the MCTest by Richardson et al. [37]. It consists of a set with 660 stories of approximately 150 to 300 words and about 2,600 multiple choice questions.

Table 2.1: Comparison between Reading Comprehension datasets.

Corpus	Number of Questions	Text Source	Questions
MCTest	2,640	Grade school level stories	Multiple choice
SQuAD	107,785	Wikipedia articles	Based on text segments
MS MARCO	1,010,916	Bing user queries	User queries and relevant passages
NewsQA	119,633	CNN news articles	Based on article title and summary
TriviaQA	95,956	Generated by trivia enthusiasts	Independent of text passages
RACE	97,687	Middle and high school level examinations	Multiple choice
NarrativeQA	46,765	Summaries of books and movies	Non-localized questions

It was crowd-sourced, and the writers were to write stories that young children could reason about. It required very little previous world knowledge, since every question was answered with information extracted from one or multiple sentences in each text. The following text is an example extracted from the data:

"One morning, Elena woke up, much like she did every day. She threw the covers on the floor and rolled out of bed, yawning hugely. She walked to the window and said, "Hello there, Mr. tree!" at the big tree in the yard. It waved its branches back at her. She walked over to her fish bowl next. "Hello there, Mr. Fish!" But wait. Where was Mr. Fish? The bowl was empty–oh, the rocks and water and tiny castle were all there alright, but the pretty blue fish with the long shiny tail was nowhere to be seen. Elena was very worried. She liked Mr. Fish very much. She looked all around her desk, but here wasn't there. Then she looked on the floor behind the desk–and there he was! He was covered in dust bunnies and not moving. Elena picked him up and put him back in the bowl. And what do you know? He shook himself off and started swimming around again!"

1. multiple: What is the very first thing Elena does after waking up?
   (a) she says hello to the tree
   (b) * she throws the covers on the floor
   (c) she says hello to the sun
   (d) she gets out of bed...
2. one: At what time of day does this story take place?
   (a) Before the tree
   (b) At the end of the day
   (c) The story doesn't say
   (d) * Morning
3. multiple: What happened to Mr. Fish in the end?
   (a) He got put back in the bowl, but he was dead
   (b) He started swimming around in Elena's hand
   (c) He got put in a new bowl

(d) * He got put back in his bowl and started swimming around again

4. multiple: What was missing from the fish bowl?

   (a) the rocks

   (b) * Mr. Fish

   (c) the water

   (d) the tiny castle

The answers marked (*) are the correct ones. The main issue with this corpus is its size: too few stories of small size and too few questions. Nowadays, it makes more sense to use this dataset as an evaluation set, instead of a training one.

One of the most relevant corpora used in RC, especially for Question Answering (QA) tasks, is the Stanford Question-Answering Dataset (SQuAD) [34]. It is a crowd-sourced dataset based on approximately 500 Wikipedia articles of various themes, generating more than 107,000 pairs of questions and answers. This corpus comes to address the need for datasets that have a considerable size, with thousands of questions, allowing the training of deep learning models. It presents a great variety of answers, with categories like Common Noun Phrases, Dates, Locations, Adjective Phrases, among others, distributed as seen in Figure 2.4. This implies a diverse collection of questions in terms of syntactic structure and, consequentially, the amount of reasoning that it might take to obtain the right answer.

In fact, during the creation of this dataset, it was reinforced that questions should diverge from the source segments, by using different words and expressions, instead of just producing the interrogative form of given sentences, in order to create compelling questions. The following example, extracted from the corpus, has words in the question that are not present in the sentence where the answer lies:

Segment of sentence: "... At Saint-Evroul, a tradition of singing had developed and the choir achieved fame in Normandy ..."

Question: "What tradition were the Saint-Evroul monks known for? "

In the example, the expression "achieved fame" was changed to "known for". Aside from this detail, we can also the the usage of the word "monks" that is not used in the seen in the selected sentence. However, the article is related to the Abbey of Saint-Evroul and in the context paragraph we can see that word appearing more than once:

Context Paragraph: "Normandy was the site of several important developments in the history of classical music in the 11th century. Fécamp Abbey and Saint-Evroul were centres of musical production and education. At Saint-Evroul, a tradition of singing had developed and the choir achieved fame in Normandy. After entering into a violent quarrel with William II of Normandy, Robert de Grantmesnil had been forced to flee to Rome in January 1061 and thence to the court of Robert Guiscard in Salerno,

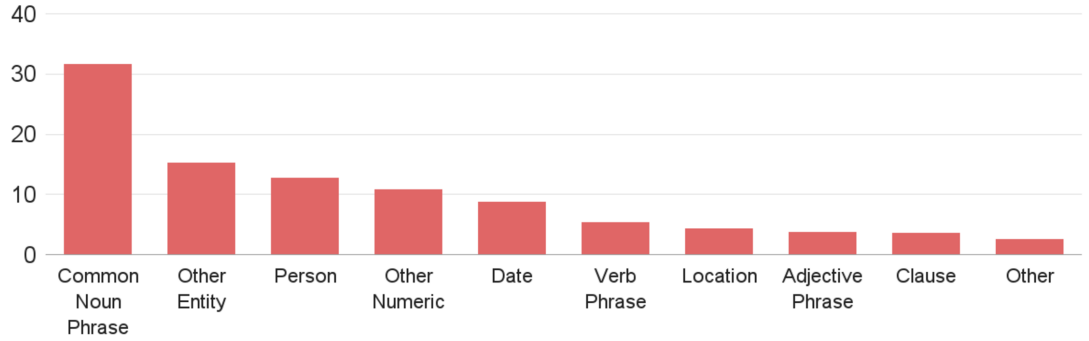

Figure 2.4: Distribution of answers in the SQuAD dataset [33]

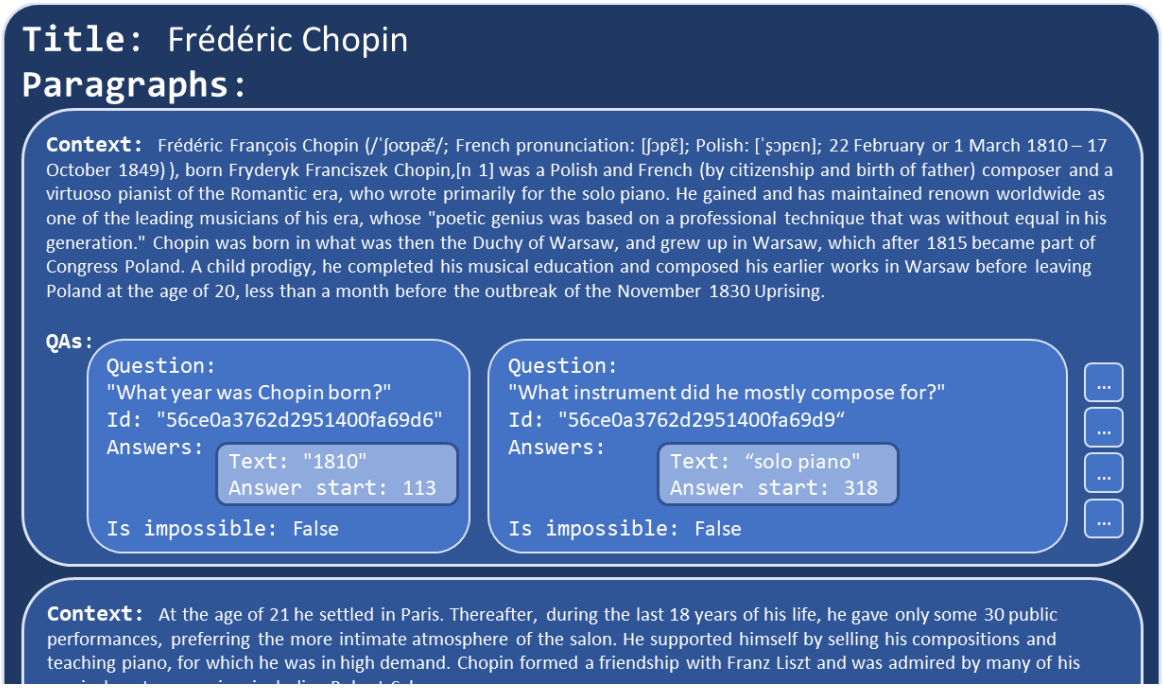

Figure 2.5: Graphical representation of the organization of SQuAD 2.0.

taking with him eleven of his *monks*, including his nephew Berengar. In his time, Saint-Evroul was famed for its musical programme and these eleven *monks* brought its musical traditions to the abbey of Sant'Eufemia (now part of the town of Lamezia Terme and different from Sant'Eufemia d'Aspromonte) in Calabria, a foundation of the Guiscards, of which Robert became abbot."

In the next example, two entities are referred to by using different words that do not appear in the text, which implies that, to answer the question, one must understand a concept not introduced in the source text.

Segment of sentence: "... The European Parliament and the Council of the European Union have powers of amendment and veto during the legislative process..."

Question: "Which governing bodies have veto power?"

Recently, a second version of SQuAD was released (SQuAD 2.0) [35], with the intent of overcoming certain weaknesses of the previous version. The improvement lies in the addition of more than 50,000 questions that do not have a correct answer in the paragraph but are closely related to its content. Each unanswerable question is presented with plausible answers. This was expected to improve question-answering models by forcing them to understand that there is not any span of the text that can actually answer certain questions.

SQuAD 2.0 is organized as shown in Figure 2.5, more specifically the first paragraph in the article that refers to Frédéric Chopin. Each article has its title and is divided by the paragraphs that constitute the whole text. In its turn, each paragraph contains a set of questions associated to the text, in this case called Context. These questions are identified as being impossible to answer or not. Each question not only contains the question text, but also the answer and an index that points to the position in the paragraph where that answer is present. In case the question cannot be answered, it does not present either the answer texts or the initial position. Instead, it has a list of plausible answers.

Another large-scale RC dataset that is also human-sourced is the Microsoft Machine Reading Comprehension (MARCO) [29]. The first release contained 100,000 queries, which were real questions issued to the Bing search engine, where a user was expecting a specific answer. This is one of the key features that makes this dataset interesting: the questions represent interactions between humans and

systems. The answers are also generated by humans, based on 10 different text passages given by the search engine in answer to the given query. Each query is classified by its intent. The second version of this dataset has 1,010,916 unique real queries, approximately 182,000 well-formed answers and about 35% of this dataset consists of questions without answer.

Example: *"A company is incorporated in a specific nation, often within the bounds of a smaller subset of that nation, such as a state or province. The corporation is then governed by the laws of incorporation in that state. A corporation may issue stock, either private or public, or may be classified as a non-stock corporation. If stock is issued, the corporation will usually be governed by its shareholders, either directly or indirectly. url:* `http://www.wisegeek.com//what-is-a-corporation.htm`*"*

Q: what is a corporation?

Query type: DESCRIPTION

Answers: A corporation is a company or group of people authorized to act as a single entity and recognized as such in law.

NewsQA [45] is very similar to Stanford Question-Answering Dataset (SQuAD) when it comes to the number of question-answer pairs. It is also human-sourced. This dataset is based on more than 10 thousand news articles from CNN. The answers are based on spans of arbitrary length, instead of just word or entity extraction, and it was encouraged to design the pairs with a certain degree of syntactic and lexical diversity: the questioners were to ask the questions based on the title of the article and its summary points. As we can see in the example, this dataset presents some questions that cannot be answered by the contents of the source article.

Example: *Vancouver, British Columbia (CNN) – The tragic death of a trainer at Sea World last week revived a number of long simmering questions. While we still grapple with "how did this happen?" the central question for many revolves around the role of large mammals – like Tilikum the killer whale – in zoos and aquariums: Should they be there or not? Animals in zoos, aquariums and museums play an important and powerful part in our cultural and formal educational processes. Humans are inherently interested in nature. We are not very far removed from a time when being knowledgeable about nature was vital to life; you either knew how to find your dinner or you were dinner. Today, with well over 50 percent of our populations living in cities, we are rapidly becoming divorced from the realities of the animal world. The dialogue we see in the media, read on blogs and hear in conversation makes it clear that many people have lots of ideas about what's happening in our natural world, much of it not correct. This lack of knowledge is concerning in a world beset by environmental problems, where species are disappearing at an alarming rate. We need people to understand the changes taking place in our natural systems and appreciate that each of our actions has an impact. More interest and knowledge, not less, is essential. Zoos and aquariums provide access and a vital connection to the world of wildlife and our environment, helping to foster an understanding of nature and how it works, and an appreciation for why it matters.*

- **Q:** What makes zoos and aquariums vital to education?
  - **Answer 1:** Animals in zoos, aquariums and museums play an important and powerful part in our cultural and formal educational processes.
  - **Answer 2:** the world of wildlife
- **Q:** What does the death of a whale trainer raise questions about?
  - **Answer 1:** Should they be there or not?
  - **Answer 2:** number of long simmering
- **Q:** What does the close up creature seeing makes to people?

- **Answer not in story.**

Unlike news stories, fictional ones, such as books and movies, possess a rich vocabulary, various events and concepts. Stories can also be very interesting when they are self-contained: even though they have a diverse set of entities and events, they are all contained in a certain domain that allows a reader to fully understand a narrative. The NarrativeQA dataset [21] consists of stories with human-generated natural questions with an average length of approximately 10 tokens, mostly starting with "WH-". These questions were based on the stories summaries, but the dataset supplies both the summary and the full story. The answers to these questions might require several spans of the document, instead of focusing on just one sentence. It counts with 1,567 stories and 46,765 question-answer pairs.

Example:

- Kind: Movie script
- Source: `http://www.imdb.com/scripts/Lord-of-the-Rings-Return-of-the-King.html`
- Summary: *"Gandalf leads Aragorn, Legolas, Gimli, and King Théoden to Isengard where they reunite with Merry and Pippin. With Saruman defeated, Gandalf retrieves Saruman's palantír. Overcome by curiosity, Pippin steals a glance into the seeing-stone, and suffers a mental attack from Sauron himself. Gandalf deduces that Sauron will attack Gondor's capital Minas Tirith, so he rides there to warn them, taking Pippin with him because Sauron thinks Pippin is the ring bearer. (...)"*
  - **Q:** Who does Sauron believe has the One Ring at the beginning of the story?
    - **Answer 1:** Pippin.
    - **Answer 2:** Pippin.
  - **Q:** Why do the Army of the Dead follow Aragon?
    - **Answer 1:** He promises to release the Army of the Dead from their curse.
    - **Answer 2:** He is the rightful heir of Isildur.
  - **Q:** What does Frodo lose in his fight with Gollum?
    - **Answer 1:** His finger.
    - **Answer 2:** The Ring.

TriviaQA [17] is a corpus composed by triples (Question, Answer, Evidence). There are more than 95,000 Question-Answer pairs, and each question has more than 4 words, and, in average, less than 14. As it contains trivia questions, it is guaranteed to have varied content and complex types of questions. For every question-answer pair, it is given an average of about six passages that confirm the answer to such question. This evidence was extracted from a search engine based on the existing pair, which means that it might not contain the exact same words or structure, or even the information needed to fully answer a given question, implying a higher level of reasoning.

Example: *"The Dodecanese Campaign of World War II was an attempt by Allied forces to capture the Italianheld Dodecanese islands in the Aegean Sea following the surrender of Italy in September 1943, and use them as bases against the German-controlled Balkans. The failed campaign, and in particular the Battle of Leros, inspired the 1957 novel The Guns of Navarone and the successful 1961 movie of the same name."*

Q: The Dodecanese Campaign of WWII that was an attempt by the Allied forces to capture islands in the Aegean Sea was the inspiration for which acclaimed 1961 commando film?

A: The Guns of Navarone

Finally, RACE [23] is a large-scale RC corpus which is composed by information extracted from English examinations for Middle and High School students in China. It has about 28,000 passages, which generated approximately 100,000 questions, all of them written by human experts with the intent to analyse the RC skills of students. In the next example, we can see that the questions are multiple choice. Each question is identified by type. It involves different degrees of reasoning to answer the multiple types of questions, and it is very diverse in terms of content. Considering the nature of our research, the issue of this dataset is that some questions are cloze-styled, meaning that they are not proper questions, but sentences that were fragmented, with the answer filling a blank space.

"Do you love holidays but hate gaining weight? You are not alone. Holidays are times for celebrating. Many people are worried about their weight. With proper planning, though, it is possible to keep normal weight during the holidays. The idea is to enjoy the holidays but not to eat too much. You don't have to turn away from the foods that you enjoy. Here are some tips for preventing weight gain and maintaining physical fitness: Don't skip meals. Before you leave home, have a small, low-fat meal or snack. This may help to avoid getting too excited before delicious foods. Control the amount of food. Use a small plate that may encourage you to "load up". You should be most comfortable eating an amount of food about the size of your fist. Begin with soup and fruit or vegetables. Fill up beforehand on water-based soup and raw fruit or vegetables, or drink a large glass of water before you eat to help you to feel full. Avoid high-fat foods. Dishes that look oily or creamy may have large amount of fat. Choose lean meat. Fill your plate with salad and green vegetables. Use lemon juice instead of creamy food. Stick to physical activity. Don't let exercise take a break during the holidays. A 20-minute walk helps to burn off extra calories."

1. **Q:** Which of the following can NOT help people to lose weight according to the passage? (Question type: detail reasoning)
   (a) Eating lean meat.
   (b) * Creamy food.
   (c) Eating raw fruit or vegetables.
   (d) Physical exercise.
2. **Q:** Many people can't control their weight during the holidays mainly because they _. (Question type: paraphrasing)
   (a) *can't help eating too much
   (b) take part in too many parties
   (c) enjoy delicious foods sometimes
   (d) can't help turning away from foods.
3. What is the best title of the passage? (Question type: summarization)
   (a) How to avoid holiday feasting.
   (b) Do's and don'ts for keeping slim and fit.
   (c) * How to avoid weight gain over holidays.
   (d) Wonderful holidays, boring experiences.

2.3 Evaluation Metrics used in Question Generation

Automatic evaluation metrics are widely used in NLP tasks. While human evaluation is more reliable to infer the quality of a certain generated text, automatic metrics are easy to compute and have a larger degree of availability. The most commonly used metrics are based on word overlap, and most were originally meant to evaluate the performance of machine translation systems. However, they cannot capture semantic diversity.

Bi-Lingual Evaluation Understudy (BLEU) [31] compares n-grams between the hypothesis and reference sequences. Its computation is similar to precision. Each n-gram weighs the same during computations. The BLEU-N score is based on the n-grams of length N. Scores are obtained for each example in the set, and then averaged. The metric seems to favour shorter sentences and it is known to not correlate that well with human perception.

Metric for Evaluation of Translation with Explicit Ordering (METEOR) [6] is said to have higher correlation with human evaluation. It is calculated with the harmonic mean of unigram precision and recall, where the recall weighs more than the precision. Precision is the number of true positive results divided by the number of all positives obtained. On the other hand, recall is the number of correct positives obtained divided by all the true positives. Each unigram match is not only based on exact match, but also stemming, paraphrases and synonyms.

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [25] is a set of metrics that were intended to evaluate automatic summarization. ROUGE-L, for instance, computes the F-measure based on Longest Common Subsequences (ROUGE-L) between hypothesis and reference sequences.

Skip Thoughts [20], which is a neural networks model that encodes sentences into embeddings and then is able to predict the preceding and following sentences after decoding. This might be interesting to capture semantic relatedness. Then, it uses cosine similarity to obtain the score between reference and candidate.

Embedding Average just computes the average of the embeddings that compose the hypothesis question, computing a sentence-level embedding. Once again, it uses cosine similarity.

Vector Extrema [9] computes the sentence-level embeddings by taking the most extreme value, for each dimension, of the word vectors in the whole sentence and then use that value. The similarity is once more measured with cosine similarity. This method is thought to stand out the most informative, unique words, while ignoring the most common ones.

Greedy Matching Score, which is the only metric that does not measure sentence-level embeddings. Each word from both reference and candidate sequences will be greedily matched based on cosine similarity of the word embeddings. Then, the total score is averaged. Matching word embeddings will take notice of tokens that share semantical characteristics.

As we will see in section 2.4, these metrics are used in QG research, sometimes by themselves, but sometimes paired with human evaluation.

2.4 Neural Approaches to Question Generation

After the work of Rus et al. [38], which defined the text-to-question task, other approaches for the QG have appeared over the past years. Predominantly, rule-based approaches were used to transform declarative structures into an interrogative, but the performance of these models relies on the quality of those hand-crafted rules and templates. On the other hand, neural solutions for QG do not rely on written rules, as they learn patterns, while achieving state of the art results.

The first relevant Neural QG approach, by Serban et al. [40], had the intent of synthetically generating a Question-Answer dataset with 30 million factoid questions. Their model was trained with the

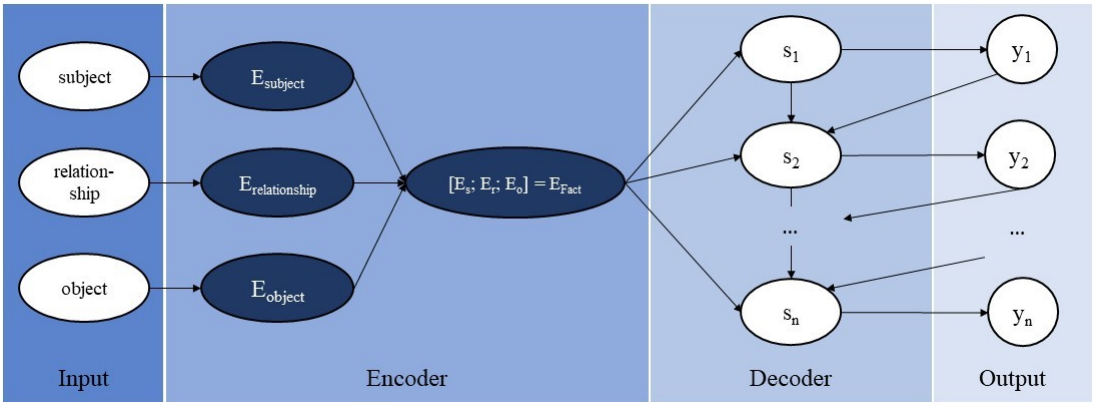

Figure 2.6: Representation of the neural QG model for structured text input by Serban et al. [40].

SimpleQuestions dataset, which consists of a total of 108,442 questions paired with Facts that have the structure of a tuple with three atoms: (subject, relationship, object). The idea was to translate these tuples into a question composed by both the subject and relationship, while the object represented the answer. The solution was developed with the encoder-decoder architecture. The Encoder was responsible for the encoding of each atom individually into an embedding, using a vector of the size of the vocabulary and an embedding matrix already learned before, not tunable during the encoding. The input given to the decoder is the concatenation of the encodings of the three atoms that compose each fact. The decision of using an already learned embedding matrix, independent of the training set, is based on the idea that, otherwise, it would deliver a poor performance when encountering unseen words in any of the atoms.

The solution to finding unknown words in each question-fact pair could be divided in two steps. First, parsing the questions for overlapping words in the subject portion of the fact and then, replacing these words with specifics tags serving as a placeholder. This last step has two variants: either use the same <UNK> placeholder in every unknown word, or have 60 different tags, for instance, <location-tag>, and place them according to the context in the sentence. Then, the decoder uses a GRU RNN with Attention on the encoder output to obtain the question. The model must learn to deal with the previously mentioned tags, by replacing them with the subject string. The structure of the model is shown in Figure 2.6. To train the neural model, it used the optimization of the log-likelihood using Adam, with early stopping based on the METEOR score for the validation set. The model was evaluated using automatic metrics: BLEU, METEOR, Embedding Greedy and Sentence Similarity. The first two automatic metrics support the use of a single placeholder tag, while sentence similarity has higher score while using multiple placeholders. The highest BLEU score obtained was 35.6, while the best METEOR is 35.38. The Embedding Greedy score was 77.01. They also performed human evaluation, where human evaluators considered the generated questions on par with the original human written ones, leading to the conclusion that neural models could, indeed, be used to create RC models.

While the previous model was successful in generating questions, the input had to have a very specific structure, which is not what is intended for the end-to-end task of QG. We want to take an unstructured text and acquire a question related to its content. New approaches followed that tried to resolve this challenge.

2.4.1 From Unstructured Text Sequences

Zhou et al. [51] conducted a study to generate a question from a text passage that contained the correspondent answer. The input was enriched with features like Part-of-Speech (POS) and Named

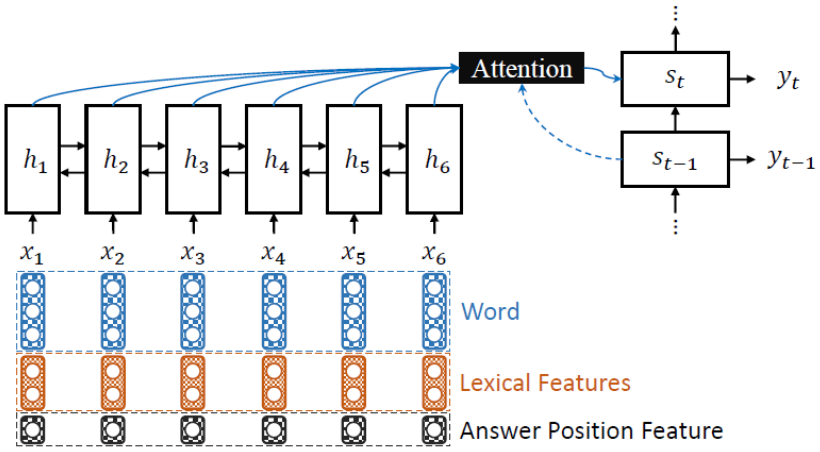

Figure 2.7: Representation of the neural QG framework by Zhou et al. [51].

Entity Recognition (NER) tags. It also uses information about the answer position in the input sequence, identified with the BIO tag: *B* is the beginning of the answer, *I* is for words that belong to the answer and finally *O* for every other word that does not belong to it. The embedding vectors for each of these components were concatenated. The encoder was composed by a bidirectional GRU, allowing the input to be read in forward and backward order. It is responsible for the creation of two hidden vectors, the forward and backward, that will also be concatenated for the decoder to process.

The decoder is composed by a GRU layer with Luong attention [26]. It is initialized with the backwards representation and each decoder state is joined with each encoder hidden state, returning an attention importance score. To deal with unknown words, a copy mechanism first suggested by Gülçehre et al. [11] was implemented and it consists in a pointing approach: a copy switch uses the current decoder state and the context vector and returns the probability of copying a word from the input sequence. To decide what word should be copied, it is used the same attention mechanism described above. A graphical representation shows the model in Figure 2.7. This model was trained using SQuAD, and to optimize the loss function they also use Adam, with dropout based on the BLEU score and gradient clipping.

The evaluation results were reported with Human Evaluation and on the BLEU-4 metric, where it obtained 13.29.

The model by Du et al. [7] was developed to generate natural questions from a given input that could either be a sentence or a full paragraph and both these solutions were compared against each other. To encode a sentence, an encoder based on bi-LSTM was used. For every time step, it would generate the forward and backward hidden states, that concatenated would be used to obtain the attention-based encoding of the sentence. Another bi-LSTM encoded the paragraph, considering a predefined length threshold to focus on a certain part of an otherwise possibly oversized input sequence. The decoder receives the output generated by the previous encoder layers concatenated, and uses a single LSTM cell. The new state is created with the previous hidden state and previously generated word.

For pre-processing, the tokenization and sentence splitting were executed with the Stanford CoreNLP [27]. Training was accomplished with the SQuAD, but not exactly with Question-Answer pairs. Instead, the sentence with the answer was located in the paragraph, and it would be selected based on the constrain of having at least one non-stop word in common. The copy mechanism here is as simple as replacing the unknown token with the one in the input sequence that has the highest attention score.

The system was compared to multiple baseline systems with the BLEU, METEOR and ROUGE scores, outperforming all the considered baseline systems. The values can be seen in Table 2.2. The

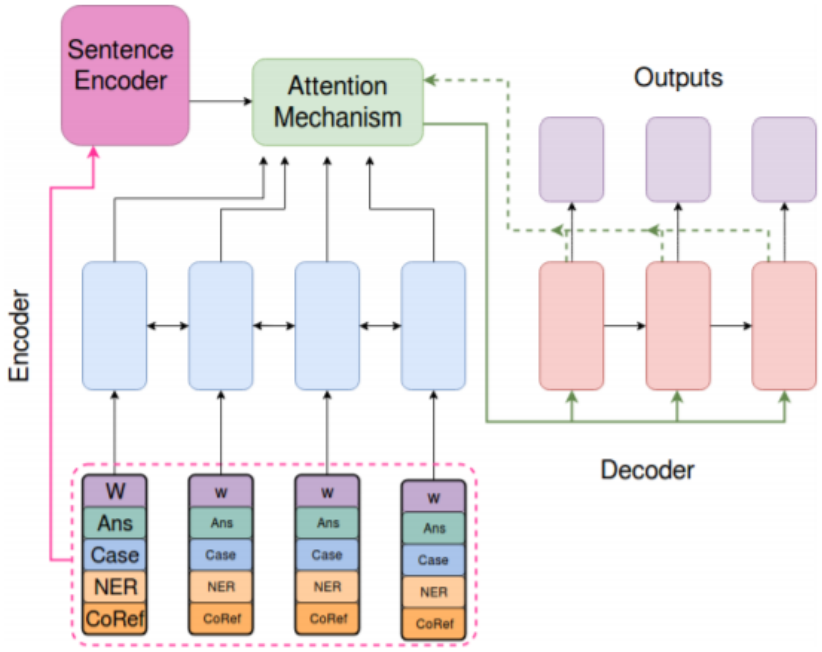

Figure 2.8: Diagram of the neural QG model by Harrison and Walker [12].

model benefits from the addition of pre-trained embeddings but delivers smaller scores when the paragraph encoding is introduced.

Yuan et al. [50] developed a model that had many similarities to the previous ones: using bi-LSTM with attention mechanism and the same pointer-softmax copying mechanism by Gülçehre et al. [11]. The encoder receives input from two sources: the document and the answer to the desired question, in the form of embedding vectors. It uses a binary feature to identify the words that belong to the answer.

This is the first model to mention the concept of teacher forcing, whose function is to make the decoder generate words during the training phase by using the source sequence, instead of the model output. The most important novelty of this work is the introduction of Reinforcement Learning (RL) to the training process, in an attempt to tune the model by combining the negative log-likelihood with policy gradient optimization. There were multiple suggestions of possible rewards closely related to the quality of the questions, such as the score obtained by passing the generated questions to a pre-trained Question-Answering model, or the fluency score given by a language model. They also experimented using BLEU score as a reward.

The results were reported concerning the scores mentioned before, plus the F1. The best BLEU score was 10.5, obtained when the RL technique used the Question-Answering reward. Apart from the Fluency measure, all other scores are improved by the application of policy gradient optimization methods, which indicates a lower probability for generating less frequent or even unknown words. One of the reported issues of this model consists in the frequent swapping of similar entities present in the text.

Harrison and Walker [12] used a similar architecture, shown in Figure 2.8, consisting of Encoder-Decoder with attention and copy mechanisms. It is one of the cases that uses the answer encoding, more specifically, a binary feature showing if a given token is part of the answer or not. The input counts with NER and casing encodings. It also adds co-reference labels, that are measured during the preprocessing time. It uses the context of the input text to augment it with useful information. For instance, in a sentence like "Mary goes shopping and forgets her wallet", aside from the POS features, it adds information regarding the word "her", explicitly showing who it means (Mary).

Table 2.2: Best reported values, for the most common automatic metrics, obtained in previous neural QG research. Table adapted from Pan et al. [30].

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE
Du et al. [7]	43.09	25.96	17.50	12.28	16.62	39.75
Zhou et al. [51]	-	-	-	13.29	-	-
Yuan et al. [50]	-	-	-	10.50	-	-
Harrison and Walker [12]	-	-	-	14.39	19.54	43.00
Kim et al. [18]	-	-	-	16.20	19.92	43.96
Wang et al. [48]	-	-	-	10.80	-	-

Once again, SQuAD is used and the results are reported with BLEU-4 (14,39), ROUGE (43.00) and METEOR (19.24). While their model outperform previous QG systems, it seems that the usage of co-reference features hurts the model performance.

Kim et al. [18] observed that using answer encoding alongside the sentence could result in useless questions that end up using words from the answer in the text. It also points that copy mechanisms could achieve the same effect. To counter this issue, it suggests using a separate RNN to encode the answer. That information is concatenated with the sentence encodings and passed to an Attention-based Decoder.

The results obtained were reported with the same metrics as Harrison and Walker [12], outperforming their solution, reaching the highest BLEU obtained so far, with a score of 16.20.

Table 2.2 assembles the models that are going to be described in this section, showing the scores that each reported. Some of the models reported more metrics, and others focused essentially on human evaluation. In this table, we show the ones that are useful to our research.

2.4.2 Question Generation Combined with Special Modules

As mentioned before, one of the potential uses of QG systems might be supporting other tasks, not only in terms of data augmentation, it might also improve the ability to answer questions. The neural model designed by Wang et al. [48] was supposed to ask and answer questions, defending that this joint-learning might be beneficial for both tasks. This is executed by alternating the input data between question answering and question generation examples, both using embeddings with word and character-level encodings. The embeddings are complemented with information about the presence of each word in the document and the condition, which can either be the question or the answer. The structure does not differ much from the previous solutions, applying an attention-based, sequence-to-sequence model, with bi-LSTM in the case of the encoder, and a pointer-softmax decoder. In the training phase, this model also applies teacher forcing and beam search in an attempt to increase the fluency of the generated questions. The reported results affirm that, even though the QA benefits from the joint-training, QG

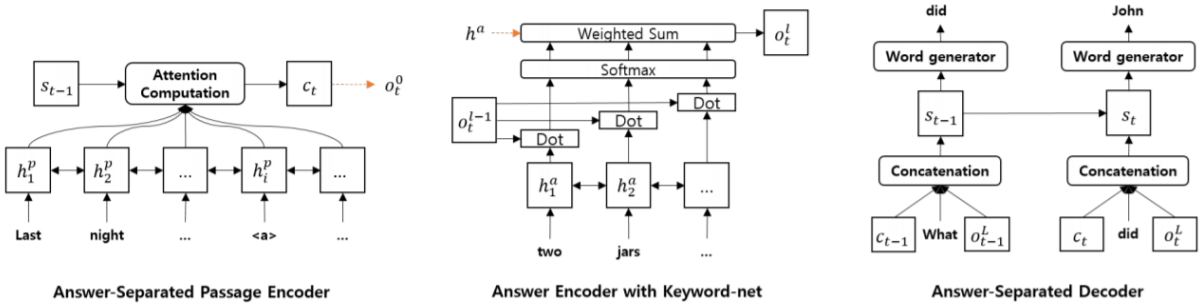

Figure 2.9: Visual representation of the architecture of Kim et al. [18].

results are worse, by a small decrease, especially when it comes to automatic metrics: the QG system scores with BLEU-4 10.8, and after joint training it is 10.2.

The final QG model to be studied was by Subramanian et al. [43]. This model aimed to generate questions based on a document by itself. To tackle this, in addition to the neural model already presented in the previous solutions, it is presented a new module with the goal of finding interesting entities in the text to generate questions about: the Key Phrase Detection Module. This is a viable solution because the model is trained on the SQuAD, that reports more than 50% of the answers as entities. The neural model to solve this selects a subset of entities from a list of candidates, identified by NER tools, and executing binary classification. When the NER fails to recognize entities in the text, the suggestion is using Pointer Networks to extract interesting aspects of the input. The reported results are solely based on human evaluation, and while they show fluency and semantic relevance, they are still easy to identify as machine-generated questions when compared against human generated questions.

2.5 Summary

This chapter started with a review on the fundamental concepts that are needed to understand Deep Learning, which formalized some ideas that are required to continue the work on neural QG, but also to design and execute our solutions.

The chapter mentions various alternatives that could be used in terms of data. There are many corpora that possess text and questions that could have been chosen for this research.

Finally, we explored recent work on neural QG. It is a challenging task comparing these models against each other by solely reviewing their reported results. The task of QG often presents linguistic divergence that cannot be captured solely by the metrics suggested in these works. Thus, even if a model is delivering acceptable questions, it can be misinterpreted because of semantical variances. Some projects report Human Evaluation alone, others combined with BLEU, METEOR or the F1 score. Others suggested estimating the quality of the obtained questions by passing them through language models to evaluate the perplexity and fluency, or even pre-trained QA models to verify is the generated questions can be automatically answered.

3

Automatic Question Generation

Inspired by the increasingly usage of Deep Learning techniques for Natural Language Processing (NLP), we built our question generation models using the same principles. As mentioned in previous chapters, the main goal for this project lies in developing a solution, making use of state-of-the-art neural techniques, that will allow the generation of questions based on a text sequence of arbitrary size.

Given as input a text sequence x, the model should output a question y, related to the content of the original text.

The idea was to implement a quite simple model and then improve it with certain techniques, while assessing the importance of such upgrades in terms of improvement of the generated questions. Some parameters were kept untouched from model to model, and throughout this section they will be mentioned only when changes occur.

The models described in the following sections were executed with Texar [1], v0.1.0. which is an open-source toolkit that supports many text generation tasks. It was developed on top of Tensorflow and it allows the implementation of various NN architectures, with ease on both design and training. The solutions described in the previous sections of this chapter were based on multiple modules available in this tool and, therefore, many of the hyperparameters available for each module were left with their default values.

First, Section 3.1 details the solution that was executed with Recurrent Neural Networks (RNNs), based on the research already seen in Section 2.4. Then, the solution based on the Transformer architecture is presented. Finally, we describe in Section 3.3 one final Question Generation (QG) solution that does not rely on neural networks, but will serve as baseline for the experiments executed and described in Chapter 4.

3.1 RNN-based Solutions

RNNs are widely used in Sequence-to-Sequence (Seq2Seq) [44] prediction tasks, whose objective is, as the name suggests, to obtain a sequence of text based on another given as input, like in machine translation. These problems are usually modeled with the Encoder-Decoder architecture. It consists in having one or more RNNs working as the Encoder, encoding the input into a fixed-size vector representation. Then, the Decoder, which is also based on RNNs, uses these vectors to generate the output

[1] https://texar.io

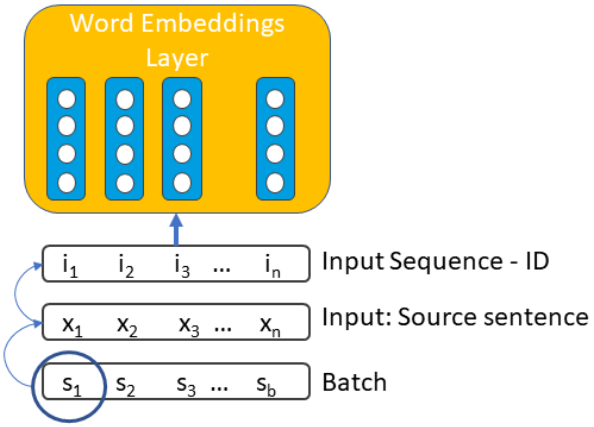

Figure 3.1: Input processing in the seq2seq models.

sequence. The gradients are propagated through the whole network. Encoder-Decoder resolves the issue of having input and output vectors of different domains with different dimensions.

The task at hand falls into this category of generating a sequence given another, therefore we will design our models based on the Encoder-Decoder concept. In the remaining part of this chapter, we will explore this paradigm by executing it in the simplest way, and upgrading it with specific modules.

3.1.1 Simple

The simplest model, which will be called Simple for now on, is our starting foundation. This model is a sequence-to-sequence, consisting of an encoder-decoder architecture.

The model receives a sentence and it generates a question. The sentences passed to the model are tokenized and represented as a vector with a constant size equal to the longest sequence found in the data. Each word is replaced with an identification number and the rest of the vector is padded with zeros to keep the predefined size.

The first layer constituting the Encoder of the network is the Word Embeddings layer. As we can see in Figure 3.1, it is responsible for mapping the previously mentioned indexes into fixed sized embedding vectors. These vectors' dimensionality is d = 256 and they are randomly initialized.

Then, we have a Bidirectional Long Short-Term Memory (LSTM), so the input can be read both in forward and backward order. The Encoder receives as input the embedding vectors described above. It shares the same dimensionality. The LSTMs both have the same configurations. The size of each of the output (dense) layers is 128. Dropout was applied to the output hidden states of these layers. Together, the concatenation of both forward and backward hidden states produces a vector.

With this setup, the Encoder collects, from the entire sequence, information that will be propagated to the rest of the network. The Encoder outputs the above vector, which will be called the context vector from now on. It encapsulates information from the original sentence. A visual representation of the Encoder is shown in Figure 3.2.

The Decoder will predict the output at each timestep, which means that it will generate the question word by word. It is composed by a single LSTM cell and the generation of each word is based on a hidden state and the representation of the word in the previous position. Teacher forcing is applied in the Decoder, which means that instead of using the previously generated word, it uses the target sequence, from the train data, at each timestep. This allows the model to learn the correct sequence, instead of being penalized by its own mistakes.

The Decoder generates a state based on ground truth and the hidden state of the previous RNN

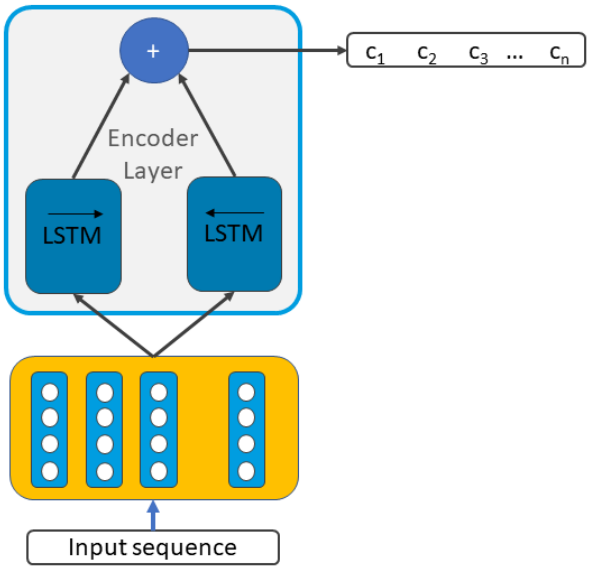

Figure 3.2: Encoder structure.

layers. In the first iteration, the Decoder is initialized with the context vector given by the Encoder. The output state of the Decoder is a real-valued vector with the size of the vocabulary. It is transformed with a $softmax$ function so that each position is filled with the probability of a certain word fitting in that part of the sentence. Figure 3.3 shows the entire pipeline of the Simple model, as it was explained.

The learning process is divided in two phases: training and inference. The dataset is divided in small batches, that in this case are of size b = 64, that are used to optimize the loss function and update the weights of the model. It computes the softmax cross entropy. It uses the Adam optimizer with default settings and learning rate $\eta = 0.001$.

During the training part of the learning process, it uses a greedy strategy to select the output words, which means that it will choose the word with highest probability. At the end of each epoch, it enters in the inference phase. The data is switched to a validation set. Here, it uses beam search to produce the outputs. Given a certain beam size, that in this case is bs = 10, the model keeps track of that many best options. This allows the model to recover from any possible mistakes during word generation.

The generated questions will be compared with the target ones, using the automatic evaluation metric BLEU, which measures word overlaps. Throughout the learning process, the highest BLEU score is saved, and when one epoch manages to update the model in a way that this evaluation step returns a higher score, the model is updated with these new weights. Otherwise, the results obtained during that epoch are ignored. Either way, the training process repeats.

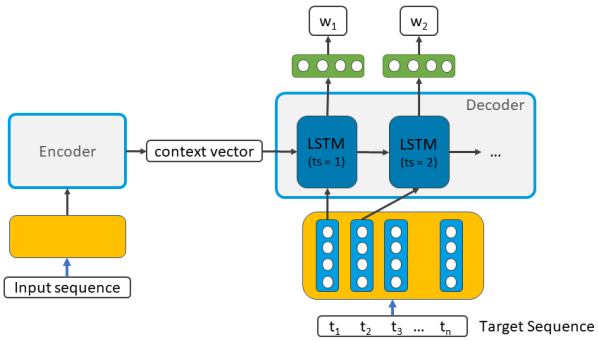

Figure 3.3: The complete structure of the Simple model, with detail on the Decoder.

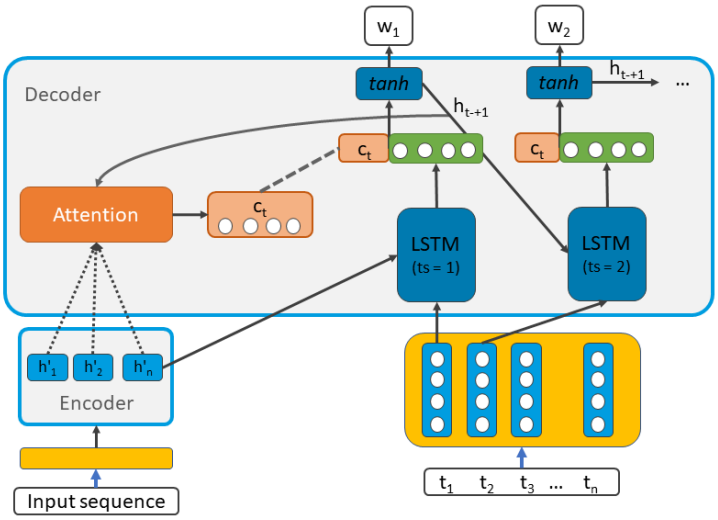

Figure 3.4: The previous model, improved with attention.

In our research, we used Early Stopping in our training. When the model stops learning for 7 epochs, we stop the training and consider it over. In the case of Simple, it only ran for 2 epochs. The highest BLEU score obtained during evaluation of the validation set was 5.04. We discuss this short training process in the next chapter, by observing the experimental results.

3.1.2 Attention

In the previous section we saw the Encoder producing a vector that contains all the information in the input sequence. This is problematic when dealing with long sentences, because it is difficult to keep track of information related to the beginning of the sequence. Attention is used to solve this problem, as it makes the model focus on specific parts of the input when it is decoding, instead of looking at it as a whole.

The idea of alignment is crucial to understand attention. It consists in finding the parts of the input that are relevant to the current position in the output. Given the current hidden state of the decoder h_t and all the hidden states of the encoder h'_s, it is possible to obtain the normalized weights, computed with Equation 3.1, which will influence the decoding process.

$$\alpha_t = \frac{\exp(\text{score}(h_t, h'_s))}{\sum_{s'=1}^{S} \exp(\text{score}(h_t, h'_s))} \tag{3.1}$$

A new vector is added with the Attention mechanism: c_t. It is computed at each decoding timestep, using the weighted sum of all the hidden states of the encoder h_s.

$$c_t = \sum \alpha_t h_s \tag{3.2}$$

Luong attention [26] was applied to the Simple model. The decoder is initialized with the last hidden state of the encoder. At the current timestep, the output of a single LSTM cell is concatenated with the weighted context vector. The result of this operation is then passed to a feed-forward neural network with a $\tanh$ activation. The neural network is trained with the rest of the parameters of the network. This will output a new hidden state and the word at each timestep.

$$h_{t+1} = \tanh(W_c[c_t : h_t]) \tag{3.3}$$

The training process went on for 31 epochs, and the best BLEU score obtained during evaluation

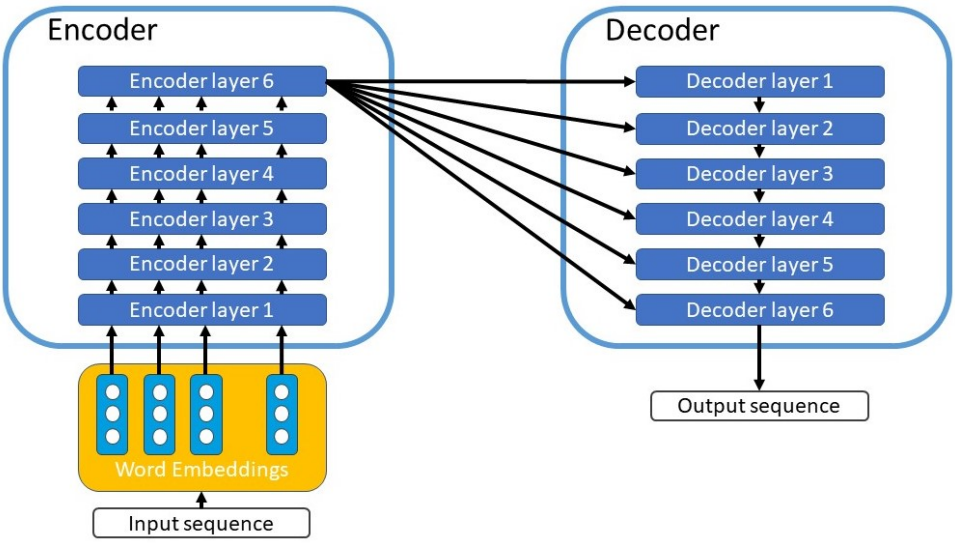

Figure 3.5: The Transformer pipeline, with 6 identical layers stacked on both the Encoder and Decoder.

was 12.06.

To this model we later applied pre-trained word embeddings from GloVe. We chose the word embeddings based on the Common Crawl corpus, with 840 million tokens available, represented with vectors of 300 features. This version of the model ran for 37 epochs, and the best BLEU score was 12.3.

3.2 Transformer-based Solutions

The Transformer model was proposed by Vaswani et al. [46]. It is a parallelized solution that focuses on attention and it is meant to speed up the training process of sequence-to-sequence tasks.

It also consists of an Encoder-Decoder paradigm, where each of these layers is composed by N versions of the same sub-layers, stacked on top of each other, as shown in Figure 3.5, but without sharing weights. Following the original Transformer design, each has N = 6.

Figure 3.5 shows that the model receives the input and output in the form of word embeddings. We will be exploring the case where they are initialized randomly and when they are initialized with the same pre-trained word embeddings described in the previous model (GloVe with 300 features).

To keep information about the location of each word in the input, it uses position encodings. They are added to the input embeddings that will be passed to the first encoder and decoder sub-layers.

The embeddings are fed to the first Encoder layer. Each of the words in the input sentence is individually given to the self-attention layer. The encoding of each position will have information from the rest of the input. In this case, we used a dimensionality of 256 in these layers. It uses a Dropout rate of 10% on the input word and position embeddings.

In the Transformer context, self-attention is supposed to attend to important words in the input sequence while encoding a certain position in the sentence. The inputs of the self-attention layer are read "side-by-side", instead of being sequentially done, like with RNNs.

Self-attention produces three vectors, sharing the same dimensionality of $d = 64$, for every word: Query, Key and Value. They are obtained by multiplying the embedding vectors with trainable matrices. These vectors are then used to calculate the score that will weigh the input sequence.

The score is obtained with the dot product of the Query and Key vectors and then normalized with softmax (Equation 3.4).

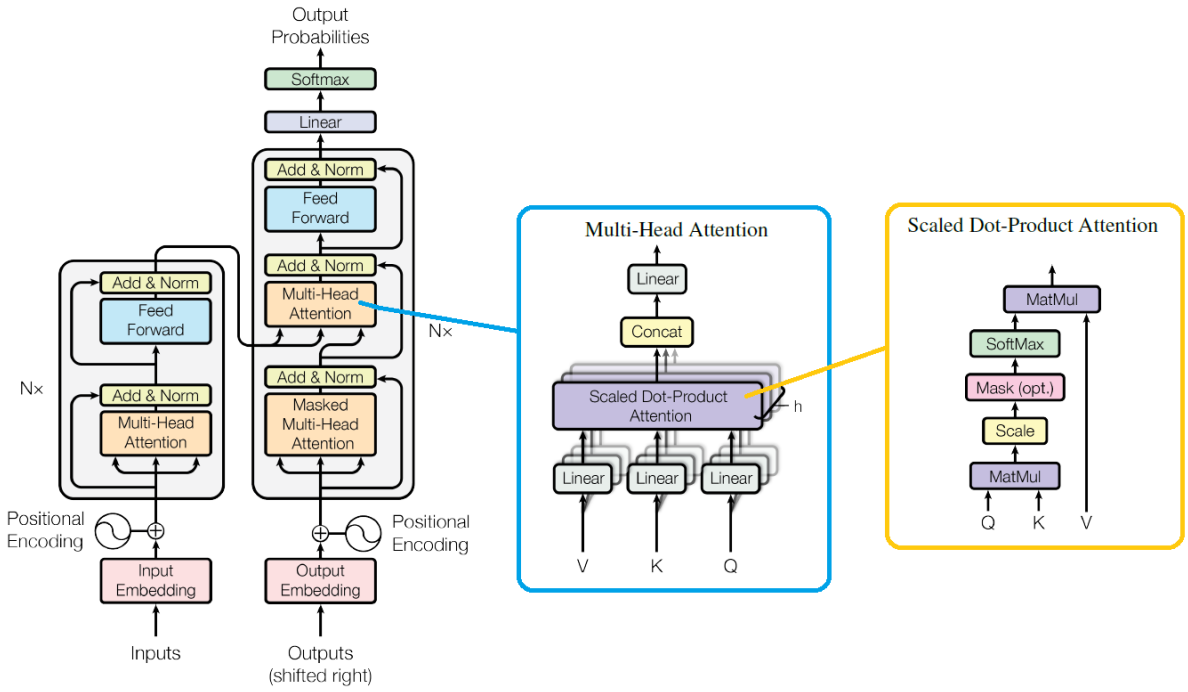

Figure 3.6: The Transformer model, as presented in Vaswani et al. [46]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3.4}$$

One important aspect of this architecture is the Multi-Headed attention, seen in Figure 3.6, where the self-attention sub-layer should be. This means that the model keeps track of more than one set of the matrices mentioned previously (in the original paper, $h = 8$ times the set of Keys, Values and Queries matrices). These are initialized randomly and then trained independently from each other. The attention is calculated for all those "heads", who are then concatenated and scored to give origin to the vector that is passed to the next layer. In Equation 3.5, $head_i$ is computed with Equation 3.4. W^O is also trained simultaneously with the network.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \tag{3.5}$$

Finally, the Multi-Headed self-attention sub-layer returns the sum of the weighted value vectors, which is fed to the next sub-layer: a Feed Forward Network (FFN) with Rectifier Linear Unit (ReLU) activation (Equation 3.6). Every position in the input goes individually through this FFN.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3.6}$$

The output of each FFN is used on the next encoder layer, until it reaches the final one in the stack.

The Decoder has almost the same structure of the Encoder. Each layer possesses the Multi-Headed attention and the FFN sub-layers. Then, it has an intermediate layer between these two, which serves as the attention layer that allows the Decoder to focus on specific parts of the input when decoding. The Multi-Headed attention sub-layer is masked, so that the self-attention cannot attend to future positions in the sequence, and instead looks just at what was already output.

In the decoding phase, the output of the uppermost Encoder layer is given to this intermediate attention sub-layer in every Decoder layer, as seen in Figure 3.5. The main difference between the bottom

and middle attention sub-layers is that the one in the middle uses the values received from the Encoder in the Key and Value matrices. Bottom-up, the Decoder output is passed to the next layer, until it reaches the final sub-layer and generates an output word.

The output of the final Decoder layer is passed to a Linear layer, which converts that output vector into another with the size of the considered vocabulary. With a $softmax$ layer, it turns the values into probabilities of the most probable word for that location. We use beam search, with beam size equal to 5.

Figure 3.6 shows the full Transformer architecture, as shown in the original paper. It shows that every sub-layer is followed by a layer normalization step.

In terms of the used Optimizer, we followed the original choice by Vaswani et al. [46]: Adam, with learning rate scheduling, to variate during the training:

$$\eta(step\_num) = dim_h^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3.7}$$

This means that the learning rate will increase linearly for the first warm up steps and then decrease them. In the equation, $dim_h = 256$, which is the dimensionality of both input and output. The $warmup\_steps = 16000$ represents the first training steps where the learning rate should increase linearly. Finally, $step\_num$ is the current training step.

The Transformer model, with and without pre-trained word embeddings, stopped training at the 20th epoch.

3.3 Dependency Parsing Solution

The final idea presented in this chapter is not related to deep learning, but it should work as a baseline of this research. The technique starts with analyzing the grammatical structure of an input sequence, obtaining the relationships between the words composing a sentence. To improve our results, we skipped the process of padding punctuation with whitespaces and the addition of the start-of-sentence and end-of-sentence symbols, allowing the sentences to resemble its original form.

We use spaCy[2] [16] to obtain such relations. spaCy is an open-source library for NLP in Python and Cython. Among other things it features tokenization, part of speech tagging, named entity recognition and labeled dependency parsing. From each input sequence, the root of the utterance and its respective subjects are extracted, as shown in the following examples:

Example 1: Stephen King is an American author of horror, suspense and fantasy.

Root = is

Root-Lemma = be

Subject = King

Example 2: King has published 58 novels and six non-fiction books.

Root = published

Root-Lemma = publish

Subject = King

With this information, the questions are generated by filling in the blanks of one of the following sentence structures:

- What **ROOT SUBJECT** ?
- What does **SUBJECT ROOT-LEMMA**?

[2] https://spacy.io

The first structure is used when the root of the sentence belongs to the verb "to be", and that verb is used as it is found in the sentence. The bottom structure is used for all the other cases, but the root is reduced to its lemma. The subject that is returned by the dependency parsing tool is used as it is found. Thus, the examples given previously would fit in such structures:
Example 1: What is King?
Example 2: What does King publish?
This solution is quite simple and it does not require previous training, unlike our deep learning models described in the following subsection. While it is guaranteed to produce questions, it is easy to predict that it will come with a few issues on its own.

For once, there is not a large variety of questions that this model can generate, as they all start the same way: *What*. Then, the second structure will always generate questions in the present tense for the third singular person: *What does*, which might produce incorrect questions.

On the other hand, this model is not capable of generating questions based on broken, unreliable input. If a sentence is incomplete or even incorrect, spaCy will not be able to parse the input in order to obtain dependencies in such sentences. There is also an issue with complex sentences, where the automatic extraction of features might result in questions that are either uninteresting or that do not make any sense. All these issues can be observed and are better explained in Chapter 4, where the results will be discussed.

3.4 Summary

This chapter exposed our efforts in developing neural models for QG. We start with an RNN model, and show how it is structured in terms of architecture and components. The Transformer, which is based entirely on attention, is also explained in detail.

While it is globally accepted that pre-trained word embeddings improve text generation tasks, we decided to test these solutions with and without them, so that their influence could be evaluated in the overall performance of these systems.

Finally, we introduced a baseline method that uses dependency parsing and hand-crafted question structures so that we could perform a direct comparison between Neural models and a much simpler solution.

In Chapter 4, we describe the corpus and then we observe how each model behaves. They will be tested and compared against each other in order to assert which is the most appropriate solution for QG.

4

Experimental Evaluation

In this chapter, we show the tests and respective results obtained by each model described in Chapter 3. It is our intention to explore the questions that each model generated and determine which provide the best ones. However, it is not easy to reach such conclusions because there is no right formula to evaluate the quality of questions.

It is important to define what constitutes a good question, especially in the context of this project. As mentioned previously, it is expected that the question is well-formed. According to Faruqui and Das [8], a question can be classified as well-formed if:

- It is grammatical;
- It does not contain spelling errors;
- It is explicitly a question.

We can follow these guidelines when evaluating our questions, but we are also interested in having them relate to the sentences they were based on. Finally, we should take into consideration the original questions associated to each specific sentence and see how far from these are the ones obtained.

Taking all these constraints into consideration, there does not seem to be any available method that could provide a concrete measure to analyse our results. While there are automatic measures widely used to evaluate various Natural Language Processing (NLP) tasks, even in question generation research, most of these cannot accurately capture their true value. As mentioned in earlier chapters, one idea can be expressed in a wide combination of words. Taking into consideration word overlaps between original questions and machine output ones can be useful but also very limited. Thus, to evaluate our models we will take into consideration the most common machine translation metrics.

Finally, we will show how human evaluators perceive a selection of questions. While this method can be very subjective, it can also be very comprehensive. Fluent humans can judge the grammatical quality of a question, while also being capable of easily capturing paraphrasing and synonyms, which is crucial to evaluate the semantic quality of questions. It might also be easier for humans to appraise the relation of a given sentence-question pair.

This chapter is organized as follows: first and foremost, the data used to train and test the models is described in Section 4.1. Then, Section 4.2 exposes an overview of the questions produced by our systems and its main traits and flaws. It is followed by the automatic evaluation in Section 4.3, where

Table 4.1: Example of unanswerable question from SQuAD 2.0.

Context paragraph: "The story focuses on series protagonist Link, who tries to prevent Hyrule from being engulfed by a corrupted parallel dimension known as the Twilight Realm. To do so, he takes the form of both a Hylian and a wolf, and is assisted by a mysterious creature named Midna. The game takes place hundreds of years after Ocarina of Time and Majora's Mask, in an alternate timeline from The Wind Waker."
Question: "What land does Ocarina serve to protect?"
Is impossible: True
Plausible answers: "Hyrule"
Answer start: 67

the chosen metrics are described, and the results are compared not only against themselves but also with other Question Generation (QG) work that was already reviewed in Chapter 2. Then, Section 4.4 describes the human evaluation process and its outcome. Finally, Section 4.5 summarizes our findings.

4.1 Datasets and Experimental Evaluation

In Chapter 2, we explored multiple available candidates that could be used as our dataset. The SQuAD 2.0 was chosen for this research.

For this specific job, however, we start by discarding the unanswerable questions. This is justified with the essence of our goal, which is to create questions that are related to a given snippet of text. The example in Table 4.1 was extracted from the SQuAD 2.0 dataset, and it shows a paragraph and a question that is impossible to answer in that context.

In the example, while the question seems related to the content of the paragraph, and it is syntactically correct, it does not mean anything. In this particular context, the entity *Ocarina* is mistakenly placed where *Link* should be, so this question cannot be seen as correct. If we considered these wrong questions, we could be feeding our models with noise, which could influence the learning process negatively.

Since the data is divided by articles, and each article is separated into paragraphs, the pre-processing of our dataset starts by separating each paragraph in sentences. This was executed using the Punkt Sentence Tokenizer for English sentences, from NLTK Data[1]. For this research we will accept incorrect input sentences. If we expect humans to use a system that generates questions, this system should be flexible enough to accommodate user mistakes. Therefore, we will consider all the sentences, even if the automatic sentence tokenizer fails to capture a complete and correct sentence from the paragraph.

Each sentence is given numerical attributes that identify the position of the beginning and end of the sentence in the paragraph, which is an integer ranging from 1 to the full length of the paragraph. Then, the questions are handled: as mentioned before, the process begins by ignoring the ones identified as unanswerable. Then we produce the pairs consisting of sentence and question. It is easy to locate the sentence that answers a question because the data gives the number that identifies the position of the answer in the paragraph. It can occur that the answer spans multiple segments in the sentence, and by choosing the sentence automatically, it is possible that sometimes the input sequence that is fed to the models is the incorrect one. The chosen sentence and the question will then be prepared for its usage.

[1] `https://kite.com/python/docs/nltk.tokenize.punkt`

The preparation includes the removal of diacritics in the whole text, to ensure that any non-English words did not possess strange codifications. For instance, in the example shown in Figure 2.5, the words “Frédéric François” were changed to “Frederic Francois”. While this does not correspond to the original words, this change still allows the texts to be quite understandable. Another important aspect was punctuation: even though it was kept in the text, it was decided that it should be padded with whitespaces. Finally, all the words were lowercased. All questions and sentences in the set were surrounded by start and end of segment tokens.

Thus, the data used to train the question generation models has the following specifications:

The full train set has 86,820 answered questions, specifically 78,138 questions in the training set and 8,682 questions in the validation one. The vocabulary is composed by all the tokens that appear in the text: words, numerals and punctuation. The training set longest sentence is 325 tokens long, while the longest question has 42 tokens. The train vocabulary, which is a list of all the tokens present in every text sequence in this set (questions and sentences), consists of 73,364 unique words. The vocabulary of the source side has 69,243 unique tokens, while the one on the target side has 36,732.

The test set has 5,927 answered questions, and its longest sentence is 163 words long, while the length of the longest question is 33. The vocabulary of the test set is 15,210 unique words long.

Out of the 73,364 unique tokens that appear in the training data (both source and target), 18,587 tokens do not appear in the GloVe pre-trained word embeddings. This is 25.34% of the tokens that had to be initialized randomly when the models were enhanced with pre-trained word embeddings.

4.2 Output Overview

All the systems, developed with the specifications described in Chapter 3, generated a certain output. Looking at specific details of the results can quickly give us a perception of the usefulness of some models. Here, we will look at some examples of generated questions and then we will select the models, which will be considered for more specific and concrete evaluation methods.

For the following sections, the systems will be mentioned by the following names:

- Dependency parsing – the baseline as described in 3.3;
- RNN-Simple – Sequence-to-sequence RNN-based model described in 3.1.1;
- RNN-Attention - RNN-based system with attention characterized in 3.1.2;
- RNN-GloVe – Attention, but with pre-trained word embeddings portrayed at the end of Section 3.1.2;
- Transformer – The Transformer upgrade, specified in 3.2;
- Transformer-GloVe – The Transformer using the same pre-trained word embeddings mentioned above.

We will focus on each model and explore its outputs. For simplicity, the symbols that represent the start of sentence and end of sentence will be omitted in most examples and only mentioned when relevant.

4.2.1 Dependency Parsing Output

The system based on the dependency parsing technique, which works as the baseline for this research, output questions that are presented in Table 4.2. The table shows five sentences that were given to the model, the original questions associated to each sentence and the produced result.

Table 4.2: Examples of questions produced by Dependency Parsing.

1	Sentence: jacksonville is the principal city in the jacksonville metropolitan area, with a population of 1,345,596 in 2010. Original question: what was the population jacksonville city as of 2010? Generated question: What is jacksonville?
2	Sentence: another important monument, the statue of little insurgent located at the ramparts of the old town, commemorates the children who served as messengers and frontline troops in the warsaw uprising, while the impressive warsaw uprising monument by wincenty kucma was erected in memory of the largest insurrection of world war ii. Original question: who does the statue of little insurgent commemorate? Generated question: What does monument commemorate?
3	Sentence: the tallest building in downtown jacksonville' s skyline is the bank of america tower, constructed in 1990 as the barnett center. Original question: what distinction does the bank of america tower hold? Generated question: What is building?
4	Sentence: prince louis de conde, along with his sons daniel and osias, [citation needed] arranged with count ludwig von nassau-saarbrucken to establish a huguenot community in present-day saarland in 1604. Original question: in what year was the agreement to allow the saarland settlement reached? Generated question:
5	Sentence: approximately 1,000 british soldiers were killed or injured. Original question: how many casualties did british get? Generated question:

The first question appears to be well-formed and relates to the given input. This question does not match the original one, but it seems that the answer is available in the sentence.

While the second question cannot be classified as well-formed, it is easy to understand what it is trying to convey, and looking at the original question, it seems that the same answer could satisfy both questions, as they are asking approximately the same thing.

The third question contains words that appear in the text. However, it does not mean anything: it is not a well-formed question, it does not relate to the original question and it cannot be answered. There are other questions in the output that resemble this one, because of their vagueness. Some of those appear multiple times, like “What is there?” and “What was this?”.

Finally, the last two examples did not generate any questions. Many sentences in the test set could not generate a question. Previously, when we described the idea behind this method, we specified the usage of an automatic dependency parser that focused on the root and subjects of the given sentence. What is happening in these discarded cases could be any of the following:

1. The input sentence is not correct, and the automatic parser fails.
2. The parser succeeds the first step but fails to find elements that fit in the blanks of the designed sentence structures.

Example 4 in Table 4.2 falls into option 1. The sentence of the fourth example has a piece of text that does not belong to any normal sentence: “[citation needed]”. This could be enough to disrupt the grammatical structure of a sentence, when running an automatic parser, justifying the amount of non-generated questions.

Finally, we present a random set of other 20 questions generated with this baseline:

- What does mongols send?
- What is result?

- What does locals frequent?
- What does decline pave?
- What was one?
- What does they have?
- What does kublai pursue?
- What are there?
- What does government achieve?
- What does disobedients choose?
- What does receivers receive?
- What are there?
- What does parliament elect?
- What does problems show?
- What does relativity offer?
- What does brothers run?
- What does system fail?
- What does law have?
- What does percent dispense?
- What is there?

Once again, it should be noted the lack of variety in these questions. When this model generates something, it will always belong in the category of "WHAT" questions, because it follows the specific sentence structure explained in the previous chapter. However, while this model is far from being the perfect question generation solution, it provides acceptable results and it will be considered for the next evaluation phases.

4.2.2 RNN-Simple Output

The next model to be verified is the RNN-Simple one. While Table 4.7 shows that it has delivered questions for all the given sentences, as soon as we analyse the results, we can see that it performed poorly. This model generates the same question for every sentence:

Question: what is the name of the city ' s population ?

These results are questionable, as we are presented with very different inputs (some already shown in Table 4.2), so some variations were to be expected. The repeated generation of this sentence, that appears syntactically correct, over the whole dataset can be related to the characteristics of the model itself, already exposed in the previous chapter: It was a simple encoder-decoder architecture, based on RNNs, with no attention mechanism. We are reporting beam search results with a significant beam size. It was mentioned in the model description that the training process stopped suspiciously early. It is possible that specific parameters were lacking, which could justify these results.

Since this model performs poorly, we will not be considering it for the next stages of evaluation.

Table 4.3: Examples of questions produced by RNN-Attention.

1	Sentence: between the lobes on either side of the mouth , many species of lobates have four auricles , gelatinous projections edged with cilia that produce water currents that help direct microscopic prey toward the mouth . Original question: what are auricles ? Generated question: what do the three blocks of the lobes of the lobes of ?
2	Sentence: the logic for this is believed to be that the prime number intervals between emergences make it very difficult for predators to evolve that could specialize as predators on magicicadas . Original question: what is the logic behind the cicadas prime number evolutionary strategy ? Generated question: what idea is used to evolve species that can be used to evolve ?
3	Sentence: last year , 121 , 200 tonnes of pears and 270 , 000 tonnes of tomatoes were produced . Original question: how many tonnes of tomatoes does victoria produce ? Generated question: how many tonnes of plants would be produced from a year ?
4	Sentence: the rhine gorge between rudesheim am rhein and koblenz is listed as a unesco world heritage site . Original question: what is the rhine gorge listed as ? Generated question: what is listed as the unesco world heritage site ?
5	Sentence: approximately 1 , 000 british soldiers were killed or injured . Original question: how many casualties did british get ? Generated question: how many british soldiers died during the attack ?

4.2.3 RNN-Attention Output

The RNN-Attention model returns a question for every given sentence in the input data. Table 4.3 shows some of the results obtained with this system, sharing the same structure used in the previous table.

In the first two examples, it is noticeable that the model sometimes delivers questions that are not well-formed, most notably containing cycles within the text. In the first example, there is the consecutive repetition of the segment “of the lobes” at the end of the sentence, while in the second example we see “used to evolve” twice, separated by another text segment. Nonetheless, the questions possess words (or concepts, like “evolution” being changed to “evolve”) from both source and target texts.

The third example starts the same way as the original question, as if asking the same thing. Yet the idea of “pears and tomatoes” was changed to “plants”, allowing the question to be different, but still feel related to the given input sentence. The last section of the question is posed with an unnatural, even incorrect way but it is easy to understand what it is being expressed and the answer can be found in the text.

The next example appears to be very successful. Not only is the question well-formed, it is very close to the content of both sentence and gold question. In fact, the answer to the model-question can be found in the text of the original question.

The sentence in the final example will be observed for every other model in the present analysis. We cannot generalize their quality by comparing them with one single sentence, but we might have some insights when looking at their differences. This well-formed question is clearly related to the input sequence. It is interesting to see that the model adds a fragment to the question that cannot be found in the text: “died during the attack”, which is not mentioned before. It is also important to notice that the replacement of the concept “being killed” by the verb “to die” proves to be a successful conversion from passive to active verb forms. In this specific case, the reference to soldier causalities (death and injuries) present in the input sentence is shortened, as the generated question only mentions soldier deaths. While the overall comprehension of the question is not affected by this replacement, the correlation to the input in a way that the generated question can be answered by the information present

in the sentence may depend on the evaluation criteria.

With this model, there is a variety of questions, that go beyond the “WHAT”-type that we saw repeated with the dependency parsing model, as it is shown in this random sample of questions that the model output:

- what kind of occurs at the double digital tracks of the double arena ?
- what is the magnitude of the sichuan species ?
- what did the states that created to the resource of french and british power ?
- what is a huge need for free workers ?
- what defined the " regular legislative " ?
- what was the concept of honor first introduced ?
- what is the charge of the human interference to ?
- when did the extinction of the mesozoic end ?
- what is the name of a village at modern day that is a village at modern day this century ?
- what is another name for the international data network ?
- what do the people identify with ?
- where did the new mail east times raise their tax tax ?
- what do the three blocks of the lobes of the lobes of ?
- what was the general order of the central india company ?
- how many tonnes of plants would be produced from a year ?
- who led the development coalition ?
- what did john headline in 1872 ?
- what animal is the compositions composed of ?
- what do the two ways of the type of paper uses by testing ?
- what do these bacteria lack ?

4.2.4 RNN-GloVe Output

The examples shown in Table 4.4 correspond to the ones produced when the RNN-Attention model was enhanced with GloVe pre-trained word embeddings. Starting with the British soldiers example, this model also generated a well-formed question that shares the answer of the target question. The questions both models produced are very similar, but here it did not introduce new unknown information to the text like we saw in the RNN-Attention model. Instead, it uses the words “were killed”, just like the source text. We do observe the usage of the word “troops” instead of “soldiers”.

The second example appears to have a confusing structure, making it difficult to comprehend what is being asked, just by itself. Nonetheless, it seems relatable to the content of both sentence and original question, and we can assume that they share the same answer: four. It also uses quotation marks correctly, while changing one of the words that are surrounded by them (“places” to “cultures”).

Table 4.4: Examples of questions produced by RNN-GloVe.

1	Sentence: approximately 1,000 british soldiers were killed or injured. Original question: How many casualties did british get? Generated question: how many british troops were killed ?
2	Sentence: in aristotle ' s view , the terrestrial sphere contained four elements that come to rest at different " natural places " therein . Original question: how many elements did aristotle believe the terrestrial sphere to be made up of ? Generated question: the sphere shows how many elements to appear at different " natural cultures " ?
3	Sentence: today , this acceleration due to gravity towards the surface of the earth is usually designated as and has a magnitude of about 9 . 81 meters per second squared (this measurement is taken from sea level and may vary depending on location) , and points toward the center of the earth . Original question: where was the measurment for the standard gravity on earth taken ? Generated question: what form is it taken by the earth ' s surface ?
4	Sentence: in actuality , warsz was a 12th / 13th - century nobleman who owned a village located at the modern - day site of mariensztat neighbourhood . Original question: what did warsz own ? Generated question: in what century was there a village ?

The next example shows an understandable question that, while not perfect, it is noticeable the correct usage of the possessive case, with the apostrophe. It has words that appear in the source, like “earth” and “surface”, but the answer to this question is not present in the text, as there is no mention to the “earth’s form” in the sentence.

The final example of the RNN-GloVe model seems well-formed, with content similar to the input sentence. It is a vague question, because there have been many villages for many centuries, but we can assume what is being transmitted by the question after reading the text, easily finding the answer there.

- what is the first - word sea determined ?
- what form is it taken by the earth ' s surface ?
- what did waitz attempt to create ?
- what kinds of people are needed in high wages ?
- when did the phrase " rule " occur ?
- who claimed for a license for the libyan revolution ?
- what are the positive charges in all eukaryotes ?
- what type of climate happen the ability to spread out across the height ?
- why is the most accurate critic that uses a fixed set of rules to determine what ?
- what is the purpose of the international data communications station ?
- what ' s the common end of the common end of oxygen on earth ?
- how could the sun avoid to avoid marine tax positions ?
- what are cilia ?
- what was the name of the building in the secretariat of the civil council ?

Table 4.5: Examples of questions produced by Transformer

1	Sentence: approximately 1,000 british soldiers were killed or injured. Original question: how many casualties did british get ? Generated question: how many soldiers were killed in the war ?
2	Sentence: the university is also home to the university of chicago press , the largest university press in the united states . Original question: what is the name of the largest university press in the u . s . ? Generated question: what is the name of the largest newspaper ?
3	Sentence: spreading throughout the mediterranean and europe , the black death is estimated to have killed 30–60 % of europe ' s total population . Original question: how much of the european population did the black death kill ? Generated question: how much of the population is there in the world ?
4	Sentence: using handheld gps devices and programs like google earth , members of the trio tribe , wholive in the rainforests of southern suriname , map out their ancestral lands to help strengthen their territorial claims . Original question: what do tribes use google earth and gps for ? Generated question:

- how much would they have of neptune ' s bomb ?
- who led the high strike ?
- which invention made a paper protected by the royal society ?
- what is the name of the state blooming flower ?
- where was the popular war fought ?
- in different alleles , the term , lack and what other cultural activity led to the offer of the population ?

In this random sample taken from the output of the RNN-GloVe model, it is possible to see a variety of question types, ranging from “WHAT” to “WHO” and “HOW”, among others. We can also see a few questions with cycles or strange structures.

4.2.5 Transformer Output

Some results obtained with the Transformer model appear in Table 4.5. Once again, the question about the British soldiers is analyzed. It is very similar to the ones presented before: it focuses on asking how many were killed. Just like the RNN-Attention model, there is information that is not in the source or in the target texts, introducing the idea of being killed “in the war”. Once again, “war” is very close to the essence of the sentence, and depending on the relation criteria, it might not disrupt the meaning of the question in this particular case. So far, this is the only example that does not specify the nationality of the soldiers, dropping the word “British”.

The next example seems well-formed too. Although both questions seem very similar, the Transformer one is significantly more vague than the original. It does not specify the country (“U.S.”) and it drops the word “university”. The word “press” was replaced with “newspaper”, which are similar concepts, but not quite synonyms of each other. It is still easy to understand what the question is conveying.

The third example has both questions starting the same way, with the generated one omitting the word “european”. Then, the rest of the sequences diverge, as the original specifically mentions the black death killing, just like in the source sentence. When taking the content of the example into consideration, it seems that the generated question is asking how many people are left, adding at the end the concept of

“the world”, instead of restricting the black death to Europe like in the sentence. Even if it is a well-formed question, it seems quite odd and even vague, given the input.

The final example is one of the cases where the model did not output a question. Out of this small sample, it has the longest sentence, and that could be one of the reasons for the lack of output question.

The next questions are a random sample of the resultant questions of the Transformer model:

- what is the name of the area of norfolk island ?
- what was the name of the ottoman empire ?
- what is the term for " black " ?
- what is the name of the constitution ?
- what was the first step in the book of species ?
- what is the term used to refer to the body ?
- what type of climate has been found in the triassic ?
- what was the name of the old dominican order ?
- what was the name of the company that the u . s . ?
- what is the name of neptune ’ s life ?
- what was the name of the first public service ?
- how many species of species are there
- what was the name of the government ’ s government ?
- how many people were killed in the war
- what was the name of the royal institute of the city ?
- what is the term used to describe the sun ?
- what is the name of the buddha ’ s life ?
- who was the leader of the enlightenment ?
- what was the name of the group of norfolk island ?
- what was the name of the invasion of france ?

The most notable thing about this sample is that most of the “WHAT” questions are followed either by “was the name” or “is the term”. Nonetheless, there are other questions that start with “How” and “Who”, showing some variety of question type. Some of the questions do not possess a punctuation at the end of the sequence.

Table 4.6: Examples of questions produced by Transformer-GloVe.

1	Sentence: approximately 1,000 british soldiers were killed or injured. Original question: how many casualties did british get ? Generated question: how many people were killed in the french ?
2	Sentence: he states that maps " contributed to empire by promoting , assisting , and legitimizing the extension of french and british power into west africa " . Original question: who besides the british colonized africa ? Generated question: what was the name of the british isles ?
3	Sentence: the concept of duty was first introduced by watt in order to illustrate how much more efficient his engines were over the earlier newcomen designs . Original question: who invented the notion of a steam engine ' s duty ? Generated question: what was the name of the first digimon ?
4	Sentence: a deterministic turing machine is the most basic turing machine , which uses a fixed set of rules to determine its future actions . Original question: what is generally considered to be the most basic iteration of a turing machine ? Generated question: what is the term used to describe the " ?
5	Sentence: a job where there are few able or willing workers (low supply) , but a large need for the positions (high demand) , will result in high wages for that job Original question: what type of wages result from jobs where there is low supply but high demand ? Generated question:

4.2.6 Transformer-GloVe Output

The results of the final model are in Table 4.6. Out of every “British soldiers” example that have been analyzed so far, this is the only one that should not be considered well-formed, and thus, the weakest question. It starts with "how many", like the others, including the original question. Then, the replacement of “British soldiers” was generalized to “people”, decreasing the relation between input and output. Finally, it adds new information, which had already happen, but this time it is very specific and incomplete, as something appears to be missing after “french”. The short sentence does not provide enough information to infer if something French is involved in this context, so there is a considerable possibility that the model added untruthful information to the question.

Examples 2 and 3 are very similar examples. They are well-formed, but quite different from the original questions. Both start with “what was the name of the”, and then they add a word that appears in sentence (“british” and “first”, respectively). In the case of the example 2, the word “british” also appears in the target question. Then, in both cases, the question has one single last word, that has nothing to do with the sentence or the original question.

The final examples are unsuccessful ones. Example 4 produced a question that is incomplete and misuses the quotation marks. It has barely any resemblance with the sentence or original question, aside from the fact that both start with “what is”. Example 5 did not produce a question, just like the previous Transformer model.

- what is the main source of the sahara ?
- what was the name of the political party ?
- what is the name of the earliest known as ?
- what is the range of the bacteria ?
- what was the name of the city ' s second studio ?
- what is the name of the earth ?

- what was the name of the bbc ' s first satellite ?
- what is the term for a species ?
- what did the u . s . rule ?
- how many people were killed in 2010 ?
- what was the name of gaddafi ' s father ?
- in what year did von neumann publish his theory ?
- what is the average number of the sahara ?
- what was the name of the battle of the battle ?
- what is the name of the west of the native american population
- what is the term used to describe the word " ?
- what was the name of the chinese ?
- what was the name of the battle of china
- who was the founder of the sun ?
- when did the war begin ?

In this sample of 20 random questions generated by the Transformer-GloVe model, there are many "What is the name"-questions and its variations. Other question types going beyond "WHAT" can also be seen. Even though most of the Transformer-GloVe examples analyzed before had a somewhat negative appraisal, we can see many examples that seem to be successful in this set.

4.2.7 General Overview

It should be reinforced that the previous examples are a small sample of the generated questions. While they manage to show some of the capabilities of the studied models, they do not represent the whole resultant set, that contains thousands of questions. Therefore, we will be exploring other metrics to analyze them from different perspectives.

Table 4.7 shows the amount of questions obtained when the models were given the test set as input. The first column has the names of each model and the second shows the percentage of questions generated, given the 5,927 input sentences of the test set. It is clear that Dependency Parsing, Transformer and Transformer-GloVe models did not generate all of the expected questions, with Dependency Parsing being the one producing the least number of questions.

Since we were able to eliminate the RNN-Simple model based on question repetition, we will analyze this trait in the other models. In Table 4.8, the second column shows the value of unique input sentences

Table 4.7: Percentage of generated questions, based on input.

System Name	% of generated questions
Dependency Parsing	49.89
RNN-Attention	100.00
RNN-GloVe	100.00
Transformer	84.42
Transformer-GloVe	88.35

Table 4.8: Number of questions and sentences repetitions in the data used and produced by each model.

System Names	Repeated Sentences	Sentence Repetitions	Repeated Questions	Question Repetitions
Dependency Parsing	21.95 %	30.71 %	21.75 %	40.45%
RNN-Attention	22.10 %	31.20 %	22.12 %	31.38 %
RNN-GloVe	22.10 %	31.20 %	22.10 %	31.40 %
Transformer	22.20 %	31.61 %	16.92 %	67.71 %
Transformer GloVe	22.82 %	31.50 %	14.49 %	74.03 %

that appear more than once in data (and, in this case, they can be associated to different target questions). The fourth column shows unique questions that appear at least twice in the output of our models. This means that if we were including RNN-Simple, it would only have 1 question (0.017%) in the fourth column. The number of repeated sentences is quite consistent between models, which makes sense because they are considering the same input sentences. The sentences that did not generate questions are not considered in this table.

The third and fifth column show the number of lines in the data (sentences and questions, respectively) that are repetitions. So, in the Simple model, the rightmost column would show 100%, as it always generated that one sentence for the whole input set. Once again, we can see the column related to the sentences with very similar values throughout the different models, because of the input coherence.

The Dependency Parsing and both RNN models show similar values in the "Repeated" columns. It is possible to infer that the model will behave consistently and produce the same question whenever it is fed with the same input sequence. This behaviour is further emphasized for RNN-Attention and RNN-GloVe in the "Repetitions" columns, as both columns show similar values. Here, the Dependency Parser model reveals to have more repetitions. This was already mentioned when the inputs were analyzed, as sometimes the parser chooses the word “there” as subject of the root and generates questions like “What is there?” for very different input sequences.

In the Transformer cases, we can assume that they are having a more erratic behaviour. In the "Repeated" columns of Table 4.8, the fact that we have a smaller value in the number of unique questions, in comparison to the number of sentences, seems to indicate that when a text is given to the model, it will not generate the same sentence every time. This is not necessarily a bad thing, as it could be an indication of variety. However, the "Question Repetitions" columns shows that more than half of the results are repetitions of the same questions. As seen in both Transformer models, there are many questions that are a variation of “what was the name of” or “what is the term for”, sometimes followed by an apparently random term. It seems that these models are not capable of generating a grand variety of questions.

This section gave some insights on the overall performance of each of the models and allowed us to exclude one: RNN-Simple. We will now explore the results obtained by automatic evaluation.

4.3 Automatic Evaluation

As mentioned before, in Section 2.3, most previous work on QG uses automatic metrics to evaluate the performance of models and the quality of the produced questions. They are easy to obtain computationally and are widely available.

The metrics reported in this section were obtained using the nlg-eval evaluation package by Sharma et al. [41]. Aside from the metrics described before, we will also report the values obtained for the CIDER metric:

Consensus-based Image Description Evaluation (CIDER) [47] is a protocol originally created to

Table 4.9: BLEU-{1,2,3,4}, METEOR, ROUGE-L and CIDER scores for every considered system.

System Names	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDER
Dependency Parser	0.8909	0.1384	0	0	4.5262	5.2213	5.1739
RNN-Attention	39.3362	25.268	13.1763	7.9029	29.3371	40.4695	43.0177
RNN-GloVe	39.1382	24.9798	13.0097	7.8027	29.2349	40.1844	41.9173
Transformer	38.8333	24.6561	12.0306	7.0544	28.5332	40.0584	18.7706
Transformer-GloVe	37.7911	23.829	11.3757	6.5936	28.1891	39.6555	15.3649

evaluate candidate sentences that could describe a certain image, regarding a given set of image descriptions. This measure is based on Term Frequency Inverse Document Frequency (TF-IDF), as it is computed for every n-gram and then the average cosine similarity between the candidate and references, which manages to capture importance and salience. This metric uses higher order n-grams to capture grammatical properties and semantics. In this specific case, this metric is used not based on many references for one candidate, but on the many sentences that constitute the corpus.

First, we analyze the scores of the automatic metrics based essentially on n-gram overlaps. The first column shows the model names, while the following report the scores from BLEU-1 to BLEU-4. The rightmost columns present the values obtained for the metrics METEOR, ROUGE-L and CIDER.

Starting with the BLEU scores, it is quite visible in Table 4.9 that the baseline model is the one who returns the worst values, which are barely higher than 0.

The next observation is that all other models are very close to each other. The difference between RNN-Attention and RNN-GloVe is practically non-existent, but it seems to indicate that the model suffers a little with the usage of pre-trained word embeddings, as observable by the small decrease, never higher than 0.4, in the score.

The same goes for the Transformer models, but the distance between them is slightly more noticeable. The model that does not use pre-trained word embeddings returns a higher score, with a difference of, at most, 1.1 in the score.

It seems that the models based on Recurrent Neural Networks (RNNs) perform better than when using the Transformer, showing a small decrease in the score. In line with these results, it appears that the RNN-Attention model is the best out of every question generation solution. However, it is important to emphasize that BLEU is not a precise measurement, and that this small differences in scores are inconclusive when it comes to the true quality of the question generation models.

When looking at the remaining scores, METEOR, ROUGE-L and CIDER, it is clear that once again, the baseline is the one coming behind every solution, with values close to 5 for all the metrics, indication of its poor behaviour.

METEOR presents scores that resemble the BLEU results, with the RNN-based models having similar values, and the Transformer ones also being quite identical between them. In both cases, the best scores are obtained without using pre-trained word embeddings. Between the two options, it seems that the question generation process performs better when using the RNN models, but the discrepancy is not that relevant, as the difference is smaller than 1. According to METEOR, it is the RNN-Attention model that achieves best performance.

The same conclusions can be reached when analyzing ROUGE-L scores. Here, the difference between the Transformer and RNN models is even smaller, but even so, it is still the RNN-Attention that performs best.

So far, the results seem consistent, but also very uncertain because every model presents similar scores. Yet, when we look at CIDER, we see a significant drop from the RNN-based solutions to the Transformer ones as their scores are less than half of the first ones. This result is not surprising: CIDER is heavily based on TF-IDF and it will favour uncommon n-grams that match the reference question,

Table 4.10: Comparison between the highest scores obtained by our systems and other existing QG solutions.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE
Du et al. [7]	**43.09**	**25.96**	**17.50**	12.28	16.62	39.75
Zhou et al. [51]	-	-	-	13.29	-	-
Yuan et al. [50]	-	-	-	10.50	-	-
Harrison and Walker [12]	-	-	-	14.39	19.54	43.00
Kim et al. [18]	-	-	-	**16.20**	19.92	**43.96**
Wang et al. [48]	-	-	-	10.80	-	-
Our best system	39.34	25.27	13.17	7.90	**29.34**	40.47

while reducing the importance of frequent ones. As mentioned before, the Transformers generate many questions that are the repetition of the same sequence (i.e. "What was the term for. . . ") and then a somewhat random word. It was also proved that most of the resultant output was repetitions of a certain set of questions. This could be influencing the weighing of n-grams in the data, which justify this variation.

Before examining the metrics based on word embeddings, these results are compared against the ones reported in the QG literature, mentioned before in Section 2.4. These values can be found in table 4.10, as well as the highest scores obtained by our models, which was RNN-Attention.

Most of the solutions reported BLEU-4, the smallest going as high as 10.5. Our best reported BLEU-4 is 7.9, by RNN-Attention. It seems to be quite far from the expected performance. The only model to report other BLEU metrics is Du et al. [7]. The scores for BLEU-2 are very close to the ones that our models obtained, as their best system reports 25.96 and our best score is 25.57. The other values of BLEU have a difference of more than 3. Regarding ROUGE, our systems only manage to outperform [7], but it is still almos 4 points behind the highest value. Despite this, our METEOR scores are almost the double of the 16.62 reported by Du et al. [7], and still approximately 10 points above the current best model, by Kim et al. [18].

This inconsistence between the BLEU/ROUGE values and METEOR could be related not only to question similarity, between reference and our generated hypothesis, but also with the way that these metrics were computed. While this is not a precise comparison, since metrics that rely on n-gram tokenization may differ according to the used tools, it can still give some degree of resemblance with the state-of-the-art.

Finally, we analyze the metrics that are based on word embeddings, where:

A Skip Thoughts

B Embedding Average

C Vector Extrema

D Greedy Matching Score

With these methods, there is an increase in the values of the baseline, not coming too far from the other models. It is still the one providing the worst results for every metric.

As with most of the previous methods, the models do not show a relevant disparity between values. RNN-Attention still delivers the highest scores, and the ones with pre-trained word embeddings the lowest.

With this analysis we could only reach two important conclusions. First and most obviously is that the baseline model is the worst solution for question generation, as it shows the smallest values for every tested metric. Then, it was confirmed with CIDER that the Transformer models are very repetitive in

Table 4.11: The scores for (A) Skip Thoughts Cosine Similarity, (B) Embedding Average Cosine Similarity, (C) Vector Extrema Cosine Similarity and (D) Greedy Matching Score.

System Names	A	B	C	D
Dependency Parser	30.6844	54.7902	38.8988	65.3319
RNN-Attention	95.5461	88.9232	51.6036	78.8058
RNN-GloVe	95.3897	88.7582	50.9495	78.4806
Transformer	95.4866	88.2715	48.8883	78.6063
Transformer-GloVe	95.4431	88.1888	47.8914	78.2778

terms of output questions. The RNN-Attention model seems to be the one with better performance, but only by a very small, almost insignificant, margin. Thus, the automatic evaluation results are uncertain as we were not able to confidently identify differences between solutions. Therefore, we will proceed to human evaluation, while dropping the baseline.

4.4 Human Evaluation

To further inspect the capabilities of these models, we perform human evaluation on the generated questions. Given that we are trying to assert which of these models is more suitable to this task, we asked 5 fluent English speakers to observe our questions.

We chose 20 sentences from the test set to develop the survey. These sentences were selected so that every model had generated a question using it as input. The survey was executed on Google Forms. Before starting the survey, the evaluators were informed about the purpose of the study and the questionnaire, with the following text:

The following survey belongs to a research in the field of Natural Language generation, more specifically Question Generation. We developed systems that, given a sentence, will generate a question. Our goal is for our questions to be correct, which means that we are expecting acceptable syntax and grammar according to the English language. We also intend for our questions to be relevant, that is, related to the context of the given sentence.

Every text sequence in this survey will be lower-cased (for instance, "World War II" will be "world war ii", and this is correct in this context). Punctuation might appear surrounded by white spaces, and that is not a problem either (for instance: "what (how , when) ?" instead of "what (how, when)?").

Your function is to analyze 20 sentences and respective generated questions. You will be asked to state if you think the question is correct and relevant, according to the definitions above. Each pair should be evaluated independently.

Then, we ask you to sort that same set in order of preference: Mark with 1 the most adequate in that context, while 5 represents the one you find the least suitable. Please classify every question, even if none seems correct or relevant to you.

Your feedback is very important for the evaluation of our systems. Thank you very much for taking the time to answer this survey.

The users are informed about the idea of correctness and relevance in the context of our research. Then, they are presented with the interface shown in Figure 4.1. Every sentence has a set of five questions, where one is the Gold reference question, while the others are from the RNN-Attention, RNN-GloVe, Transformer and Transformer-GloVe models. Each question is placed randomly, and identified with a number from 1 to 5. The placement order in each example was saved separately. For each question, the user must read and consider if it is correct or relevant, regarding the sentence.

Before proceeding to the next example, they must order every question, where the first is the most suitable in the terms of this research, by filling the table shown in Figure 4.2.

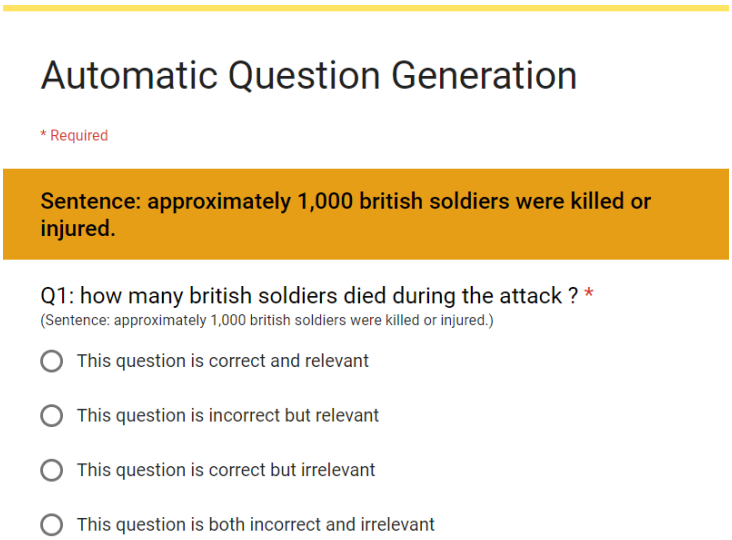

Figure 4.1: Survey interface: correctness and relevance.

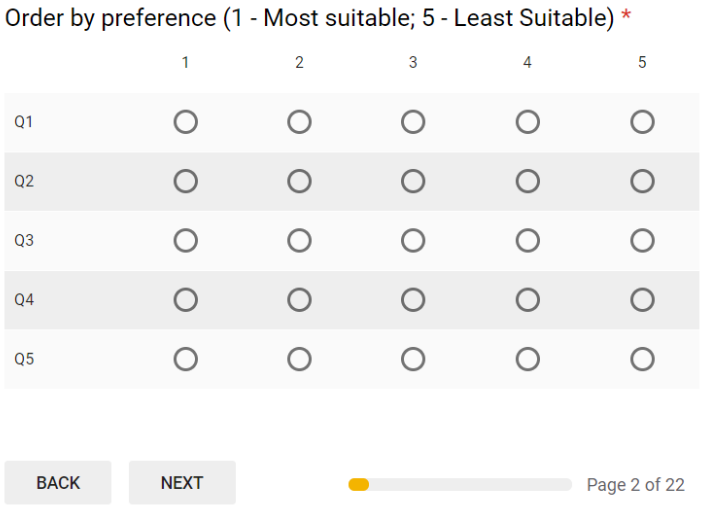

Figure 4.2: Survey interface: preference.

In Table 4.12, the second column presents the number of times the questions were evaluated as correct. The fourth does so, but with the classification of relevance. Both third and fifth columns show the amount of times all the human evaluators agreed on the classification of the 20 questions of each system, in terms of correctness and relevance, respectively. For instance, if four human evaluators considered every question by the same system incorrect, and the other thought every question was correct, it would show 0% in C-Concordance. On the other hand, if every evaluator classified every question, generated by the same system, as relevant, it would have a 100% score in R-Concordance.

Out of the reference questions, presented in the second line of the table, 76% seem to be correct and relevant an approximate amount. Human evaluators agreed approximately half of the time on their answers. This could be related with what was mentioned previously in Section 4.1: automatic tokenization of sentences could have selected the wrong section of the text for said question. The example with most concordance on its irrelevance shows this:

- **Sentence:** he embarked on a plan for the 1758 campaign that was largely developed by loudoun .

Table 4.12: Percentage of correct and relevant answers, as selected by users, and their agreement on the classification.

Question origin	Correctness	C-Concordance	Relevance	R-Concordance
Gold	76 %	45 %	77 %	50 %
RNN-Attention	66 %	25 %	31 %	45 %
RNN-GloVe	64 %	40 %	36 %	35 %
Transformer	83 %	60 %	15 %	70 %
Transformer-GloVe	67 %	55 %	7 %	70 %

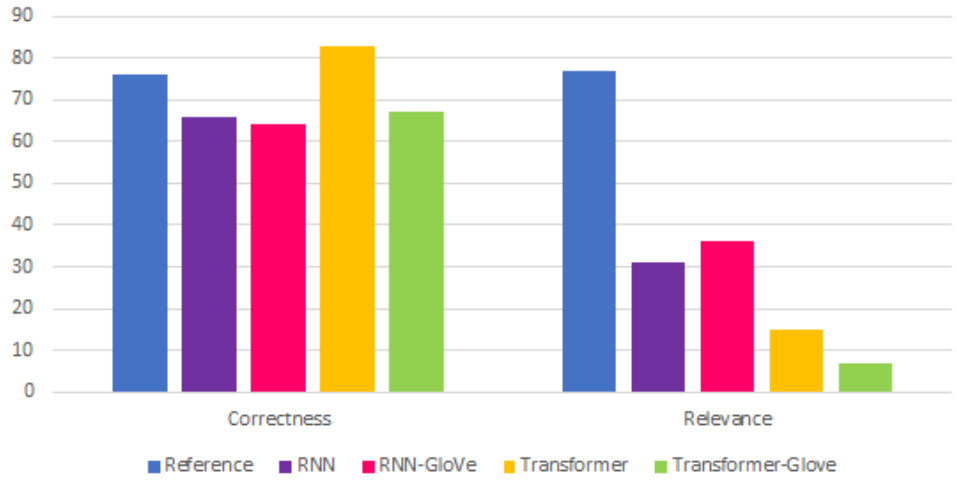

Figure 4.3: Percentage of correct and relevant answers, as selected by users, and their agreement on the classification.

- **Question:** who did abercrombie replace as commander in chief ?

In terms of relevance, the human-generated questions achieved by far the best value.

Regarding the values of the QG solutions, it is noticeable that the correctness of all the models is above 50%, which means that most of the time, users think these questions are correct. However, in terms of relevance, the results are very low.

The RNN-Attention model obtained the best scores in the previous evaluation step, by an almost insignificant margin. In this case, however, its correctness only manages to surpass the RNN-GloVe model, but once again the difference is not very relevant. In respect to relevance, it achieves a better result than both Transformer models. It is the model with lowest concordance in correctness out of all the considered questions and it is still low in the case of relevance: in general, users do not seem very confident and do not agree about these questions.

While RNN-GloVe has the lowest level of correctness, it achieves the highest value of relevance of all the solutions. Similar to the RNN-Attention, the values of concordance are very low.

The Transformer questions are the ones that are considered the most correct, surpassing even the Human questions. Relevance, on the other hand, drops to 15%, which reinforces the idea of good random questions, which was already concluded in the previous sections of the this chapter. The level of correctness in Transformer-GloVe decreases to valus similar to the RNN-based solutions. Relevance is very low. In terms of concordance, users agree in most of the examples of both Transformer solutions.

This analysis proves that, while models seem very similar when passing through automatic evaluation, they show differences for the human eye. The questions produced by RNN-based solutions are ambiguous in terms of correctness and relevance. Even so, they seem less correct than the Transformer, but more similar to the given sentences. The Transformer models generate correct, but irrelevant

Table 4.13: Percentage of preferred questions, where 1 is considered the best and 5 the worst.

System Name	1	2	3	4	5
Gold	61 %	21 %	7 %	3 %	8 %
RNN-Attention	9 %	29 %	22 %	23 %	17 %
RNN-GloVe	16 %	24 %	35 %	10 %	15 %
Transformer	13 %	13 %	19 %	27 %	28 %
Transformer-GloVe	1 %	13 %	17 %	37 %	32 %

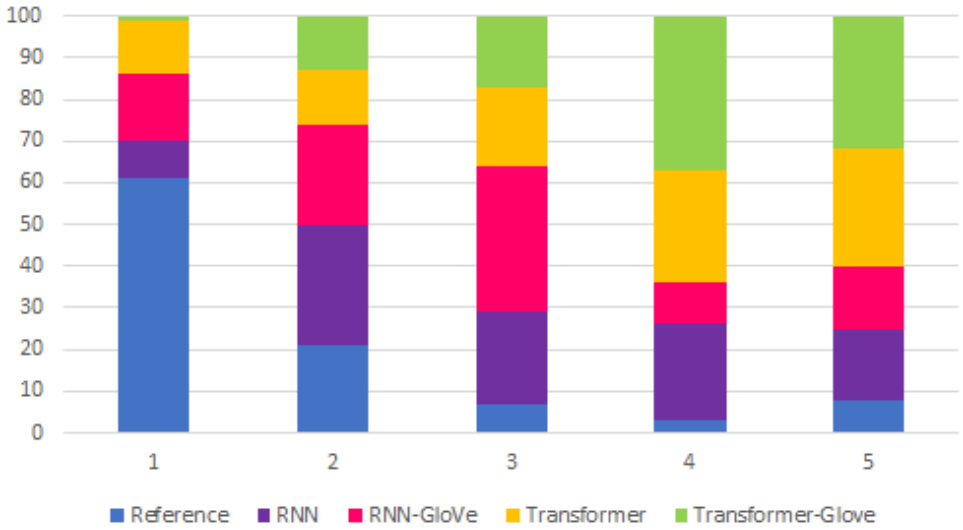

Figure 4.4: Percentage of correct and relevant answers, as selected by users, and their agreement on the classification.

questions.

Regarding preference, the reference question is the one that is preferred most of the time. It is placed the least amount of times in the bottom three classification places (18%).

Out of the solutions, RNN-GloVe is the one that is classified most of the times as the best question. The RNN-based solutions are both selected approximately 40% of the time as the two top-most questions. The Transformer is selected as top 2 question 26%. There is only one instance where one user selected the Transformer-GloVe as the best question.

The question that is selected as the worst most of the times is generated by the Transformer-GloVe, that is chosen for two bottom classification values 69%. Next comes the Transformer, showing more positive evaluations, but not that higher.

Figure 4.4 allows a direct comparison between results. It appears that, even though users considered the Transformer model the most correct, relevance weighed more in the users preference, as the Transformer models are the least preferred. Both RNN-based solutions scored similarly, in this case with RNN-GloVe being better classified most of the times: this could also be related with the weight that user put on relevance.

4.5 Discussion

In this chapter, we investigated the experiments executed with the QG models described in Chapter 3. In a first analysis, the RNN-Simple model was eliminated. This goes accordingly with previous research on Neural Network (NN), since attention mechanisms are known to improve drastically the performance of sequence-to-sequence tasks.

The other RNN-solutions generated interesting questions. In terms of automatic evaluation, the results were close to the ones obtained in the literature, even surpassing those in certain metrics. While they could not achieve the same level of correctness or relevance that the reference questions did, they were still positively evaluated by humans.

Both solutions saw a decrease of quality, specially in the automatic evaluation section, when using the pre-trained word embeddings by GloVe. This is something that goes against what would be expected. However, upon further inspection, we saw that there were many tokens that had to be initialized randomly, since there was no word embedding for that token. One can conclude that 25% of unknown tokens being randomly initialized can damage the performance, and that the number of matches has to be higher for these embeddings to be useful. Other issue with word embeddings could be the initialization: maybe using values of similar words instead of random values would penalize less the training process.

Then, we concluded that the Transformer models were incapable of generating questions for every sentence in the data. They are also repetitive and the questions were not relevant, as in being related with the source sentence. Even so, the questions were considered correct by human evaluators and the automatic evaluation placed this solution higher than the baseline. This architecture is considered the state of the art in Machine Translation, and there were high expectations for its performance in this QG project. The fact that it is lacking could be associated with various implementation aspects related with lack of experience with this novel concept. Maybe, with further inspection, certain hyperparameters could be tuned to increase its capabilities and achieve better results in QG.

5

Conclusions

5.1 Contributions

In this work, neural techniques were used to generate natural language questions from a given text sequence as input. The studied models comprising Recurrent Neural Networks (RNNs) with Attention mechanisms, like the state-of-the-art of this specific task, and the Transformer architecture were able to generate distinct questions from the ones present in the training dataset.

After automatic and human evaluation, the direct comparison between RNNs and the Transformer is not conclusive given the large number of questions generated by both models that often did not share words with the reference questions, hence leading to low BLEU scores. Manual inspection of the results showed several problems in terms of textual coherence (e.g., generations with repeated words), but nonetheless many of the questions that were generated were correct, relevant and even interesting.

The automatic evaluations show results close to the state of the art in Question Generation (QG) and human evaluators gave a positive score in terms of correctness and preference.

These evaluation results show that the Transformer architecture can be a promising alternative to the state-of-the-art models used in question generation and the further development of its implementation could lead to very accurate question generation tools in the future.

5.2 Future Work

With further efforts over this introductory work on QG with neural networks, we believe that there is still great room for performance improvement of these models that could be achieved, for instance, by

- Using other approaches for pre-trained word embeddings of unknown tokens. Instead of initializing them randomly, using a similarity function;
- Experimenting other Word Embeddings, like BERT;
- Using reinforcement learning on training, like policy gradient to maximize the BLEU reward;
- Tuning different hyperparameters that were left with the default values.

Bibliography

[1] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.

[2] Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. Automatic question generation for vocabulary assessment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 819–826, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: $10.3115/1220575.1220678$.

[3] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*, 2016.

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[5] Sérgio Curto, Ana Cristina Mendes, and Luisa Coheur. Question generation based on lexico-syntactic patterns learned from the web. *Dialogue & Discourse*, 3(2):147–175, 2012.

[6] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[7] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*, 2017.

[8] Manaal Faruqui and Dipanjan Das. Identifying well-formed natural language questions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 798–803, 2018.

[9] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. Bootstrapping dialog systems with word embeddings. In *NIPS, Modern Machine Learning and Natural Language Processing Workshop*, volume 2, 2014.

[10] Yoav Goldberg. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726, 2015.

[11] Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *CoRR*, abs/1603.08148, 2016.

[12] Vrindavan Harrison and Marilyn Walker. Neural generation of diverse questions using answer focus, contextual and linguistic features. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 296–306, 2018.

[13] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[14] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301, 2015.

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

[16] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[17] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551, 2017.

[18] Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. Improving neural question generation using answer separation. *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[20] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[21] Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328, 2018.

[22] Hidenobu Kunichika, Tomoki Katayama, Tsukasa Hirashima, and Akira Takeuchi. Automated question generation methods for intelligent english learning systems and its evaluation. In *Proc. of ICCE*, 2004.

[23] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683, 2017.

[24] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer, 2012.

[25] Chin-Yew Lin. Rouge: a package for automatic evaluation of summaries. In *In Proceedings of the Workshop on Text Summarization Branches Out*, 2004.

[26] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[27] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[29] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016.

[30] Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. Recent advances in neural question generation. *ArXiv*, abs/1905.08949, 2019.

[31] 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, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: $10.3115/1073083.1073135$.

[32] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[33] Pranav Rajpurkar. The stanford question answering dataset. `https://rajpurkar.github.io/mlx/qa-and-squad/`, 2019. Last accessed: 2019-04-29.

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100.000+ questions for maching comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[35] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

[36] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[37] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. *Proceedings of the 2013 Conference on Emprical Methods in Natural Language Processing (EMNLP)*, 2013.

[38] Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 251–257, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[39] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: $10.1109/78.650093$.

[40] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 588–598, 2016.

[41] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

[43] Sandeep Subramanian, Tong Wang, Xingdi Yuan, and Adam Trischler. Neural models for key phrase detection and question generation. *CoRR*, abs/1706.04560, 2017.

[44] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[45] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[47] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726, 2014.

[48] Tong Wang, Xingdi Yuan, and Adam Trischler. A joint model for question answering and question generation. *CoRR*, abs/1706.01450, 2017.

[49] PJ Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[50] 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 *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, 2017.

[51] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer, 2017.