UPPSALA
UNIVERSITET

UPTEC IT 21003
Examensarbete 30hp
30 mars 2021

Large-Context Question Answering with Cross-Lingual Transfer

Markus Sagen

Institutionen för informationsteknologi

Department of Information Technology

Abstract

Large-Context Question Answering with Cross-Lingual Transfer

Markus Sagen

Models based around the transformer architecture have become one of the most prominent for solving a multitude of natural language processing (NLP) tasks since its introduction in 2017. However, much research related to the transformer model has focused primarily on achieving high performance and many problems remain unsolved. Two of the most prominent currently are the lack of high performing non-English pre-trained models, and the limited number of words most trained models can incorporate for their context [53, 54]. Solving these problems would make NLP models more suitable for real-world applications, improving information retrieval, reading comprehension, and more. All previous research has focused on incorporating long-context for English language models. This thesis investigates the cross-lingual transferability between languages when only training for long-context in English. Training long-context models in English only could make long-context in low-resource languages, such as Swedish, more accessible since it is hard to find such data in most languages and costly to train for each language. This could become an efficient method for creating long-context models in other languages without the need for such data in all languages or pre-training from scratch. We extend the models' context using the training scheme of the Longformer architecture and fine-tune on a question-answering task in several languages.

Our evaluation could not satisfactorily confirm nor deny if transferring long-term context is possible for low-resource languages. We believe that using datasets that require long-context reasoning, such as a multilingual TriviaQA dataset, could demonstrate our hypothesis's validity.

Handledare: Philipp Eisen, Anders Arpteg
Ämnesgranskare: Joakim Nivre
Examinator: Lars-Åke Nordén
UPTEC IT 21003
Tryckt av: Reprocentralen ITC

Contents

1 Introduction **2**
1.1 Purpose and Goals 4
1.2 Research Questions 4
1.3 Thesis Outline 5
1.4 Miscellaneous 5

2 Background **6**
2.1 Natural Language Processing 6
2.1.1 Tokenizers 6
2.1.2 Input embedding 7
2.1.3 Language Modeling 7
2.1.4 Encoder-Decoder Architectures 8
2.1.5 Question Answering Tasks 8
2.2 Transfer Learning 9
2.3 The Transformer Model 10
2.3.1 An Overview to the Attention Mechanism 10
2.3.2 Multi-Head Attention 12
2.4 Pre-trained Transformer-based Language Models 13
2.5 Extending Transformer Models for Long-term Context 15
2.5.1 Efficient Pre-trained Transformer Architectures 17

3 Methodology and Evaluation **19**
3.1 Design of Long-context Multilingual Models for QA 19
3.1.1 Training a Long-context Language Model 20
3.1.2 Fine-tuning on Downstream Task 21
3.2 Evaluation 22
3.2.1 Evaluating Language Models and Efficient Transformers 22
3.2.2 Evaluating Question-Answering Models 23
3.3 Methods for Reducing Memory and Training Time 25
3.4 Tools, Frameworks and Experiment Environment 26

4 Results **28**
4.1 Result for the Language Models 29
4.2 Result for the Monolingual QA Model 30
4.3 Result for the Multilingual QA Model 30
4.4 Fine-Tuning a Monolingual model on a Multilingual Dataset 33

5 Discussion **34**

6 Conclusion and Future Work **36**

List of Figures

2.1 Figure depicting the Vaswani et. al. transformer encoder-decoder model [65]. The figure is lifted from the Vaswani et. al. paper *Attention is All You Need* [65] with slight alterations to better illustrate key concepts. 10
2.2 Illustration of how bidirectional (left) and unidirectional (right) attention applies to a sentence. The bold lettered word indicates which word is currently attended/predicted. The coloring illustrates the importance or attention assigned to every other word given the current word. Darker colors indicate higher importance is placed on those words to predict the current word. . . 11
2.3 Figure depicting the Vaswani et. al. transformer encoder-decoder model with the multi-head and dot product self-attention [65]. The figure is lifted from the Vaswani et. al. paper *Attention is All You Need* [65] with slight alterations to better illustrate key concepts. 12
2.4 Figure depicting categories used by Tay et. al in their paper Efficient Transformers: A Survey [63], to categorize various efficient transformer architectures 16
2.5 Figure from the Longformer paper [3] illustrating: a) regular self-attention, b) sliding window attention, c) dilated sliding window attention, and d) is a combination of the sliding window attention from b and c, combined with global attention. 17

4.1 This figure depicts the training and evaluation loss when extending the context of a RoBERTa base and XLM-R base model for 6000 iterations. The loss is the negative log-likelihood between the label of the masked out token and the predicted token. 29
4.2 The training and evaluation loss for the XML-R and XLM-Long models, when trained on the XQuAD dataset. The loss is the negative log-likelihood between the predicted start and end tokens and the actual start and end tokens. 31

6.1 This is an illustration of how the encoder-decoder attention is applied when translating an English quote by the physicist Richard Feynman into Swedish. More vibrant coloring indicates more attention is placed on those words, and more colors for a word, indicates it attends to multiple words. The figure above is generated using the open-source library Bertviz using an XLM-R model. 44
6.2 Figure depicting the Vaswani et al transformer encoder-decoder model [65] 45

List of Tables

2.1 A comparison between popular pre-trained transformer models [37]. 15

3.1 The table depicts the training and validation datasets used for each task. We group the evaluation based on both the context length (regular or concatenated) for the different QA datasets and if it is a monolingual or multilingual dataset. For the validation set of MLQA, the amount of available validation data is percentage based and we have therefore stated the number above as the average number of validation samples for each language. 24

4.1 An overview of the pre-trained models and monolingual QA datasets used for fine-tuning in English only. 28

4.2 An overview of the pre-trained models and multilingual QA datasets used for fine-tuning and evaluating long-context transferability. 29

4.3 The SQuAD 1.1 dataset was evaluated using the regular SQuAD dataset (Regular) or the concatenated SQuAD dataset from three contexts (SQ3). The Number to the right of each dataset describes the maximum sequence length used for each model. 30

4.4 Comparative table between the baseline XLM-R base model presented in the paper *Unsupervised Cross-lingual Representation Learning at Scale* [13, 35] and our trained base models: XLM-R and XLM-Long. The models were trained on the English SQuAD 1.1 dataset and evaluated using a zero-shot cross-lingual transfer on MLQA. 31

4.5 Evaluation of the XLM-R and XLM-Long model on variations of the XQuAD dataset. Regular is the regular XQuAD dataset, XQ3 is a concatenated XQuAD dataset following the same scheme as the SQ3 dataset for SQuAD. The first table describes the EM score for the respective models and datasets and the F1 score. 32

4.6 Fine-tuning an XLM-R and RoBERTa model on the XQuAD dataset and comparing the results. 33

Chapter 1

Introduction

The transformer architecture has become one of the most prominent in natural language processing (NLP) since its introduction in 2017 [65]. The primary component of the transformer-based architecture is the use of an attention mechanism. Using the attention mechanism, the model learns the relevance for each word in a sentence in relation to other words in the same or other sentences. This novel concept has allowed transformer-based models to achieve state-of-the-art performance in a wide number of tasks such as sentiment analysis, machine translation, and question answering (QA) [3, 15]. Training deep learning models on a specific task, such as QA, requires a vast amount of labeled data. Since someone is required to label the correct answer in the text for supervised tasks such as QA, which is costly, these datasets generally contain fewer samples than needed to train a machine learning model with high performance. A large part of the transformer-based model's success is its ability to train on unlabeled and easily available data and then fine-tuned on labeled and task-specific data. This two-step process is called *pre-training* and *fine-tuning*. It allows the models to first learn language-specific and general features before learning how to solve a specific task.

A drawback of these models is that the memory requirement and computing time grows quadratically with the input text's length. For practical reasons, most language models have truncated the input text length to restrict the memory required to train and evaluate them. Instead of processing longer sentences, most models either partition the text into segments up to the maximal context size and process these one at a time with or without stride/overlap or ignore text past the truncation. This is a problem for several reasons. Firstly, depending on the truncation scheme, the source text's information is left out, potentially essential for the task. Secondly, by partitioning the context and processing them separately, critical context and retention between far away sentences are lost. This makes transformer models less useful in practice, such as answering questions in a text on page 20 using information presented on page 1. Finally, storing and utilizing these models would require steep hardware requirements unless the transformer's attention mechanism has a reduced memory and computational cost.

One of the recent research areas for NLP is methods for reducing the transformer network's attention computation cost, sometimes referred to as *efficient, long-term* or *long-context transformers*. These model tackle various aspects of the problem of incorporating long-term context, discussed more in detail in § 2.5. However, a drawback of these models is that research mainly has been constricted to English, and most efficient transformers need to be trained from scratch using long-context datasets. This is very computationally expensive, and such datasets may not be available in other languages, especially low-resource languages.

Research of transformer-based models has primarily been made on for the English language or other so-called high-resource languages [29, 54]. Among the over 7000 languages globally, ten of these comprises 76.9% of the internet presence and English alone for 25.9%.[1] A language such as Swedish for instance is considered a *low-resource* language because of the limited number of articles on the internet as a whole, few speakers and limited number of high quality datasets.

Having language models in the language one speaks determines one's access to information and education. Technologies such as spell-checking [60], digital keyboards, search engines [64], and more are still lacking support for many of the worlds 7000 languages. However, because of the high cost, time, data, and hardware required to train such language models, many countries, companies, and private users lack valuable tools that high-performant NLP models could provide.

Therefore, a commonly used solution is to train a massive language model on several, even hundreds of languages instead of only one and then allow others to use these pre-trained multilingual models and fine-tune them on a specific task and language. Commonly used multilingual models include XLM-R and mBERT.[2] Since these models require such a vast amount of data in several languages, the training of these are primarily made by large technology companies such as Google or Facebook.

Improving long-context reasoning and having more and better low-resource language models are commonly cited as two of the most prominent open questions in NLP [52, 53, 54] and could potentially have enormous ramifications for the accessibility and practical applications of future NLP model. The Swedish innovation agency Vinnova has granted a vast amount of funding[3] to RISE and its partners, among them Peltarion, to develop state-of-the-art language models for use by Swedish agencies and public sectors.[4,5] Currently, both the Swedish agency for public employment and taxes use NLP models to improve the usability, security, and utility for its employees and users. By improving Swedish language models and their use-cases, these services could be further enhanced to serve the users better, smarter, and simplify their employees' workload.

Both training long-context models and multilingual models are very time-consuming and require vast amounts of data for training. Even more so when combined, since most long-context models and multilingual models are trained from scratch. However, finding long-context datasets and training in multiple languages are costly and may not be available. To our knowledge, we have yet to see studies of incorporating long-context for multilingual models - even more so without retraining the whole multilingual model from scratch on all languages [41, 63]. Our aim is to investigate practical and cost-effective methods for making models process longer context in low-resource languages.

[1]https://www.internetworldstats.com/stats7.htm
[2]https://bit.ly/3p2rip4
[3]https://www.vinnova.se/en/p/language-models-for-swedish-authorities/
[4]https://www.ri.se/sv/vad-vi-gor/projekt/sprakmodeller-for-svenska-myndigheter
[5]https://bit.ly/3iNs0VD

1.1 Purpose and Goals

In this thesis, we examine if extending the context of a pre-trained multilingual model is possible if it has only been pre-trained with a longer context in English. We also investigate how long-context is affected for downstream tasks such as QA and if extending long-context in one language benefits or harms evaluation in another language. Since most languages lack high-quality annotated and task-specific datasets (especially with long-context), we will concatenate existing mono- and multilingual QA datasets to simulate longer context.

Since it is costly and challenging to train multilingual models and especially with large context from scratch, we will instead try to extend the context of a multilingual model, but only for English. This could potentially enable a long-context and multilingual model trained on plentiful English data without the need to retrain the model on long-contexts in every language. We use the Longformer pre-training scheme for extending the context for the multilingual model. To verify that our training corresponds to that of the Longformer authors', we will also train a monolingual model with a long-context and compare it to the results reported in their paper. These pre-trained models with extended contexts will then fine-tuned and evaluated on monolingual and multilingual QA datasets. Since there is currently no long-context multilingual datasets, we will create our own by concatenating a multilingual dataset to simulate a longer context. By evaluating multiple different models with varying maximum context lengths and multilingualism, we hope to ascertain how extending the context affects the model performance and its transferability between languages.

Our initial belief is that large-context or long document reasoning can, to some extent, be transferred in a multilingual setting from one language to another language. However, we believe that the resulting improvements on zero-shot cross-lingual evaluation would be marginal using this technique. We also assume that fine-tuning the multilingual models with an extended context in the target language could significantly improve performance.

1.2 Research Questions

1. What are efficient and practical methods for introducing large-contexts aware transformer models to low-resource languages?

2. Can large-context be cross-lingually transferred to a downstream task without large-context training for the target language, and if so, is the Longformer training scheme appropriate?

3. How can we design existing multilingual QA datasets so that we can evaluate the cross-lingual transfer of large-context since no such datasets exist yet in languages apart from English?

4. Can a multilingual model be trained to incorporate longer context in English without harming other languages' performance?

5. Will extending the context in one or multiple languages retain the same performance it had before extended pre-training on short context datasets, such as SQuAD or XQuAD?

1.3 Thesis Outline

- The thesis starts by presenting general terms in natural language processing and the transformer network in Chapter 2. A more detailed description is then presented about the transformer architecture's inner workings. A special emphasis is placed on the attention mechanism, its drawbacks, and potential solutions.
- In Chapter 3 we present an overview of the experimental setup and evaluation methods used for the respective task and dataset. The chapter concludes with an overview of the tooling, frameworks, and methods used to train efficient transformers given a limited hardware budget.
- Chapter 4 presents the experimental results and is discussed in greater detail in Chapter 5 in regards to the research questions.
- The thesis concludes in Chapter 6 with a summary and discussion of future works.

1.4 Miscellaneous

This thesis and the subsequent research questions came together based on the limitations observed in previous work on information-retrieval systems [5] and were echoed as some of the most important and open research questions for NLP by Anders Arpteg of Peltarion and other NLP experts.[6]

All code used for this thesis is available on Github via open-source. [7] For questions regarding the thesis, code or other, feel free to add an issue in the Github repository or contact Markus.John.Sagen@gmail.com.

[6] https://ruder.io/4-biggest-open-problems-in-nlp/
[7] https://github.com/MarkusSagen/Master-Thesis-Multilingual-Longformer

Chapter 2

Background

This section describes the necessary background information for modern deep learning-based NLP models - specifically transformer-based models. We start by presenting a general overview of some essential concepts from natural language processing relating to how text is processed and interpreted by computers. We then present an overview of the standard transformer architecture as presented by Vaswani et al. A particular emphasis is placed on the concept of attention, the underlying mechanism that has made transformer architecture so successful, and some of its drawbacks. We conclude by describing some proposed solutions to resolve the memory requirement of the transformer's attention mechanism.

2.1 Natural Language Processing

Natural language processing (NLP) is the study of processing natural (human) languages with computers. These models aim to learn the underlying structure, syntax, or other linguistically concepts for a language or specific task by training on massive text datasets (corpus) in the target language and task.

The training of such a model usually consists of three steps. First, separating and grouping words into a more computationally efficient representation, called tokenization. These tokens are then transformed using a word embedding. This allows for the tokens to be described in greater detail and facilitates the use of mathematical operations. The final step is to train a language model on the word embedded tokenized dataset.

2.1.1 Tokenizers

Tokenization is the process of transforming a human-readable text into a smaller sub-string of characters, called tokens [20]. When computers represent text, it is all stored as one long sentence of text. A tokenized sentence can be fed into various NLP procedures to gain more insight from the text, such as morphological analysis, wordclass tagging, parsing, sentiment analysis, and more.

Several methods exist for tokenizing text. A naive method would be to split each sentence based on space-separated words. However, this approach has several limitations:

1. This approach only works for languages using the Latin alphabet with spaces separate words.
2. Even for whitespace-separated languages, several words do not follow this structure — for instance, concatenation of words, negation of words, etc.

3. Words with the same spelling may have multiple different meanings.

4. Representing every possible word or even a fraction of them is costly since there are far more words than characters in a language. Webster's dictionary reported in 1993 that it had 470,000 entries for the English language.[1]

5. Representing tokens on a character-based level is more efficient than entire words since only 26 letters of the English alphabet. However, this low-level representation often fails to capture the full structure and relational interplay between words in a language.

Ideally, tokenization should be language-independent, fast, and an effective trade-off between word level- and character level representation. In deep learning, finding the most effective tokens to split words into is learned by training the tokenizer [32] on a language modeling task (See § 2.1.3).

2.1.2 Input embedding

Input embedding relates to the concept of representing words or subsets of words in some abstract representation, which facilitates comparison between inputs in some abstract sense. In contrast to images, where each pixel can be represented as grayscale intensity values between 0 and 255, words have historically had more arbitrary and varying representations making it more difficult for comparisons.

One method for representing words that facilitate comparisons is word embedding. Word embeddings are a learned vector representation of words that allows mathematical comparisons to be applied. This allowed for addition and subtraction of words: 'Given the word King, subtract man and add woman, the equivalent word would be Queen'; or dot-product to find the similarity between words. One drawback of these traditional statistical word embeddings was that words spelled the same way would be represented as one vector.
For a language model to accurately represent languages, it must differentiate between words and draw conclusions based on its context. This is something transformer-based models (See § 2.3) are doing. Instead of a word embedding, transformer models are trained to learn a *contextual embedding*, which assigns each word a representation based on its context [37]. These input embeddings have been shown to yield a broader language understanding [15, 36, 69] and are transferable between different NLP tasks and languages [16, 23, 55].

2.1.3 Language Modeling

A language model is a statistical distribution over a sequence of words or tokens. Given a token-corpus of known words it assigns a probability to each possible subsequent sequence of words/tokens [30] $(t_1, t_2, \cdots, t_n)$ in a given language L *i.e.,* a predictive model of the most likely next words.

$$\begin{aligned} \mathrm{P}(t_1, t_2, \cdots, t_n) &= p(t_1)p(t_2|t_1)\cdots p(t_n|t_1, t_2, \cdots, t_n) \\ &= \prod_{i=1}^{n} p(t_i|t_1, t_2, \cdots, t_{i-1}) \end{aligned} \tag{2.1}$$

Training a language model can be done on a large unlabeled dataset to learn a general language understanding. These models can then be fine-tuned on specific downstream tasks

[1] https://www.merriam-webster.com/help/faq-how-many-english-words

with labeled data. The classical language modeling objective Equation (2.1) learns a left-to-right context for a language model *i.e.,* the conditional probability of all preceding tokens $t_1, t_2, \cdots, t_{i-1}$ for token t_i. Since this traditional language modeling objective only makes predictions based on previously seen words, it is referred to as auto-regressive or unidirectional. This language modeling objective is well suited for language generation.

Another common learning objective is the so-called *masked language model* objective [15], where words/tokens in a sentence are masked-out with some probability, and the learning objective is to predict the masked out tokens. It does this by analyzing the context surrounding the masked-out word and assigning probabilities to the most likely missing words. This language modeling objective enables the model to learn left-to-right and right-to-left or bidirectional reasoning, which is more suited for sentence-level tasks such as text classification, named entity recognition, sentence analysis, and question answering.

2.1.4 Encoder-Decoder Architectures

The Encoder-Decoder architecture is a neural network architecture aimed at sequence modeling. In most neural architectures, the input and or output are fixed dimensions; however, many NLP tasks such as sentiment analysis, text classification, and extractive question answering have a variable size of the input but a fixed size. The sequence modeling task is to map an input sequence into an output sequence, both with arbitrary lengths. Traditional network architectures fail to capture this behavior. This meant if a network was trained with texts of a certain length would need to text with other lengths were used. These models are called sequence-to-sequence models, presented in a 2014 paper [61].

The Encoder-Decoder architecture consists of two neural networks: an encoder that takes in some input sequence and encodes it to a fixed-length lower-dimensional vector representation of the text; and a decoder that takes as input the vector from the encoder and reconstructs the text sequence as output. The encoder and decoder are trained jointly, where the goal is to encode and reconstruct a target sequence as closely as possible.

One of this architecture's limitations was that the encoder's fixed-length hidden state was not sufficient to represent the whole input sequence, especially for longer sequences. The introduction of *attention*, first used in RNNs (another network architecture for sequential data), allowed the decode to not only generate a sequence for the last hidden state of the encoder but to 'attend' to each the hidden states of the encoders h_i for each hidden state of the decoder s_j [2, 8]. We will describe attention in more detail in other parts of the report, primarily *multi-head attention* - the attention mechanism used by the transformer models § 2.3.1.

2.1.5 Question Answering Tasks

Question Answering (QA) is the task of finding the correct answer (if it exists) to a given question from some knowledge base [24, 33, 70]. Question-answering is a task commonly cited as requiring both an in-depth language understanding and reading comprehension [25, 28]. This has made QA tasks common for evaluating general language understanding in multiple languages and measure how well a language model can retrain information over a long context. Generally speaking, QA tasks are classified as either:

1. Extractive Question Answering (EQA)
   Where the task is to find the span, word for word, containing the most probable answer

to a question in the text *i.e.,* for all tokens in the text, find the most likely tokens to be the start and end to the answer and return all tokens within that span.

2. Abstractive Question Answering (AQA)
   The aim is to provide an abstractive answer to the question *i.e.,* answering the question not by extracting an exact passage but rather generating its own answer to the question based on some context.

QA tasks are also divided into domains, depending on where the context for answering the questions can be found:

1. Closed-domain/book Question Answering (cdQA)
   If the answers are localized to one domain, such as medical, legal, etc.

2. Open-domain/book Question Answering (odQA)
   If the answers are in multiple domains, general knowledge questions, or similar.

2.2 Transfer Learning

Transfer learning is the general umbrella term referring to transferring knowledge gained from one task, domain, or language and transferring it to another. This enables the model to learn from previous tasks and solve another task faster, using less training data. In the context of NLP, the most well-known use of transfer learning is pre-training a language model. However, multiple examples of transfer learning exist and are broadly categorized as:

Domain adaptation is the process of training on a similar task and problem, but which might be easier in some sense [17]. This is common if the target dataset is small or has little to no annotated data.

Cross-lingual learning is the process of training in one language and using the knowledge gained in another language. This is common for low-resource language; to train in English and fine-tune in another target language with less available training data.

The effect of transfer learning might yield high to low improvements depending on a multitude of factors. For instance, training more or less with data for the target task will greatly improve performance, and it is therefore common to categorize how much training has been done on the target task:

Regular transfer learning trains on both the source and target task using all training data available.

Few-shot transfer learning allows training on a few data samples for the target task.

Zero-shot transfer learning does not allow for any training on the target task.

Additionally, factors such as the types of tasks, data quality, model architecture, and similarity in data distributions between source and target will greatly affect the benefit of transfer learning.

2.3 The Transformer Model

Transformer based networks (by Vaswani et. al.) [65] has become one of the de facto architectures for solving most NLP task. Since its introduction in the seminal paper *Attention is All You Need*, there have been several pre-trained models such as BERT [15], GTP-3 [6], and RoBERTa [38] which have achieved state-of-the-art performance across a wide range of tasks. These models' success is largely from its use of the so-called attention mechanism (See § 2.3.1). It enables deep learning models to selectively attend to input and output sequences of various lengths and how important each word in the input was to predict each word in the output.[2]

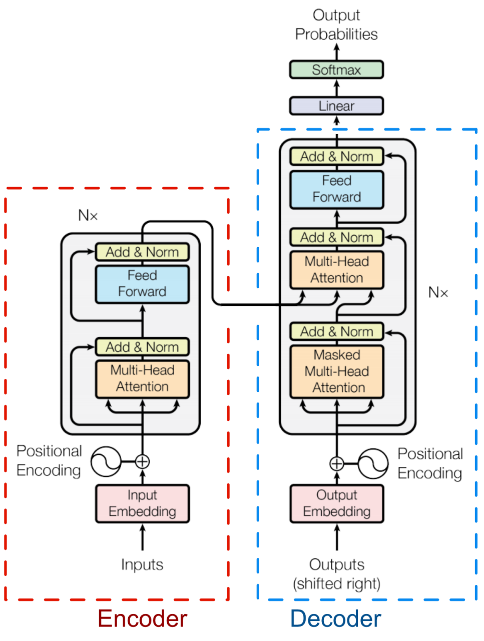

Figure 2.1 Figure depicting the Vaswani et. al. transformer encoder-decoder model [65]. The figure is lifted from the Vaswani et. al. paper *Attention is All You Need* [65] with slight alterations to better illustrate key concepts.

In the original paper, Vaswani et al. presented the transformer as a new network architecture using an encoder-decoder architecture consisting of self-attention layers. The introduction of self-attention allowed parallel training of the networks on longer sequences of arbitrary length and replaced the recurrence module previously used in RNNs.

A transformer model consists of a stack of several encoders and decoders. Each encoder and decoder module comprises a multi-head self-attention layer (See § 2.3.2), additive and normalization layer; and a feed-forward layer. For the decoder, there is also a specialized *masked multi-head attention*. There is also positional encoding and embedding for the input and output of the transformer. Models based around the transformer architecture may have different configurations and variations on these configurations.

2.3.1 An Overview to the Attention Mechanism

The concept of attention is inspired by how we as humans selectively attend to certain things in detail while ignoring others. For images and texts, certain regions bear more information

[2] https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

than others, and neighboring pixels or words are often highly correlated - meaning, it is often possible to infer the same general understanding by attending to a subset of the object instead of the whole. This *attention mechanism* is learned during training [37].

In the transformer architecture, the mechanism of self-attention is implemented differently in the encoder and decoder modules and serves different purposes. The encoder block uses a bidirectional self-attention where each word is allowed to attend to every other word; in the decoder block, it uses a unidirectional self-attention; meaning a word in a sentence can only attend to by preceding words. The difference between the two self-attention mechanisms is illustrated in Figure Figure 2.2 and are closely related to the masked language modeling and classical language modeling objective respectively § 2.1.3.

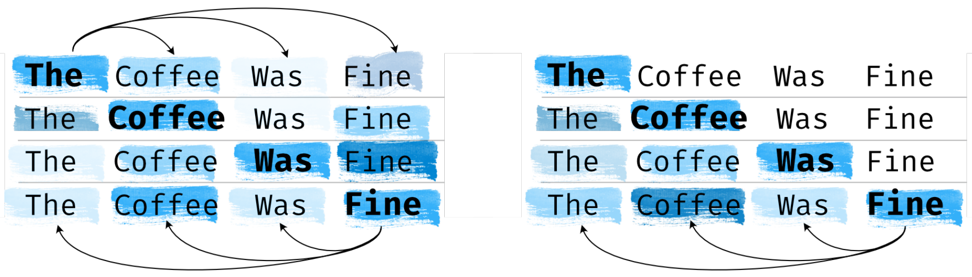

Figure 2.2 Illustration of how bidirectional (left) and unidirectional (right) attention applies to a sentence. The bold lettered word indicates which word is currently attended/predicted. The coloring illustrates the importance or attention assigned to every other word given the current word. Darker colors indicate higher importance is placed on those words to predict the current word.

The mechanism of applying attention is an all-to-all comparison between an input and an output. The goal is to selectively learn which words are most critical from the input to predict each word in the output. We train the model to identify each token's relevance in the input (source) to predict each token in the output (target) sequence [39]. In this way, we force the model to learn grammatical constructions and inherent structures of languages that can then be used to solve specific problems.

The attention mechanism was first introduced as a method to better retain information for long sequence in natural machine translation [2]. Instead of building a single context vector for the last hidden state of the encoder, the attention mechanism allowed the target sequence to directly attend to the source sentence. For every token k_t of the target sequence, a weight $w_{t,i}$ is assigned to all the tokens in the source sequence $x_1, x_2, \cdots, x_N$. Assigning these weights are analogous to an information retrieval system, where given a key k_t to search for in a sequence X we compare how closely all possible candidates x_i match to k_t. The most common similarity function used for attention is the *dot-product* or *cosine similarity*. The attention weights $w_{t,i}$ between each token in the source x_i and the key is calculated as [19]:

$$w_{t,i} = \text{Softmax}(k_t, x_i) = \frac{\exp\ \text{dot}(K, x_i)}{\sum_j \exp\ \text{dot}(K, x_j)} \tag{2.2}$$

$$v_t = \sum_i w_{t,i} x_i \tag{2.3}$$

Where a v_t is the context vector corresponding to the attention scores for the source token/key k_t. Softmax is applied to normalize the weights. We have purposely left out certain aspects from the original attention formulation presented by Bahdanau et al. [2] to better illustrate the core underlying principle of how attention weights are calculated. The following section will describe how the transformer's attention mechanism is implemented and its current limitations.

2.3.2 Multi-Head Attention

The transformer network, presented in the paper *Attention is All You Need* [65], replaced modules commonly used in previous network architectures such as recurrent or convolutional layers. Instead, it uses only self-attention layers to capture dependency between the input's tokens by applying attention to the tokenized input with itself. This allows transformers to model sequence to sequence tasks without any recurrence modules while increasing its performance and allow for parallel training.

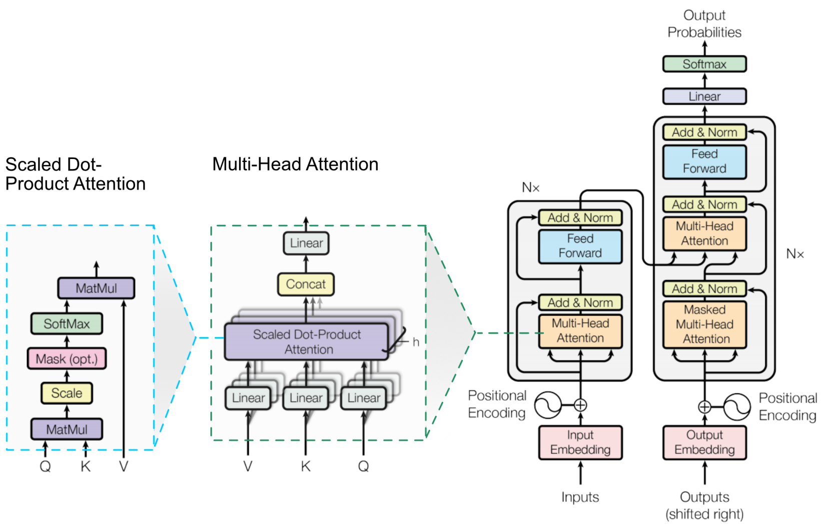

Figure 2.3 Figure depicting the Vaswani et. al. transformer encoder-decoder model with the multi-head and dot product self-attention [65]. The figure is lifted from the Vaswani et. al. paper *Attention is All You Need* [65] with slight alterations to better illustrate key concepts.

At the heart of the transformer models, success is the *multi-head self-attention mechanism*. It reuses concepts from information retrieval systems by generating three different vector representations with randomly initialized weight created from the input: key K, value V, and query Q. The key and value vectors are sent as input to the self-attention layer and the output as the query. By borrowing concepts from information retrieval, the problem of learning to assign attention weights to each word can instead be viewed as: given something to search for (query), match the closest keywords (key), and return the most similar results based on your inquiry (value). This mapping or self-attention mechanism between the words in the sequence and their relevance is learned during training. The general formulation for the attention mechanism can then be reformulated as:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

This is also called the *scaled dot-product attention*, which projects the expected keys from the source (Keys) onto the target output (Query). The query, key, and value matrices are created by multiplying the input sequence X with randomly initialized weights W to that $Q = W^Q X$, $K = W^K X$, and $V = W^V X$ are learned during training by updating the weights W^Q, W^K, W^V.

Instead of applying self-attention once to learn one representation on the data, Vaswani et. al. found that by creating multiple scaled-dot product attentions each using randomly initialized weight matrices, these attention weights could then learn multiple word alignments of a sequence. By concatenating multiple such dot-product self attentions, the model could learn a more general and complete representation of the words. They called this *multi-head self-attention* [41, 65]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \cdots, \text{head}_h)W^O \tag{2.5}$$

$$\text{head}_i) = \text{Attention}(Q, K, V) \tag{2.6}$$

Where each attention head, head$_i$, is the scaled dot-product attention of the query, key, and value and randomly initialized weights and W^O is another randomly initialized weight matrix learned during training. The number of heads h denotes the number of parallel attention layers computed and concatenated into a single *multi-head attention layer*. The standard transformer network uses six stacked encoder- and six stacked decoder blocks, where each such block consists of a multi-head self-attention layer.

A downside with the multi-head self-attention mechanism is the quadratic time and memory complexity by computing these matrix multiplications. Since Q, K, V are all matrices generated from a linear projection of the input text of length N and each token attends to every other token, the memory and computational complexity of these operations or, more specifically, the QK^T matrix multiplication is $\mathcal{O}(n^2)$ [63]. Because the time and memory complexity grows quadratically, several models have self-imposed a maximum sequence length of 512 or 1024 tokens, which a model can process at a time. We will elaborate on methods for combating these issues in § 2.5.

2.4 Pre-trained Transformer-based Language Models

During training, the transformer model learns high-level representations for the text (text embedding). Pre-trained transformer-based models, such as BERT [15] have been shown to learn general language understanding by training on massive unlabeled datasets in a self-supervised manner and fine-tune on a downstream task (see § 2.2).

Because of the transformer model's different learning objectives, the model itself can be used in different modes by composing a network of either: only encoder blocks, only decoder blocks, or both encoder and decoder blocks. Each composition specializes in solving different tasks because of how the self-attention is configured and learned differently in the encoder and decoder block [37]:

Encoder-only uses the self-attention where the input can attend to both past and future tokens, making it suitable to process information for classification, question-answering, and extractive summarization. The self-attention mechanism may be auto-regressive [63], but is most often bidirectional and trained with an MLM objective. Notable models include BERT, RoBERTa, mBERT, and XLM-RoBERTa (XLM-R).

Decoder-only has a restriction to only attend to previous tokens in the input. This makes it suitable for generating new text based on previously seen text, such as text-generation. The self-attention mechanism must be auto-regressive. Notable models include: GPT-2 and GPT-3.

Encode-Decoder combines both the benefits of an encoder and decoder and is usually used when generating or producing new information from existing text, much like the decoder-only, but can learn more complex patterns. These models are often used in machine translation, abstractive summarization, and abstractive question answering. Notable models include BART and T5.

Below are some commonly used pre-trained transformer models: either encoder-only, decoder-only, or encoder-decoder architectures. We also describe their differences and potential improvements from previous transformer-based models.

BERT stands for Bidirectional Encoder Representations from transformers [15] and set a new state-of-the-art (SOTA) across multiple NLP tasks and became a benchmark to measure all other models against. Instead of the base transformer configuration of an encoder-decoder structure, the BERT model uses several stacked encoder layers. Each layer learns various aspects of a language and task. Their seminal paper introduced several novel ideas, including a new language modeling task, *masked language modeling* (MLM). MLM is an alternative language modeling objective (See § 2.1.3), where a percentage of words are masked-out from a sentence, and the objective is to predict the masked out words based on the context to other words. With a traditional language modeling objective, each token in the sequence is used to predict the next in an auto-regressive manner (See Equation (2.1)), as used in the transformer models GPT-2 [47] and GPT-3 [7]. BERT's authors showed a language modeling objective that could be utilized to learn a bidirectional language representation by forcing each word to be used when predicting masked out words. They trained BERT on a large corpus of data collected from the BookCorups dataset and crawled Wikipedia pages.

The authors also proposed an additional language modeling objective called *next-sentence prediction*; however subsequent papers have rejected it, demonstrating it had no positive impact on model performance [38]. They also presented key ideas of how tasks such as classification and question-answering could be performed on a pre-trained language model by replacing the final output layer or the given task and training only that layer on a downstream task.

RoBERTa or 'A Robustly optimized BERT' is a model based on the training schemes and model architecture presented in the BERT paper [38]. The RoBERTa model evaluated the different model implementation- and hyperparameter choices BERT used to determine the best method for training an even better pre-trained model. They replaced the BERT WordPiece tokenizer with a byte-level Byte-Pair-Encoding used by GPT-2, trained with much larger mini-batches, higher learning rate, removed the BERT NSP objective, and trained on more data. It set a new SOTA across multiple tasks. The RoBERTa model was trained on the same dataset as BERT: BookCorpus and Wikipedia; and two enormous datasets: CC-News and OpenWebText, which contains content from more than 10 million web-pages each. The total amount of English training data for RoBERTa was 160GB.

XLM-RoBERTa or XLM-R is a multilingual pre-trained transformer model based on the RoBERTa architecture [13]. The model was trained on 2.5 Terra Bytes (TB) of data across 100 languages, including Swedish, collected and filtered from Common Crawl. XLM-R showed substantial gains over the preceding multilingual models mBERT and XLM on downstream tasks [13]. XLM-R has a vastly better performance on multilingual tasks than RoBERTa but slightly worse on English downstream tasks.

Model	**Type**	**#Params**	**Objective**	**Datasets**
GPT-2 (base)	Decoder	117M	LM	Pages on Reddit
BERT (base)	Encoder	110M	MLM, NSP	BookCorpus, Wiki
RoBERTa (base)	Encoder	125M	MLM	RoBERTa*
XLM-R (base)	Encoder	270M	MLM	Common Crawl

Table 2.1 A comparison between popular pre-trained transformer models [37].

The different objectives noted in table 2.1 describes the different language modeling objectives used for the various pre-trained transformer-based models:

– Language Modeling (LM)
– Masked Language Modeling (MLM)
– Next-Sentence Prediction (NSP)

2.5 Extending Transformer Models for Long-term Context

One of the largest drawbacks of the transformer architecture is also its biggest strength: the self-attention mechanism. As described in § 2.3.2, when computing the self-attention weights, each encoded token in the input sequence attends to every other token *i.e.,* a computational complexity of $\mathcal{O}(n^2)$ for sequence length n. Therefore, popular models such as BERT, RoBERTa, and more have limited the pre-trained models to attend to a maximum of 512 tokens at a time, while others such as T5 and GPT-2 have a maximum token span of 1024. In theory, these models could be pre-trained with a much longer context. However, in reality, hardware limitation, growing memory consumption, and computing time have led these models to cap their maximum context [15].

Several solutions have been proposed to solve the computational complexity of transformer models, generally categorized as *efficient transformers* [63], methods centered around different notions of sparsity of the dense attention matrix. Authors Y. Tay, M. Dehghani, D. Bahri, and D. Metzler categorized five different characteristics of proposed efficient transformer models in their paper *Efficient Transformers: A Survey* [63]:

Fixed or Factorized Pattern are methods for applying self-attention to a fixed block size instead of the whole sequence and with some stride length. An input sequence of length N are chunked into blocks of length B, then for $B << N$, the computational complexity will tend toward be $\mathcal{O}(B^2)$. Other methods employ dilated sliding window attention, operating similarly to a convolution. Here, the computational complexity becomes $\mathcal{O}(N\ x\ k)$, where k is the sliding widows of some constant size. If $k << N$, when the complexity is linear. Models based around these characteristics include: Memory Compressed [48], Blockwise Transformer [46], Sparse Transformer [10], Longformer [3], and Big Bird [71]

Learnable Patterns are methods to cluster or group tokens with strong relevance and apply attention to tokens within the same clustering. The underlying principle is still to

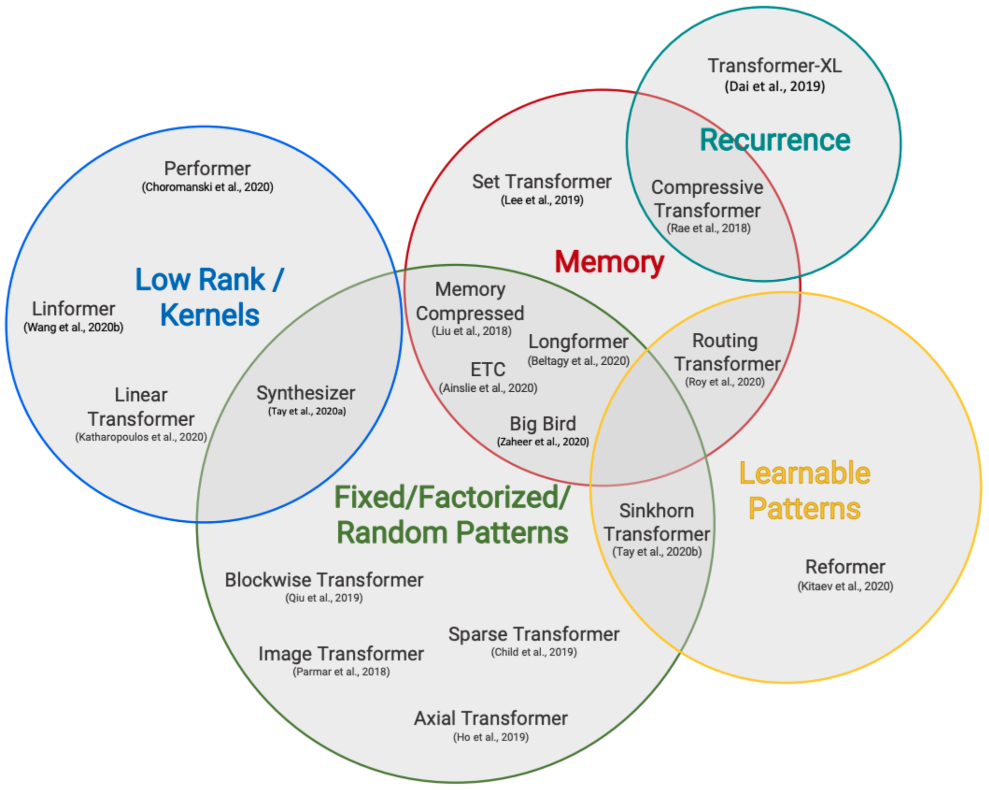

Figure 2.4 Figure depicting categories used by Tay et. al in their paper Efficient Transformers: A Survey [63], to categorize various efficient transformer architectures

learn a fixed-sized attention pattern but in a more effective manner. Tokens within the same cluster may apply full attention to every token but not to every token in other buckets. Models based around these characteristics include: Reformer [31] and Sinkhorn Transformer [62].

Low-Rank Matrices or Kernels leverage the fact that the self-attention matrix is sparse and can therefore utilize a low-rank approximation of the N x N matrix into some non-quadratic lower-dimensional representation. Models based around these characteristics include: Linformer [66] and the Performer [11].

Memory Extension is a common method utilized by several models to allow for a limited number of tokens to attend to every other token in the sequence. This attention model is commonly referred to as *global attention*. These special global attention tokens can either be learned or assigned manually. For instance, the separation token <s> or <CLS> is often assigned with global attention for downstream tasks to allow better retention between the context and the question. Models based around these characteristics include: Routing Transformer [51], Longformer [3], Big Bird [71], and Compressive Transformer [48].

Recurrence can be reintroduced back from RNNs to transformers and allows models to propagate the parsed context from previous segments for longer. These methods are often paired with *Fixed Patterns* to allow models to process some number of tokens at a time and propagate the context from previous steps to the next. Models based around these characteristics include: Transformer-XL [14] and Compressive Transformer [48]

2.5.1 Efficient Pre-trained Transformer Architectures

This section describes some of the more well known pre-trained *efficient transformers* in greater detail.

Longformer is an *efficient transformer*, which has been trained from a RoBERTa model checkpoint and is unique among effective transformers since other models are trained from scratch. This means a Longformer conversion can be applied to all RoBERTa based models, such as RoBERTa, XLM-RoBERTa, etc. However, the authors of the Longformer paper [3], I. Beltagy et al. also states that the general principle they presented can be applied to any transformer-based model.

Instead of a dense-attention matrix, the Longformer uses three self-attention window arrangements to capture longer dependency. These three attention patterns are in turn:

1. *Sliding window attention*
   Sliding window attention allows tokens within a limited window span w to attend to other tokens. This is essentially a convolution applied over all tokens, where attention is applied within the fixed sliding window size.
2. *Dilated sliding window attention*
   A dilated sliding window is a sliding window attention, where the attention window has been dilated only to apply attention to every jth token. By applying dilated attention, the model can learn features for longer gaps in the sequences. The authors suggest using different dilation heads in the multi-head self-attention to capture a more diverse long-term context better.
3. *Global attention*
   For Question Answering (QA), the text's answer needs to map to the question posed. For longer sentences, not having global attention in certain instances severely reduces the performance of the model performance. By allowing a certain number of tokens k, such as the starting-, separating and end tokens, to have global attention, the mapping for these tasks drastically improves while keeping the number of tokens with dense self-attention to a minimum.

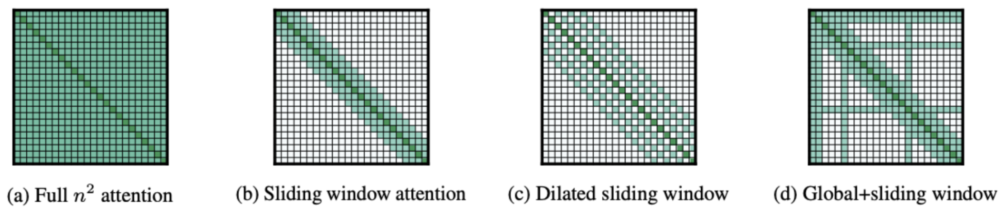
(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

Figure 2.5 Figure from the Longformer paper [3] illustrating:
a) regular self-attention, b) sliding window attention, c) dilated sliding window attention, and d) is a combination of the sliding window attention from b and c, combined with global attention.

The Longformer can start its training from a RoBERTa model and extend its context to be less computationally expensive. The window size for the diluted sliding window w is set to the maximum sequence length a RoBERTa model can attend to *i.e.,* 512 tokens. While the Longformer has a memory and time complexity of $\mathcal{O}(n(w+k))$, in practice, it is only an improvement if the sequence length n is much greater than 512, since w is a constant value of the checkpointed models maximum sequence length (512).

Reformer claims to be able to process the longest sequences out of all *efficient language models*, with a maximum sequence length of 1-0.5 million tokens. It does with while claiming to be used on most high-end computers (1 GPU accelerator and 8GB of ram) [31]. The Reformer introduced two novel concepts to ensure a better memory and time complexity over the transformer architecture, reducing it to $\mathcal{O}(n \log n)$. First, it uses a *locality sensitivity hashing (LSH)*, a hashing function to map similar inputs to the same or closely related hashes. Entries with the same hash are grouped into the same buckets, and attention is applied to the entries in the same bucket and not to all entries, as is the case in the traditional transformer architecture.

Secondly, it uses *reversible layers*. Training a deep learning model usually means storing activation functions in memory for backpropagation. However, this is memory-wise very expensive. The Reformer instead recomputes the input and output tensors only during the training, using the same technique presented for the ResNet architecture [18] and use the difference between these to approximate the gradients in the intermediate layers. This ensures that the model only needs to store the activation's for the input and output layer once[3], instead of N times.[4]

Linformer proposes to project the key and value matrices K, V from a $n\ x\ d$ dimensional matrix dependent on the input size, to low-rank matrices K', V' of dimension $k\ x\ d$, via a projection matrix [66]. The matrix multiplication needed to compute the attention score: Softmax(QK^T), where QK^T is of dimension $n\ x\ n$ then becomes QK'^T, which is of dimension $n\ x\ k$. Since k is a constant, the overall attention computation becomes $\mathcal{O}(n)$.

Transformer-XL in constant to the previously discussed *efficient transformer* architectures, does not sparsify the dense-attention matrix [14]. Instead, it reintroduces recurrence into the transformer by allowing information to flow from the current segment computed to the nodes preceding it for a limited number of nodes back. By reintroducing recurrence and computing the attention of the network in chunks, the Transformer-XL claims[5] compared to transformers: a) Can learn dependencies which are 450% longer; b) up to 1,800 times faster inference time; c) One of the best perplexity scores out of all the *efficient transformer* models.[6] However, because the dense-attention calculations are not sparsified, the computational complexity remains $\mathcal{O}(n^2)$.

[3]https://huggingface.co/transformers/model_doc/reformer.html
[4]https://huggingface.co/blog/reformer
[5] https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html
[6]https://paperswithcode.com/sota/language-modelling-on-wikitext-103

Chapter 3

Methodology and Evaluation

This section will describe the method used to train and evaluate the pre-training used for the extended language model and fine-tuning for the downstream QA task. We start by describing our methodology and the design choices made for evaluating if cross-lingual transfer is possible. This is followed by a description of the training procedure, evaluation, and datasets used in relation to how it best would answer our research questions. This followed with an in-depth description of the problems with training vast and memory-intensive models, hardware limitations, and methods that can be used to combat these problems. The chapter concludes with a description of the tooling, libraries, and hardware used to run the experiments.

3.1 Design of Long-context Multilingual Models for QA

This section will describe the process and decisions made in designing, training, and testing long-context multilingual QA models. We start by pre-training an extended context language model using the Longformer training scheme. We train both a monolingual and multilingual model with extended context. This is to ensure that the results we measure can be compared accurately to that presented by the Longformer authors for a monolingual model. These models are then fine-tuned and evaluated on monolingual and multilingual QA datasets with varying context length and multiple types of pre-trained transformer models. This way, we hoped to verify that our long-context training scheme worked as expected, that the models achieved equally well on short and long-context datasets, and confirm if cross-lingual transfer was possible. Using pre-existing and trusted pre-trained transformer models also allowed us to compare more accurately how our own long-context dataset affected model performance.

The aim was to investigate efficient methods for extending the context for low-resource models. However, training such models in multiple languages is expensive and may not be available, depending on the language. We concluded that training a multilingual or even bilingual long-context model from scratch was infeasible given the currently available time-constraint and lack of long-context non-English datasets. Instead, we decided to investigate if long-context could be extended from a pre-trained multilingual model, such as XLM-R. To our knowledge and as of this writing, there is only one such efficient transformer architecture: the Longformer. Using the same datasets, training scheme, and hyper-parameters as the Longformer model, we hoped to investigate if extending the context on one language, English, also improved other languages' performance on tasks that require long-context.

Since no-one has yet to investigate this, to our knowledge, we also wanted to verify that extending the context in one language did not harm the performance in other languages, neither when evaluated on datasets with short contexts nor with long. If we could showcase that long-context training in one language does not disrupt the performance in other languages. This could then enable subsequent long-context training on the already extended multilingual model by training in other languages, such as Swedish, when data becomes available to improve the performance in that language further.

To verify our research questions, we trained both a monolingual and multilingual long-context model using the same pre-training scheme as the Longformer authors. This enabled us to compare our pre-training results with those reported by the Longformer authors since they only trained a monolingual model. We chose to evaluate on the QA task to measure long-context cross-lingually because it has previously been shown to be a good indicator of language understanding [25] and long-context reasoning [3, 14, 28]. Long-context datasets do exist in English, but not in other languages currently, but on the other hand, there are plenty of multilingual datasets with shorter context. We, therefore, decided to construct a QA dataset with a longer-context artificially. This would then let us evaluate if long-context worsens the performance and how it varies between different models.

3.1.1 Training a Long-context Language Model

The goal was to train a QA model with a longer context on low-resource languages, such as Swedish. However, since there is no publicly available dataset for Swedish QA, we instead decided to evaluate a general long-context low-resource model on a QA task. We, therefore, knew that we needed to use a multilingual model. However, almost all *efficient transformers* (See § 2.5) have a modification of their architecture in such a way that they must be retrained from scratch. The only model we found which could extend the context on a pre-existing transformer model was Longformer. We reasoned that extending the context of a pre-trained multilingual language model would be more practical and take less time than using an efficient transformer model and training it for multiple languages.

Training

To extend the context for both the monolingual and multilingual models, we use a modified training script provided from the Longformer Github repository.[1] Since no one to our knowledge had trained a multilingual model, we, therefore, chose to rely on the same hyper-parameters for training the multilingual model as the monolingual. This was primarily because of the project's limited time budget, and we would have otherwise investigated the best hyper-parameters for the multilingual models.

We used the same hyper-parameters as the Longformer authors reported in their paper. We used warm-up steps of 500, a learning rate of $3 * 10^{-5}$, AdamW optimizer with hyper-parameters $\epsilon = 1 * 10^{-6}, \beta_1 = 0.9, \beta_2 = 0.999$, a L_2 weight decay of 0.01, a dropout probability of 0.1, and mixed-precision training. We set the per GPU training batch size to 1 and a gradient accumulation to 64. This ensures that the total training batch size is 64, as recommended in the Longformer paper. We also set the new maximum sequence length to 4096 or any multiple of 512. We train a monolingual RoBERTa model using the Longformer scheme to compare the available pre-trained Longformer models and then train an XLM-R model

[1] https://bit.ly/36WNdYr

using the same hyper-parameters. We experimented with other hyper-parameters but found those to be the best.

3.1.2 Fine-tuning on Downstream Task

Once long-context pre-training was completed, we fine-tuned the models for extractive QA. The extended context models and regular transformer models were fine-tuned on monolingual and multilingual QA data following the SQuAD-format. We used a concatenated version of SQuAD (SQ3) and XQuAD (XQ3) to evaluate how well these models performed on longer contexts (See § 3.2.2). This was done by restricting the models to only be allowed to process a maximal sequence length of 512 or 4098 and then compare the results. We hoped to determine if long-context cross-lingual transfer was possible and how the model performance was affected in other languages and for datasets with shorter context by having an extensive comparison between different choices of datasets, languages, and context lengths.

Training

We experimented with different hyper-parameters for fine-tuning and settled on using similar hyper-parameters as Huggingface's fine-tuning script for SQuAD.[2] For the regular QA models, we set the maximal sequence length of 512 tokens, convert all text to lowercase, and applied truncation for text extending beyond the 512 token limits. We set the learning rate to $3 * 10^{-5}$, with the AdamW optimizer and an Adam ϵ of $1 * 10^{-6}$. We train for three epochs with mixed-precision training, a stride of 0 and a total per device training batch size of 32. For the monolingual models, we fine-tuned a base RoBERTa, base Longformer, and our own base RoBERTa model converted to a Longformer. For the multilingual model, we trained a base XLM-R model, and our own base XLM-R converted to an XLM-Longformer. For the multilingual models, the gradient computations used for backpropagation became too large, and we, therefore, decreased the batch size to 4, and gradient accumulation increased to 8. The fine-tuning and evaluation was repeated five times using different seeds.

For fine-tuning the large context models, with a maximum sequence length of 4096 tokens, we used the extended context datasets SQ3 and XQ3 for monolingual and multilingual evaluation. SQ3 and XQ3 are datasets created by us with a longer context from the monolingual SQuAD and multilingual XQuAD datasets, respectively. The longer context of these datasets are created by concatenating three unique contexts for the respective dataset (See § 3.2.2). We did this since there were no available multilingual datasets with long-term context, which was required to answer our research questions. Since both of these are SQuAD-formated, we could reuse the training script from the monolingual and multilingual fine-tuning. We ran two different experiments using these datasets and with the same training parameters and the regular QA fine-tuning described above. However, we changed the maximum sequence length to be evaluated with 512 and 4096 tokens to better assess the effects of evaluating long-context datasets. We reduced the per GPU training batch size to 1, increasing the gradient accumulation to 32. We also used gradient checkpointing to fit gradient updates in memory when training.

[2] https://bit.ly/3tDTK3U

3.2 Evaluation

The respective model's evaluation is made using the de facto standard for their respective tasks: Perplexity (PPL) and BPC for language modeling and exact match (EM) and F1 score for extractive question answering.

3.2.1 Evaluating Language Models and Efficient Transformers

Perplexity is a standard metric for measuring the performance of a trained language model. It measures the uncertainty from choosing the most likely next sequence for a learned distribution or how well a predictive model learns a target distribution. When training a language model, we train it on a sample text corpus with a distribution Q. The aim is to learn the true underlying distribution of the text[3], such that when predicting the next words in a new test sample dataset (empirical distribution) P, then Q and P should be distributions as closely related as possible.

The perplexity score (PPL) can also be viewed from the perspective of information theory. Viewed from this lens, PPL represents the average number of bits needed to encode any outcome P using the learned optimized encoding from P, plus the average number of bits needed to encode any outcome Q using the encoding learned from P [26]. The PPL is defined as:

$$\text{PPL} = 2^{H(P,Q)}$$

Where $H(P,Q)$ is the Shannon entropy or the cross-entropy between the empirical distribution P and the sample distribution [57]. However, comparing perplexity score between LMs are not that straight forward as comparing model accuracy on downstream tasks, since lower PPL does not necessarily mean a model with higher performance.

This is for several reasons: firstly, different language models encode the learned tokens differently; some learn an encoding on a character level (BPC), and others learn on a (sub)word level (BPW), and because words can have a lower encoding than combining encoded letters separately, comparing models using different encodings are challenging. Secondly, classical language models are auto-regressive, meaning they predict the next word given all the previous words. However, most modern transformer-based LMs are bidirectional *i.e.,* are using a masked language modeling objective(See § 2.1.3). Because bidirectional models use context to the left and right of the words it predicts, those models tend to have a lower PPL than classical language models. Chip Huyen, formerly of NVIDIA [26] therefore suggests for other researchers to include both the PPL score and BPC or BPW, depending on the LM's symbol type, for better comparisons between different LMs.

If the dataset is tokenized using a character-based tokenization and the loss function used for training the model was the cross-entropy, then we can calculate the BPC score for a trained language from its final evaluation loss value $\mathcal{I}$ as:

$$\text{BPC} = \frac{\mathcal{I}}{\ln(2)} = \frac{\log_e(\mathcal{L})}{\log_e(2)} = \log_2(\mathcal{L})$$

Since the loss returned for the PyTorch and Tensorflow libraries are using the negative log-likelihood *i.e.,* the natural logarithm of the true loss ($\mathcal{L}$). To display the true BPC, which is a binary representation, we, therefore, change the basis using the logarithmic rules.

[3] https://huggingface.co/transformers/perplexity.html

In practice, there seem to be conflicting opinions on whether or not a lower perplexity score for a language model translates to a better performing model on downstream tasks. As stated in two papers *RoBERTa: A Robustly Optimized BERT Pre-training Approach* [38] and *XLNet: Generalized Auto-regressive Pre-training for Language Understanding* [69] respectively. However, it should be noted that the positive conclusions made in the RoBERTa paper may be influenced by the fact that RoBERTa is bidirectional and, therefore, had a lower BPC.

Dataset

The dataset used for extending the context is the Wikitext-103 dataset [3]. Wikitext-103 [42] is one of several datasets used for training language models on English data. It consists of a list of featured articles extracted from Wikipedia, where <UNK>-tokens replace rare word-level tokens if they appear less than thrice in the entire dataset. Its name is derived from the number of word-level tokens - 103 million, and with a total vocabulary size of 267,735 [42]. We chose this dataset primarily because of its length and previous usage by the Longformer authors.

3.2.2 Evaluating Question-Answering Models

The standard evaluation used for extractive QA models follows from metrics used by the authors of the SQuAD paper [49]: Exact match (EM) and F1 score. Exact match measures the percentages of exact matches made when predicting the answer span in the text. F1, on the other hand, is a trade-off between precision and recall and measures the average overlap between the predicted and actual answer.

Dataset

The most commonly used dataset for question-answering tasks is the Standford Question Answering Dataset (SQuAD v1.1). A monolingual English dataset crowdsourced from 100k question and answer pairs from Wikipedia. The tasks are to extract the most likely span in the text to answer a question posed [15, 50]. The answer is the minimal span between the token with the highest probability for a start and the highest probability for the end token. Datasets following the same general structure are sometimes referred to as *SQuAD-formated* or *SQuAD-like* datasets.

We chose to evaluate the monolingual models using the SQuAD datasets since it serves as a standard benchmark for QA models and would make comparisons with other models easier. This way, we could potentially identify problems early on before fine-tuning on multilingual data. We chose not to evaluate on other English datasets with a longer context, such as TriviaQA, due to time constraints and our focus on evaluating long-context transferability for multilingual models. If, however, TriviaQA had been available in other languages, this would have been our dataset of choice.

To simulate QA models with longer contexts, we took the context from pairs of three SQuAD formatted datasets and concatenated them together. However, since SQuAD formatted datasets may have several questions using the same context, we filtered out duplicate contexts to avoid the same context ever being concatenated together. We did this since questions could have otherwise potentially had the correct answers in multiple places - and since SQuAD is pre-trained on a shorter context, it would always find the answer in the first of the concatenated segments. Filtering out duplicate contexts reduces the original SQuAD dataset to about a fifth of its original size. We call this reduced and concatenated SQuAD based dataset *SQ3*. We used this dataset when training and evaluating monolingual and multilingual long-context QA.

For the multilingual QA, we used the dataset XQuAD [1], a multilingual SQuAD formatted dataset used for evaluating zero-shot cross-lingual transfer. The dataset consists of a small subset of the English dev dataset, 1190 samples translated to ten other languages. We chose XQuAD because of its wide adoption as a cross-lingual evaluation for QA and a relative scarcity of other multilingual QA datasets - especially with long-context. The dataset TyDi QA [12] was initially the dataset we tried to used since it posed harder to answer questions from real users and in their native language. However, the SQuAD formatted dataset had a concise context (about 150 tokens on average), and the extended context variant of TyDi QA used a different formating than that of SQuAD. SQuAD formatted datasets expect to find a minimal span of the answer in the text. However, for TyDi QA, the task is to find if any sentence answers a given question and, in that case, which sentence. Because of the significant difference between these two datasets, the fine-tuning and evaluation scripts would have needed to be entirely different. However, we would still have to fine-tune on multilingual SQuAD datasets to compare with other researchers' results for these multilingual models. Our initial experiments on the TyDi QA dataset highlighted that accessing if the model performed well or not and why was also much harder than for a SQuAD-formated dataset. Because of all these reasons and the time constraint, we chose not to proceed with TyDi QA.

Regular Context

Type	Train	Samples	Validation	Samples
Monolingual	SQuAD	87599	SQuAD	10570
Multilingual	SQuAD	87599	XQuAD	1190
Multilingual	SQuAD	87599	MLQA	~512

Concatenated Context

Type	Train	Samples	Validation	Samples
Monolingual	SQ3	18895	SQ3	2067
Multilingual	SQ3	18895	XQ3	250

Table 3.1 The table depicts the training and validation datasets used for each task. We group the evaluation based on both the context length (regular or concatenated) for the different QA datasets and if it is a monolingual or multilingual dataset. For the validation set of MLQA, the amount of available validation data is percentage based and we have therefore stated the number above as the average number of validation samples for each language.

Finally, we evaluate our multilingual models, XLM-R, and our multilingual model with extended context XLM-Long on the MLQA dataset. MLQA [34] is a multilingual SQuAD formatted dataset consisting of highly parallelizable instances from seven languages - 12000 samples from English and 5000 for every other language. Having parallel instances between languages allows for a fairer comparison, faster learning, and requires less annotated data. However, for our purposes, we use the MLQA dataset for zero-shot cross-lingual evaluation only, meaning that we only train using the SQuAD dataset [13, 68]. We evaluate against the MLQA since it allows us to compare our pre-training of the XLM-R and XLM-Long models with the Facebook research group's results on an XLM-R base model. We did not evaluate against XLM-R base on XQuAD since the baseline listed in the paper was an XLM-R large model and generally performs much better than an XLM-R base model.

3.3 Methods for Reducing Memory and Training Time

State-of-the-art (SOTA) models in NLP are models with hundreds of millions to billions of parameters trained on large datasets on large distributed systems with high-end GPUs. Generally, using larger batches, more GPUs with larger RAM are the easiest ways to train large models while achieving a low and convergent training/evaluation loss [4]. However, this may not be possible.

Both multilingual models and long-context transformers are very computationally demanding models to train, especially when combined. Therefore, this project's essential requirement was to train these models within the memory constraint set by hardware and do so in a reasonable time-frame without sacrificing model performance.

Below are some techniques commonly used to reduce the memory requirement for deep learning models and techniques for improving training time. These methods were mentioned briefly in § 3.1.1 and § 3.1.2 but discussed in greater detail here.

GPUs and Distributed Training are methods for facilitating training across multiple GPUs to spread out the workload to fit larger batches or faster training. The authors of the transformer network *Transformer-XL* scaled their training to 128 GPUs[4], reducing the training time from two weeks to 18 minutes. This method is beneficial if possible since it reduces training time, allows for larger batch sizes and few downsides. However, there is a diminishing gain from an increased number of GPUs per Amdahl's law.

Batch Size is one of the most important hyperparameters to tune in deep learning models. Often larger batch sizes of 32, 64, and 128 are used to allow computational speed up [4, 58]. Using larger batches leads to poor generalization, while smaller batches start training faster and may have faster convergence but tend to have a more unstable loss curve and might not converge to a local minimum at all [40, 59]. Generally, there exists an optimal batch size to ensure fast convergence and the global minimum loss, which are dependent on the deep learning model, dataset, and optimizer used among other factors [22].

Gradient Accumulation is a method for accumulating gradients when training on low memory GPUs, where a smaller batch size can be used. This strategy simulates using a global batch size of $B = bg$, by splitting the samples into a mini-batch size of b. During the backpropagation step, gradients are calculated and accumulated for g mini-batches. The averaged accumulated gradients are used to updated the model weights. This acts as if the full global batch size bg had been used for training. A drawback of gradient accumulation is that it increases training time compared to training with the full batch size.

Mixed (16-bit) Precision refers to the use of 16-bit and 32-bit floating-point types and is only available as of writing on NVIDIA GPUs.[5] Running floating-point operations as 16-bit floats are generally faster and require less memory. NVIDIA research showed that while some high-performance computing requires 32-bit or higher precision, deep learning applications are less sensitive to this small floating-point accuracy loss. In theory, we could use only half-precision calculations to achieve a theoretical 8x speed-up,

[4]Transformer-XL Distributed training https://github.com/cybertronai/transformer-xl
[5]https://docs.nvidia.com/deeplearning/performance/mixed-precision-training

but in practice, those calculations may suffer from gradient underflow when performing backpropagation. Therefore, a mixture of single precision and half-precision is used and yields a 2-4x speed-up.[6],[7],[8]

Gradient Checkpointing (GC) is a method claiming to reduce memory overhead by 10x, but increase training time by about 20% [9, 21]. Instead of recomputing the gradients for each layer when performing backpropagation, they are instead 'checkpointed' at certain intervals in the network. The computations can be started from these checkpoints instead when gradients need to be recomputed and propagated through the network, which is much more memory efficient.[9]

Regularization Methods are a collection of techniques meant to prevent overfitting when training a model. Below are listed some of the commonly used methods for deep learning models. `Early stopping` stops the training early if overfitting occurs, `batch normalization` normalized the values in a batch to scalar values between $(0, 1)$, and *dropout*, where neurons are deactivated in the network with a binomial probability p. When applying dropout, the model is forced to learn other connections and representations to solve the task, making the whole model learn the task and not individual neurons.

There are also some additional methods that we did not use:

Optimizer is often recommended to be Adam for deep learning tasks as a starting point [56], but it depends. However, if after trying all other memory reductive methods and nothing has reduced the computational cost sufficiently, one can replace the Adam optimizer with the SGD optimizer to reduce the gradient update's computations further.

Adapters are small pre-trained bottleneck layers inserted within each hidden layer of a transformer model. Instead of pre-training a language model and then fine-tuning on a large dataset for each possible task, adapters allows one to freeze the pre-trained model's parameter and only train the adapter layers while achieving the same accuracy in a shorter time [43, 45]. Additionally, since adapters are modular and task-specific, these can, in turn, be combined to improve the training on the same task but in a different language [44] and shared with others. Because of the recent release of the module and lack of pre-trained adapters as of writing, we decided not to use these.

3.4 Tools, Frameworks and Experiment Environment

Throughout this project, we used many libraries and programs to simplify training, maintain results from training, and maintain reproducibility.

We used the Huggingface Transformers library [67] version 3.4.0 for their pre-trained transformer models, tokenizers, and general training loop. We used the Huggingface datasets library version 1.1.3 to import and manipulate datasets for training and evaluation. We also used the deep learning framework PyTorch version 1.7.0 for additional data manipulation, custom data loaders, mixed-precision operations, distributed training, and utility functions for the training. The versions specified are not hard requirements except for Huggingface

[6] https://blog.paperspace.com/mixed-precision/
[7] https://docs.fast.ai/callback.fp16.html
[8] https://www.tensorflow.org/guide/mixed_precision
[9] https://github.com/cybertronai/gradient-checkpointing

transformers since creaking changes occurred in older and newer versions needed for running the experiments. For converting transformer models into Longformer type models, we based or training script on the conversion script provided by the Longformer authors Beltagyet al. in their Github repository.[10]

We used Tensorboard to capture the loss and other metrics from each experiment and to monitor results. We also created log files for each run to capture the output from training and evaluation. The results from the Tensorboard were then imported as CSV-files and plotted with Matplotlib using a modified 'seaborn-deep' color scheme.

We ran the experiments on two of Peltarion's remote servers with CUDA-enabled GPUs: one server with 24GB GPUs for monolingual QA training and another with 48GB GPUs for converting models into Longformers and multilingual training.

Smaller Server for Monolingual model

- 12 Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz
- NVIDIA TITAN RTX, 24 GB
- CUDA Toolkit 10.2
- Ubuntu 18.04.3 LTS

Larger Server for Multilingual model

- 8 Intel(R) Xeon(R) Gold 5222 CPU @ 3.80GHz
- NVIDIA Quadro RTX 8000, 48 GB
- CUDA Toolkit 10.2
- Ubuntu 18.04.3 LTS

We ran each model inside a Docker container for reproducible results and isolation of concerns, initialized with five different seeds for each experiment.

[10]https://github.com/allenai/longformer/blob/master/scripts/convert_model_to_long.ipynb

Chapter 4

Results

To evaluate our hypothesis: that long-context can be transferable to other languages by fine-tuning in one language only. We split the comparison of the models into two sets of two, based on if the model was monolingual or multilingual and if the model had been pre-trained using the Longformer pre-training to process longer sentences. These pre-trained models were then fine-tuned on multiple different QA datasets with different languages and context lengths.

The two tables below Table 4.1 and Table 4.4 describes an overview of the different models and datasets used for fine-tuning. The left-most column in the two tables described the pre-trained model used for fine-tuning and whether it was a monolingual English model or a multilingual model. The other columns describe the datasets used for fine-tuning. The number next to the dataset indicates the number of tokens the self-attention window could attend as a maximum. A model that has been trained into a long-context model can attend to more than 512 token. Therefore, for those models, a number is assosiated to long context datasets to indicate how many tokens the models could attend to during fine-tuning. If no number is next to the dataset, then the attention window is the default 512 tokens.

Fine-Tuning on Monolingual QA Datasets

Model	SQuAD	SQ3 (512)	SQ3 (4096)
Monolingual Models			
RoBERTa	X	X	-
Longformer	X	X	X
RoBERTa-Long	X	X	X
Multilingual Models			
XLM-R	X	X	-
XLM-L	X	X	X

Table 4.1 An overview of the pre-trained models and monolingual QA datasets used for fine-tuning in English only.

In Table 4.1, Longformer is the pre-trained Longformer released by the Longformer authors, and the RoBERTa-Long is the RoBERTa model we pre-trained from scratch using the same hyperparameters and base model as the Longformer authors. In Table 4.1 and Table 4.2, the model XLM-L is a multilingual XLM-R model that we pre-trained using the Longformer authors pre-training scheme - making it a long-context multilingual model.

Fine-Tuning on Multilingual QA Datasets

Model	XQuAD	XQ3 (512)	XQ3 (4096)	MLQA
Multilingual Models				
XLM-R	X	X	-	X
XLM-L	X	X	X	X

Table 4.2 An overview of the pre-trained models and multilingual QA datasets used for fine-tuning and evaluating long-context transferability.

4.1 Result for the Language Models

For the language models, the goal was to extend the context for a multilingual transformer model. However, since there are no such models to our knowledge as of writing, we strived to verify each process's validity and compare our results with once presented by other researchers. Therefore, we trained a monolingual model according to the Longformer pre-training scheme and compared our created RoBERTa to Longformer with the AllenAI research groups released Longformer model. The figures below depict the training and evaluation loss for the trained LMs. Running the Longformer pre-training scheme for 6000 gradient updates took five days and 13 hours for the RoBERTa-Long model and six days and 22 hours for XLM-Long.

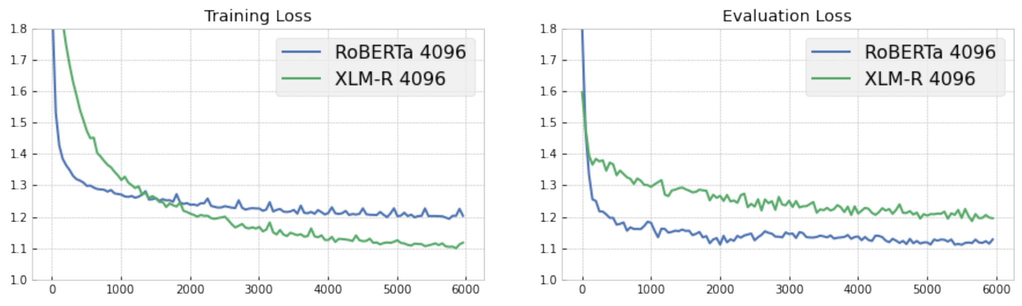

Figure 4.1 This figure depicts the training and evaluation loss when extending the context of a RoBERTa base and XLM-R base model for 6000 iterations. The loss is the negative log-likelihood between the label of the masked out token and the predicted token.

The bit-per character(BPC) score was calculated for each language model with extended context. Our RoBERTa model converted using the Longformer training scheme achieved a final BPC of **1.725** after $6K$ iterations and the converted XLM-RoBERTa a BPC of **1.604**. We compare this to the results presented by the Longformer authors (See table 5 of the Longformer paper [3]) when trained on the same Wikitext-103 dataset:

1. A regular pre-trained RoBERTa base model has a BPC of **1.846**
2. A Longformer base model trained for $2K$ iterations has a claimed BPC of **1.753**.
3. A Longformer base model trained for $65K$ iterations has a claimed BPC of **1.705**.

Our results indicate that the training scheme used follows closely to the one reported by the Longformer authors, characterized by smooth and converging training and validation loss at around $2K$ iterations of the RoBERTa-Long model and the similar BPC scores' reported for Longformer and our RoBERTa-Long. We can also see that the recommended early stopping of $2K$ gradient updates for a RoBERTa based Longformer corresponds well to how the model's training stabilizes near this point. However, the XLM-R based Longformer does not converge even past $6K$ iterations, and the validation loss is noticeably higher than for the monolingual model.

4.2 Result for the Monolingual QA Model

Fine-tuning each respective model on SQuAD, we compared how well the monolingual and multilingual models perform on the standard SQuAD 1.1 dataset, here called *Regular (512)* using a maximum sequence length of 512. Depending on the models' max length, these models were then trained and evaluated on SQ3 utilizing a maximum of 512 or 4096, described in Table 4.3 as SQ3 (512) and SQ3 (4096) respectively. The mean and standard deviation for the EM and F1 score is shown from five runs with different seeds.

	SQuAD 1.1					
	Regular (512)		SQ3 (512)		SQ3 (4096)	
Model	EM	F1	EM	F1	EM	F1
RoBERTa	84.3 ± 0.2	91.2 ± 0.1	81.4 ± 0.3	88.1 ± 0.1	-	-
Longformer	$\mathbf{85.1 \pm 0.2}$	$\mathbf{91.4 \pm 0.2}$	$\mathbf{84.8 \pm 0.2}$	$\mathbf{91.0 \pm 0.3}$	$\mathbf{84.3 \pm 0.2}$	$\mathbf{90.8 \pm 0.2}$
RoBERTa-Long	83.8 ± 0.2	90.8 ± 0.1	78.7 ± 0.2	83.4 ± 0.1	76.8 ± 0.1	81.8 ± 0.2
XLM-R	$\mathbf{81.9 \pm 0.3}$	88.7 ± 0.1	76.3 ± 0.3	83.0 ± 0.1	-	-
XLM-Long	81.9 ± 0.5	$\mathbf{89.1 \pm 0.1}$	$\mathbf{76.7 \pm 0.1}$	$\mathbf{83.5 \pm 0.0}$	$\mathbf{70.2 \pm 0.1}$	$\mathbf{77.3 \pm 0.1}$

Table 4.3 The SQuAD 1.1 dataset was evaluated using the regular SQuAD dataset (Regular) or the concatenated SQuAD dataset from three contexts (SQ3). The Number to the right of each dataset describes the maximum sequence length used for each model.

From the evaluations depicted in Table 4.3 we note that the pre-trained Longformer model from AllenAI performs the best across all datasets and with a marginal decrease in accuracy on datasets with extended context. We observe the same trend for our own RoBERTa model converted to a Longformer, called RoBERTa-Long, but with an overall lower score. The monolingual models yield a much higher overall score compared to the multilingual models, which is to be expected [13, 15, 68]. We also observe a noticeable improvement of the XLM-Long on the regular XQuAD dataset compared to the XLM-R and RoBERTa-Long model. We believe this to be mainly attributed to the additional English pre-training training. The XLM-Long seems to drop significantly across all evaluations and context lengths compared with the other long-context models Longformer and RoBERTa-Long. We believe that this could be attributed to numerous factors: using the same hyperparameters for XLM-Long as for the RoBERTa-Long when pre-training; the XLM-R base model may be under-parameterized when extending its context, etc. We will elaborate on this in more detail in the subsequent Chapter.

4.3 Result for the Multilingual QA Model

For the multilingual dataset, we similarly evaluated each model XLM-R and XLM-Long on the multilingual dataset XQuAD. We applied the same concatenation scheme used for the monolingual QA to XQuAD and called the dataset XQ3. The figure below depicts the training and evaluation loss for the monolingual and multilingual models.
We evaluated both the XLM-R and XLM-Long on the MLQA dataset. We compare our results with the baseline presented in the XLM-R paper [13]. By comparing our results

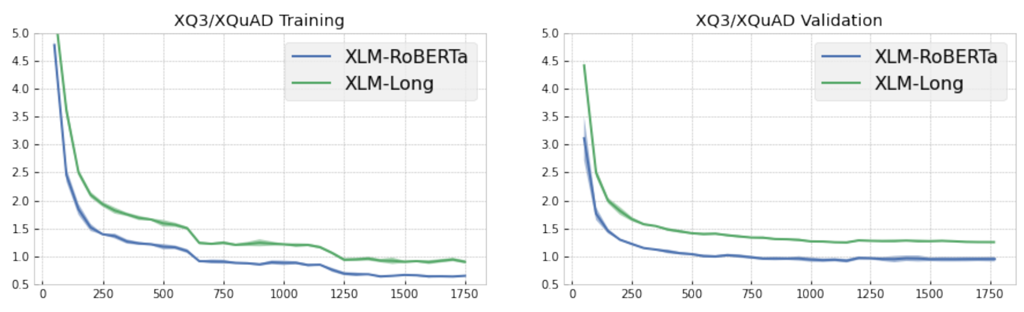

Figure 4.2 The training and evaluation loss for the XML-R and XLM-Long models, when trained on the XQuAD dataset. The loss is the negative log-likelihood between the predicted start and end tokens and the actual start and end tokens.

with the MLQA baseline results, we hoped to verify the correctness of fine-tuning and pre-training of our XLM-R and XLM-Long against trained XLM-R base models. We noted that our evaluations on the MLQA dataset correspond closely to the MLQA authors' results. This seems to suggest that our long-context training does not worsen the performance when fine-tuning on multilingual QA and that our choice of hyperparameters are satisfactory.

	MLQA							
Model	AR	DE	EN	ES	HI	VI	ZH	**AVG**
EM Score								
`BASELINE`	**36.6**	**46.7**	64.6	49.6	42.9	44.7	39.3	46.3
XLM-R	33.6	45.1	65.0	**50.0**	**46.2**	**45.1**	**43.2**	**46.9**
XLM-Long	31.7	45.3	**65.3**	49.7	45.6	44.8	41.8	46.3
F1 Score	-	-	-	-	-	-	-	-
`BASELINE`	**54.9**	60.9	77.1	67.4	59.4	64.5	61.8	63.7
XLM-R	51.9	59.9	78.1	**69.3**	**62.1**	**65.0**	**66.2**	**64.6**
XLM-Long	51.4	**61.0**	**78.4**	67.8	60.3	62.9	51.9	61.9

Table 4.4 Comparative table between the baseline XLM-R base model presented in the paper *Unsupervised Cross-lingual Representation Learning at Scale* [13, 35] and our trained base models: XLM-R and XLM-Long. The models were trained on the English SQuAD 1.1 dataset and evaluated using a zero-shot cross-lingual transfer on MLQA.

Finally, we compared how XLM-R and XLM-Long performed when evaluated on the regular XQuAD dataset and XQ3 with varying the maximal sequence length each model can process at a time. We note that the XLM-R and XLM-Long perform well on the XQuAD and XQ3 datasets with minor differences between the models, indicating that extended English training does not worsen the training or evaluation on datasets with shorter context.

	XQuAD - EM Score					
	Regular (512)		XQ3 (512)		XQ3 (4096)	
Language	XLM-R	XLM-Long	XLM-R	XLM-Long	XLM-R	XLM-Long
Arabic (AR)	49.6 ± 0.9	**49.9 ± 0.6**	**47.3 ± 1.5**	46.0 ± 1.5	-	36.6 ± 1.6
German (DE)	59.3 ± 0.4	**59.7 ± 1.1**	**52.7 ± 1.7**	51.7 ± 1.8	-	46.4 ± 2.0
Greek (EL)	**55.6 ± 0.5**	55.1 ± 1.3	43.8 ± 0.2	**46.2 ± 0.8**	-	36.7 ± 1.1
English (EN)	73.0 ± 0.8	**73.3 ± 0.0**	**70.3 ± 0.8**	70.1 ± 0.2	-	64.2 ± 1.6
Spanish (ES)	57.5 ± 0.9	**57.7 ± 0.6**	**51.7 ± 0.4**	51.2 ± 2.0	-	45.8 ± 1.5
Hindi (HI)	50.1 ± 1.4	**51.2 ± 0.7**	**51.0 ± 1.1**	**51.0 ± 0.0**	-	36.4 ± 1.6
Russian (RU)	**57.5 ± 0.5**	56.8 ± 0.2	**54.4 ± 0.8**	54.0 ± 0.4	-	45.8 ± 0.6
Taiwanese (TH)	**49.2 ± 1.3**	47.6 ± 1.2	**40.1 ± 1.2**	38.4 ± 2.4	-	24.1 ± 0.9
Turkish (TR)	**49.8 ± 1.0**	49.7 ± 0.8	**52.3 ± 1.1**	52.1 ± 1.7	-	40.9 ± 2.2
Vietnamese (VI)	**54.4 ± 0.2**	53.3 ± 1.2	**47.1 ± 0.0**	44.7 ± 0.2	-	37.6 ± 1.9
Chinese (ZH)	53.3 ± 1.1	**54.4 ± 0.8**	50.4 ± 0.4	**51.9 ± 2.3**	-	42.1 ± 1.5

	XQuAD - F1 Score					
	Regular (512)		XQ3 (512)		XQ3 (4096)	
Language	XLM-R	XLM-Long	XLM-R	XLM-Long	XLM-R	XLM-Long
Arabic (AR)	65.2 ± 1.1	**66.1 ± 0.9**	**62.4 ± 1.4**	61.9 ± 0.4	-	51.3 ± 1.4
German (DE)	**74.8 ± 0.5**	74.7 ± 0.7	**67.7 ± 1.1**	66.9 ± 0.8	-	59.7 ± 1.0
Greek (EL)	**72.5 ± 0.3**	72.3 ± 0.3	58.6 ± 0.3	**61.1 ± 0.5**	-	48.9 ± 1.6
English (EN)	**84.0 ± 0.6**	83.8 ± 0.1	81.3 ± 0.3	**81.6 ± 0.5**	-	74.7 ± 1.3
Spanish (ES)	**75.8 ± 0.4**	75.6 ± 0.3	69.7 ± 0.1	**71.1 ± 0.6**	-	60.8 ± 1.5
Hindi (HI)	66.1 ± 0.7	**67.0 ± 0.3**	65.1 ± 1.1	**65.2 ± 0.1**	-	47.1 ± 1.5
Russian (RU)	**73.6 ± 0.3**	73.0 ± 0.4	**68.7 ± 0.1**	68.5 ± 0.6	-	59.7 ± 0.6
Taiwanese (TH)	**60.6 ± 1.0**	58.8 ± 0.8	**49.7 ± 1.3**	49.1 ± 1.1	-	33.5 ± 0.8
Turkish (TR)	**66.1 ± 0.2**	65.8 ± 0.6	**66.6 ± 0.1**	65.7 ± 0.6	-	55.5 ± 0.9
Vietnamese (VI)	**73.6 ± 0.2**	72.6 ± 0.2	**67.6 ± 0.3**	65.4 ± 0.0	-	54.3 ± 1.3
Chinese (ZH)	**63.3 ± 0.5**	62.9 ± 0.6	60.8 ± 1.0	**62.2 ± 1.1**	-	51.7 ± 2.7

Table 4.5 Evaluation of the XLM-R and XLM-Long model on variations of the XQuAD dataset. Regular is the regular XQuAD dataset, XQ3 is a concatenated XQuAD dataset following the same scheme as the SQ3 dataset for SQuAD. The first table describes the EM score for the respective models and datasets and the F1 score.

4.4 Fine-Tuning a Monolingual model on a Multilingual Dataset

In order to verify that multilingual models were needed to perform well on multilingual QA tasks, we fine-tuned a monolingual RoBERTa model and a XLM-R model on the XQuAD dataset and compared the results. As expected, using a multilingual model showcased drastic improvements when fine-tuning on a multilingual downstream task. This verified that using a Longformer model for low-resource languages was not a viable method and that investigating multilingual models with longer contexts was needed to make low-resource language models more practically viable.

	XLM-R		RoBERTa	
Language	EM	F1	EM	F1
Arabic (AR)	**49.6 ± 0.9**	65.2 ± 1.1	1.93 ± 0.21	5.78 ± 0.65
German (DE)	**59.3 ± 0.4**	**74.8 ± 0.5**	28.35 ± 1.18	43.19 ± 1.15
Greek (EL)	**55.6 ± 0.5**	**72.5 ± 0.3**	2.55 ± 0.26	5.72 ± 0.42
English (EN)	73.0 ± 0.8	84.0 ± 0.6	**75.10 ± 0.10**	**86.07 ± 0.22**
Spanish (ES)	**57.5 ± 0.9**	**75.8 ± 0.4**	40.53 ± 1.82	57.84 ± 1.27
Hindi (HI)	**50.1 ± 1.4**	**66.1 ± 0.7**	1.88 ± 0.10	4.73 ± 0.69
Russian (RU)	**57.5 ± 0.5**	**73.6 ± 0.3**	2.77 ± 0.48	7.14 ± 0.28
Taiwanese (TH)	**49.2 ± 1.3**	**60.6 ± 1.0**	1.85 ± 0.25	4.41 ± 0.57
Turkish (TR)	**49.8 ± 1.0**	**66.1 ± 0.2**	6.97 ± 0.42	14.33 ± 0.60
Vietnamese (VI)	**54.4 ± 0.2**	**73.6 ± 0.2**	6.39 ± 0.61	12.60 ± 0.79
Chinese (ZH)	**53.3 ± 1.1**	**63.3 ± 0.5**	3.28 ± 0.49	6.52 ± 0.23

Table 4.6 Fine-tuning an XLM-R and RoBERTa model on the XQuAD dataset and comparing the results.

Chapter 5

Discussion

In this section, we reflect on the results from the respective experiments. We will start by discussing trends noted from a decreasing performance between evaluation on the different datasets, tasks, and varying context-length. We will discuss possible limitations and choices made for the evaluations and suggest possible improvements. We will also discuss the performance difference between the monolingual and multilingual models and how the extended long-context models could be improved.

We observed that the Longformer based training scheme for extending the context behaves as expected. Both the Longformer and RoBERTa-Long perform similarly in terms of the BPC. We also observed that the training and evaluation loss converged around the same number of gradient updates reported by the Longformer authors. We noticed a lower performance on downstream task using our RoBERTa-Long compared to the pre-trained Longformer. We believe that this is because we stoped our long-context training at 6000 gradient iterations, whereas the Longformer authors trained theirs for 65000 iterations. This seems to highlight that even though the training and evaluation loss converged and the authors of the paper recommended 3000 iterations as an early-stopping point, training for longer does seem to improve downstream performance.

However, when extending the multilingual model's context, the training loss did not converge within the set number of training iterations. This means that the XLM-Long model could be improved further for longer contexts in English. When training the models for extended context, the XLM-R models use the same hyper-parameters for our monolingual RoBERTa-Long. By not having a convergent loss and non-optimized hyper-parameters for the multilingual model, it is possible that the performance on downstream tasks could be improved further. However, there seem to be conflicting options [38, 69] in the matter, where the RoBERTa authors claimed that a smaller improvement in pre-training could result in a noticeable improvement when fine-tuning on a downstream task, while the XLM authors argue that is not clear causation between the two.

From the evaluations made on the SQuAD dataset (See Table 4.3) using different configurations (SQuAD, SQ3 (512), SQ3 (4096)), we see a general trend of drop in performance between SQuAD and the concatenated filtered dataset SQ3 (512), when dividing and processing segments in maximum lengths of 512. This is expected since even though the maximum context length for most SQuAD samples is 512 tokens or below, the filtering process needed to guarantee that the same context is newer concatenated means filtering out questions that uses the same context. This means that the training and testing datasets with longer context are $\frac{1}{5}$ of the original dataset, meaning that the performance will be lower since there is less training data. It also means that there are more possibilities for where the answer (minimal span) can be or a higher chance to make incorrect predictions when using a longer context.

An additional drop in performance is observed for the extended long-context models when setting the maximum sequence length from 512 to 4096. This echoes the result of the Transformer-XL and Longformer paper: model performance will degrade with longer sequences. It seems plausible that if three contexts are concatenated together, each with an average maximum sequence length below 512, we would assume a higher performance if each segment is processed 512 tokens instead of the whole concatenated context. We believe an additional contributing factor is that SQuAD does not require the model to retrain information over long passages of text, something the dataset HotpotQA and TriviaQA are designed to test. We believe that if we would run the same evaluations for the extended models, then the results would be much lower when limiting the models to a maximum sequence length of 512 and higher when using the full sequence. We also assume that by creating a longer context, it has more segments to predict where the answer is, decreasing the performance.

It has been noted that training a monolingual model on one language only tends to yield better performance for the monolingual model [13, 68]. We suspect this to be from *under-parameterization* - that the model has not enough parameters to fit the source language optimally. For instance, for the pre-trained RoBERTa and XLM-R, the XLM-R base model contains about twice as many parameters as the RoBERTa base model but trained on 99 additional languages and with a nearly five times larger tokenization vocabulary [13, 38]. By increasing the number of parameters *i.e.,* using the large model and not the base model, we will see an improvement by 5 to 10 units across all the languages [13, 35]. We also note that since the multilingual datasets contain a fraction of English examples in the test set, compared to SQuAD, we also expect the performance to decrease when evaluating on multilingual SQuAD-formated datasets. Other contributing factors are likely to be that the same training and fine-tuning parameters were used between the monolingual and multilingual models. To achieve the best possible result and comparison, we would run hyper-parameter tuning for the respective models. However, we do not expect the multilingual models to increase their performance significantly with slightly better choices of hyper-parameters.

We observe only a minor difference between the XLM-R and XLM-Long model in evaluation on SQuAD and SQ3. We believe that the marginally higher score for XLM-Long is because of the additional pre-training on English. Noticeably, we observe nearly the same EM and F1 score for the two models for both regular XQuAD and XQ3. Interestingly, we note that the near-linear decrease in EM and F1 score observed for the monolingual models in Table 4.3 also holds true for the multilingual models in Table 4.5. We believe this is because of the same reason that we observed a lower score between SQ3 and SQuAD - namely that XQ3 has a training and testing dataset one fifth compared to XQuAD.

Since we could not observe a definitive increase in performance between having a larger or shorter attention window on the extended multilingual dataset SQ3 and XQ3, we can not conclude that long context can be transferred to a low-resource language. We believe that our hypothesis could be answered by acquiring a multilingual dataset that can not be answered by exact sentence extraction with a short attention window. Instead, we would investigate translating a dataset that requires multi-sentence reasoning, such as TriviaQA [27, 28].

Chapter 6

Conclusion and Future Work

This thesis aimed to investigate methods to efficiently and practically extend the context of transformer-based models in low-resource languages. We reasoned that pre-training a multilingual or even a bilingual language model for any low-resource language would be too costly and time-consuming. We therefore decided to investigate if long-term context can be cross-lingually transferred by training in a high-resource language such as English. Based on our results, we can neither confirm nor deny if long-term context can be transferred, since current multilingual datasets for QA does not require retention beyond 512 tokens to answer most task - even when the context is extended artificially. We managed to showcase that the results of the XLM-R and our long-context XLM-R performed nearly identically and that the long-context models performed equally as well as a regular pre-trained transformer based model on datasets with short context lengths. We also showcased that our results corresponded with those reported by other researchers when using the same dataset and models.

We did not manage to definitively conclude if long-context can be transferred from a high-resource language to a lower-resource language without additional long-context training in the target language within the set time limit for the project. In regards to our research questions, we could conclude that:

1. The Longformer training scheme was an effective method for extending the context of a multilingual model for a single language.
2. We could not conclude if cross-lingual transfer is possible. We believe that this is primarily because of the QA datasets we created and used for fine-tuning and evaluation.
3. Our method for concatenating existing datasets to create longer contexts artificially could not be shown to measure long-term reasoning in multilingual models. We believe that using a more complex dataset that requires multi-sentence reasoning, such as TriviaQA [28] and translating it to multiple languages [27], would allow us to conclude the validity of our hypothesis.
4. Training a multilingual model and extending the context in one language did not significantly worsen the model's performance in other languages for downstream multilingual QA. This seems to hold true both for datasets with shorter contexts and those with artificially extended context.
5. We did not manage to confirm if a multilingual model with an extended context in English could be trained using the Longformer scheme in an additional language to achieve a better cross-lingual transfer. We believe the primary reason for this is the current lack of long-context datasets in languages besides English.

For future work, the primary interest would be to use long-context datasets in languages other than English. However, to our knowledge, there is no such dataset currently. To confirm our hypothesis's validity, we would therefore primarily focus on either translating an existing long-context dataset such as TriviaQA or WikiHop into another language or (ideally) if such a low-resource dataset could be created. A promising prospect is the Vinnova funded research project called *SuperLIM* [1] - A Swedish equivalent to the English evaluation metric GLUE and SuperGLUE, where the aim is to create a large corpus of text for evaluating language models on various NLP tasks. One of the datasets proposed by Peltarion to include in SuperLIM is a long-context dataset for question-answering. Ideally, when the SuperLIM is released, we could then evaluate low-resource datasets that require long-context reasoning using our models and definitively determine if long-context cross-lingual transfer is possible.

We would also focus on finding the XLM-R and XLM-Long models' optimal parameters for pre-training of extended context and fine-tuning on downstream tasks. We noted that training and evaluation loss did not converge for the XLM-Long training, indicating that the BPC and evaluation on downstream tasks potentially can be improved further. We believe that by fine-tuning on a multilingual dataset that requires long-context reasoning, we could definitively answer all of our research questions.

If evaluation could be done on a more complex dataset in a low-resource language, we would investigate how larger language models affect long-context transferability. Suppose there is a minor difference in evaluating such a long-context dataset between using a long and short maximum sentence length; in that case, it could be interesting to investigate if it is because of under-parametrization the initial hypothesis is incorrect. On the other hand, if the assumption is correct and we can observe a noticeable difference, it would be interesting to observe such a model's performative limit.

If cross-lingual transfer of long-context could be confirmed, we would investigate if a multilingual model with an extended context in English could be trained with the same Longformer training scheme in another language. If true, this could prove to be a computationally affordable and efficient method for training a long-context model in a low-resource language without sacrificing performance. This is because it allows the training to start from a pre-trained language model and requires much less long-context data, and trains faster than training an efficient transformer model from scratch.

We would also like to investigate different avenues than using a multilingual model and extending its context. The efficient transformer model Reformer claims to handle training models with a maximum sequence length of 0.5 million on an 8GB GPU. An interesting approach could be to investigate if it is more usable and practical to train a multilingual Reformer model. This model could be multilingual, have a very long context, and have a small memory footprint. If training a language model were not so expensive, this would have been our preferred choice to investigate instead of the one presented in this thesis.

Although QA is generally considered a good indicator for cross-lingual transferability and requires a comprehensive language understanding [12], only evaluation on QA tasks may not express full language comprehension. It could well be that the models overfit for the specific tasks of QA [12] and we would therefore like to fine-tune and evaluate the models on other tasks to ascertain general language comprehension better.

[1]https://www.vinnova.se/p/superlim-en-svensk-testmangd-for-sprakmodeller/

Bibliography

[1] Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations, 2020.

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

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]*, April 2020.

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures, 2012.

[5] Alexander Bergkvist, Nils Hedberg, Sebastian Rollino, and Markus Sagen. Surmize: An online nlp system for close-domain question-answering and summarization, 2020.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[8] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*, 2019.

[9] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[10] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[11] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[12] Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages, 2020.

[13] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[14] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019.

[16] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation, 2019.

[17] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015.

[18] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *arXiv preprint arXiv:1707.04585*, 2017.

[19] Alex Graves. Generating sequences with recurrent neural networks, 2014.

[20] Gregory Grefenstette. Tokenization. In *Syntactic Wordclass Tagging*, pages 117–133. Springer, 1999.

[21] Audrūnas Gruslys, Remi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time, 2016.

[22] Fengxiang He, Tongliang Liu, and Dacheng Tao. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems*, pages 1143–1152, 2019.

[23] John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://www.aclweb.org/anthology/D19-1275.

[24] Dichao Hu. An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 432–448. Springer, 2019.

[25] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization. *arXiv:2003.11080 [cs]*, September 2020.

[26] Chip Huyen. Evaluation metrics for language modeling. *The Gradient*, 2019.

[27] Tim Isbister and Magnus Sahlgren. Why not simply translate? a first swedish evaluation benchmark for semantic similarity, 2020.

[28] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

[29] Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. The state and fate of linguistic diversity and inclusion in the nlp world, 2020.

[30] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer, 2019.

[31] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv e-prints*, pages arXiv–2001, 2020.

[32] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[33] Chia-Hsuan Lee and Hung-Yi Lee. Cross-lingual transfer learning for question answering. *arXiv preprint arXiv:1907.06042*, 2019.

[34] Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. MLQA: Evaluating Cross-lingual Extractive Question Answering. *arXiv:1910.07475 [cs]*, October 2019. Comment: To appear in ACL 2020.

[35] Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. Mlqa: Evaluating cross-lingual extractive question answering, 2020.

[36] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations, 2019.

[37] Qi Liu, Matt J. Kusner, and Phil Blunsom. A Survey on Contextual Embeddings. *arXiv:2003.07278 [cs]*, April 2020.

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

[39] Aditya Malte and Pratik Ratadiya. Evolution of transfer learning in natural language processing. *arXiv:1910.07370 [cs]*, October 2019.

[40] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[41] Madison May. A Survey of Long-Term Context in Transformers. https://www.pragmatic.ml/a-survey-of-methods-for-incorporating-long-term-context/, March 2020.

[42] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[43] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[44] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020.

[45] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*, 2020.

[46] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

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

[48] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[49] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[50] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv:1606.05250 [cs]*, October 2016. Comment: To appear in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)
Comment: To appear in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[51] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020.

[52] Sebastian Ruder. 10 Exciting Ideas of 2018 in NLP. https://ruder.io/10-exciting-ideas-of-2018-in-nlp/, December 2018.

[53] Sebastian Ruder. 10 ML & NLP Research Highlights of 2019. https://ruder.io/research-highlights-2019/, January 2020.

[54] Sebastian Ruder. Why You Should Do NLP Beyond English. `http://ruder.io/nlp-beyond-english`, 2020.

[55] Sebastian Ruder, Ivan Vulić, and Anders Søgaard. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631, Aug 2019. ISSN 1076-9757. doi: 10.1613/jair.1.11640. URL `http://dx.doi.org/10.1613/jair.1.11640`.

[56] Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley – benchmarking deep learning optimizers, 2020.

[57] Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64, 1951.

[58] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018.

[59] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[60] Claudia Soria, Valeria Quochi, and Irene Russo. The dldp survey on digital use and usability of eu regional and minority languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[61] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[62] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020.

[63] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient Transformers: A Survey. *arXiv:2009.06732 [cs]*, September 2020.

[64] Daan van Esch, Elnaz Sarbar, Tamar Lucassen, Jeremy O'Brien, Theresa Breiner, Manasa Prasad, Evan Crew, Chieu Nguyen, and Francoise Beaufays. Writing across the world's languages: Deep internationalization for gboard, the google keyboard, 2019.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. Comment: 15 pages, 5 figures.

[66] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[67] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*, July 2020.

[68] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer, 2020.

[69] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[70] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[71] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. *arXiv:2007.14062 [cs, stat]*, July 2020.

Appendix

The Transformer in Action

To illustrate how the transformer architecture works, we use machine translation from English to Swedish as an example.

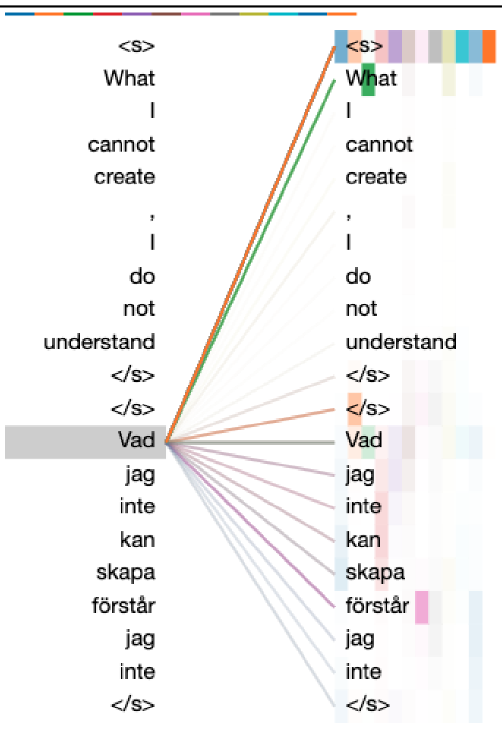

Figure 6.1 This is an illustration of how the encoder-decoder attention is applied when translating an English quote by the physicist Richard Feynman into Swedish. More vibrant coloring indicates more attention is placed on those words, and more colors for a word, indicates it attends to multiple words. The figure above is generated using the open-source library Bertviz using an XLM-R model.

After each Multi-head attention layer, an additive and normalization layer is used in both the encoder and decoder, followed by a feed-forward layer with another additive and normalization layer. The feed-forward layer takes the vector representation of the word and converts it to the correct dimensions. The normalization layer decreases the training time and reduces the risk of overfitting [65].

1. English and Swedish words are converted from text into an embedding (See § 2.1.2) and composed with a positional encoding, which is used to indicate the ordering of the words in each sequence.

2. For the encoder block, the encoded English sentence is passed through a multi-head attention block, which assigns a normalized weight or *attention score* for how important every other word in the context of anyone (See Figure 2.2).
3. The resulting multi-head operation outputs an attention weight matrix, which is additively applied to the encoded English Sentence and normalized.
4. The resulting matrix is then passed through a feed-forward network and normalized and sent to combined multi-head attention in the decoder. The entire sentence from the encoder block is processed before being passed along to the decoder (Step 6).
5. For the decoder, the Swedish sentence is first encoded and composed with a positional encoding and passed to a *masked multi-head attention* block. The multi-head self-attention layer in the decoder is unidirectional, meaning it only attends to the previously seen tokens in a sentence. This ensures that the model learned to predict new words only based on what it has previously seen.
6. The final attention layer is the encoder-decoder self-attention and is applied from the input in the encoder (English) to the output in the decoder (Swedish). This is where the mapping between the two languages are learned, whereas the previous two self-attention blocks learned the semantic representation for English and Swedish separately.
7. The output is finally passed through a linear layer and a Softmax output layer that yields a probability over all Swedish tokens the English token could be translated into. The Swedish token with the highest probability is then passed along with all the previously translated Swedish tokens as input for the decoder (Step 5). This process is repeated until the whole sentence has been translated.

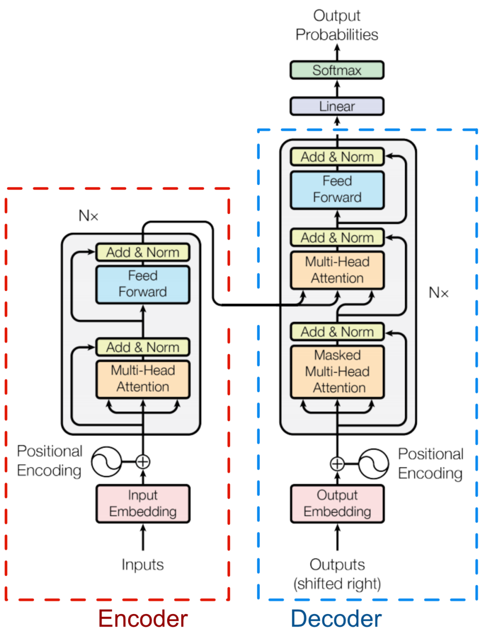

Figure 6.2 Figure depicting the Vaswani et al transformer encoder-decoder model [65]