Evaluation of methods for question answering data generation
- Using large language models

Utvärdering av metoder för skapande av fråge-svar data

Alexander Bois
Daniel Bissessar

Supervisor: Ehsan Doostmohammadi
Examiner: Marco Kuhlmann
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.
Abstract

One of the largest challenges in the field of artificial intelligence and machine learning is the acquisition of a large quantity of quality data to train models on.

This thesis investigates and evaluates approaches to data generation in a telecom domain for the task of extractive QA. To do this a pipeline was built using a combination of BERT-like models and T5 models for data generation. We then evaluated our generated data using the downstream task of QA on a telecom domain data set. We measured the performance using EM and F1-scores. We achieved results that are state of the art on the telecom domain data set.

We found that synthetic data generation is a viable approach to obtaining synthetic telecom QA data with the potential of improving model performance when used in addition to human-annotated data. We also found that using models from the general domain provided results that are on par or better than domain-specific models for the generation, which provides possibilities to use a single generation pipeline for many different domains. Furthermore, we found that increasing the amount of synthetic data provided little benefit for our models on the downstream task, with diminishing returns setting in quickly. We were unable to pinpoint the reason for this. In short, our approach works but much more work remains to understand and optimize it for greater results.
Acknowledgements

First and foremost, we would like to thank Henrik Holm and Fitsum Gaim Gebre at Ericsson for providing the opportunity, valuable insights, and guidance throughout the work on this thesis. We also want to thank our supervisor Ehsan Doostmohammadi and examiner Marco Kuhlmann at Linköping University for their feedback and encouragement during the thesis work.
# Contents

Abstract ................................................................. iii  
Acknowledgments ......................................................... iv  
Contents ................................................................. v  
List of Figures ......................................................... vi  
List of Tables ......................................................... vii  
Terms and abbreviations ........................................ viii  

1 Introduction ........................................................ 1  
1.1 Motivation ......................................................... 1  
1.2 Aim ................................................................. 3  
1.3 Research questions ............................................. 3  

2 Theory ............................................................... 4  
2.1 Task at hand ...................................................... 4  
2.2 Neural networks ............................................... 7  
2.3 Language Models ............................................... 12  
2.4 Related Work ................................................... 16  

3 Method ............................................................ 21  
3.1 Conceptual framework .......................................... 21  
3.2 Experimental setup ............................................ 26  

4 Results ............................................................ 33  
4.1 Pipeline model set evaluation ................................ 33  
4.2 Pipeline criteria evaluation ................................... 36  
4.3 Final evaluation ................................................. 37  
4.4 Model size experiment ........................................ 38  

5 Discussion ........................................................ 40  
5.1 Results ............................................................ 40  
5.2 Method ............................................................ 43  
5.3 The work in a wider context ................................. 48  

6 Conclusion ........................................................ 49  
6.1 Research questions ............................................. 49  
6.2 Future work ..................................................... 50  
6.3 Final conclusion ................................................ 50

Bibliography ........................................................... 51
List of Figures

2.1 The transformer encoder-decoder architecture ........................................ 10
2.2 Scaled dot-product attention architecture ........................................... 11
2.3 Multi-head attention architecture ....................................................... 11
2.4 Visual representation of the pre-training and fine-tuning concept, picture from Devlin et al. ................................................................. 13
2.5 The input embedding of BERT, picture from Devlin et al. ....................... 14

3.1 The data generation pipeline ................................................................. 22
3.2 Model set evaluation pipeline ............................................................... 25
3.3 Final evaluation pipeline ...................................................................... 26

4.1 Tables 4.9 and 4.10 visualized. ............................................................... 39
List of Tables

2.1 Presents the QA models and amount of data used for fine-tuning .................. 20

3.1 TeleQuAD2 leader board ................................................................. 26
3.2 The data sets used in this thesis ......................................................... 27
3.3 The model sets used during model set selection used for the pipeline ........... 32

4.1 The downstream model performance for the model set evaluation using a TeleRoBERTa model .......................................................... 34
4.2 The downstream model performance for the model set evaluation using a RoBERTaBASE model .......................................................... 34
4.3 Statistics regarding the lengths of answers and questions in the synthetic data sets 35
4.4 Statistics about questions in synthetic data sets ...................................... 35
4.5 The downstream model performance for the data quality evaluation .......... 36
4.6 Cutoff scores for the MS1 evaluation .................................................... 36
4.7 The downstream model performance for the final evaluation using a TeleRoBERTa model .......................................................... 37
4.8 The downstream model performance for the final evaluation using a RoBERTaBASE 37
4.9 Downstream performance for RoBERTaLARGE .................................... 38
4.10 Downstream performance for DistilRoBERTa ........................................ 38
Terms and abbreviations

3GPP 3rd generation partnership project.

AE Answer Extraction.

AI Artificial Intelligence.

BERT Bidirectional Encoder Representations from Transformers.

BPE Byte Pair Encoding.

ELECTRA Efficiently Learning an Encoder that Classifies Token Replacements Accurately.

EM Exact Match.

F1 The harmonic mean of the precision and recall.

GPT-3 Generative Pre-trained Transformer 3.

LM language model.

LSTM Long Short-Term Memory.

ML Machine Learning.

MLM Masked Language Model.

NLP Natural Language Processing.

NN Neural network.

NSP Next Sentence Prediction.

QA Question Answering.

QG Question Generation.

RNN Recurrent Neural Network.

RoBERTa Robustly Optimized BERT Pretraining Approach.

RQ Research Question.

span a segment of text.

SQuAD2 Stanford Question Answering Dataset 2 (version 2.0).

TeleQuAD2 Telecom Question Answering Dataset 2 (version 2.0).

TfIdf Term Frequency-inverse Document Frequency.
One of the largest challenges in the field of Artificial Intelligence (AI) and Machine Learning (ML) is the acquisition of a large quantity of quality data to train models on. In the research area of Natural Language Processing (NLP) this data can take many forms. When working with the subsection of Question Answering (QA), where the task is for the model to answer a question posed to the model, a common type of data are question-answer sets. In this thesis, we will focus on QA tasks where the answer is a segment of text from reference text, or as it also is called a span of text. This means that in this thesis we will focus on reading comprehension styled extractive QA.

In the NLP domain, the state-of-the-art models used are typically large general language models (LM) that have been fine-tuned for specific tasks or domains. Many models in the NLP domain become increasingly better at their specific task with more data to fine-tune the large pre-trained language models with [30]. This does however require data sets specific to the task and domain at hand, and that is the challenge this thesis is focused on. The data generation is focused on data in the telecom domain. What sets this domain apart from other domains is the technical jargon, formal language, and the use of a lot of abbreviations.

The thesis builds on a previous thesis conducted by Holm [11] at Ericsson, which is also where our thesis is conducted. In the thesis, by Holm, the domain-specific and general performance were evaluated for a BERT-like model trained using an ELECTRA training approach. Holm states that the study suffered from the insufficient size of target domain data sets. This observation lead to a problem description, which motivated the start of this thesis [11]. Thus, this thesis aims to evaluate methods for generating extractive QA data sets specific to the telecom domain to improve the performance of large language models with fine-tuning.

1.1 Motivation

Many machine learning models are trained using supervised learning, which is a technique where the model updates or sets its parameters based on correcting itself according to data
which is presented as pairs of input-output. The model thus tries to come as close to the correct output based on the input and is corrected when it is making mistakes. Since the data sets of input-output pairs are used as reference material for the model to learn a task, such data sets are often created by domain experts to get as good quality of the pairs as possible. This process is called labeling or annotating a data set [27].

The annotation is a resource-demanding process for any task in machine learning. To exemplify this in the field of NLP, one can look at the task of annotation for a QA data set. A typical extractive QA data set [9, 22, 23] has two inputs and one output. The first input is a context document, which is a text about which questions can be asked, and the second input is a question about the context document. The output is a span of a text representing the answer to the question based on the context. Typically, the context document is always present during the annotation process, and the process aims to complete the data set with either a question and an answer or one of them depending on if the other one is present already. Thus, the annotator must read the context document and construct the answers and/or questions. This process requires a lot of resources since it involves processing natural text, expertise, and imagination, which all contribute to poor scalability [17].

The need for larger fine-tuning data sets, that are distinct for the task or domain, in combination with the poorly scalable process of creating domain-specific sets creates a need for machines to handle the annotation of data, with good enough quality to improve the results of the re-trained models. Comparing different methods for the creation of task- and domain-specific data sets could benefit many actors [30].

Related work that has shown promising results in creating QA data sets automatically using a general pre-trained language model like BERT, GPT-3 and others [2, 8, 19, 26]. To our knowledge, there has not been any research into comparing the performance of models trained on general domain data and technical domain data for the generation of QA data sets. While generating QA data for the general domain has shown promise, there is no research done regarding the generation of QA data for niche domains.

There already exists good language models for different tasks, but most are trained in the general domain. When comparing the data used to fine-tune more general models, Ericsson has seen a great difference in the language used for the general models compared to data used for models in a telecom domain setting [11]. This motivates the creation and usage of domain-specific data for improvements in domain-specific language models. The difference in language is partly due to the jargon used in the domain, for example, “cell” in a telecom setting has a very different meaning to "cell" in biology, criminology, and other domains.

The value of researching this is to make a plausibility study into domain-specific data generation for future use cases, with the novelty of using pre-trained domain-specific language models for data set generation.
1.2 Aim

The thesis aims to evaluate different approaches to data generation in a telecom domain for the task of extractive QA. To evaluate these methods we have access to a human-annotated data set and a baseline comparison of F1-score and Exact Match (EM) score of other models of the same domain and task. The data sets created using the different approaches will be used to fine-tune larger models and compare that model’s scoring to the established baseline with the defined metrics. The evaluation will be based on the downstream task of QA in the telecom domain.

1.3 Research questions

1. What possible approaches can be used to generate synthetic data sets for extractive QA, given context documents?

This research question will help us reach the thesis aim by providing methods to generate data for the given task and domain. This question will be answered in chapters 2 & 3.

2. How do the models trained on synthetic data generated from a general domain pipeline perform compared to the models trained on synthetic data generated from a pipeline pre-trained on telecom domain data?

This question aims to provide insight into what effect the pre-training domain of a generation model has on the quality of generated telecom data.

3. How do the models trained on synthetic data in addition to human-annotated data perform compared to the baseline model trained only on human-annotated data?

Answering this research question will give insight into which method shows potential and feasibility of use in a real-world application. This question will be answered through experimentation.
2 Theory

In this chapter both a theoretical background for the scope of the study is provided, as well as a study of related works concerning the task at hand. This is done to establish a framework for the development of the data generation approaches in the method (chapter 3).

2.1 Task at hand

The field of NLP is focused on the processing of natural language by computers and addressing a plethora of tasks and challenges that computers have when dealing with natural language.

2.1.1 Question Answering

One of the tasks faced in the field of NLP is QA. QA is a task performed by computer programs to give answers to questions posed to it by humans in the form of natural language.

The type of QA that will be in focus in this thesis is called reading comprehension style QA. This style answers a question about a passage of text [23, 22]. This type of QA has mainly been tackled with two types of models which are extractive models and generative models [8].

Extractive models

Extractive models take as input a passage of text or context document and a question about the context document. As output, the model gives the answer in the form of a span of text from the context document. Thus, this model’s answer is a part of the context document. The extractive model that will be used in this thesis is only using one context document for reference when answering the question. Examples of models that give their answers in this form are models based on BERT [7].
Generative models

Much like the extractive models for the reading comprehension type QA, the generative models take a context document and a question about that passage as inputs. The difference is that, based on the inputs, the model generates a free-form answer. This is different from the extractive models in the sense that the answer does not have to correspond to a specific span in the passage of text taken as input [8]. Examples of models that use this to answer questions are GPT-3 [5] and T5 [21].

2.1.2 Transfer learning

Recent models that have achieved state-of-the-art for many different NLP tasks rely on transfer learning [7, 21]. Transfer learning is the technique by which a model is pre-trained on a data-rich task to develop the model’s general “understanding” or model a language and then fine-tune the model within certain domains or tasks that are more specific.

2.1.3 Evaluation metrics

When working with QA tasks several metrics are used to judge the performance of a model, depending on which kind of QA is performed. These metrics will be presented and explained here.

F1-score

The macro-averaged F1-score is a metric regularly used in connection to classification tasks in NLP and is also a useful metric for evaluation in QA tasks [23]. The F1-score is the harmonic mean of the precision and recall and is calculated with this formula [13]:

\[
F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\] (2.1)

The precision and recall are in turn computed using three terms [13]:

- **True positives (TP)**, when the predicted value and the true value are of the same class or both are positive. For extractive QA this is the number of positional units (e.g. words or tokens) of a span that are shared between the true answer span and the predicted answer span.

- **False positives (FP)**, when the value is predicted as positive (belonging to the observed class), but the true value is negative (belonging to another class). For extractive QA this is the number of positional units that appear in the predicted answer span that are not present in the true answer span.

- **False negatives (FN)**, when the value is predicted as negative (belonging to another class), but the true value is positive (belonging to the observed class). For extractive QA this is the number of positional units that are present in the true answer span but not in the predicted answer span.

Using these three terms the metrics are calculated according to the following formulas [13]:

\[
\text{precision} = 2 \times \frac{TP}{TP + FP}
\] (2.2)

\[
\text{recall} = 2 \times \frac{TP}{TP + FN}
\] (2.3)
2.1. Task at hand

Exact match score

The exact match score is, as the name suggests, a score based on if the answer span produced by the model is an exact match to the answer span in the data set. Thus if an answer span is correct for every position the score for the question is one, otherwise it is zero even if there is just a small error for the position of the span. If any span is predicted for a non-answerable example the score is also zero [23].

2.1.4 Text normalization

Natural language can contain a lot of variation which leads to models having to account for a lot of variability. To make it easier for the models a common task to use is text normalization. The task of normalizing text includes several different processes which can be applied to text to make it more convenient or standard form, which then will be easier to handle for models [13].

One of these processes is called tokenization. Tokenization aims to demarcate the text into units called tokens. The form of these tokens depends on the tokenization chosen. One simple example could be to divide a text based on whitespace, then the tokens would be words and symbols that are divided by whitespace. The algorithm or model that performs this process is normally referred to as a tokenizer [13].

Byte Pair Encoding tokenization

Byte Pair Encoding (BPE), used in the context of NLP, was first introduced by Sennrich et al. [25] and is an adaptation of a simple data compression technique. The algorithm takes the most frequently occurring pair of bytes in a sequence and replaces the pair with a single byte that is unused. Sennrich et al. applied this as a tokenization technique but for Unicode characters instead of bytes creating a tokenization strategy that used sub-word units as tokens. Radford et al. [20] argues that using Unicode instead of byte requires a very large vocabulary, and is not competitive against whole-word tokenization. Thus, Radford et al. adapted the algorithm to use bytes again, but in a way that made the compression more effective and added minimal fragmentation of words across multiple tokens in the vocabulary. This type of tokenization is used for several models, including RoBERTa [33] and GPT-3 [5].

To give an example of this one could imagine that a pre-tokenization process has identified all unique words in a text, let’s say by using whitespace tokenization, and counted their respective frequency. This could result in a list looking like this:


Starting with a base vocabulary consisting of the unique characters of the words leaves us with the vocabulary of: "b", "g", "h", "n", "p", "s", "u".

Starting the first iteration the words would be split into symbols based on the vocabulary.

("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

The BPE algorithm then counts the frequency of each possible symbol pair. The most frequent pair in this example would be "u" followed by "g", appearing 20 times. Thus, the first merge rule the tokenizer learns is to group "u" followed by "g". Also, the symbol "ug" is added to the vocabulary. This leaves us with the new list:
2.2 Neural networks

Neural networks have been established as the state-of-the-art approaches in sequence modeling and transduction problems, and thus also the state of the art in the area of QA [29].

There are several variations in architecture for these networks, but here we will be looking at using NNs to create sequence-to-sequence models. This type of modeling is used for several purposes, but we will be delimiting ourselves to the NLP space. This type of model has a sequence of inputs and also outputs a sequence.
2.2 Neural networks

2.2.1 Recurrent Sequence Modeling

In a recurrent sequence-to-sequence model, a sequence is used as input and is fed into an NN referred to as an encoder, and the output \( c \) of that encoder feeds into another NN referred to as a decoder which outputs another sequence. The output of the encoder \( c \) is a fixed-length vector, thus placing a bottleneck on the amount of information this architecture is able to model. The fact that the architecture requires \( c \) to be a vector places further limitations on the model, for example, a matrix of the hidden states of the encoder cannot be used as an input for the decoder.

Another shortcoming of this architecture is that the computation of the model is by definition sequential, which limits the sequence lengths of training examples because of memory constraints. It also inhibits the use of parallel computation, as the output from the previous state is required to make operations on the current state. These two factors limit both the inputs to the architecture, but also make NN encoder-decoder architectures slow to train due to inefficiency.

RNN Encoder-Decoder architectures

Recurrent neural networks are feedforward neural networks rolled out over time and deal with sequence data, i.e. the input has a defined ordering.

An issue with RNNs comes with the fact that they use gradient propagation. When exposed to large amounts of time steps the RNN suffers from the vanishing gradients problem. This occurs when the gradient becomes vanishingly small and thus parameter weight updates also become vanishingly small, preventing the network from learning further. There have been different attempts to combat this issue, and some of the more successful ones which have achieved competitive results are described below [6, 10]. They also put less emphasis on information early in a sequence because of their sequential work, and in general have a bias towards recent information [6, 10].

Gated RNN Sequence Modeling

Gated RNN sequence models are similar to regular RNN sequence models but use a more sophisticated activation function to combat the vanishing gradients problem [6]. Here, instead of updating the activation function using the gradient, the activation function is linearly interpolated between the previous activation function and the candidate activation function [6].

LSTM Encoder-Decoder architectures

In an attempt to remedy the vanishing gradients problem an alternative to the RNN encoder-decoder architecture is used. Here, the neurons in the previous architecture are replaced by LSTM networks. This allows the LSTM network to avoid the vanishing gradients problem, train more efficiently, and perform well on tasks requiring knowledge from input steps that are many more steps behind than what an RNN could handle [10].

2.2.2 Attention

If we were to give one of the previously mentioned architecture a book as a context, for example, a book on neural networks; and we would give the model the question of "What does RNN stand for?" the model would in layman’s terms read the entire book and from that
knowledge answer the question [3]. This approach is however quite different from how a human would approach this problem, as they would likely just look up “RNN” in the book and try to find the information in just one sentence, and this idea of trying to attend to the right contextual information is also possible to combine with neural networks, as demonstrated by Bahdanau et al.[3].

To do this every word $x_i$ is given a context vector, defined by concatenating a forward and a backward hidden state. This context vector is dependent on a sequence of annotations $h_1..h_T$ where an encoder has mapped the input sequence. Thus, every annotation $h_i$ contains information about the entire input sequence, but there is a strong focus on the $i$-th word giving the model the previously mentioned attention.

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$  \hspace{1cm} (2.4)

This weight $a_{ij}$ for each annotation $h_j$ is calculated with the following formula

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$  \hspace{1cm} (2.5)

where

$$e_{ij} = a(s_{i-1}, h_j)$$  \hspace{1cm} (2.6)

is a model for alignment, where the inputs are scored based on position $j$ and the output match at position $i$. Thus $s_{i-1}$ is the hidden state of the RNN before the time $i$.

Here, if we examine the formula 2.5 for attention proposed by Bahdanau et al. we can see from the summation that the mechanism is additive [3]. The same can be said for the formula for $c_i$ 2.4 which is also additive and thus a feed-forward network. Later Luong et al. proposed a multiplicative version of attention, replacing the feed-forward network with a dot product and also making the attention mechanism multiplicative [18]. This approach introduces simpler computations and is thus the more common approach since its introduction [4].

### Self-Attention

In 2017 Lin et al. introduced the concept of self-attention, building on the previous work done in the field of attention. Here, instead of using a vector for attention, a 2-D matrix is used with every row in this matrix representing attention on different parts of the sentence [16]. As a side effect, this also introduces an easy way of visualizing what parts of the sentence are specifically encoded into the embedding [16]. This new method produced very compelling results and improved the attention mechanism used in language models [16].

### 2.2.3 Transformers

As it turns out, sequential models despite many improvements attempting to tackle the fundamental problems of them, were not the way forward. In 2017 Vaswani et al. proposed that "Attention is all you need" in a paper by the same name which introduced the Transformer model [29]. This model proposed a new model architecture composed of only attention mechanics, and thus removed both convolutions and recurrence mechanics and the presented issues with them [29]. This model architecture achieved several state-of-the-art results and laid the groundwork for many of today’s state-of-the-art models [29].
2.2. Neural networks

As displayed in figure 2.1 of the architecture it can be observed that this architecture still uses the concept of encoder-decoder design, but as previously stated uses attention to draw global dependencies between input and output [29]. This structure also allows for a significant increase in potential parallelization, and at the time achieved state-of-the-art results after just 8 hours of training on eight P100 GPUs [29]. The new mechanism that can be observed from the visualization of the structure 2.1 is that of the multi-head attention blocks, which are created by using what is called scaled dot-product attention Vaswani et al.

Figure 2.1: The transformer encoder-decoder architecture [29]

Scaled Dot-Product Attention

The scaled dot-product attention used in the transformer architecture builds upon the multiplicative attention mechanism proposed by Luong et al. but uses a scaling factor to prevent issues with the softmax function due to the possibility of extremely small gradients [29]. The input contains values of dimension $d_v$ and also queries & keys with the dimension of $d_k$. The queries are contained in the matrix $Q$, the values in the matrix $V$, and the keys in the matrix $K$. Thus, the output of the attention mechanism can be computed with the following formula

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$ (2.7)

This implementation is computationally more efficient since it can exploit matrix multiplication code of high efficiency and quality and with the scaling factor, it avoids gradient issues. A graphical representation of this module in the architecture can be seen in figure 2.2.
Multi-Head Attention

Using the previously introduced scaled dot-product attention move forward to our multi-head attention module in the transformer architecture. Here, to further improve parallelization the values, keys, and queries are linearly projected $h$ times with different linear projections that are learned. This results in final output values of dimension $d_v$, which as can be observed in figure 2.3 are concatenated.

Another reason for using multi-head attention is that it enables averaging, while a single head attention mechanism would inhibit this. Thus the multi-head attention can attend to information originating from multiple different sub-spaces at differing positions at the same time. Thus the output of the multi-head attention mechanism can be calculated using the following formula

$$ MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O $$

(2.8)

where $head_i$ can be derived from

$$ head_i = Attention(QW^Q_i, KW^K_i, VW^V_i) $$

(2.9)

and the parameter matrices are

$$ W^Q_i ∈ R^{d_{model} × d_k}, W^K_i ∈ R^{d_{model} × d_q}, W^V_i ∈ R^{d_{model} × d_v}, W^O ∈ R^{hd_v × d_{model}} $$

(2.10)
2.3. Language Models

The transformer encoder

If we examine the encoder part of the transformer architecture we can see that every layer of the encoder has sub-layers that have a residual connection to the previous sub-layer and a layer normalization of that connection [29]. This means that the output for every sub-layer can be formulated as

\[ \text{LayerNorm}(x + \text{SubLayer}(x)) \] (2.11)

where the SubLayer function is the function implemented in the sub-layer itself, for example, multi-head attention. The computed and normalized attention is then fed forward through a feed-forward neural network, which has the same sub-layer procedures before being fed forward to the next layer.

The transformer decoder

The transformer decoder has the same sub-layers in the last two sub-layers except for the encoder feeding into the multi-head attention sub-layer function. It also has a first sub-layer with a masking function to ensure that predictions are for a position \( i \) can only depend on position previous to \( i \) [29].

Encoder-decoder stacks in the transformer architecture

As can be observed in figure 2.1, every layer in the encoder and decoder is repeated \( N_X \) times. In the original article by Vaswani et al. both the encoder layer and decoder layer are repeated 6 times before feeding into the linear and softmax layers for final predictions [29]. We can also observe that before inputs are fed into the transformer architecture they are embedded and given positional encoding [29]. The positional encoding is needed since the removal of recurrence and convolution also removes the idea of the relative and absolute position of words in the input [29]. The embeddings are used to convert input and output tokens to vectors of dimension \( d_{\text{model}} \) [29].

2.3 Language Models

In this section, we will present a couple of relevant LMs that are used by related works 2.4. LMs come in many different shapes and sizes. The relevant LMs for this thesis are all based on the transformer architecture; using either the whole transformer or parts of it.

2.3.1 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a contextualized language representation model. It is a model that when building a representation of language (i.e. a word or token embedding) takes into account the context in which the word appears. Another advantage of BERT is the ability to pre-train the model using unlabeled text data to tune the weights to create the language representation or “understanding” language. BERT can be easily adapted for a wide range of downstream tasks by adding an output layer on top of BERT, using the BERT embeddings as input to the task-specific output layer [7]. The training of BERT is thus divided into two separate phases: the first is pre-training which focuses on creating the language representation, and then comes fine-tuning which is done for the extra layers and is task-specific. This concept can be seen in figure 2.4. When BERT was introduced it established itself as the state-of-the-art for eleven different NLP tasks [7] and has been used and adapted continuously since.
2.3. Language Models

Figure 2.4: Visual representation of the pre-training and fine-tuning concept, picture from Devlin et al. [7].

Architecture

The architecture of BERT is based on the encoder part of the Transformer (section 2.2.3). Devlin et al. [7] presented two different BERT models in their paper: BERT\_BASE & BERT\_LARGE. There are differences between the two models in terms of the number of stacked encoders ($L$), the amount of self-attention heads ($A$), and the hidden size ($H$). For these different amounts and sizes BERT\_BASE has $L = 12$, $A = 12$, and $H = 768$. This, resulted in 110M parameters to tune during training. BERT\_LARGE on the other hand has $L = 24$, $A = 16$, and $H = 1024$. The resulting amount of parameters end up at 340M. The input sequence of this architecture is 512 tokens.

Tokenization and input embeddings

BERT is using WordPiece (section 2.1.4) for tokenization, with a vocabulary size of about 30,000 words. The first token in the input token sequence is always a special $[\text{CLS}]$ token, which during training is used for the next sentence prediction (NSP) task, described in section 2.3.1, and during fine-tuning is used as a representation of the whole sequence that could be used e.g. a classification task. To separate the sentence pairs used during pre-training, another special $[\text{SEP}]$ token is used. A token sequence following the form just described is what is used as input to BERT. With this input BERT creates input embeddings by summarizing three different embeddings: token, segment, and position. This can be seen in figure 2.5. The token embeddings are the embeddings for each token created by the WordPiece tokenization. The segment embedding is used to indicate whether a token is in segment A or Segment B, i.e. where pairs of sentences are used as input each sentence is a segment. The position embedding marks the absolute position of a token in a sequence.
2.3. Language Models

During the pre-training of BERT, the model was trained on two unsupervised tasks at the same time. The tasks are **Masked Language Model (MLM)** and **Next Sentence Prediction (NSP)**. These tasks are unsupervised since they do not depend on labeled data, and use the original text as ground truth to compare the task predictions. The data used were the BookCorpus (800M words) [32] and English Wikipedia (2,500M words).

MLM is a *cloze* styled task, where the model is tasked with filling in or predicting the tokens which are masked in the input. To do this a softmax layer at the end of BERT, with the same amount of neurons as there are words in the input embedding vocabulary is used. The tokens are masked at random during training, and 15% of the tokens are masked. But, to mitigate the problem of creating a mismatch between the pre-training and fine-tuning each masked token only has an 80% chance of being replaced by `[MASK]`, 10% are replaced by a random token, and 10% are not replaced at all. This is because the `[MASK]` token itself does not occur in the fine-tuning. The final hidden vectors for the masked tokens are fed into a softmax over the initial WordPiece vocabulary. The MLM task is trained using cross-entropy loss for the prediction of the masked tokens. Since the whole sequence is processed at the same time, giving the encoder access to both the right and left context during the MLM task BERT becomes bidirectional in its representation of language.

For NSP, the model takes in two sentences and is tasked with determining if the second sentence logically follows the first. This helps the model understand context over different sentences. The data used for this task are examples of sentence pairs that are either positive or negative. A positive pair is sampled by using a sequence of two sentences from the unlabeled data. A negative pair is made by sampling one sentence and then another from another part of the unlabeled data. The positive-negative split used during training is 50% [7].

**Fine-tuning**

To use BERT for various specific NLP tasks, the model has to be fine-tuned for that specific task. An output layer needs to be added, of chosen complexity or design, to the output of the BERT model to cater to the task. After that has been added the fine-tuning is done [7]. For the task of QA supervised learning is used to train the model. When fine-tuning the model all parameters are tuned, only the output layer parameters are trained from scratch. Compared to pre-training, fine-tuning is relatively fast [7].

The input to the model for QA tasks is in the following format: `[CLS]` *question* `[SEP]` *context*. 
2.3. Language Models

To produce an output Devlin et al. only introduce two vectors \( S \in \mathbb{R}^H \) and \( E \in \mathbb{R}^H \). These represent the weights applied to each token position \( T_i \) with the dot product. After that, a softmax activation is applied over all the tokens in the context to produce a probability distribution over the tokens, as is shown in eq. 2.12.

\[
P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}
\]  

(2.12)

The analogous equation is utilized for the end position. The scoring of the candidate span is defined as \( S \cdot T_i + E \cdot T_j \), from position \( i \) to \( j \). The span with the maximum scoring, and where \( j \geq i \), is used as the prediction. The training objective for this task is the sum of log-likelihoods of the correct start and end positions [7].

2.3.2 Robustly Optimized BERT Pretraining Approach

Robustly Optimized BERT Pretraining Approach (RoBERTa) was presented by Zhuang et al. [33] and is essentially a reproduction of the study by Devlin et al. [7] in which BERT was presented. Zhuang et al. explore some different design choices for the pre-training of a BERT-like model. Two design choices during pre-training of the model was to remove the NSP task and to dynamically mask tokens for the MLM task. The dynamic masking of tokens takes place when feeding each sequence of tokens into the model, improving the pre-training since the model can see the same sequence but masked differently during different epochs of the training. Zhuang et al. also found that increasing the batch size improved the model slightly. Zhuang et al. used 10 times as much data compared to the original BERT during pre-training and longer sequences. Another change is the use of a different tokenizer compared to BERT. The tokenizer used for RoBERTa was a byte level BPE described in section 2.1.4 with a vocabulary size of 50,000 words. This change of tokenizer did not show any improvement, but the Zhuang et al. argues that the byte level BPE has the benefit of being able to encode any input.

2.3.3 Text-to-text Transfer Transformer

The Text-to-Text Transfer Transformer model or T5 was introduced by Raffel et al. [21]. The authors introduced this model as a unified framework for various transfer learning techniques within NLP, converting a plethora of tasks into a text-to-text format. This text-to-text approach takes, for every task, the input as text and the output is a new text, making it possible to apply the same model, objective, training procedure, and decoding process for every NLP task [21].

Architecture

Raffel et al. chose to use an architecture which is based in large on the full transformer presented by [29], described in section 2.2.3. The authors made slight modifications to the transformer by removing the layer norm bias and using a different position embedding scheme [21]. In the study the authors tried out several different model sizes where the T5\text{BASE} model uses the architecture described above, with \( L = 12 \) layers of the decoder and encoder, \( A = 12 \) attention heads, and \( H = 768 \) as the hidden size, resulting in roughly 220M parameters. The T5\text{BASE} model has similar stats to the BERT\text{BASE}. The authors also tested sizes larger than the T5\text{BASE} with up to 11B parameters. The authors acknowledged that these models are not an option when there is a limit on computational resources and thus also tested a T5\text{SMALL} model with \( L = 6 \), \( A = 8 \), and \( H = 512 \) resulting in about 60M parameters. T5 uses SentencePiece to encode the text as WordPiece tokens [14, 21].
Pre-training

For the pre-training of the T5 model Raffel et al. created a dataset called the Colossal Clean Crawled Corpus, which is 745 GB of “clean” natural text. The cleaning was mainly done using filtering heuristics, and also a model which predicted if the text was English or not. During the study, the authors tested different pre-training objectives and found that a model trained with a BERT-styled masked language objective worked best. This method is unsupervised. Since the model is text-to-text, standard maximum likelihood with cross-entropy loss is used for training, both for pre-training and fine-tuning.

Fine-tuning

Raffel et al. found that the most effective approach to fine-tuning the model for specific NLP tasks was to update all parameters of the model [21]. Since T5 is a text-to-text model no additional task-specific layers are added to the model, but the task-specific “instructions” are added to the input text. For example for a translation task, the input could be “translate from English to German: that is good” and the corresponding output example would be “das ist gut”. For an example of question generation, the input could be “answer: Apples are fruits context: Apples are fruits and carrots are vegetables” with an example output of “What is an apple?”. In this way, the model learns the specific task. The “instructions” work to prime the model for a specific task, so that one model could potentially handle several different tasks.

2.3.4 Generative Pre-trained Transformer 3

Generative Pre-trained Transformer 3 (GPT-3) is an auto-regressive language model which was presented by Brown et al. [5]. One conceptual difference between BERT and GPT-3 is in the training approach. As described in section 2.3.1 BERT uses pre-training to model the language and is then fine-tuned for specific tasks, while GPT-3 does not need to be fine-tuned the same way to achieve comparable results. GPT-3 is able to achieve competitive performance compared to the state-of-the-art fine-tuned models when released. Brown et al. explains this by the sheer size of GPT-3, giving it a good enough model of language to understand and perform using in-context learning. In-context learning is different from supervised learning in the way that in-context learning does not involve any updates to the weights of the model at inference time. Instead, the model is shown some few examples of labeled data to prime it for the specific task, then it should be ready to produce a result in that task. The idea when developing GPT-3 was to introduce a model which was task agnostic and did not require large labeled data sets for fine-tuning.

2.4 Related Work

In this section works that relate to the area studied will be presented to give context about the base on which this thesis builds and to present techniques that can be used from previous work. Several different works spanning different aspects of the QA data set generation task at hand will be presented.

When studying the related works, several distinct tasks were identified when generating QA data sets. Thus, we will present the contributions from several works regarding these tasks.
2.4. Related Work

2.4.1 Answer Extraction

Alberti et al. use BERT to accomplish Answer Extraction (AE) [2]. They do this by fine-tuning BERT to model the function

\[
p(a|c; \theta_A) = \frac{e^{f_j(a,c;\theta_A)}}{\sum_{a'\in a_{\text{a}}} e^{f_j(a',c;\theta_A)}} \tag{2.13}
\]

This makes the fine-tuned model choose the span \(a\) which has the highest probability among spans \(a_{\text{a}}\) over \(c\). The span \(a\) & \(a_{\text{a}}\) has a maximum length of 32 WordPiece tokens. The function \(f_j\) (Eq. 2.14) is the token representation produced by the BERT model, where \(a = (s,e)\) and \(BERT(c)[i]\) is the BERT representation for the \(i\)’th token in \(c\). \(MLP_j\) is representing the output layer added to BERT for fine-tuning as described in section 2.3.1. This layer used by Alberti et al. is a multi-layer perceptron with a single hidden layer.

\[
f_j(a,c;\theta_A) = MLP_j(\text{CONCAT}(BERT(c)[s], BERT(c)[e])) \tag{2.14}
\]

Puri et al. [19], inspired by Alberti et al., took a similar approach to fine-tune a BERT model to do AE. Their \(MLP_j\) consisted of one hidden layer with a hidden size twice as big as the model’s hidden size, adding a ReLU nonlinearity, and a projection from activations to logits.

In the case of both Puri et al. and Alberti et al., the input to the BERT model is the tokenized context, excluding the question, and the fine-tuning is trying to predict the correct answer span and comparing that to ground truth.

2.4.2 Question Generation

When it comes to text generation there are a variety of options to choose from when it comes to different models. Here we will present some approaches used for generating questions. Alberti et al. [2] investigated two different options. The first was an approach where they fine-tuned a BERT model, while the second was to utilize a whole transformer model and pre-train it from the start. Both approaches will be described here.

The first approach chosen by Alberti et al. was a simple adaptation of the publicly available and already pre-trained BERT model, making it work with the same objective as a decoder. The motivation behind this choice was that it did not require any additional pre-training and no additional parameters had to be trained from scratch. They fine-tuned the model on examples from data sets like SQuAD2 (for detail see section 3.2.2). The model was fine-tuned as a left-to-right LM according to eq. 2.15.

\[
p(q|a,c;\theta_Q) = \prod_{i=1}^{L_Q} p(q_i|q_{1:i-1}, a,c;\theta_Q) = \prod_{i=1}^{L_Q} \frac{e^{f_Q(q_{1:i-1}, a,c;\theta_Q)}}{\sum_{q'_i} e^{f_Q(q'_{1:i-1}, a,c;\theta_Q)}} \tag{2.15}
\]

Alberti et al. defined and computed \(f_Q(q_{1:i}, a,c;\theta_Q)\) as seen in equation 2.16, where \(W_{\text{BERT}}\) is the word piece embedding matrix used in BERT.

\[
f_Q(q_{1:i}, a,c;\theta_Q) = BERT(q_{1:i-1}, a,c)[i-1] \cdot W_{\text{BERT}}^T \tag{2.16}
\]

Alberti et al. also introduced a new token type id to separate the context, answer span, and the previously generated question tokens used as input to the model, helping the model make a distinction between the three. These tokens work by marking question tokens as 0, context tokens as 1, and answer tokens as 2. To further separate the input, the context and answer tokens are separated from the question tokens using a separation token. Alberti et al. also pad or truncate the question being input to the model to the length of \(L_Q\) as to not
give any hints for the model of the expected length of the question. The model is trained by making use of an attention mask that forces all the attention weights to zero for all the input except the previously generated question tokens. The actual QG is done through iterative greedy decoding using $\text{argmax}_{q_i} f_Q(q_1, ..., q_i, a, c)$.

As mentioned earlier Alberti et al., also tried another approach to QG which is based on fully pre-training and fine-tuning a sequence-to-sequence transformer. For the pre-training the encoder part was trained identically to how BERT is pre-trained, described in section 2.3.1. As for the decoder it was trained to output the next sentence. The fine-tuning of this model was done in the same way as described in the paragraph above, using $(c, a)$ as input, $q$ as output. For the actual QG, they sampled from the decoder using both beam search and Monte Carlo search. Alberti et al. found that the approach involving pre-training and fine-tuning the Transformer provided the best results on the downstream QA task.

Sultan et al. [26] opted for using the fine-tune only approach presented by Alberti et al. with some modifications. Instead of using BERT, they used the pre-trained RoBERTa model, because it is pre-trained on more text. The fine-tuning regime is the same as for Alberti et al. described above, using the loss function seen in eq. 2.17.

$$\text{loss}_i = -\log p(q_i|q_{1:i-1}, a, c)$$ (2.17)

The QG also works similar to the approach of Alberti et al., but Sultan et al. found that Top-p nucleus sampling to sample the next question token had the best effect on downstream QA tasks.

### 2.4.3 Roundtrip filtering

Roundtrip filtering is a method presented by Alberti et al. [2] used to increase the accuracy for QA tasks when the data used for fine-tuning is generated using other LMs. They found that roundtrip filtering overall increased accuracy for the downstream task. Alberti et al., Dong et al., and Puri et al. utilize this step when filtering their generated QA data sets. All papers use a BERT model fine-tuned for QA for this task [2, 8, 19].

The roundtrip filtering is using a model fine-tuned for QA. As input, the context document and the generated question are used. The QA model tries to answer the question and thus produces an $\text{answer}_{\text{new}}$. This $\text{answer}_{\text{new}}$ is then compared to the $\text{answer}_{\text{extracted}}$. Alberti et al. made the filtering criteria to require an exact match between $\text{answer}_{\text{extracted}}$ and $\text{answer}_{\text{new}}$. Alberti et al. could not give a formal explanation as to why this filtering improves the accuracy in the downstream task, but rather showed it did improve the accuracy overall.

A variation to the filtering presented by Alberti et al. comes from Dong et al. [8]. Dong et al. adjusted the filtering criteria. Dong et al. expressed the criteria as a F1-score instead, where an exact match would equal a score of 1. This made it possible to be more lenient to deviations between $\text{answer}_{\text{extracted}}$ and $\text{answer}_{\text{new}}$. Dong et al. adjusted the criteria to discarding the produced question-answer pair if F1 between $\text{answer}_{\text{extracted}}$ and $\text{answer}_{\text{new}}$ was less than F1 = 0.6.

### 2.4.4 Non-answerable questions

Rajpurkar et al. [22] when evaluating SQuAD2, described in section 3.2.2, also evaluated models trained on computer-generated non-answerable questions. Rajpurkar et al. divides the approaches for generating non-answerable questions into Tfidf and rule-based methods.
The rule-based method is focused on altering the question about a specific context, according to a couple of rules, so that the question becomes unanswerable when using the original context. Rajpurkar et al. used a set of rules, presented by Jia and Liang [12], that replaced entities and numbers with similar words and replaced nouns and adjectives with WordNet antonyms. Another set of rules was used by Dong et al. [8] which used two rules: first substitute the question entities with the same entities within the context; second they insert the negative *not* behind the verbs *be, do, have*, and modal words.

The TfIdf method, applied by Rajpurkar et al., on the other hand, pairs questions with new contexts, hoping that the question becomes unanswerable in the new context. This was done using TfIdf scores to measure the likeness between the original and the new context [22]. Alberti et al. used a simpler approach than the TfIdf approach. Instead of comparing the likeness of the true context and another potential context, they simply took another context, that was not overlapping with the original context, from the same Wikipedia page and matched the question with the new context [2].

Rajpurkar et al. found that no matter which of their approaches was chosen, TfIdf or rule-based, the non-answerable questions produced were much easier for the model to identify than human-created non-answerable questions. Rajpurkar et al. concluded that the non-answerable questions generated are too easy for the model to identify [22].

### 2.4.5 Question Answering for evaluation

In the related work, when training the QA model for downstream evaluation, a difference in the size of the model, the size of the data set, and the type of data used can be observed. For their fine-tune only approach Alberti et al. fine-tuned their QA model first on 3M SQuAD2-like samples, with 2.4M answerable samples and 0.6M non-answerable. This model was then fine-tuned on SQuAD2 before evaluation [2]. The resulting performance can be seen in table 2.1.

For their full pre-training approach Alberti et al. generated 50M samples for the model to be pre-trained on and then fine-tuned it using SQuAD2. For these models, the performance was reported on the dev set of SQuAD2. This model was evaluated on the SQuAD2 dev set with a resulting score of $EM = 85.1$ and $F1 = 87.9$ [2].

Dong et al. differs in both the model used and the amount of data used. The amount of data used ended up at around 5M answerable questions and 4M non-answerable resulting in a total of 9M samples that are SQuAD2-like [8]. The performance of their QA model can be seen in table 2.1.

Puri et al. wanted to compare their approach to Alberti et al. and Dong et al., with the difference of not generating non-answerable questions. Thus Puri et al. trained two QA models. The first was a BERT$_{LARGE}$ model, which is the architecture used by Alberti et al., fine-tuned on 3M answerable samples and then on SQuAD2. The second model was a larger BERT-like mode with 345M parameters, fine-tuned first on 3M samples and then on 8M more, to compare with the state-of-the-art established by Dong et al. [19]. All these results are reported in table 2.1.
Table 2.1: Presents the QA models and amount of data used for fine-tuning used by different authors. The EM and F1 scores from the evaluation on the SQuAD2 dev split are also presented. The best performance is highlighted and is achieved by a large model which has seen the most samples, all of which are answerable.
This chapter presents the methodology and experimental setup used to answer the research questions. First, a conceptual overview of the method will be presented together with the evaluation method. After going through the conceptual overview, the experimental setup will be presented in detail.

### 3.1 Conceptual framework

The aim of this report, presented in section 1.2, is to evaluate different methods of generating synthetic telecom domain QA data.

To help reach the thesis aim three research questions were posed in section 1.3:

1. What possible approaches can be used to generate synthetic data sets for extractive QA, given context documents?
2. How do the models trained on synthetic data generated from a general domain pipeline perform compared to the models trained on synthetic data generated from a pipeline pre-trained on telecom domain data?
3. How do the models trained on synthetic data in addition to human-annotated data perform compared to the baseline model trained only on human-annotated data?

To answer RQ 1. several methods were considered in section 2.4 from which some will be used for this thesis. What could be seen was that generating QA data typically were divided into sub-task, e.g. AE, QG, and others. For each sub-tasks, different models fine-tuned for the specific sub-task were used. RQ 2. aims to investigate if there are benefits to using models trained for a specific domain when generating data for that domain or if more general training suffices. RQ 3. aims to investigate whether using synthetic data in addition to human-annotated data is beneficial compared to using only human-annotated data when fine-tuning a model for a specific task and domain.
3.1. Conceptual framework

3.1.1 Data generation method framework

To generate synthetic data a pipeline is proposed, which is an abstraction containing module which conduct separate tasks needed to generate synthetic data, presented in section 2.4. The pipeline takes text as input and produces an extractive QA data set as output. To get an overview of the pipeline and its tasks a step-by-step visualization is presented in text, but also accompanied by a more visual representation in figure 3.1 with the modules being represented by the boxes. The components to keep track of are: context (C), answer (A), question (Q), and predicted answer (A'). The process will be presented in steps (0-5):

0. Input: C
   1. C → A : Extract answer from C.
   2. C, A → Q : Using C & A generate Q.
   3. C, Q → A' : Using C & Q extract A'. (a normal QA task)
   4. A ↗ A' : Filtering of proposed Q & A based on condition over A & A'.
   5. +C, Q, A' : Add C, Q, A' to the final data set.

In section 3.2 the details will be presented pertaining to steps (0-5) mapping them as: step 0 - Input data, step 1 - Answer Extraction, step 2 - Question Generation, step 3 - Question Answering, step 4 - Roundtrip consistency, and step 5 - Final data set.

For each module, 1-3, different and separate LMs will be used to achieve the specific task. It is also possible to vary the model in each module. Varying the underlying models for the different tasks make it possible to generate several data sets produced under different circumstances, which can then be compared to find the best setup for the generation of data. This will be used to answer RQ 2, since we can vary the models in the different modules to be either trained in the telecom or general domain.

![Diagram of the data generation pipeline and its inherent task-specific modules. The purple boxes represent the usage of a separate LM fine-tuned for each task. In the blue boxes, the different criteria which can be adjusted for some pipeline modules/steps are presented.](image_url)

Figure 3.1: The data generation pipeline and its inherent task-specific modules. The purple boxes represent the usage of a separate LM fine-tuned for each task. In the blue boxes, the different criteria which can be adjusted for some pipeline modules/steps are presented.
Pipeline example

For extra clarification of how this pipeline works in action, here is an example. We start of with a context, for this example we will use the following:

LTE (Long Term Evolution) or the E-UTRAN (Evolved Universal Terrestrial Access Network), introduced in 3GPP R8, is the access part of the Evolved Packet System (EPS). The main requirements for the new access network are high spectral efficiency, high peak data rates, short round trip time as well as flexibility in frequency and bandwidth.

We pass this context to our first language model, fine-tuned for answer extraction. This model will return many extracted answers, where we can choose at what certainty score we keep extracted answers. We also filter out some common oddities that occur, like the start of the next sentence etc. An example of an extracted answer is:

*Evolved Universal Terrestrial Access Network*,

Which would be filtered to:

*Evolved Universal Terrestrial Access Network*

Next, our second language model fine-tuned for question generation is passed both the context and all the extracted and accepted answers. When we pass our example answer, one possible question we could receive is:

*What is E-UTRAN?*

After this we pass the context and the generated question to another model, fine-tuned for question answering. Here the model will try to answer the question provided using the context, and one answer could be:

*Evolved Universal Terrestrial Access Network*

Which is the same answer that was previously extracted. In that scenario our roundtrip consistency would accept the answer and the QA-pair along with the context would be added to the final data set. However, the answer could also be:

*the access part of the Evolved Packet System*

This is also a valid answer to the question, but too different from our original extracted answer and thus it would be rejected from the final data set because of roundtrip consistency.

3.1.2 Evaluation framework

To answer research questions RQ 2 and RQ 3 the experimentation and evaluation will be divided into two separate parts, each one with the aim to answer one of the questions. The evaluation method used for both is a downstream task evaluation.

Downstream task evaluation

The pipelines will be evaluated using a downstream task. To evaluate pipelines using this evaluation method, the generated data is used to fine-tune a model for a specific domain and
task, using both synthetic data and human-annotated data. The reason why both synthetic and human-annotated data are used is to see how additional synthetic data, of presumed lower quality, will affect the downstream model performance. If improvements are shown, one only needs to annotate limited amounts of human-annotated data for fine-tuning, in addition to using the synthetic data. In the case of this thesis, the downstream task is extractive QA within the telecom domain. This evaluation method can also be used to compare the generation methods to each other based on the effect their generated data has on the downstream task performance of the downstream model.

Models

The model architectures that will be used as downstream models are a TeleRoBERTa model and a RoBERTa\textsubscript{BASE} model. These models are chosen to be able to compare to other models performing the same task in a telecom domain.

Metrics

For the evaluation, two metrics will be used: F1-score and EM. These metrics will be used to compare the performance relative to each downstream model and the benchmark model. The F1-score and EM were presented in section 2.1.3.

Pipeline configuration evaluation (RQ 2.)

To answer RQ 2. the pipeline will be used with different underlying models used in the modules, thus generating several synthetic data sets created using different configurations of the pipeline. The ensemble of models used for a specific pipeline instance we will call the model set, which is a set of several models containing one model for each module 1-3 of the pipeline. Other parameters used in the pipeline that are not LMs will be called criteria. This approach was inspired by Sultan et al. which uses several different configurations, of what we call a pipeline, to evaluate the respective performance of data generation techniques [26]. The possible models which are part of a model set for a specific module of the pipeline might have different architectures, will be pre-trained on different types of data, but will be fine-tuned for the module specific task at hand in a similar fashion. A visual representation of the method is presented in figure 3.2.

The synthetic data sets generated will be used to fine-tune two different downstream QA model per data set produced. The performance of these downstream models on a telecom QA task will represent how the different synthetic data set generation pipeline instances perform in comparison to each other. The data used for fine-tuning the QA models will not only consist of synthetic data but the models will first be fine-tuned on synthetic data and then on human-annotated data.

Due to time and computational constraints, the synthetic data sets produced for comparison and evaluation of the different pipeline configurations will be smaller than the data sets presented in section 2.4.5. The pipelines will each get 200k contexts to generate their data sets from. We chose 200k to end up with enough samples in the synthetic data sets, intending to end up at around 100k complete samples in the data sets produced. This approach will capture the entirety of the pipeline’s ability to improve downstream performance, given a set amount of contexts, by both considering quality and quantity of the data produced.

Due to the constraints, we will not evaluate all different combinations of model sets and criteria. Instead, the choice of the best model set and the optimizing of criteria will be done
3.1. Conceptual framework

separately. First, the best model set will be chosen, through evaluation, then using this model set one criterion will be optimized. The best model set together with the optimized criteria is the best configuration for the generation of synthetic data.

Due to the above mentioned time constraints only the score cutoff will be optimized. To optimize the score cutoff a heuristic method will be used where a sample of the data will be used to generate QA-pairs using our pipeline. When the amount of generated QA-pairs starts to decline we will use that score as our optimal cutoff point. This is to generate the synthetic data as efficiently as possible, due to every extra accepted extracted answer being an extra question to generate.

![Diagram of model set evaluation pipeline](image)

Figure 3.2: Model set evaluation pipeline. 200k contexts are provided to a pipeline using one of four different model sets for the pipeline modules. The pipeline produces a synthetic data set that is used to fine-tune a downstream QA model, which is evaluated on the task of telecom QA. The pipeline whose data produce the best downstream performance is selected for further use in the experiments. The orange boxes represent three steps, as can be seen on the right, which are: the generation of data, fine-tuning of downstream model, and evaluation of that model performance on the downstream task.

**Final pipeline evaluation (RQ 3.)**

To answer RQ 3., RQ 2. must first have been answered, resulting in which of the pipeline configurations was best at generating synthetic data. Moving forward the best configuration for the pipeline will be used to generate a large synthetic data set. The aim is to have a comparable amount of samples to the studies presented in section 2.4.5, which is about 3M-11M samples.

This large data set, in addition to human-annotated data, will be used to fine-tune a downstream QA model. The resulting model’s performance on a QA task will then be compared to a baseline QA model’s performance on the same task. The result will help answer RQ 3.
3.2 Experimental setup

In this section, we will present the experimental setup for this thesis in detail. First, some models trained for the telecom domain are presented. These models will be used as a foundation for many model instances in the experiments. Secondly, the data sets used to train the pipeline models, the data used for the generation, and the data used for evaluation are introduced. Thirdly, for each module of the pipeline, the models and criteria will be shown and explained. Lastly, we will discuss how the model sets and criteria will be selected.

3.2.1 Telecom domain model

TeleRoBERTa

The TeleRoBERTa model is a RoBERTaBASE model that was taken from Huggingface. It was pre-trained on the MLM task using English text from five data sets: BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories. All these data sets total 160GB of text. This model then continued pre-training with additional 21,5GB of text data from the telecom domain, mainly 3GPP-specs data, resulting in TeleRoBERTa [24].
3.2. Experimental setup

Baseline model

The baseline model used for the final pipeline evaluation, described in section 3.1.2, is a TeleR-roBERTa model which has been fine-tuned for extractive QA on two data sets: the train split of SQuAD2 and the train split of TeleQuAD2. This model, pre-trained on KELM data, is the current state-of-the-art for the telecom domain QA task using the TeleQuAD2 test split for evaluation. This can be seen in table 3.1.

3.2.2 Data sets

In this thesis project, we will be using several different data sets for the generation of synthetic QA sets, training, and evaluation. These will be described below and presented in table 3.2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Domain</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP-context</td>
<td>Unlabeled</td>
<td>Telecom</td>
<td>60GB JSON files</td>
</tr>
<tr>
<td>SQuAD2</td>
<td>QA</td>
<td>Non-specific</td>
<td>150k QA-sets</td>
</tr>
<tr>
<td>TeleQuAD2</td>
<td>QA</td>
<td>Telecom</td>
<td>4.2k QA-sets</td>
</tr>
</tbody>
</table>

Table 3.2: The data sets used in this thesis

3rd Generation Partnership Program (3GPP)

The 3rd Generation Partnership Program or 3GPP is a partnership program uniting seven standard development organizations active in the telecom industry. 3GPP hosts a large number of specifications and other technical documents specific to the telecom industry. Ericsson has compiled a large amount of these documents that they deem high quality into a data set of contexts, which in this thesis we will refer to as the 3GPP-context data set. In total, this data set is about 60GB of data consisting of JSON files in unlabeled context documents [1, 9].
SQuAD2

Stanford Question Answering Data set 2 or SQuAD2 is a question answering data set from Stanford University consisting of questions from Wikipedia articles and should be answered with a span of text. The complete set consists of 100 000 question-answer pairs together with 50 000 questions that can not be answered from the text provided. This introduces additional complexity to the model by requiring the model to know when it can not answer a question. SQuAD2 will be used to fine-tune models both for the pipeline modules and for the downstream task evaluation [22].

An example context from SQuAD2:
Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny’s Child. Managed by her father, Mathew Knowles, the group became one of the world’s best-selling girl groups of all time. Their hiatus saw the release of Beyoncé’s debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles “Crazy in Love” and ”Baby Boy”.

Example questions for the given context:

When did Beyonce leave Destiny’s Child and become a solo singer?

Ground Truth Answer: 2003

What areas did Beyonce compete in when she was growing up?

Ground Truth Answer: singing and dancing

In what R&B group was she the lead singer?

Ground Truth Answer: Destiny’s Child
TeleQuAD2

Telecom Question Answering Data set or TeleQuAD2 is a SQuAD2-like dataset with domain-specific questions related to the telecom industry. It is designed to evaluate reading comprehension and consists of both answerable and non-answerable questions. It consists of over 4200 questions which were manually annotated by a few (less than 10) employees at Ericsson from paragraphs from the 3GPP-context data set [9].

An example context from TeleQuAD2:

5.5.4 Urban canyon The urban canyon model is switch selectable. When switched on, the model modifies the AoAs of the paths arriving at the subscriber unit. It is for use in both the urban macro and urban micro scenarios. Urban-canyons exist in dense urban areas served by macro-cells, and for at-rooftop micro-cells. When this model is used, the spatial channel for all subscribers in the simulated universe will be defined by the statistical model given below. Thus for the SCM channel generation steps given in Clause 5.3, Step 9 is replaced with steps 9a-d given below, which describe the AoAs of the paths arriving at the subscriber in the urban canyon scenario. The following procedure is used to determine the subscriber mean AoAs of the six paths. This model does not use a building grid, but assigns angles based on statistical data presented in the figures below. The procedure is defined in terms of the subscriber terminal: Step 9a. Select a random street orientation from: U(0, 360) which also equals the direction of travel for the UE.

Example questions for the given context:

<table>
<thead>
<tr>
<th>Question</th>
<th>Ground Truth Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>How is the urban canyon model selected?</td>
<td>switch selectable</td>
</tr>
<tr>
<td>Which scenarios is the urban canyon model used?</td>
<td>both the urban macro and urban micro scenarios</td>
</tr>
<tr>
<td>What is SCM?</td>
<td>&lt;No Answer&gt;</td>
</tr>
</tbody>
</table>

3.2.3 Input data

The data used as input to the pipeline is text data from the 3GPP-context data set. The pipeline takes as input an iterable object of text strings, each representing a context. The data is processed and filtered according to some simple heuristics. For example, the context can not be too short since there is a lower chance for both the AE model and the QG model to produce a relevant example for the data set. To filter out inappropriate contexts we apply a simple rule-based heuristic controlling the length of the context. As a minimum length of the context, we use 50 characters in the string. If the context is shorter than the minimum length the context is disregarded.

Since most documents start with formalities like document name and other structures in common for many documents a filter was also applied to only use paragraphs from further down in the document. We choose to ignore the first 9 lines of each document.

3.2.4 Answer Extraction

For the pipeline module used for AE, four different models will be considered. All of these are based on the RoBERTa architecture, with the differences being the data used for domain-
specific pre-training and task-specific fine-tuning. The tokenizers used are the same corresponding to the specific models used.

Models

The models and the data they are fine-tuned on are presented here.

**RoBERTa - SQuAD2**: Here the base model of RoBERTa\textsubscript{BASE} will be used, with the fine-tuning consisting of training on the SQuAD2 data sets train split.

**RoBERTa - SQuAD2 + TeleQuAD2**: Here the base model of RoBERTa\textsubscript{BASE} will be used, with the fine-tuning consisting of training on firstly the SQuAD2 data sets train split and then secondly the TeleQuAD2 data sets train split.

**TeleRoBERTa - SQuAD2**: Here the base model of RoBERTa\textsubscript{BASE} will be used, with pre-training being performed on the 3GPP-context data set, thus making it a TeleRoBERTa model. Then fine-tuning is performed using the SQuAD2 data sets train split.

**TeleRoBERTa - SQuAD2 + TeleQuAD2**: Here the base model of RoBERTa\textsubscript{BASE} will be used, with pre-training being performed on the 3GPP-context data set, thus making it a TeleRoBERTa model. Then fine-tuning is performed firstly using the SQuAD2 data sets train split and secondly using the TeleQuAD2 data sets train split.

Training

The models used in this section are fine-tuned on various data sets, presented above. This is done by using traditional QA data sets without the questions as input data. When fine-tuning the non-answerable questions of both SQuAD2 and TeleQuAD2 were excluded since we are only interested in actual answers.

Filtering of extracted answers

The filtering approach for the extracted answers is divided into several steps.

**Score filtration**: When computing the extracted answers each answer is given a score by the model indicating how confident the model is in the extracted answer. This score is not on a defined scale and the models, in general, have different distributions in their values. Therefore this cutoff point is going to be optimized heuristically.

**Stop sign filtration**: When extracting answers parts of the following sentence are sometimes included. This behavior is unwanted and to combat it filtration is performed by taking the text before a full stop if one is present. Filtration is also performed if an unclosed parenthesis is included in an answer, where the part before the extracted answer is kept.

**Subset filtration**: The different answers extracted for a specific context are compared to each other by examining if answers have the same start or stop-tokens, and are thus subsets of each other. If an answer is a subset of another answer, the longer answer is kept.

**Similarity filtration**: The different answers extracted for a specific context are compared to each other using `SequenceMatcher` from the `difflib`-library in Python. The answers are then filtered out if the returned value is greater than the similarity threshold, where the longer answer is kept.
Leading & trailing symbol filtration: At the end of the filtration process trailing symbols that are considered junk are filtered out, however only from the start and end of an answer. This is to ensure that the answer is still a span of the context. The symbols are whitespace and comma.

Criteria

The adjustable criteria in the filtering of extracted answers are the score cutoff and the similarity threshold. As stated earlier the score cutoff is the criteria that will be optimized heuristically during the experimentation.

3.2.5 Question Generation

For the QG, two different models will be considered. The models are both based on the model architecture of T5\textsubscript{SMALL} with the difference between them is which data they have been fine-tuned for the QG task. The model used was taken from HuggingFace and had already been pre-trained and fine-tuned for QG using SQuAD2, thus we continued the fine-tuning of the model [28].

When fine-tuning the T5 models the input to the model was a sequence, which represents the context. Within this sequence, the answer span was highlighted using a special marker (<hl>) added around the span. This can be considered the “instruction” part of the T5 input described in section 2.3.3. This highlighting is meant to make the model answer aware and condition the question to be generated based on the answer. The correct output is the question corresponding to the context and answer from the data set. The data used for fine-tuning varies, to be able to offer alternative models to use in the pipeline. One of the models fine-tuned for this task used the train split of the SQuAD2 data set. The other model was a continuation of the first, being fine-tuned further on the train split of the TeleQuAD2 data set. When fine-tuning the non-answerable questions of both SQuAD2 and TeleQuAD2 were excluded since the task at hand is only interested in actual questions.

3.2.6 Roundtrip consistency

For the roundtrip consistency filtering the answer from the AE module of the pipeline is used for comparison. That answer is then compared to the answer generated by the QA model in this module and an F1-score is computed. If this F1-score is higher than our threshold parameter, then the answer from the QA model is kept as the answer to the question. If the F1-score is under the threshold, the QA pair is discarded.

Models

The models used for the QA task which produce the new answer candidate are the following:

**RoBERTa - SQuAD2 + TeleQuAD2**: Here the base model of RoBERT\textsubscript{BASE} will be used, with the fine-tuning consisting of training on firstly the SQuAD2 data set and then secondly the TeleQuAD2 data set.

**TeleRoBERTa - SQuAD2 + TeleQuAD2**: Here the base model of RoBERT\textsubscript{BASE} will be used, with pre-training being performed on the 3GPP-context data set. Then fine-tuning is performed firstly using the SQuAD2 data set and secondly using the TeleQuAD2 data set.
3.2. Experimental setup

Training

All the models used in this section are available to us already pre-trained on the mentioned data. The models were trained on the training set of the mentioned data sets.

Criteria

The only criteria in this module is the minimum F1-score where answers are deemed acceptable. A number between zero and one. This criteria can be changed, but during our experiments will be kept at 0.6.

3.2.7 Non-answerable questions

As established in the section 2.4 there are two approaches commonly used for converting generated questions to non-answerable questions. These are either rule-based or TfIdf based, however, both of these methods have issues associated with them. Both of these methods are rather ineffective at generating questions that are not easy for the model to determine whether or not they are answerable. Because of this, we selected to not generate non-answerable questions in our pipeline.

3.2.8 Model set selection

When selecting the best model set for the pipeline the model sets presented in table 3.3 will be evaluated. Since the purpose of this evaluation is to look into the pipeline’s ability to produce synthetic telecom-specific domain data the data for fine-tuning models have been chosen to represent the spectrum from no telecom-specific training to as much as currently possible.

<table>
<thead>
<tr>
<th>Model set</th>
<th>AE</th>
<th>QG</th>
<th>Roundtrip consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS1</td>
<td>RoBERTa₅</td>
<td>T₅</td>
<td>RoBERTa₅</td>
</tr>
<tr>
<td>MS2</td>
<td>TeleRoBERTa₅₆+T</td>
<td>T₅₆+T</td>
<td>TeleRoBERTa₅₆+T</td>
</tr>
<tr>
<td>MS3</td>
<td>RoBERTa₅₆+T</td>
<td>T₅₆+T</td>
<td>RoBERTa₅₆+T</td>
</tr>
<tr>
<td>MS4</td>
<td>TeleRoBERTa₅</td>
<td>T₅</td>
<td>TeleRoBERTa₅</td>
</tr>
</tbody>
</table>

Table 3.3: The model sets used during model set selection used for the pipeline. Each model set contain the different models used for the different tasks to be performed in the generation pipeline. This is done to measure the impact of pre-training and fine-tuning domain of the models on the generated data. The subscripted letters (S & T) to the models denote on which data they have been fine-tuned on for their respective task. S represents the SQuAD2 data set and T represents the TeleQuAD2 data set.

When running the evaluation of model sets the criteria will be set to initial values. These values are chosen to match the QA-pairs per context ratio in the TeleQuAD2 data set. The QA-pairs per context ratio is regulated using the score cutoff when filtering the extracted answers in the AE module of the pipeline. These initial values will be decided by running small experiments using 10000 contexts. The score cutoff will be adjusted to heuristically achieve the desired amount of QA-pairs per context.

3.2.9 Hardware and software specifications

For all computation (pre-training, fine-tuning, evaluation, and data generation) a remote Microsoft Azure cluster was used. The cluster had access to a single GPU (Tesla P100-PCIE-16GB). The main software tools used include Python (version 3.8.8), PyTorch (version 1.11.0), HuggingFace transformers (version 4.18.0), and HuggingFace datasets (version 2.0.0).
This chapter presents the results collected by experimentation. The main focus will be on the experimentation described in chapter 3, but also some additional exploration of the data and the pipeline will be presented.

4.1 Pipeline model set evaluation

To be able to choose the model set with which to proceed in the experimentation we fine-tuned two different downstream models, one based on TeleRoBERTa and the other based on RoBERTa$_{\text{BASE}}$. The choice to use these two models was made to capture the synthetic data’s impact on downstream performance for models pre-trained in different domains. Each downstream model was first fine-tuned using only synthetic data. After that, the performance of the model was evaluated, to capture the effect of only the synthetic data. Some models then continued fine-tuning using human data sets, either being only fine-tuned on TeleQuAD2 or first on SQuAD2 and then TeleQuAD2. For robustness, the experiment was repeated three times using different seeds. The results for the TeleRoBERTa-based models are presented in table 4.1, and for the RoBERTa$_{\text{BASE}}$-based models in table 4.2. The metrics presented are the EM and F1-scores attained when evaluating the model performance on the downstream task using the evaluation split of TeleQuAD2.
### 4.1. Pipeline model set evaluation

<table>
<thead>
<tr>
<th>Model:</th>
<th>TeleRoBERTa</th>
<th>RoBERTa BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fine-tuning data</td>
<td>EM1</td>
</tr>
<tr>
<td>MS1</td>
<td>46.71, 46.86, 45.73</td>
<td>46.43</td>
</tr>
<tr>
<td>MS2</td>
<td>47.55, 47.69, 46.66</td>
<td>47.30</td>
</tr>
<tr>
<td>MS3</td>
<td>46.37, 47.45, 47.40</td>
<td>47.07</td>
</tr>
<tr>
<td>MS4</td>
<td>43.62, 44.31, 43.87</td>
<td>43.93</td>
</tr>
</tbody>
</table>

|        | EM1 | EM2 | EM3 | EM\text{AVG.} | F1_1 | F1_2 | F1_3 | F1\text{AVG.} |
| MS1 → T | 58.59, 58.49, 57.85 | 58.31 | 78.91 | 79.63 | 78.89 | 79.14 |
| MS2 → T | 59.42, 59.57, 58.44 | 59.14 | 79.06 | 79.30 | 79.06 | 79.14 |
| MS3 → T | 58.88, 58.83, 60.26 | 59.32 | 78.58 | 78.53 | 79.61 | 78.91 |
| MS4 → T | 58.98, 59.43, 60.03 | 59.13 | 79.08 | 79.37 | 79.12 | 79.19 |

Table 4.1: The downstream model performance for the model set evaluation using a TeleRoBERTa model as the downstream task model. The MS\# data sets are the synthetic data sets produced by the different pipelines, each using different model sets. S represents the SQuAD2 data set and T represents the TeleQuAD2 data set. The arrows represent in what order the fine-tuning was conducted for the models.

<table>
<thead>
<tr>
<th>Model:</th>
<th>TeleRoBERTa</th>
<th>RoBERTa BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fine-tuning data</td>
<td>EM1</td>
</tr>
<tr>
<td>MS1</td>
<td>46.76, 47.35, 47.06</td>
<td>47.06</td>
</tr>
<tr>
<td>MS2</td>
<td>47.84, 48.09, 47.94</td>
<td>47.96</td>
</tr>
<tr>
<td>MS3</td>
<td>49.21, 49.12, 49.17</td>
<td>49.17</td>
</tr>
<tr>
<td>MS4</td>
<td>44.55, 46.37, 45.98</td>
<td>45.63</td>
</tr>
</tbody>
</table>

|        | EM1 | EM2 | EM3 | EM\text{AVG.} | F1_1 | F1_2 | F1_3 | F1\text{AVG.} |
| MS1 → T | 60.01, 60.30, 60.40 | 60.24 | 79.72 | 80.69 | 80.38 | 80.26 |
| MS2 → T | 59.13, 59.13, 58.59 | 58.95 | 79.69 | 79.10 | 79.11 | 79.30 |
| MS3 → T | 58.73, 59.62, 60.01 | 59.45 | 78.54 | 78.46 | 78.97 | 79.29 |
| MS4 → T | 59.57, 59.76, 60.06 | 59.80 | 79.69 | 80.01 | 79.91 | 79.87 |

|        | EM1 | EM2 | EM3 | EM\text{AVG.} | F1_1 | F1_2 | F1_3 | F1\text{AVG.} |
| MS1 → S → T | 63.25, 64.57, 64.28 | 64.03 | 82.21 | 83.39 | 83.1 | 82.90 |
| MS2 → S → T | 65.16, 65.21, 64.72 | 65.03 | 83.52 | 83.55 | 83.43 | 83.50 |
| MS3 → S → T | 64.43, 64.13, 65.55 | 64.70 | 83.32 | 83.26 | 84.16 | 83.58 |
| MS4 → S → T | 64.47, 64.87, 64.67 | 64.67 | 83.51 | 83.82 | 83.62 | 83.65 |

Table 4.2: The downstream model performance for the model set evaluation using a RoBERTa BASE model as the downstream task model. The MS\# data sets are the synthetic data sets produced by the different pipelines, each using different model sets. S represents the SQuAD2 data set and T represents the TeleQuAD2 data set. The arrows represent in what order the fine-tuning was conducted for the models.
4.1. Pipeline model set evaluation

4.1.1 Synthetic data set exploration

To gain insight into the synthetic data sets some statistics and smaller experiments were run. The results from these will be presented in this section.

Sample statistics

When generating synthetic data it is of interest to look further into the statistics of the data sets generated. In Table 4.3 statistics about each of the four different model set data sets are presented together with the TeleQuAD2 data set statistics, with a focus on the length of answers and questions.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Statistics</th>
<th>MS1</th>
<th>MS2</th>
<th>MS3</th>
<th>MS4</th>
<th>TeleQuAD2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total amount of samples</td>
<td>159412</td>
<td>93833</td>
<td>93910</td>
<td>86061</td>
<td>4113</td>
</tr>
<tr>
<td></td>
<td>Average answer length</td>
<td>46.54</td>
<td>46.28</td>
<td>47.37</td>
<td>44.19</td>
<td>50.79</td>
</tr>
<tr>
<td></td>
<td>Median answer length</td>
<td>47</td>
<td>47</td>
<td>48</td>
<td>45</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Shortest answer</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Longest answer</td>
<td>133</td>
<td>124</td>
<td>121</td>
<td>120</td>
<td>2069</td>
</tr>
<tr>
<td></td>
<td>Average question length</td>
<td>61.06</td>
<td>65.63</td>
<td>66.06</td>
<td>69.30</td>
<td>50.89</td>
</tr>
<tr>
<td></td>
<td>Median question length</td>
<td>156</td>
<td>61</td>
<td>62</td>
<td>66</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Shortest question</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Longest question</td>
<td>195</td>
<td>195</td>
<td>196</td>
<td>201</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 4.3: Statistics regarding the lengths of answers and questions in the synthetic data sets. All numbers except the total amount of samples represent the number of tokens.

Question analysis

In Table 4.4 an analysis of what type of questions are represented in the data sets is presented, and also what amount of questions are missing a question mark. The analysis counted the occurrence of the question words in the left column. Worth noting is that some questions contain several of the question words, and thus the sum of all amounts of the words overstates the total amount of samples. The same analysis was done for the TeleQuAD2 data set and is also included in Table 4.4. These results are useful to show the difference in questions posed by the different pipelines compared to each other and to humans.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Type of questions</th>
<th>MS1</th>
<th>MS2</th>
<th>MS3</th>
<th>MS4</th>
<th>TeleQuAD2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total amount of samples</td>
<td>159412</td>
<td>93833</td>
<td>93910</td>
<td>86061</td>
<td>4113</td>
</tr>
<tr>
<td>What</td>
<td>134442 (84.3%)</td>
<td>74754 (79.7%)</td>
<td>75392 (80.3%)</td>
<td>72709 (84.5%)</td>
<td>2761 (67.1%)</td>
<td></td>
</tr>
<tr>
<td>When</td>
<td>18355 (11.6%)</td>
<td>15386 (16.4%)</td>
<td>15687 (16.7%)</td>
<td>14366 (16.7%)</td>
<td>308 (7.5%)</td>
<td></td>
</tr>
<tr>
<td>How</td>
<td>9526 (6.9%)</td>
<td>7310 (7.8%)</td>
<td>7028 (7.5%)</td>
<td>4371 (5.1%)</td>
<td>511 (12.4%)</td>
<td></td>
</tr>
<tr>
<td>Where</td>
<td>5709 (3.6%)</td>
<td>1780 (1.9%)</td>
<td>2213 (2.4%)</td>
<td>1821 (2.1%)</td>
<td>252 (6.1%)</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>1565 (0.1%)</td>
<td>182 (0.2%)</td>
<td>1618 (1.7%)</td>
<td>1345 (1.6%)</td>
<td>113 (2.7%)</td>
<td></td>
</tr>
<tr>
<td>Which</td>
<td>835 (0.5%)</td>
<td>829 (0.9%)</td>
<td>837 (0.9%)</td>
<td>811 (0.9%)</td>
<td>149 (3.6%)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>547 (0.3%)</td>
<td>331 (0.3%)</td>
<td>409 (0.4%)</td>
<td>255 (0.3%)</td>
<td>113 (2.7%)</td>
<td></td>
</tr>
<tr>
<td>Whose</td>
<td>64 (0.0%)</td>
<td>60 (0.0%)</td>
<td>69 (0.0%)</td>
<td>70 (0.0%)</td>
<td>1 (0.0%)</td>
<td></td>
</tr>
<tr>
<td>No question mark</td>
<td>7257 (4.6%)</td>
<td>6256 (6.7%)</td>
<td>6783 (7.2%)</td>
<td>8057 (9.4%)</td>
<td>23 (0.6%)</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Statistics about which kind of questions are asked, based on the occurrence of the question words in the samples for the synthetic datasets MS1-4 and the TeleQuAD2 data set.
4.2 Pipeline criteria evaluation

Data quality

To look further into each pipeline’s difference in the quality of generated data, a small experiment was conducted, using an equal amount of data from each synthetic data set. The purpose of the experiment was to capture the pipeline’s ability to produce qualitative data, not having interference of quantity on the performance. A subset of 80k samples from each model set data set was sampled, and used for the first fine-tuning. For the model, we opted for using a RoBERTa\textsubscript{BASE} model as the downstream model to be fine-tuned. For this experiment, three runs were conducted with different seeds to achieve more robust results.

<table>
<thead>
<tr>
<th>Model:</th>
<th>Fine-tuning data</th>
<th>RoBERTa\textsubscript{BASE}</th>
<th>(\text{F1}_{\text{AVG.}})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EM\textsubscript{1}</td>
<td>EM\textsubscript{2}</td>
<td>EM\textsubscript{3}</td>
</tr>
<tr>
<td>MS1\textsubscript{80k}</td>
<td>47.06</td>
<td>46.81</td>
<td>47.40</td>
</tr>
<tr>
<td>MS2\textsubscript{80k}</td>
<td>48.72</td>
<td>48.87</td>
<td>48.38</td>
</tr>
<tr>
<td>MS3\textsubscript{80k}</td>
<td>48.23</td>
<td>49.07</td>
<td>48.58</td>
</tr>
<tr>
<td>MS4\textsubscript{80k}</td>
<td>45.72</td>
<td>45.93</td>
<td>45.44</td>
</tr>
<tr>
<td>MS1\textsubscript{80k} (\rightarrow) T</td>
<td>59.52</td>
<td>60.35</td>
<td>60.06</td>
</tr>
<tr>
<td>MS2\textsubscript{80k} (\rightarrow) T</td>
<td>58.93</td>
<td>59.72</td>
<td>59.27</td>
</tr>
<tr>
<td>MS3\textsubscript{80k} (\rightarrow) T</td>
<td>59.91</td>
<td>59.37</td>
<td>59.96</td>
</tr>
<tr>
<td>MS4\textsubscript{80k} (\rightarrow) T</td>
<td>58.54</td>
<td>59.08</td>
<td>59.42</td>
</tr>
</tbody>
</table>

Table 4.5: The downstream model performance for the data quality evaluation using a RoBERTa\textsubscript{BASE} model as the downstream task model. The \(MS\#_{80k}\) data sets are the subset (80k samples) of the synthetic data sets produced by the different pipelines. \(T\) represents the TeleQuAD2 data set. The arrows represent in what order the fine-tuning was conducted for the models.

4.2 Pipeline criteria evaluation

To evaluate the cutoff score, in the AE module of the pipeline, to use for the final evaluation of the data generation pipeline a heuristic method was used where different cutoff scores were used on the same sample of 10k contexts. A pipeline using model set 1 was given the same 10k context to generate from. In table 4.6 the number of questions produced from the contexts is presented for the different levels of the cutoff score.

<table>
<thead>
<tr>
<th>Cutoff score</th>
<th>Questions produced</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>7870</td>
</tr>
<tr>
<td>3.5</td>
<td>9368</td>
</tr>
<tr>
<td>3.6</td>
<td>8767</td>
</tr>
<tr>
<td>3.75</td>
<td>7586</td>
</tr>
<tr>
<td>4.0</td>
<td>5065</td>
</tr>
</tbody>
</table>

Table 4.6: Cutoff scores for the MS1 evaluation with the number of questions generated from the same sample of 10k contexts.
4.3 Final evaluation

For the final evaluation, all available context data from the 3GPP-context data set was used to produce a large synthetic data set. The model set chosen for the final evaluation was model set 1. This was because of model set 1 achieving good results, in combination with producing the largest amounts of data with the highest quality according to our experiments. We used a cutoff score of 3.5. The size of the final data set ended up at around 4.4M samples. Using this data set fine-tuning was done with two models, TeleRoBERTa and RoBERTa_{BASE}, using the full synthetic data set. Similar to when evaluating model set performance the model was evaluated after only being fine-tuned on synthetic data, after being fine-tuned further on TeleQuAD2, and being further fine-tuned on SQuAD2 & TeleQuAD2. The performance for the TeleRoBERTa model when evaluating on the telecom QA task is presented in table 4.7, and for the RoBERTa_{BASE} model in table 4.8.

<table>
<thead>
<tr>
<th>Model: TeleRoBERTa</th>
<th>Fine-tuning data</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinalData</td>
<td>47.74</td>
<td>71.94</td>
<td></td>
</tr>
<tr>
<td>FinalData → T</td>
<td>55.89</td>
<td>77.46</td>
<td></td>
</tr>
<tr>
<td>FinalData → S → T</td>
<td>62.81</td>
<td>82.22</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: The downstream model performance for the final evaluation using a TeleRoBERTa model as the downstream task model. The FinalData data set is the synthetic data set produced by the pipeline using model set 1. S represents the SQuAD2 data set and T represents the TeleQuAD2 data set. The arrows represent in what order the fine-tuning was conducted for the models.

<table>
<thead>
<tr>
<th>Model: RoBERTa_{BASE}</th>
<th>Fine-tuning data</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinalData</td>
<td>47.94</td>
<td>71.89</td>
<td></td>
</tr>
<tr>
<td>FinalData → T</td>
<td>57.36</td>
<td>78.55</td>
<td></td>
</tr>
<tr>
<td>FinalData → S → T</td>
<td>63.40</td>
<td>82.60</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.8: The downstream model performance for the final evaluation using a RoBERTa_{BASE} model as the downstream task model. The FinalData data set is the synthetic data set produced by the pipeline using model set 1. S represents SQuAD2 and T represents TeleQuAD2. The arrows represent in what order the fine-tuning was conducted for the models.
4.4 Model size experiment

To investigate how the size of the downstream model might affect the impact of using synthetic data, two different size models were trained using the whole or parts of the FinalData data set. To capture the effect using a larger model RoBERTa\textsubscript{LARGE} was used and for a slightly smaller model, DistilRoBERTa was used. The models were trained on parts of the FinalData data set, and intermediate instances were saved during the training. This was done to investigate how fine-tuning on different amounts of synthetic data might affect the downstream model performance. The results for the RoBERTa\textsubscript{LARGE} model are presented in table 4.9 and the results for the DistilRoBERTa model are presented in table 4.10. In figure 4.4 the EM and F1-scores from table 4.9 & 4.10 are visualized.

<table>
<thead>
<tr>
<th>Model:</th>
<th>RoBERTa\textsubscript{LARGE}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuning data</td>
<td>EM</td>
</tr>
<tr>
<td>FinalData\textsubscript{100K}</td>
<td>48.53</td>
</tr>
<tr>
<td>FinalData\textsubscript{200K}</td>
<td>46.42</td>
</tr>
<tr>
<td>FinalData\textsubscript{300K}</td>
<td>45.24</td>
</tr>
<tr>
<td>FinalData\textsubscript{400K}</td>
<td>45.29</td>
</tr>
<tr>
<td>FinalData\textsubscript{500K}</td>
<td>45.53</td>
</tr>
<tr>
<td>FinalData\textsubscript{600K}</td>
<td>45.53</td>
</tr>
<tr>
<td>FinalData\textsubscript{700K}</td>
<td>44.60</td>
</tr>
<tr>
<td>FinalData\textsubscript{800K}</td>
<td>46.27</td>
</tr>
<tr>
<td>FinalData\textsubscript{900K}</td>
<td>46.07</td>
</tr>
<tr>
<td>FinalData\textsubscript{1M}</td>
<td>46.61</td>
</tr>
</tbody>
</table>

Table 4.9: The downstream model performance for an experiment of impact for different sizes of downstream models. In this table, the results from using a RoBERTa\textsubscript{LARGE} model as the downstream task model are presented. The FinalData data set is the synthetic data set produced by the pipeline using model set 1. The model was trained on different amounts of data from the FinalData set, the number of samples is written in the subscript to the data set name.

<table>
<thead>
<tr>
<th>Model:</th>
<th>DistilRoBERTa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuning data</td>
<td>EM</td>
</tr>
<tr>
<td>FinalData\textsubscript{400K}</td>
<td>46.27</td>
</tr>
<tr>
<td>FinalData\textsubscript{800K}</td>
<td>46.61</td>
</tr>
<tr>
<td>FinalData\textsubscript{12M}</td>
<td>44.95</td>
</tr>
<tr>
<td>FinalData\textsubscript{16M}</td>
<td>45.24</td>
</tr>
<tr>
<td>FinalData\textsubscript{2M}</td>
<td>45.29</td>
</tr>
<tr>
<td>FinalData\textsubscript{24M}</td>
<td>45.34</td>
</tr>
<tr>
<td>FinalData\textsubscript{28M}</td>
<td>44.60</td>
</tr>
<tr>
<td>FinalData\textsubscript{32M}</td>
<td>46.71</td>
</tr>
<tr>
<td>FinalData\textsubscript{36M}</td>
<td>46.03</td>
</tr>
<tr>
<td>FinalData\textsubscript{4M}</td>
<td>46.32</td>
</tr>
</tbody>
</table>

Table 4.10: The downstream model performance for an experiment of impact for different sizes of downstream models. In this table, the results from using a Distilled RoBERTa model as the downstream task model are presented. The FinalData data set is the synthetic data set produced by the pipeline using model set 1. The model was trained on different amounts of data from the FinalData set, the number of samples is written in the subscript to the data set name.
4.4. Model size experiment

**RoBERTa\textsubscript{LARGE}**

<table>
<thead>
<tr>
<th>Data</th>
<th>EM &amp; F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>200k</td>
<td>80</td>
</tr>
<tr>
<td>300k</td>
<td>75</td>
</tr>
<tr>
<td>400k</td>
<td>70</td>
</tr>
<tr>
<td>500k</td>
<td>65</td>
</tr>
<tr>
<td>600k</td>
<td>60</td>
</tr>
<tr>
<td>700k</td>
<td>55</td>
</tr>
<tr>
<td>800k</td>
<td>50</td>
</tr>
<tr>
<td>900k</td>
<td>45</td>
</tr>
<tr>
<td>1M</td>
<td>40</td>
</tr>
</tbody>
</table>

**DistilRoBERTa**

<table>
<thead>
<tr>
<th>Data</th>
<th>EM &amp; F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>800k</td>
<td>90</td>
</tr>
<tr>
<td>1.2M</td>
<td>85</td>
</tr>
<tr>
<td>1.6M</td>
<td>80</td>
</tr>
<tr>
<td>2M</td>
<td>75</td>
</tr>
<tr>
<td>2.4M</td>
<td>70</td>
</tr>
<tr>
<td>2.8M</td>
<td>65</td>
</tr>
<tr>
<td>3.2M</td>
<td>60</td>
</tr>
<tr>
<td>3.6M</td>
<td>55</td>
</tr>
<tr>
<td>4M</td>
<td>50</td>
</tr>
</tbody>
</table>

Figure 4.1: Tables 4.9 and 4.10 visualized.
5 Discussion

In this chapter, we will be discussing both the results and the method of the thesis; as well as commenting on the replicability, reliability, and validity of the study. We will also be discussing our sources, and how this thesis fits into a wider context.

5.1 Results

In this section, the results of the thesis will be discussed, with the purpose to present a perspective on the results that were obtained, what they might mean, and why these results were obtained.

5.1.1 Pipeline model set evaluation

Importance of F1 & EM-score

When evaluating how the different model sets perform when used in our data generation pipelines there are several metrics to consider. The ones we have chosen are F1 and EM-score, however, this choice turned out to raise an issue. When evaluating our model sets, some model sets improved EM-score more than they improved F1-score and vice versa. This led to the difficult choice of which metric to prioritize. We reasoned that the F1-score should be prioritized higher due to a few factors.

Firstly, when analyzing the task of QA in the telecom domain it can be observed that the answers are generally longer than general domain answers. This means that it should be harder for models to achieve an exact match since the spans of tokens are longer, and achieving an exact match requires a perfect answer matching every token. This leads to us prioritizing an answer that is mostly correct instead of requiring perfection.

Observing the data used to train and evaluate performance, the TeleQuAD2 data set, we also considered a few factors. One of these is that the data set was created by a few Ericsson employees, often in their spare time between other tasks. We find that there is not a lot of quality control for this data set and this could lead to lower quality in the data set which
would further influence our choice of prioritizing F1-score due to faulty answers or questions debilitating EM-score further. Inconsistencies in the data can be a result of annotators might define the ideal answer span differently.

Likeness of results

When observing the results of the different evaluations the performance of the different model sets are very similar, and most of the time the difference is less than one percentage point. This made the choice of the final model difficult. It also provides a challenge when answering the second research question, and even though the fine-tuning was done using different seeds to increase the robustness of the results drawing a decisive conclusion is difficult.

One thing that can be observed is that the size of the data set generated by MS1 is substantially larger than the other data sets due to the difference in model set in regards to extracting of answers and passing roundtrip consistency. This model set is the model set that has been trained on the least amount of telecom domain data, and it seems that this allowed the model to be more creative for lack of a better term. This is interesting if we look at the results since this could imply that the domain-specific fine-tuning provides a quantity-quality problem where there might be an optimal amount of training for the pipeline where both quality and creativity are maximized. However, the results from our data quality study at least partially debunk this idea.

Were the results expected?

From our results, we concluded that MS1 was the best model set in our evaluation. However, this result could be considered quite unexpected as this is the model set with no telecom-specific pre-training or fine-tuning at all. But, as previously mentioned this is the model set that performed best through our evaluation, as well as the model set that performed the best during our data quality ablation study (section 4.1.1). One factor to this that we have discussed is that this model set seems to be more creative, and we observed this from the large increase in the quantity of the data generated from this model set. This creativity might not only be expressed in the quantity of the data since the results of the data quality ablation study (section 4.1.1) seems to indicate that the quality of the data generated from MS1 was also of the highest quality. This indicates that the QA-pairs generated are also more creative and thus of higher quality, but this is something that would require more research.

Another result that could be considered unexpected is that both MS3 and MS4 performed poorly when compared to the other model sets. This is unexpected since these models were our middle ground in terms of the telecom domain data the models saw. The expectation was that these models would perform somewhere in the middle of MS1 and MS2 but this was not the case. One thing that could be stated is that all model sets except MS1 use the T5 model with fine-tuning on TeleQuAD2 and that this model could be the culprit of the results. Therefore it would be interesting to run some other model sets through the pipeline, where an example could be using TeleRoBERTa fine-tuned on the SQuAD2 data set and the TeleQuAD2 data set for the answer extraction and roundtrip consistency but keep the regular T5 model for the question generation to see if this combination of general domain and telecom domain provides better results.

Performance improvement

Observing the results of the evaluations using both TeleRoBERTa and RoBERTa for the downstream task of QA and comparing this to the other results of similar models on the Tele-
QuAD2 data set something interesting can be observed. This is, while our generated data improves performance for both of the models for the evaluation, the increase is much more significant when the RoBERTa model is fine-tuned. This can partially be explained due to the generated data being the only telecom specific data RoBERTa has been exposed to except for the training part of TeleQuAD2.

Likeness of data sets

If we look at the question analysis in section 4.1.1 it can be observed that the distribution for the data sets generated by the pipeline through the different model sets are similar, except for MS1. As previously stated, the main difference here is that MS1 is the only model set that does not use the T5 model fine-tuned on TeleQuAD2 for question generation. This should not impact the number of questions generated that is dictated by the answer extraction model, however, it seems to indicate that the questions generated are of lower quality since more QA-pairs are lost in the roundtrip consistency step through all of the model sets except MS1 which seems to indicate a lower quality of questions generated.

Advantages of general domain model sets

If our results correctly imply that general domain model sets provide equal or better results than domain-specific model sets then that has interesting implications. First of all, it would imply that the domain-specific data in the contexts provided to the pipelines is enough for adaptations between domains, which would allow a general-domain pipeline to generate synthetic data for a wide array of domains without modifications. This might reduce the need for domain pre-training by just feeding the pipeline with raw domain-specific text data used for generation.

One data set to rule them all

If the results of the fine-tunings of both TeleRoBERTa and RoBERTa are compared it can be stated that the performance is similar, if not better for the RoBERTa model even though TeleRoBERTa has been exposed to an additional 21.5 GB of telecom domain data. This might be related to the model not being large enough to achieve higher performance, but if this is not the case it means that our pipeline might enable models to both adapt to a task and a domain at the same time. This could eliminate the need for time-consuming and expensive pre-training sessions with additional fine-tuning and enable us to generate one data set to rule them all of a specific task and domain. However, even though the pipeline seems to work as well or better with general domain models the pipeline would still need to be adapted for every specific task.

Time and computational efficiency

As previously mentioned, comparing the performance of the fine-tuning in the pipeline model set evaluation, section 4.1, we see that the performance is very similar between the RoBERTa and TeleRoBERTa models. This is interesting as this realization could save time and computational costs for domain adaptation since no extensive pretraining is required. If we compare the total time and cost of both models (looking domain pre-training and fine-tuning), the RoBERTa model took about 8 hours for the data generation and 6 hours for the fine-tuning. If we compare this too the TeleRoBERTa model, there was about 20 hours of pre-training as well as 3 hours of fine-tuning. This difference in cost is significant, and due to the supposed adaptability of the generation pipeline to many different domains this approach could be a good approach if efficiency is important.
5.2. Method

5.1.2 Pipeline criteria evaluation

Cutoff score

When evaluating the criteria of the cutoff score a heuristic approach where different cutoff scores were used on the same sample of 10k contexts. Here, the expectation was that as we increase the cutoff score we would have fewer questions in the final data set and that the optimal point was just before the drop-off in questions. However, what we found out was that there was a point where we increased the cutoff score and the final amount of questions also increased. This leads to us achieving even better performance than during our first evaluations, with us being able to generate more questions per pipeline iteration while shortening execution time. We believe that this is because of the higher quality of the original answers leading to better QA-pairs that pass roundtrip consistency. This would however be an interesting topic to look further into.

5.1.3 Final evaluation

Looking at the results from the final evaluation it can be observed that the increased amount of data did not contribute to better results. This could either be attributed to the generated data not being of high enough quality or the model not having enough parameters to be able to use the additional data. This hypothesis is strengthened since all of the results are very similar when put in relation to the non-answerable questions the models were exposed to. If we look to the related works, other authors who have done similar studies generally use much larger models for the downstream task.

Another possible scenario is that our generated data is still not enough to train the rather large model in a satisfactory manner.

A third possible scenario is that the data generated by the pipeline is very generic, and always poses the same types of questions. This could give training on our generated data diminishing returns on the training, which would explain why training on our smaller sample and our full data set gives similar performance.

To investigate these three scenarios an additional study was conducted (section 4.4). Here a distilled RoBERTa model was fine-tuned using all of the generated data, with 10 checkpoints being saved. These ten models were then evaluated on the TeleQuAD2 data set to investigate how the amount of data affected a smaller model. We then fine-tuned a RoBERTaLARGE model with 1 million of the generated QA-pairs because of time restraints. Here was also saved 10 checkpoints and evaluated them using TeleQuAD2.

Both of these experiments had the same results, in that additional data did not grant additional performance. Because of this it seems like the model size does not impact how useful the data is. This leads us to believe that it is the third scenario that is causing the lack of performance, which is that the data generated is not of high enough quality and probably too generic. It can however be stated that the model does not decrease in performance either.

5.2 Method

In this section, the method of the thesis will be discussed, with the purpose to present a perspective on the choices made and potential improvements to the method used for this thesis.
5.2. Method

5.2.1 Non-answerable questions

One aspect of the QA task that was not implemented and evaluated was the generation of non-answerable questions and the potential effect of synthetic non-answerable questions on the downstream task performance. This was ignored because of the results from previous work showing that there is as of yet no promising approach to tackle this generation task. It is however an aspect that, if done right, could potentially improve the task adaption of a model to be more precise when training with synthetic data. It is an interesting avenue for research to find valid approaches to generating non-answerable samples, which combined with the approach presented here and in other works could benefit all research concentrating on the task of QA. The potential benefit is to further improve the training effect of the synthetic data set, and possibly improve overall model performance since the model learns earlier in training to recognize non-answerable samples.

If a good approach to generating non-answerable questions is found, or is investigated, it can either be added on as a module to the pipeline or as a completely separate process which can be applied to any QA data set (human annotated or synthetic).

There is also the question of how necessary non-answerable questions are in a generated data set, and while there is no conclusive research on the importance, research has shown that state-of-the-art results can be achieved without them. This is partially based on the assumption that the final fine-tuning on the SQuAD2 training data set has enough data for the model to understand the concept of not answering a question. As demonstrated in section 2.4.5 the inclusion of non-answerable questions does not necessarily increase performance and because of this, we decided to not convert a partial amount of the generated questions to non-answerable questions. A method that could be used is to compare the results from not converting questions at all, converting some questions while deleting the original converted questions, and converting questions while keeping the originals. This would give a better perspective on the importance of non-answerable questions while serving as an ablation study for the subject.

5.2.2 Downstream task evaluation

The choice of using a downstream task evaluation to compare the performance was chosen for several reasons. One reason was the fact that it had been used by related works earlier to measure the effect of using synthetic data, but also the difficulty in measuring the individual performance of each separate module in the pipeline, e.g. AE or QG. During our thesis, some thoughts were brought up to try to tackle the problem of individual evaluation of pipeline modules, but none were too promising. This however could be an interesting area of research. If methods for evaluating each pipeline module’s performance were developed, one could more easily improve the pipeline without relying on the whole pipeline performance through a downstream task. If methods are found to evaluate the different modules of the pipeline the downstream task could still be a helpful tool to make sure the overall result increases and the pipeline is not getting worse by locally optimizing a single part of the pipeline.

When fine-tuning the downstream model for the evaluations and experimentation we opted for using annotated data from the general domain (SQuAD2) and also from the telecom domain (TeleQuAD2). The fine-tuning also happened in sequence, first training on synthetic telecom data, then the general domain human data, and lastly the telecom domain human data. We chose to use SQuAD2 to be able to capture the non-answerable aspect of a QA task.
since those examples were not present in the synthetic data and the TeleQuAD2 data set was
demed too small. It would be interesting to see the effect of using only the non-answerable
samples from SQuAD2, as to capture that aspect of QA since the synthetic data did not in-
clude non-answerable samples, but maybe not confuse the model by re-introducing general
domain data or causing catastrophic forgetting of what the model has learned in terms of the
domain. The effect of fine-tuning using a large general domain QA data set is unknown, and
it would be valuable to measure this effect. Other constellations of data set for fine-tuning
could yield other results e.g. mixing all the samples from different data sets, or changing the
order etc.

5.2.3 Model set selection
Due to time constraints, it was not possible to try all combinations of models in the pipeline,
but considering the results that might not have yielded more clarity since the results are quite
similar. What would have been interesting, as mentioned earlier is further exploring the
effect of using the general domain T5 model, used in model set 1, in combination with e.g.
the telecom domain models otherwise used in model set 2.

5.2.4 Quality-quantity trade-off
Something to consider when doing the experiments is what ability of the pipeline is featured.
What we were looking at was the pipeline’s ability to produce data, thus both the quality
and the quantity of samples different pipelines could produce, and their ability to improve
the downstream model’s performance. This can be seen in the model set evaluation, where
each pipeline got the same contexts and the resulting data set, no matter the size, was used
for fine-tuning the downstream model.

While our chosen method does not completely ignore the quality of the data, it does not
capture the pure effect of quality or quantity. With that said, our approach is more focused
on the pipeline’s capabilities as a whole, since the quantity of samples produced and their
effect on the downstream model is arguably important and interesting when applying the
pipeline in a real-world context. The quality-quantity trade-off and the total effect of the
pipeline’s ability to improve downstream performance is especially important if there is a
limited amount of contexts to generate from.

An ablation study was conducted to look into the quality of data. In the study the same
amount (80k) of the synthetic data from each model set data set were used for fine-tuning.
This ablation study is presented in section 4.1.1. From the result of this ablation study a slight
difference in quality can be observed, with MS1 being the best.

5.2.5 Other pipeline criteria
In our experiments on the pipeline, only one pipeline criterion was evaluated. The focus was
on the cutoff score of the AE model, indicating how certain the model was of the extracted
answer. This score controlled the number of answers to proceed with. However, one criterion
found later in the pipeline was not evaluated, and that is the F1-score used as an acceptance
criterion in the roundtrip filtering module. It could be interesting to further evaluate the effect
of this criterion if set at different levels, to see the effect of the quantity of samples let through
and the quality of the samples.
5.2.6 Filtration in AE

The length of an extracted answer could affect the questions generated to an answer. One could imagine that longer answers might lead to the generation of more open-ended or general questions. If true, this would be an important aspect to consider when filtering answers, since it might have an impact on the model’s ability to handle long vs. short answers and precise vs. open-ended questions.

For this thesis when filtering answers in the AE the longer answer was preferred when two answers were too similar. This can have impacted the generated data and thus also affected the downstream model performance. This however was not looked into, and further study would be needed to both look into the validity of the assumption described above, and the impact on the synthetic data set. Another change to the filtering done in AE could be to filter by length, only accepting answers that are over a certain length threshold and below another.

5.2.7 Answer choice during roundtrip

One aspect not addressed by this thesis, when using roundtrip consistency, is why the answer from the QA model (which we have denoted $A'$) is to be preferred over the answer extracted in AE (which we have denoted $A$). The intuition of this choice is that $A'$ is made conditional on more information compared to $A$, maybe making it a better candidate.

The approaches to roundtrip consistency differ in the related works. In the paper from Alberti et al. [2], the criteria for comparison between $A$&$A'$ was EM, which meant it did not matter which of the answer candidates they used. Dong et al. [8] however use the F1-score from comparing $A$&$A'$, and keeps $A'$ if it is over a threshold ($F1=0.6$). It might be interesting to keep both answer candidates to see the contrast between them when allowing for some differences, and if the usage of one candidate has a notable effect on the generated data set over the other.

5.2.8 Model choice for pipeline modules

Here some thoughts about the models chosen for the different pipeline modules will be discussed. During the design of the pipeline, there were options and potential models to use for the different models, and we will discuss why we made the choice we did and potential improvements to our approach.

Answer Extraction

When choosing which model to use for AE it was an option to use GPT-3 instead of a BERT-like model. The argument for this option was that GPT-3 is a very large LM which could have a good effect on the resulting extracted answers. It was thought that GPT-3 should be able, with the right prompt, to perform this task since it is capable of performing a generative QA task. This idea was however not pursued due to three reasons, the first being that the answer produced by GPT-3 does not necessarily have to be a span of text from the context document, due to the answer being generated and not in the same way extracted from the context document. Secondly, it is not possible to fine-tune GPT-3 to the telecom domain, in the same manner, we did for our experiments, which would have made it hard to answer the research questions posed about the effect of domain-specific models in the pipeline. Thirdly, with the GPT-3 model it was also a question about access since a trained GPT-3 model is only
accessible through an API. Depending on speed of access and response from the API this may have made the pipeline slower than it is today.

**Question Generation**

For the pipeline model for QG, there were some options to consider when choosing a model. One option was to use a BERT-like model for this module. This was disregarded since a sequence-to-sequence model was deemed more fitting for the task of QG due to the architecture. The other option would be to use GPT-3, which had the potential to work great since it is so large and did seem to be a good candidate. The downside of using GPT-3 would be that it would not be possible to fine-tune the model to be a bit more in the telecom domain, which is why we opted for the small T5 model used in this thesis.

But considering both our results showing that the domain of the pipeline models does not seem to make a significant difference, and the possibility of obtaining better results, a potential improvement of the QG pipeline would be to use GPT-3 for the QG task.

**5.2.9 Replicability, reliability, and validity**

In regards to replicability, this study has some issues, which we will address here. The main concern lies in the fact that some resources used for experimentation are owned by Ericsson and not publicly shared. The resources are both models used, which have been trained in the telecom domain (TeleRoBERTa models), access to the TeleQuAD2 & 3GPP-context data set, and the code written to perform the experiments. To be able to recreate this experiment one would need the Ericsson resources, which limits the possibility of perfect reproduction to employees of Ericsson. As for the scripts we have tried to follow design patterns and code published on HuggingFace as inspirations for our code. While the exact implementation is the property of Ericsson, similar results should be obtained using an implementation following the principles presented in this thesis.

In terms of reliability, we tried to make the results as robust as time permitted by running each of the experiments, except the final evaluation, which included fine-tuning the downstream model three times and averaging the performance on the QA task. We consider this to have added to the robustness and reliability of the results by excluding some randomness, but not all.

When it comes to validity the method chosen leaves things to be desired. By not being able to compare the individual model performance of the pipeline models, it is hard to draw any concrete conclusions about the effect the model domain has on performance. Instead, the pipeline is evaluated as a whole. In addition to this, there is also the issue of machine learning models largely being black boxes, leaving us to simply observe the results without being able to observe how these results are obtained. This further leaves the validity of the thesis lacking since we can only guess why things are happening without the knowledge of what goes on behind the scenes.

**5.2.10 Source critique**

When reading research about topics relating to our field of study and related works in regards to the topic at hand we mainly tried to keep sources that have been peer-reviewed and accepted by journals that are regularly publishing work in the field of NLP. This has however not always been possible. Some of the papers are published using public publishing options
or by companies themselves. In these cases, we mainly looked at the authors and their associated companies to try and determine the validity of the paper. In other works where it was harder to determine we had to rely on the number of citations and thus whether other researchers deemed the paper as valid, within the field of NLP.

5.3 The work in a wider context

Propagating bias

When generating synthetic data using LMs it comes with a risk of propagating and possibly amplifying bias that is present in the models used for generation. In our work, these biases might come from different sources such as the pre-training of the models used or the fine-tuning. When we use these models to produce data from raw text the data produced might be influenced by this bias. Since the methods we present give the possibility of producing large data sets that can be used for training, the bias transferred from the generating model can propagate to the task-specific model, and be enhanced, due to the sheer size of the data sets generated.

Different biases are present in different domains and have the varying potential for harm when present in real models. Since the thesis presents methods for generating synthetic data which can be applied more generally for different domains, it is important to keep in mind the possible propagation of biases for each certain domain when used.

Energy consumption

Training model in the field of ML & NLP, which use models with many parameters, is quite expensive and consumes a lot of energy. This thesis might reduce some of the energy consumed, which was touched upon earlier in the discussion. If it is true that fine-tuning a model using synthetic data can prime a model for a domain equally or better than pre-training the model in that domain, it could be beneficial since the pre-training a model might consume more energy in total compared to generating and fine-tuning a model.
The purpose of this thesis was to investigate and evaluate approaches to data generation in a telecom domain for the task of extractive QA. To do this we built a pipeline using a combination of BERT-like models and T5 models for data generation. We then evaluated our generated data using the downstream task of QA on a telecom domain data set.

6.1 Research questions

Here we will attempt to answer our research questions. As a reminder to the reader the RQs were the following:

1. What possible approaches can be used to generate synthetic data sets for extractive QA, given context documents?

2. How do the models trained on synthetic data generated from a general domain pipeline perform compared to the models trained on synthetic data generated from a pipeline pre-trained on telecom domain data?

3. How do the models trained on synthetic data in addition to human-annotated data perform compared to the baseline model trained only on human-annotated data?

6.1.1 Research Question 1

RQ1 was posed with the purpose of presenting different methods of how to approach the task of synthetic data generation for the task of extractive QA. This RQ was thus already answered in chapter 2.4, where we present some different approaches to this problem. In short, it can be said that several previous efforts that have shown promising results use LMs to perform sub-tasks, which combined produces a synthetic data set. We chose to employ a similar tactic in our method for this thesis.

6.1.2 Research Question 2

RQ2’s purpose was to investigate if it mattered which domain the LMs used for the generation tasks had been trained in originally, and how this would affect their ability to produce
telecom-domain data. In our study, we found that it did not matter much which domain the LMs were trained in and that the downstream model used for evaluation performed equally well or better when trained on data from models originating in the general domain.

6.1.3 Research Question 3

For RQ3 the purpose was to look into the actual effect of using synthetic data when fine-tuning a model. What was found from our study is that using synthetic data does improve the performance of a QA model. We also found that the synthetic data plays a role in adapting the model to the task and possibly domain, although the domain adaptation was not measured. It is however hard to tell exactly how much improvement can be expected and how much data is needed to improve. In terms of the amount of data, we found that a rather small amount of synthetic data was sufficient to obtain the improvement, and that using more data resulted in diminishing returns.

6.2 Future work

One interesting topic in future work would be to generate more than one question per extracted answer. This would increase how costly the pipeline is to run but could give better results and it would be interesting to see how much larger the data set would become since many questions could possibly still disappear in roundtrip consistency. It would also be interesting to see if this would give greater variance or creativity in the questions since the model’s first choice of question is the safest bet. This is however very difficult to measure.

Another interesting topic for future work is to run the pipeline using different models. Perhaps the pipeline could be improved in its quality or efficiency by using smaller or larger models in different parts of the pipeline. It would also be interesting to see how our evaluation would perform using much larger models than we used.

In the roundtrip consistency, we chose to follow the existing literature and use an F1-score of 0.6 for the threshold and to keep the new answer $A'$. It would be interesting to run the pipeline using different configurations of both thresholds and which answer to keep and evaluate on the downstream task. Roundtrip consistency in general is also a concept with a lacking amount of research, so this or any other analysis of the concept would make for important future work.

6.3 Final conclusion

To conclude, our thesis has found that synthetic data generation is a viable approach to generating synthetic telecom QA data with the potential of improving model performance when used in addition to human-annotated data. It was also found that synthetic data can be used for task and possibly domain adaptation of a model, although the domain adaptation was not measured. We found that using models from the general domain provided results that are on par or better than domain-specific models for the generation, which provides possibilities to use a single generation pipeline for many different domains. We also found that increasing the amount of synthetic data provided little benefit for our models on the downstream task, with diminishing returns setting in quickly. We were unable to pinpoint the reason for this, but we speculate that it is because of low data quality. In short, our approach works but much more work remains to understand and optimize it for greater results.
Bibliography


