LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
CENTER FOR INFORMATION AND LANGUAGE PROCESSING

Masterarbeit

im Studiengang Computerlinguistik

an der Ludwig-Maximilians-Universität München

Fakultät für Sprach- und Literaturwissenschaften

Summarization of Texts using a Pretrained Language Model

vorgelegt von
Anna Bettina Steinberg

Betreuer:	Dr. Benjamin Roth
Prüfer:	Dr. Benjamin Roth
Bearbeitungszeitraum:	09. September 2019 - 27. Januar 2020

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, den 27.Januar 2020

. . . . . .
Anna Bettina Steinberg

Abstract

Large pre-trained language models have already been proven useful for several downstream tasks. For abstractive summarization, they have either been used as input for sequence-to-sequence models or finetuned to the task specifically. This thesis explores to what extent the Transformer-based language model, XLNet, can be applied to abstractive summarization without finetuning. Results on ROUGE scores indicate that for unsupervised summarization, XLNet does not reach state-of-the-art levels. Applying a repetition penalty and possibly adding a prompt can however improve performance.

Contents

List of Figures

List of Tables

1 Introduction

Recent development in natural language processing has focused on producing models that are trained on an immense amount of data and for various tasks. The goal is to arrive at an optimal vector representation of a word which captures the word's meaning in diverse contexts. These so-called pretrained language models can then be used to finetune the model for more specific tasks, such as classification or question answering (Devlin et al., 2019, for BERT). Considering the successful performance of these pretrained language models for classification, the question arises whether these models are also apt for tasks involving natural language generation (NLG). One such task in the realm of NLG is summarization which aims to produce a condense version of a text containing its essential information. This can be either achieved by extracting important sentences from the text to form the summary or by generating a completely new, shorter text. For the latter task of abstractive summarization, previous work (Edunov et al., 2019; Khandelwal et al., 2019; Liu and Lapata, 2019) has already indicated that using and finetuning pretrained language models produces promising results.

With the creation of XLNet (Yang et al., 2019), a new way of thinking about language modeling has emerged. While previous approaches such as ELMo (Peters et al., 2018) or GPT-2 (Radford et al., 2019) trained a model by sequentially processing the sentence, XLNet has introduced the concept of permutations. Thus, the resulting embedding of a word incorporates a richer context than previous models. XLNet has beaten previous models in several downstream tasks. This thesis examines the question whether it is also superior for unsupervised summarization.

Previous work on summarization (Gehrmann et al., 2018; See et al., 2017) has identified repetition of words as a primary weakness. To mitigate this issue, this thesis adds a repetition penalty to the unsupervised system and explores different penalty schemes.

1.1 Goal of this Work

This thesis tries to answer the question whether pretrained language models can be effectively used for summarization, specifically for abstractive summarization. The pretrained language model that is used for this purpose is XLNet. This model is unique in its approach compared to previous models because of its permutational language modeling objective. The essence of the thesis consists in analyzing whether this method is particularly suitable for the task of unsupervised summarization.

1.2 Outline

This paper is structured as follows: Chapter 2 introduces related work in the realm of pretrained language models and summarization. Particularly, the language models GPT-2 (Radford (2018)) and BERT Devlin et al. (2019) are presented (Section 2.2). Previous work on the use of pretrained models for summarization is also discussed in Section 2.3. Chapter 3 covers the description of the data used in this thesis (Section 3.1.1) as well as the relevant preprocessing steps (Section 3.1.2). Furthermore, it introduces the main model of interest, XLNet, in detail in Sections 3.2.1 and 3.2.2 and demonstrates how it is applied to the summarization task (Section 3.2.4), including the different settings 3.3. The results of the experiments are provided in Chapter 4. A quantitative as well as a qualitative analysis of the results are given in Sections 4.1 and 4.3, respectively. Chapter 5 concludes this thesis and gives an outlook on future work. The code implementing the results of this thesis is published on *GitHub*.[1]

[1] `https://github.com/anbestCL/XLNetforSummarisation`

2 Related Work

2.1 Summarization

Summarization is the process of compressing a text to a brief description of its essential elements. The meaning of the text should be preserved in this process. Such a compression of a text can be achieved in two ways, either by selecting original sentences from the text containing essential information and grouping them to form a summary (*extractive summarization*) or by creating a text token-by-token from scratch (*abstractive summarization*).

In extractive summarization, the main difficulty consists in the selection process of sentences. For example, one might employ a relevance score for each sentence. This can be reinterpreted as a classification task where the goal is to divide the text into essential and non-essential sentences. To generate the summary, the selected sentences can be simply joined together, possibly using pre-selected conjunctions, or an ordering of the candidates can be performed and the top n sentences are chosen.

This thesis focuses on abstractive summarization. In abstractive summarization, a model does not only have to identify the topic and the key phrases of the text, but also has to reformulate and express the content in its 'own' words. The prevalent approach (Rush et al., 2015; Nallapati et al., 2016, for example) has been to use an encoder-decoder structure, originally developed for machine translation (Bahdanau et al., 2014). The encoder part of the architecture is tasked with creating an optimal hidden representation of fixed length of the source sentence while the decoder then generates the target sentence from this representation an output token at a time using a fixed target vocabulary. Rush et al. (2015) apply such a system to abstractive summarization and experiment with a simple bag-of-words encoder, an encoder with convolutional layers and an attention-based encoder, following the attention mechanism (Bahdanau et al., 2014). They demonstrate that given a large amount of data, it is effectively possible to train a summarization model with little linguistic features.

A common design question when working with a fixed vocabulary is how to treat rare or unseen words during testing. Instead of using a placeholder for unknown words, Nallapati et al. (2016) use a pointer-generator switching mechanism which controls whether the token is taken from the vocabulary (generator) or a token is copied from the source at a position pointed to by a pointer. See et al. (2017) extend the usage of the switching mechanism from out-of-vocabulary words and named entities (Nallapati et al., 2016) to

all words. Thus, their model learns freely when to use the pointer (See et al., 2017, see Chapter 3, Pointer-generator networks).

Furthermore, See et al. (2017) adapt the concept of coverage to summarization in order to avoid repetition. They introduce a coverage vector which is passed to the attention mechanism as additional input. They state that this way the attention mechanism is reminded of its previous decisions. Additionally, a coverage loss is defined to penalize repeated attention to the same locations. The repetition penalty used in this thesis is largely inspired by the concept of coverage and coverage loss.

Besides the pointer mechanism, another concept from extractive summarization which has been successfully adopted to abstractive summarization is the aspect of content selection. Gehrmann et al. (2018) demonstrate how restricting the training of neural models to selected portions of the text largely improves the summaries' quality without sacrificing the positive fluency aspect of abstractive summarization.

2.2 Language Models

Language modeling can be described as the task of assigning a probability to a sequence of words. Mathematically, the probability of a sequence of tokens $x_1, \ldots, x_n$ can be refactorized as a product of multiple conditional probabilites

$$p_\theta(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i | x_1, \ldots, x_{i-1}) \tag{2.1}$$

Each conditional probability represents the probability of a word in a sequence given the previous words. The θ parametrizes the probability distribution.

These probabilites can be learned by passing sequences to a deep neural network which maximizes Eq.2.1 for all sequences of the data. Such models were first trained unidirectionally (Bengio et al., 2003), i.e. the model processes tokens from left to right. ELMo, a system developed by Peters et al. (2018), combines the representations, obtained from the forward and the backward pass through the sequence, to account for bidirectional context.

While ELMo is built on recurrent neural networks using Long Short Term Memory units (Hochreiter and Schmidhuber, 1997), more advanced language models are based on the Transformer architecture (Vaswani et al., 2017). The Transformer architecture follows an encoder-decoder set-up containing several stacked layers in each block. The novum of the architecture is that it forgoes recurrence completely, but relies solely on self-attention. Each encoder layer consists of a multi-head self-attention unit and a feed-forward neural network, while each decoder layer contains a masked self-attention unit, an encoder-decoder self-attention unit and a feed-forward neural network on top. Incorporating self-attention has enabled the model to identify relevant parts of the sequence and better cope with long sequences of text (Liu et al., 2018). The Transformer structure has originally

been developed for machine translation, but has subsequently been successfully applied to language models. For example, Radford et al. (2019) build their GPT-2 language model on the decoder block of the Transformer, with the only modification of leaving out the encoder-decoder self-attention unit in each decoder layer since no encoder is attached.

The Transformer architecture is also the basis of BERT (Devlin et al., 2019). This model is strictly speaking not a language model because it does not optimize sequence probabilities as in Eq. 2.1. However, it has outperformed previous models in several language understanding tasks (see Devlin et al., 2019, Ch. 4 on experiments) and relates directly to the main model of this thesis, XLNet (Yang et al., 2019). When a sequence is passed to BERT for training, it replaces a percentage of the tokens by the '<mask>' token and another percentage by random words, thus corrupting the input. The model then aims to reconstruct the original input from the corrupted version while having access to both, left and right, context. In comparison to GPT-2, BERT is built on the encoder part of the Transformer architecture and uses self-attention instead of masked self-attention, enabling BERT to 'look ahead' in the sequence. Thus, BERT works bidirectionally while GPT-2 processes input unidirectionally. Devlin et al. (2019) argue that this left-to-right prediction structure limits GPT-2 since it does not lend itself well to sentence-level tasks for example. BERT's pretrained representations are, on the other hand, so universal that they can be finetuned to a downstream task using only one additional output layer. This is also due to BERT's specific input structure. Each input sequence starts with '<cls>' token (short for classification) whose hidden representation is used as a sentence-level representation of the input. If the input consists of a sentence pair, for example for question answering, the two sentences are split by the '<sep>'.

Masking is one of the main features of BERT, but it is also one of its limitations. The '<mask>' token is only used during training. When BERT is used in downstream tasks, it is no longer present, thus constituting a train-test discrepancy. Devlin et al. (2019) are aware of this issue and additionally they replace 10% of the tokens selected for masking with random real words. This introduces though unneccessary random noise to the data.

Furthermore, Yang et al. (2019) criticize that BERT assumes parallel conditional independence of masked tokens, conditioned on the unmasked tokens and the model parameters, during training. The model's objective is to reconstruct the masked sequence given the corrupted sequence. This conditional probability is decomposed into a product of conditional probabilities of the individual masked tokens, conditioned on the corrupted sequence, implying that the masked tokens are reconstructed separately. For an example consider the sentence

> 'A banking crisis erupted in 2012.'

If during training both, 'banking' and 'crisis', are masked, the model would treat both these masked tokens independently given the other unmasked tokens in the sequence, although there is clearly a linguistic link between them. However, this is a weak critique since, assuming that the model already implicitly incorporates information about the link

between the two masked words from the surrounding context, the actual knowledge of the token 'banking' might no longer be needed when reconstructing the token 'crisis'.

Returning to classical language modeling, XLNet overcomes BERT's shortcomings in two ways. It avoids the use of a "<mask>" token, but still captures bidirectional context by performing language modeling on permutations of the input sequence. Yang et al. (2019) adapt this concept of *permutational language modeling* from Uria et al. (2016). Since XLNet optimizes sequence probabilities using forward factorization, no indepence assumption is needed.

An important difference between langauge models such as ELMo using recurrent neural networks and Transformer layers lies in the input to the model. While recurrent neural networks are inherently aware of the order of the sequence, attention-driven structures require positional encodings as an additional input to indicate the order of tokens. Up until XLNet, token embedding and positional encoding have been coupled together. The introduction of *two-stream self-attention* lets XLNet treat token and position embedding separately. When a token is predicted, its embedding is masked, but its position encoding is not. Besides token and position embeddings, BERT and XLNet also share the concept of segment embeddings. In BERT, each sentence of a sentence-pair receives its own segment embedding, while XLNet works with *relative segment embeddings*. XLNet and its features are further detailed in Section 3.2.

2.3 Summarization using Pretrained Language Models

To apply pretrained language models to summarization, the first step is to pass the representations obtained from the language models to a classical sequence-to-sequence model to improve performance. Edunov et al. (2019) implement this by passing the ELMo representations to an attention-based encoder-decoder setup. Their best-performing system for abstractive summarization on the CNN/DailyMail dataset inputs ELMo embeddings to the encoder only, shares learned word representations with the decoder and ties input and output embeddings together. This setting outperforms See et al. (2017), whose word embeddings are trained from scratch on the same dataset. Furthermore, it surpasses results of Gehrmann et al. (2018) who finetune and then concatenate GLoVE (Pennington et al., 2014) and ELMo embeddings. Note that Edunov et al. (2019) are aware of the repetitive nature of their model and disallow repeating the same trigram when generating.

Radford et al. (2019) directly test their GPT-2 model for summarization on the CNN/DailyMail dataset by appending the hint 'TL;DR:' to each article and generating 100 tokens with top-2 random sampling. The first 3 sentences of this generated sequence serve as the summary. This approach, however, does not measure up to the results of Gehrmann et al. (2018) and performance drops significantly when the hint is left out.

Following Gehrmann et al. (2018) by introducing extractive summarization aspects to abstractive summarization, Subramanian et al. (2019) split the summarization task into

an extractive step and an abstractive step. Their corpora is mainly a collection of scientific papers. First, a hierarchical sequence-to-sequence sentence pointer identifies important sentences from the text. Then the introduction, the extracted sentences, possibly an abstract and the rest of the text are passed to a Transformer-based language model similar to GPT-2 which is trained from scratch for summarization. Subramanian et al. (2019) manage to outperform the attention-based encoder-decoder models of See et al. (2017) or Nallapati et al. (2016). Furthermore, they demonstrate a way for dealing with long-document summarization since they train on data from scientific papers.

Continuing the application of GPT-2 style architectures to summarization, Khandelwal et al. (2019) demonstrate how a Transformer language model improves on combining pretraining with attention-complemented LSTM-based sequence-to-sequence structures. When pretrained representations are passed to an encoder and/or a decoder, some parameters remain non-pretrained, specifically the encoder-decoder attention parameters. In contrast, when a Transformer language model is used, these parameters are also pretrained. Khandelwal et al. (2019) outperform approaches with special features such as a copy mechanism or a coverage loss (Gehrmann et al., 2018) on the CNN/DailyMail dataset.

Contrasting GPT-2 and BERT for summarization purposes, Rothe et al. (2019) experiment with using GPT-2 and BERT public checkpoints to warm-start Transformer based sequence-to-sequence models. Specifically, they test all combinations of GPT-2, BERT and random initialization for the encoder and the decoder. Their best model *BERTSHARE* on the CNN/DailyMail dataset consists of a BERT-initialized encoder combined with a BERT-initialized decoder. The weights are shared between encoder and decoder and additionally layer-wise attention mechanisms are implemented. This model outperforms several standard summarization models such as See et al. (2017) and Gehrmann et al. (2018).

Following the overwhelming performance of BERT-initialized sequence-to-sequence models, Liu and Lapata (2019) explore this direction further by creating a new, pretrained BERT-inspired architecture (*BERTSUM*) for the encoder and choosing a randomly initialized 6-layered Transformer as the decoder. *BERTSUM* differs from BERT in its inputs. While in the original BERT model the '<cls>' only appears at the beginning of a sequence, *BERTSUM* inserts it before every sentence of the input. The segment embeddings are also modified to occilate between embedding for segment 'A' and embedding for segment 'B' such that two neighboring sentences have two different segment embeddings. This *BERTSUMABS* architecture is trained on the CNN/DailyMail dataset with separate finetuning schedules for the encoder and decoder. The decoding process uses a beam search of size 5, implements a length penalty and blocks repeated trigrams similar to Edunov et al. (2019). No other copy or coverage mechanism is applied. *BERTSUMABS* surpasses previous abstractive models including See et al. (2017) and Gehrmann et al. (2018). It is only outperformed by *BERTSUMEXTABS*, a model with the same architecture as *BERTSUMABS* which first finetunes the encoder on an extractive summarization task and then

on the abstractive summarization task.

The approach of this thesis is most similar to the one of Khandelwal et al. (2019) in that no sequence-to-sequence set-up is used, but instead the data is directly inputed to a Transformer-based language model. While Khandelwal et al. (2019) use GPT-2, this thesis works with XLNet. A key difference is further that Khandelwal et al. (2019) train their model for the summarization task, while this thesis restrict itself to the evaluation of a pretrained language model for summarization.

3 Methodology

3.1 Data

3.1.1 Description

The dataset used throughout this thesis is one of the standard datasets tested in summarization, namely the CNN/DailyMail Dataset. Hermann et al. (2015) created the dataset and made a script available to download and process the data. For this thesis, the dataset was downloaded from a website[1] that provides the raw stories in order to employ a custom preprocessing routine.

The dataset is comprised of 312.085 newspaper articles. It contains 70% articles from the Dailymail and 30% articles from CNN. Each text consists of the original article and several manually created bullet points. These bullet points are joined together by full stops to form the summary of the article.

For the purpose of this thesis the dataset is split (Liu and Lapata, 2019) into training, development and test sets (90.154/1.218/1.093) for the CNN articles[2] and (196.961/12.148/10.397) for DailyMail articles. Apart from the splitting process no distinction is made between the two data origins. The data split mainly used in this thesis is the development set. On average, an article in the development split consists of 914 tokens of article and 67 tokens of summary. Table 3.1 lists further characteristics of the entire dataset.

dataset	origin	# articles	avg. story length	shortest story	longest story	avg. summary length	shortest summary	longest summary
train	CNN	90.154	871	30	5084	57	10	129
	DM	196.961	916	52	4726	68	5	2829
	both	287.110	902	–	–	64	–	–
dev	CNN	1.218	873	76	2623	58	19	100
	DM	12.148	919	13	3.430	68	9	883
	both	13.366	914	–	–	67	–	–
test	CNN	1.093	870	64	2570	57	13	97
	DM	10.397	912	97	2623	68	8	835
	both	11.490	908	–	–	67	–	–

Table 3.1: Descriptive statistics of dataset.

[1] `https://cs.nyu.edu/~kcho/DMQA/`, last accessed: 18.01.2020.

[2] Standard split according to Liu and Lapata (2019) is 90.266/1.220/1.093 for the CNN articles, but several articles were excluded from the development set due to missing a story part.

Table 3.1 indicates that the story and summary lengths vary between the two data origins. CNN articles are shorter on average compared to articles from the DailyMail. Particularly, summaries of CNN articles are on average 10 tokens shorter in the development dataset. The longest summary in the DailyMail development set is more than 8 times longer than the longest summary in the corresponding CNN set. This is due to the fact that the comparisons are performed on raw story text. The raw version of this specific story has html remains appended to the summary which are counted into the summary length. The difference in distributions between CNN and DailyMail articles is illustrated on the basis of the development set in Figure 3.1 and Figure 3.2 for the stories and summaries, respectively.

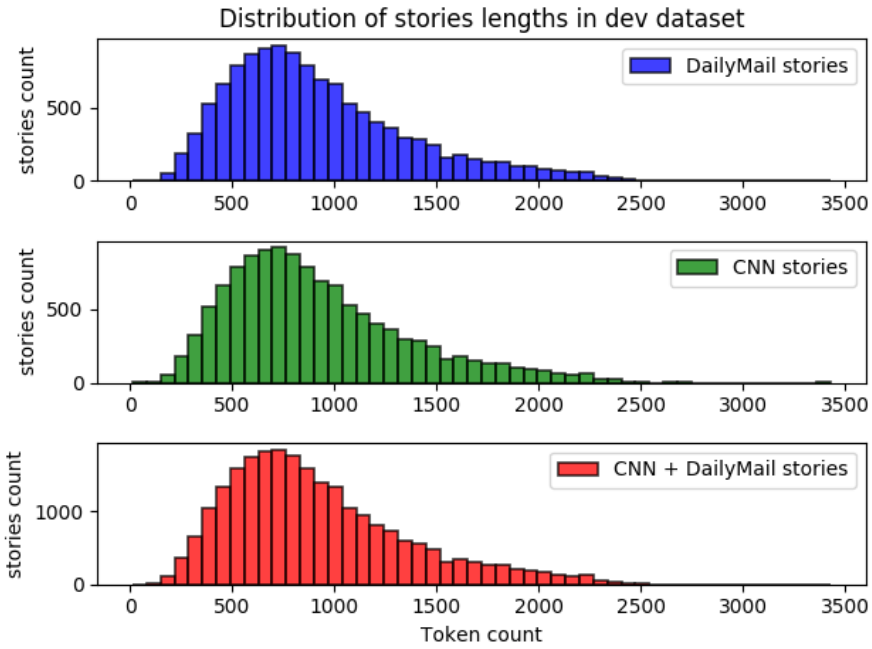

Figure 3.1: Distribution of stories' lengths in development dataset

In theory, the various articles are structured in the same way with the summary consisting of bullet points appended to the main story. Looking at the data in more detail, it becomes obvious that the beginning of the articles varies heavily. Some articles start directly with the story, others have their story preceded by a marker such as '(CNN) –'. Another category of articles provides names of authors and sometimes even time stamps of publication in the beginning of the article. Examples are provided in Table 3.2. The ¶ symbol encodes linebreaks in the raw story text.

3.1.2 Preprocessing

To objectively assess the capabilities of pretrained language models in generating summaries, minimal preprocessing is performed before passing the data to the model. Each

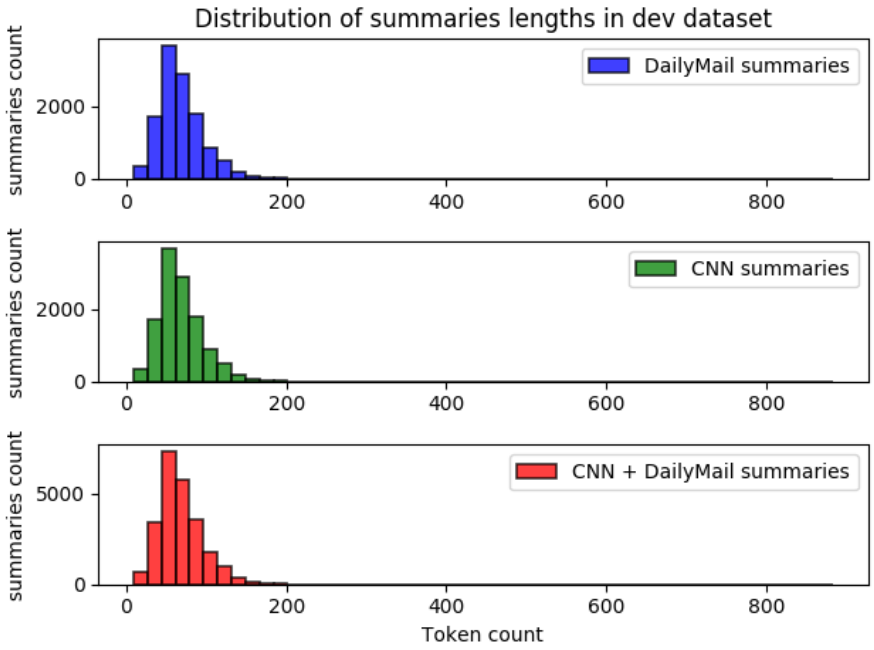

Figure 3.2: Distribution of summaries' lengths in development dataset

Example	Story beginning
1	NEW YORK (CNN) −− Candace Bushnell is a New Yorker and the author of five novels, ...
2	(CNN) −− It's a sad day for Britain's panda lovers. ...
3	By ¶Rob Preece ¶¶PUBLISHED: ¶¶07:39 EST, 23 August 2012 ¶¶| ¶¶UPDATED: ¶¶02:01 EST...
4	By BELINDA BROWN FOR THE DAILY MAIL ¶¶Why is the success of female ...
5	Brendan Rodgers has admitted the fierce rivalry between Liverpool and Chelsea means ...

Table 3.2: Examples of story beginnings in the dataset

document is split into its article and its summary using the keyword '@highlight' which separates the bullet points at the end from the main text. The summary is produced by concatenating these bullet points. For lines missing a punctuation mark at the end, in particular the bullet-point-styled sentences, a full stop is added in the preprocessing step for the text to consist of clearly separable sentences. Note that this modification also transforms the sequence in Example 3 in Table 3.2 to 'By. Rob Preece. PUBLISHED: 07:39 EST, 23 August 2012. ...'.

Stories and summaries are tokenized using a XLNet-specific tokenizer[3]. This tokenizer is based on SentencePiece (Kudo and Richardson, 2018). To see the tokenizer in action, consider the following example taken from a CNN article: the original sentence reads

'At 23 years old, Stockwell was deployed to Iraq in March 2004.'

The tokenizer converts it to the following list:

['_At', '_23', '_years', '_old', ',', '_Stock', 'well', '_was', '_deployed', '_to', '_Iraq', '_in',

[3]see XLNetTokenizer `https://huggingface.co/transformers/model_doc/xlnet.html#xlnettokenizer`, last accessed: 18.01.2020.

'_March', '_2004', '.']

For the summarization task, a summary and a story are concatenated to form the model input and then passed to the `XLNetTokenizer` to be tokenized and encoded in one step. For finetuning, the true summary is prepended to the story sequence. In the evaluation mode, the true summary is not available. Instead a universal summary length is set by a hyperparameter and then a sequence of '<mask>' tokens of length equivalent to the predetermined summary length is concatenated with the story token sequence.

Similar to BERT, XLNet makes use of several special tokens to delimit input sequences for tasks that take multiple segments of text as input. Even in one-segment settings these tokens need to be added to the input. Due to computational limitations, computations are done on input sequences with a length of 1024 tokens. In this case, the encoded concatenated input sequence is truncated to 1022 tokens and the necessary tokens '<sep>' and '<cls>' are then encoded and added to the end of the truncated input.

3.2 Model

3.2.1 XLNet : Objective

XLNet can be superficially classified as a language model that aims to predict the next token given the context of surrounding tokens in the sequence. This is detailed in Section 3.2.2. Mathematically, this can be expressed as maximizing the likelihood of a token x_t given the preceding tokens $x_1, \ldots, x_{t-1}$. The model's objective for the entire sequence $x = [x_1, \ldots, x_T]$ is to maximize the sum of the likelihoods for the individual tokens. This is described by Eq. 3.1

$$max_\theta \quad \log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t|x_1, \ldots, x_{t-1}) \tag{3.1}$$

where T is the sequence length and θ parametrizes the distribution which is assumed by the model (Yang et al., 2019, compare Eq.1, p.2). In the context of neural models, this equation can be reformulated using the hidden vector representation of the context $h_\theta(x_1, \ldots, x_{t-1})$ and the embeddings of the tokens $e(x_i)$:

$$max_\theta \quad \sum_{t=1}^{T} \log p_\theta(x_t|x_1, \ldots, x_{t-1}) = \sum_{t=1}^{T} \log \frac{\exp(h_\theta(x_1, \ldots, x_{t-1})^\top e(x_t))}{\sum_{x_i} \exp(h_\theta(x_1, \ldots, x_{t-1})^\top e(x_i)))} \tag{3.2}$$

$h_\theta(x_1, \ldots, x_{t-1})^\top e(x_t)$ computes a score for each token taking into account the context and its own embedding. To arrive at probabilities, a normalization step is performed.

3.2.2 XLNet : Features

XLNet combines three particularities in one model: it is based on the *TransformerXL* (Dai et al., 2019) structure, implements *two-stream self-attention* and uses the *permutational language modeling* approach.

TransformerXL

The TransformerXL is an extension of the *Transformer* (Vaswani et al., 2017). One of the drawbacks of the traditional Transformer architecture is that it accepts only fixed-length input and due to memory constraints, feeding very long input sequences to the network is not possible. In recurrent neural networks, long sequences could be chunked into smaller pieces. Thanks to the recurrent nature that acts as memory, it is possible to model long-range dependencies in the text. The TransformerXL reintroduces recurrence to the Transformer model at the level of segments. This means that the ouput of the previous sentence is cached and fed to the model together with the next sentence.

This modification, however, interferes with the absolute positional encodings used in the Transformer to model the order of a sequence. XLNet resolves this by suggesting *relative positional embeddings*. These embeddings take the relative distance between words into account by computing the attention score between any two words, i.e. a score is computed for words that are n words before and after the current word.

Two-Stream Self-Attention

While input to the BERT model couples token embeddings, positional encodings and segment encodings together, XLNet uses two seperate representations for each token. This stems from the following disadvantage of BERT: When a token is masked for prediction, its positional and segment encoding are also hidden. XLNet resolves this by splitting the embedding for a token x_t into a *content representation* and a *query representation*.

The content representation is derived from the standard TransformerXL architecture and consists of contextual information and position information for the context tokens $x_1, \ldots, x_{t-1}$ and the selected token x_t. Here, position information relates to the relative positional embeddings mentioned above.

The query representation for a token x_t replaces the '<mask>' token during prediction from BERT by combining contextual information for the tokens $x_1, \ldots, x_{t-1}$ (excluding x_t) and position information for x_t only. Figure 3.3 illustrates this structure.[4]

In a similar fashion as XLNet replaces absolute positional encodings from BERT with relative positional encodings from the TransformerXL model, XLNet modifies the absolute

[4]inspired by `https://towardsdatascience.com/what-is-two-stream-self-attention-in-xlnet-ebfe013a0cf3`, last accessed 16.01.2020.

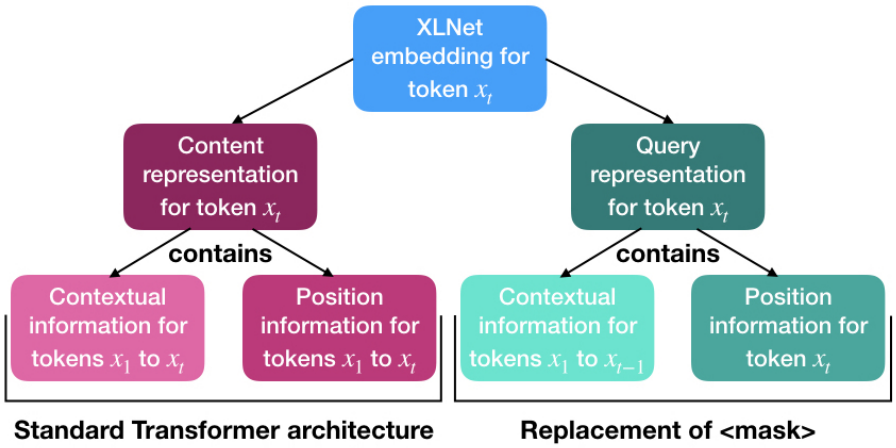

Figure 3.3: Illustration of two-stream self-attention.

segment encodings in BERT. In BERT, these embeddings are introduced to model two input segments for tasks such as question answering where a question and a context paragraph are provided. In BERT, the two parts 'A' and 'B' of such an input are divided by the special '<sep>' token and additionally the segment embedding encodes whether a token belongs to part 'A' or part 'B'. In XLNet on the other hand, the segment embedding is used to capture the information whether two tokens belong to the same segment or not. This modification also makes it possible to extend the model to allow for input consisting of more than two segments.

Permutational Language Modeling

BERT has an advantage compared to auto-regressive models in that it has access to contextual information on both sides of a token. XLNet sticks to the typical auto-regressive approach of conditioning the prediction of a token on previous tokens. By training on various permutations of the sequence, however, it manages to capture bidirectional context. Permutational language modeling implies that for a sequence of length T, all $T!$ many permutations of the sequence order are considered. Formally, let $\mathbb{Z}_T$ be the set of all possible permutations of a sequence of positions with length T. Then z_t denote the t-th element and $z_1, \ldots, z_{t-1}$ denote the preceding $t-1$ elements of a permutation $z \in \mathbb{Z}_T$. To predict a token at position t given the $t-1$ preceding tokens, one factorization order of the sequence is selected at a time. Then the conditional probability of token x_{z_t} given the $t-1$ preceding tokens $x_{z_1}, \ldots, x_{z_{t-1}}$ in the specific factorization order is optimized. Repeating this for other factorization orders leads to the t-th token having seen all other tokens in the sequence in expectation and thus capturing bidirectional context. Thus, the model's objective described by Equation 3.1 in Section 3.2.1 can be modified to (compare Yang et al., 2019, Eq.3, p.3):

$$max_\theta \quad \mathbb{E}_{z\sim\mathbb{Z}_T} = \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t}|x_{z_1},\ldots,x_{z_{t-1}})\right] \tag{3.3}$$

This approach dispenses with masking the token to be predicted and hence avoids the train-test-discrepancy of BERT.

Note that every sequence is given to the model in its original order and the positional encodings are also based on this original sequence order. Only the factorization orders are permuted. Consider the following sentence: "I like cats more than dogs". The original sequence order would be:

I, like, cats, more, than, dogs
$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$

From this sequence 6! permutations can be created. Consider one specific permutation, namely

cats, than, I, more, dogs, like
$3 \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow 2$

Prediction is performed in a sequential manner. To compute the conditional probability, only tokens to the left of the selected token can be considered, i.e. for predicting the token 'than' $\mathbb{P}$('than'|'cats') is computed, for token 'I' $\mathbb{P}$('I'|'cats', 'than') is calculated and so on.

For a visual explanation, consider Figures 3.4 and 3.5 (inspired by Yang et al., 2019, Figure 4 in Appendix). Figure 3.4 demonstrates how the token 'more' (x_4) would be predicted in the original sequence order. Only the left context and the memory from the previous sequence ($mem^{(0)}$ and $mem^{(1)}$), cached thanks to the recurrence on segment level (compare Section 3.2.2), play into the hidden vector representation of 'more'. This is repeated for all the layers. The sequence in the Figure has a length of 6, so there exist 6! many permutations of the sequence. One such permutation would be the order displayed in Figure 3.4, another the one in Figure 3.5. In the latter, the hidden representation of token 'more' is conditioned on the context tokens x_3, x_5 and x_1 to the left of x_4. Note that the factorization order impacts all layers, not only the first.

Introducing the permutational language modeling objective complicates the optimization problem of the model due to the computation of permuations and slows down convergence (Yang et al., 2019). To accelerate the optimization, the authors suggest *partial prediction* of the sequence, i.e. to predict only the last tokens of a factorization order. They split every sequence of a factorization order into a target $z_{c+1},\ldots,z_T$ and a non-target subsequence $z_1,\ldots,z_c$ and reformulate the optimization problem to maximizing the likelihood of the target subsequence given the non-target subsequence. A hyperparameter controls how many tokens are selected for prediction in a factorization order. The objective can be mathematically expressed by adapting Eq. 3.3 to

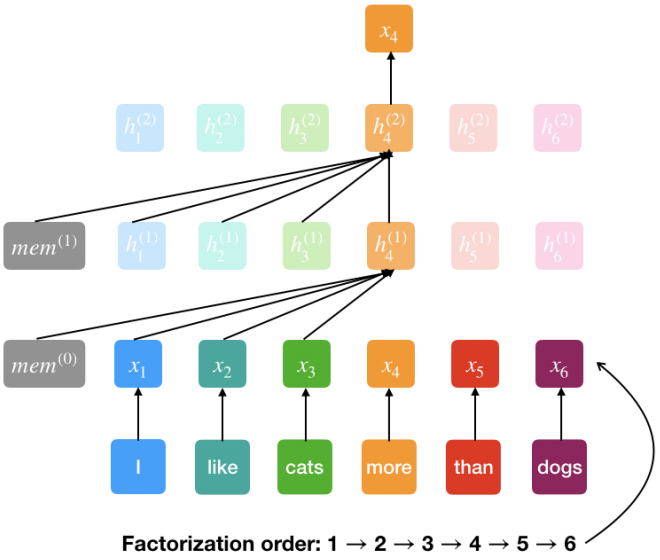

Figure 3.4: Illustration of permutational language modeling for prediction of x_4 given input sequence $x_1x_2x_3x_4x_5x_6$ and factorization order $x_1x_2x_3x_4x_5x_6$.

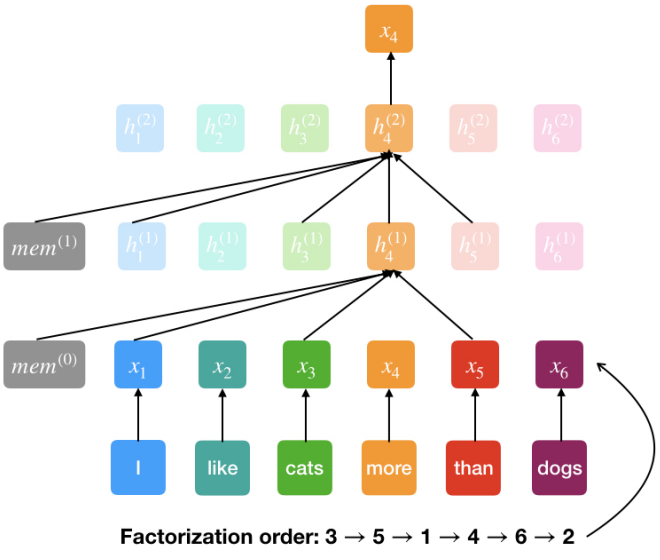

Figure 3.5: Illustration of permutational language modeling for prediction of x_4 given input sequence $x_1x_2x_3x_4x_5x_6$ and factorization order $x_3x_5x_1x_4x_6x_2$.

$$max_\theta \quad \mathbb{E}_{z\sim\mathbb{Z}_T}\left[\log p_\theta(x_{z_{c+1},\ldots,z_T}|x_{z_1,\ldots,z_c})\right] = \mathbb{E}_{z\sim\mathbb{Z}_T}\left[\sum_{t=c+1}^{T}\log p_\theta(x_{z_t}|x_{z_1},\ldots,x_{z_{t-1}})\right] \quad (3.4)$$

3.2.3 XLNet : Training

XLNet has been trained on a corpus totaling 32.89 billion subword pieces from different domains, including the English Wikipedia, e-books, newswire text data and English web pages.

The XLNet model has been trained in two configurations, XLNet-Large and XLNet-Base, and both have been made available by Yang et al. (2019).[5] The former uses the same architecture hyperparameters as BERT-Large consisting of 24 Transformer layers with 1024 hidden units and 16 attention heads. The latter consists of half the layers (12) with 768 hidden units and 12 attention heads. The sequence length is set to 512.

3.2.4 XLNet : Application to Unsupervised Summarization

Due to computational constraints, the smaller model XLNet-Base is used in this thesis. Since the code of this thesis is written using the Python package `PyTorch`, computations were performed loading a PyTorch wrapper of the XLNet model. The library `pytorch_transformers` created by the platform *HuggingFace* has made several Transformer models with extensions available in `PyTorch`, among others `XLNetModel` for XLNet.

This thesis employs the extension `XLNetLMHeadModel`[6] which adds a linear layer projecting the XLNet output to the 32000-dimensional space of the XLNet vocabulary. Thus, the output of the extended model after normalization is a probability distribution over the tokens of the vocabulary for each token in the sequence. This is exactly the structure needed for language modeling.

Unsupervised summarization implies that the model is not finetuned on the data described in Section 3.1. Instead, the pretrained model is evaluated on the specified data. As described in Section 3.1.2, in evaluation mode the input consists of a sequence of '<mask>' tokens, whose length is determined by a hyperparameter, and a sequence of story tokens. Both are concatenated, encoded by the tokenizer and truncated to a maximum sequence length. Two special tokens are lastly appended to the sequence. This sequence is passed to the `input_ids` argument of the model.

The permutational language modeling approach is not only relevant in the pretraining phase, but offers new possibilities when the model is used for evaluation or finetuning. As detailed in Section 3.2.2, the model is trained on various permutations of the order

[5] `https://github.com/zihangdai/xlnet`, last accessed: 18.01.2020.
[6] `https://huggingface.co/transformers/model_doc/xlnet.html#xlnetlmheadmodel`, last accessed: 18.01.2020.

of a sequence to capture bidirectional context. The particularity of XLNet consists in making it possible to specify the factorization order that is supposed to be used during evaluation. Specifically, this can be encoded in the `perm_mask` variable. Interpreting the summarization task as a language modeling task means that each token of the summary is generated conditioned on previous tokens of the summary, so the right context of the token up until the start of the main story is not visible during prediction. The model, however, has access to the main story text. This set-up is translated into a matrix with 0's for visible tokens and 1's for masked tokens. For an input of length n and summary length of m, $m < n$, the matrix is a $n \times n$ matrix of the following form:

$$\texttt{perm\_mask} = \left(\begin{array}{cccc|c} 1 & 1 & \cdots & 1 & 0 \cdots 0 \\ 0 & 1 & 1 \cdots & 1 & 0 \cdots 0 \\ 0 & 0 & 1 \cdots & 1 & 0 \cdots 0 \\ \vdots & \vdots & \ddots & 1 & 0 \cdots 0 \\ 0 & 0 & 0 \cdots & 1 & 0 \cdots 0 \\ 0 & \ddots & 0 \ddots & 0 & 0 \cdots 0 \\ 0 & 0 & 0 \cdots & \cdots 0 & 0 \cdots 0 \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c} a \\ a \\ a \\ a \\ a \\ a \\ a \end{array}}\right\} m \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \\ a \\ a \\ a \\ a \end{array}}\right\} n - m \end{array}$$

$\underbrace{\qquad\qquad}_{m} \quad \underbrace{\qquad}_{n-m}$

To evaluate the model, one output token is predicted at a time as in a classical auto-regressive model. The prediction is then passed to the input sequence and replaces the '<mask>' token at the predicted position in the input sequence. Thus, the next token is predicted conditioned on the prediction of the previous token. Which output token is predicted at a time is represented in the `target_mapping` argument. This variable is a one-hot-vector of size $1 \times n$ for a sequence of n tokens. It describes the position to be predicted.

Lastly, in order to pass batches of articles to the model instead of individual articles, padding has to be applied to the sequences. An `attention_mask` distinguishes pad from real tokens. A value of 0 represents a pad and value of 1 represents a real token.

3.3 Settings

3.3.1 Basic Unsupervised Prediction

To assess the capability of XLNet in summarizing text, the `input_ids`, `attention_mask`, `perm_mask` and `target_mapping` as detailed in Section 3.2.4 are passed in batches of 8 articles with a sequence length of 1024 tokens to the XLNet-Base model. The model is set to its evaluation mode.

The summary length is set to 67 tokens since this has been determined as an average summary length in the development and test dataset (compare Table 3.1). To generate a sequence of summary length, each token is generated in a separate loop. The output of the

model is a set of vectors of prediction scores with a length equivalent to the vocabulary size (default=32000), one vector for each article, but for each article in the batch the same position is predicted. Given the prediction scores $s \in \mathbb{R}^d$, where d is the size of the vocabulary, the probability of predicting the token x_i can be expressed as (inspired by Keskar et al., 2019, Eq. 1, p.4)

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}. \tag{3.5}$$

This is equivalent to normalizing the logits output by the model using the softmax function. For the purpose of this thesis, the token with the maximum probability is selected as the only candidate. Applying a beam search algorithm could be a possible extension. Since applying the softmax function only normalizes the prediction scores, the maximum function can also be directly applied on the prediction scores.

In this setting, further referred to as XLNetSUM, the index with the maximum prediction score is selected as the most likely prediction. This index replaces the "<mask>" token's encoding at that position in the input sequence. While `input_ids`, `attention_mask`, `perm_mask` arguments stay the same for all tokens of the summary sequence, the `target_mapping` has to be computed from scratch for every next token since the position of the 1 shifts to the right in each loop.

3.3.2 Unsupervised Prediction with Repetition Penalty

In this setting, further referred to as XLNetSUMPen, a rescoring of the prediction scores is performed before the maximum is identified. This stems from the observation that the model tends to repeat itself. Hence, a repetition penalty following Keskar et al. (2019) is introduced as another hyperparameter. Eq. 3.5 is modified to (inspired by Keskar et al., 2019, Eq. 2, p.5)

$$\begin{gathered} p_i = \frac{\exp(A_i)}{\sum_j \exp(A_j)} \\ \text{with} \\ A_k = \begin{cases} \exp(s_k \cdot I(x_k \in g)) & s_k > 0 \\ \exp(s_k / I(x_k \in g)) & s_k \leq 0 \end{cases} \\ I(c) = \begin{cases} \lambda & \text{if } c \text{ is True} \\ 1 & \text{else} \end{cases} \end{gathered} \tag{3.6}$$

where g is the sequence of generated tokens and λ represents the penalty. The indicator function is used to only discount scores of tokens from the vocabulary that have already been generated in the sequence g. All other scores are not modified. When applied to unnormalized prediction scores, Eq. 3.6 translates to the following code snippet:[7]

[7] slightly adapted from `https://github.com/huggingface/transformers/blob/23c6998bf46e43092fc59543ea7795074a720f08/src/transformers/modeling_utils.py#L883`, last

```
if args.repetition_penalty != 1.0:
   for i in range(input_ids.shape[0]):
      # loop through previously generated tokens
      # generated tokens replace <mask> token in input
      for previous_tokens in set(input_ids[i].tolist()[:predict_pos]):
         # next_token_logits has dimensions
         # (batch_size, num_predict, vocab_size)
         # num_predict is the number of tokens to be predicted
         # num_predict=1 for evaluation
         if next_token_logits[i, 0, previous_tokens] < 0:
            next_token_logits[i, 0, previous_tokens] *= args.repetition_penalty
         else:
            next_token_logits[i, 0, previous_tokens] /= args.repetition_penalty
```

Positive prediction scores have to be divided by the penalty factor to be discounted, while negative prediction scores have to be multiplied by the factor to be reduced in relative value. Note that Equation 3.6 differs from the equation described in Keskar et al. (2019). The latter lacks a case distinction depending on the sign of the logit scores. The adaptation is performed so that negative scores also get adequately discounted. The penalty value is set to 1.2, as is suggested by Keskar et al. (2019).

As an alternative, another penalty setting has been tested on the development data set. Instead of dividing the prediction score by the penalty which requires distinguishing between positive and negative scores, the idea is to subtract a fixed value from all scores of tokens which have already been generated. This is motivated by standard weight regularization in neural networks, for example using the Lasso penalty. Mathematically, this translates to

$$
\begin{gathered}
p_i = \frac{\exp(s_i - I(x_i \in g))}{\sum_j \exp(s_j - I(j \in g)} \\
\text{with} \\
I(c) = \begin{cases} \theta & \text{if } c \text{ is True} \\ 0 & \text{else} \end{cases}
\end{gathered}
\tag{3.7}
$$

The value of the penalty tested for this setting is set to 0.2, so the difference between the setting with and without penalty is the same for both penalty schemes. In the first penalty scheme, a penalty value of 1.0 would translate to no penalization, while a value of 0.0 achieves this for the second scheme. Results for the alternative are discussed in Section 4.4.

accessed: 18.01.2020.

4 Results

For evaluation, 10 GPUs with 11GB RAM each were available. To parallelize the process, development and test data have been split into 10 chunks, each containing 1336 and 1149 articles respectively. Evaluation of each development and each test chunk took 8,9 hours 7,7 hours, respectively.

4.1 Quantitative Analysis

Each of the two settings described in Section 3.3 are first evaluated on the development set, discussed in Section 3.1.1. For the quantitative analysis, scores according to the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric (Lin, 2004) are computed. ROUGE has been frequently used for evaluation of automatic summarization frameworks (see Section 2.3). A comparison with other papers is provided in 4.4. ROUGE-1, ROUGE-2 and ROUGE-L are computed using the Python library `rouge-score`[1].

ROUGE-1 and ROUGE-2 represent the overlap of unigrams and bigrams, respectively, of the candidare summary and the reference summary. While ROUGE-1 (ROUGE-2) recall measures the percent of unigrams (bigrams) of the reference summary which are also present in the generated summary, ROUGE-1 (ROUGE-2) precision is equivalent to the percent of unigrams (bigrams) in the generated summary which are also present in the reference summary. Here, the F-score, defined as the harmonic mean of recall and precision, is reported to assess the quality of the summaries.

ROUGE-L differs from ROUGE-1 and ROUGE-2 in that it is based on the longest common subsequence that matches in reference and generated summary. Note that it does not require consecutive matches but in-sequence matches that reflect sentence level word order. It automatically includes longest in-sequence common n-grams, so no n-gram length needs to be specified. As for the other ROUGE metrics, the F-score is reported.

Table 4.1 shows ROUGE scores for the baseline lead67 and the two settings, XLNetSUM and XLNetSUMPen. The baseline represents the comparison of the first 67 words of the article to the reference summary.

Firstly, it can be observed that neither setting is able to surpass the baseline. Instead the scores of the baseline are 2 and 1.5 times higher than the scores of the setting without and

[1] https://github.com/google-research/google-research/tree/ddc22300c4cb3223654c9a981f892dc0f6286e35/rouge, last accessed 18.01.2020.

setting	ROUGE-1	ROUGE-2	ROUGE-L
lead67	32.55	12.38	20.43
XLNetSUM	15.87	3.59	12.15
XLNetSUMPen	20.96	5.02	14.07

Table 4.1: ROUGE F-scores for baseline, XLNet evaluated without and with the repetition penalty on the development dataset

with the penalty respectively. The strong performance of the baseline is not too suprising, since most newspaper articles state the key facts of the story in the beginning of the text. It is plausible that the article beginning and the reference summary share about 1/3 of unigrams or 22 unigrams. Consider the following story beginning

> **Story 1.** Former F1 driver Michael Schumacher is no longer in a coma and has been transferred from a hospital in Grenoble, France, ...

The first bullet-point or rather the first sentence of the summary reads

> **Summary 1.** Michael Schumacher transferred to a hospital in Switzerland for rehab....

The two XLNet settings perform weakly compared to the baseline. Interestingly, adding the repetition penalty does boost performance. For the ROUGE-1 score, the boost is equivalent to a 32% increase from the setting without to the one with penalty.

Moving from ROUGE-1 to ROUGE-2 scores, a large drop in performance is detected. Bigram overlap is a more strict constraint to the summarization process since it requires two words to appear right next each other to be counted as a matching bigram.

While ROUGE-1 and ROUGE-2 are seen as measures of informativeness, ROUGE-L can be interpreted as the level of fluency Liu and Lapata (2019). Its score is again much higher for the baseline than the XLNet involved settings. The ROUGE-L scores point, however, into a similar direction as the unigram score. This suggests that a portion of the words that appear in both, generated and reference summary, also appear in a similar order, albeit there might be gaps between the words.

Table 4.2 presents results computed on the test set, generated for this thesis. Note that the articles evaluated in the test data set of Liu and Lapata (2019) are not necessarily the same as in this thesis' test set because of a different preprocessing procedure. During data loading for this thesis, the articles have, however, been shuffled such that on average the two sets which are of same size can be regarded as comparable.

The results in Table 4.2 exhibit very similar characteristics as the results for the development set. It can be concluded that the data shuffling during preprocessing has lead to a similar text distribution for both, development and test set.

setting	ROUGE-1	ROUGE-2	ROUGE-L
lead67	32.88	12.56	20.60
XLNetSUM	16.09	3.74	12.29
XLNetSUMPen	21.16	5.14	14.17

Table 4.2: ROUGE F-scores for baseline, XLNet evaluated without and with the repetition penalty on the test dataset

Model	ROUGE-1	ROUGE-2	ROUGE-L
GPT-2 (zero-shot) Radford et al. (2019)	29.34	8.27	26.58
Transformer LM (No pretraining) Khandelwal et al. (2019)	37.72	16.12	34.62
Transformer LM (With pretraining) Khandelwal et al. (2019)	39.65	17.74	36.85
BERTSHARE Rothe et al. (2019)	39.25	18.09	36.45
BERTSUMABS Liu and Lapata (2019)	41.72	19.39	38.76

Table 4.3: ROUGE F-scores for previous models using CNN/DailyMail dataset

4.2 Comparison with Other Models

The CNN/DailyMail dataset is a standard dataset used for evaluation of summarization systems. As detailed in Section 2.3, pretrained language models have been successfully used for abstractive summarization. Table 4.3 displays some of the results of previous papers. The set-up most similar to the one in this paper is displayed in the first row. Radford et al. (2019) evaluate their GPT-2 language model for summarization by appending the hint 'TL;DR:' to the end of the article, generating 100 tokens with beam size 2 and using the first 3 sentences as their summary. This results in a ROUGE-1 F-score of 29.34 which is close to the baseline of this thesis for XLNet at 32.88. Note however that the results of Radford et al. (2019) are obtained on the entire CNN/DailyMail dataset while the results of this thesis consider subsets. Comparing the results from development and test set (compare Table 4.1 and 4.2), it can be extrapolated, however, that the results would also hold for the entire data set. While GPT-2 lends itself well to summarization, XLNet seems to need more sophisticated tweaking when comparing the ROUGE-1 F-score for GPT-2 and XLNetSUMPen. It could be of an interest to explore the effect of a hint on XLNet's prediction.

Table 4.3 also clearly demonstrates that fine-tuning a Transformer(-based) model on the dataset leads to a strong improvement in ROUGE scores, even without pretraining as is visible from the ROUGE-1 score at 37.72 for the non pretrained model of Khandelwal

et al. (2019). Using pretrained embeddings then boosts the score only by 2 percentage points the most.

The last two rows of Table 4.3 make a case for returning to the encoder-decoder set-up, but building each block from Transformer layers. Recall that BERTSHARE consists of BERT-initialized encoder and decoder sharing learned representation and BERTSUMABS contains a BERT-initialized encoder and randomly initialized 6-layered Transformer decoder. Unfortunately, neither Rothe et al. (2019) nor Liu and Lapata (2019) test how BERT without the build up performs on unsupervised summarization. The scores presented in Table 4.3 for Liu and Lapata (2019) are however obtained on a same-sized data set of CNN/DailyMail data as XLNetSUM.[2] The results of Table 4.3 clearly indicate that a boost in ROUGE scores of the magnitude of factor 2 could be expected if XLNetSUMPen was to be finetuned.

4.3 Qualitative Analysis

For the qualitative analysis a sample of 20 summaries, 10 from the development and 10 from the test set were evaluated. Skimming through these summaries, it is noticeable that every summary, generated with or without penalty, starts with a full stop. The full stop is followed by space and then the first sentence starts with the first word having a capital letter. The generation of the full stop could be linked to the preprocessing step in which every line in the raw story text which does not end with a punctuation mark gets a full stop added at the end. Recall this also happens for articles which start by stating the author and time stamp, such as Example 3 in Table 3.2. If the model assigns a higher weight to tokens at the beginning of the article, then this might explain why a full stop at position 2 of the original article is produced right at the beginning of the summary.

Another explanation could be derived from an observation made by Aman Russia who tested XLNet for language generation from short prompts.[3] He significantly improved XLNet's performance by passing a padding text to the model together with the prompt to generate the output token one by one. This approach has also been incorporated into example code of *HuggingFace*.[4] The set-up of this thesis clearly differs from the generation of text from short prompts. In this thesis, the text of the article is not masked during generation thanks to the `perm_mask`,but should instead replace the need for a padding text. Following the approach using a padding text, it could thus be useful to use a couple of words from the article as a prompt. The influence of such a prompt is discussed in Section 4.4.

Examining the sample of 20 articles, several observations can be made. The model manages

[2]Since this thesis does not use the same preprocessing procedure, the articles in the two sets might differ. Considering successful shuffling, it can though be assumed that the resulting sets contain similar articles.
[3]`https://github.com/rusiaaman/XLNet-gen`, last accessed: 18.01.2020.
[4]`https://github.com/huggingface/transformers/blob/master/examples/run_generation.py#L59`, last accessed: 18.01.2020.

to select the correct named entities and key words, but it fails to correctly represent the relation between these entities. For example, for a story about a handcuffed teenager who saves a police officer while the latter is having a heart attack, the summary reads

> **XLNetSUMPen 1.** . The **police officer was arrested in jail** for criminal mischief and burglary after a heart attack. The arrest warrant was issued to the officer. The arrest warrant was issued to the officer who was being booked into jail. The arrest warrant was issued to the officer who was being booked into jail. Jailhouse: Jamal Rutledge:

Evidently, the teenager has been arrested for criminal mischief and burglary, not the police officer. In the worst case, only key words are retained, but the summary misses the main point of the story.

The model performs well in terms of syntactic structure. This can, however, lead to phrases without semantic meaning, such as

> **XLNetSUM 1.** , the prosecutor's office said. The **death** of a man's grandfather **was reported to be in the family's custody**, according to a news release from the Michigan State Police. (MORE) nn (MORE) (MORE) (MORE) MILES

The summaries' generation is clearly influenced by the fact that the language model has been trained on an immense dataset including text from various domains. This can be illustrated by the observation that the summary often includes words that are related to the domain of the text, but do not appear in the original article, such as the word 'arrested' in XLNetSUMPen 1. Another text quotes Osama bin Laden in a story about an ISIS killing, clearly dated after Osama bin Laden's death. Also, words which appear in the text can be wrongly misinterpreted as part of an idiom, such as in this case where the original story reads

> **Story 2.** Until a week ago Cardinal O'Brien, 74, had been preparing to help choose the next Pope. But last night he admitted his 'sexual conduct' had 'fallen below the standards expected of me as a priest, archbishop and cardinal'. **He said sorry** and added that he was retiring from public life.

The summary generated with penalty factor, however, contains the common colocation 'I am sorry for the inconvience'

> **XLNetSUMPen 2.** . The Pope's statement: **'I am sorry for the inconvenience** caused by this news.' The Vatican has no further responsibility for the situation in Scotland. Apology: Keith O'Brien's resignation is complete last night as Cardinal Keith O'Brien's sexual misconduct was revealed. Apology: The

Evidently, the word 'sorry' appears in a completely different context in the story than in the generated summary.

Regarding the effect of the penalty, it can be said that the penalty does improve the fluency of the text. Word-by-word repetition occurs more seldomly on segment level, synonymous

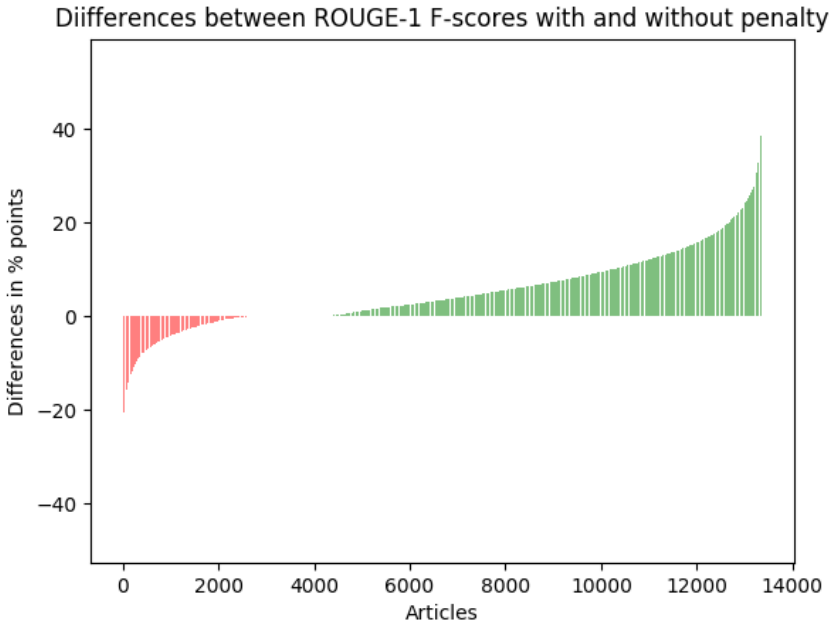

Figure 4.1: Distribution of differences in ROUGE-1 F-scores between summaries generated with and without the penalty factor.

structures can however be formed instead. Consider the following summary, generated without the penalty

> **XLNetSUM 2.** . "The "capsule" is a very popular hotel in Tokyo." "The "capsule" is a very popular hotel in Tokyo." "The "capsule" is a very popular hotel in Tokyo." "capsule"

and the following summary, generated with the penalty factor

> **XLNetSUMPen 3.** . "The Hotel of Tokyo" (CNN) – The hotel of Tokyo's most popular and well-known hotel is the Peninsula Hotel. "The hotel of Tokyo's most popular hotel, the hotel of Tokyo's most famous hotel, the hotel of Tokyo's best hotel

The phrase 'very popular' in the first summary is replaced by 'most popular' or 'well-known' or 'most famous' in the second summary.

To better grasp how the penalty factor effects the summary, the differences in ROUGE-1 F-scores for the texts, generated with and without the penalty, have been calculated for each article. Figure 4.1 displays the distributions of these differences.

Negative values, marked red, indicate that adding the penalty harmed the summary's score while positive values, colored green, point to a useful inclusion of the factor. It is noticeable that for more than half of the articles the addition of the penalty factor boosts the score. This observation is also in line with the overall results displayed in Table 4.1. Recall that overall performance of the model with the repetition penalty exceeds the setting without the factor.

Table 4.4 and 4.5 show the three cases on the right (most benefitted from penalty) and the three cases on the left extreme (most harmed by penalty) of Figure 4.1. For each example, its reference summary, lead67 and the two generated summaries are displayed.

For the examples in Table 4.4 the ROUGE-1 score improved by 51-54 percentage points. This large improvement becomes apparent when comparing the generated summaries with and without the penalty. Without the penalty, the summaries consist of repetitions of the same token 'Six' or 'The' or an entire phrase 'It is a new series of movies'. With the penalty, sentences are formed. Interestingly, ignoring the punctuation and quotation mark, the two summaries split into different versions starting from the second or third generated word. The penalty seems to enable the model to leave the loop of generating the same token over and over again. Furthermore, it is noticeable that apart from the last example, the first two examples' reference summaries and story beginnings are very close to each other. The best summary has an 62% overlap with the reference summary. This is, however, not always the case as Example 3 shows. Examining the generated summaries with penalty factor, similar observations as in the sample analysis appear. The model seems to grasp main components of the story, such as the act of 'pirating ten gay porn films online' in Example 3. However, this act is attributed to the wrong person or organisation. In Example 3, 'The World of Poker' is the subject while it should be 'Kywan Fisher'. Example 2's enerated summary with penalty would be an optimal summary if it was not for the wrong subject, 'The American Family'.

The ROUGE score difference between the summaries generated without and with the penalty in Table 4.5 ranges from 40 to 47 percentage points. This difference is slightly smaller in absolute values compared to the benefit computed for the examples in Table 4.4. This is comprehensible when looking at the summaries generated with the penalty in Table 4.5. They clearly are repetitive for the Examples 1 and 2, but they contain more variety of words than the summaries without penalty scoring in Table 4.4. Surprisingly, the best example of Table 4.4 shows a generated summary without penalty close to the generated summary with penalty in Example 1 of Table 4.5. This has to be a coincidence because nothing has been changed in the set-up which might explain this reversal of behaviour. It should be pointed out that the model clearly incorporates much information from the story beginning, as evident from the string 'UPDATED' only present at the beginning of the article in Example 3 in Table 4.5. Furthermore, the critique of incorrect association of subject and act raised for examples in Table 4.4 could be revoked by the summary of Example 2. Here, subject and object are correctly identified.

Tables 4.6 and 4.7 show the three cases with the highest and the lowest difference between the summary generated with penalty and the baseline. In Table 4.6 this positive difference ranges from 39 to 50 percentage points for the top articles. It can be concluded from Table 4.6 that the more an article's beginning differs from the reference summary, the more likely there is a large overlap between the generated and the reference summary. Considering Examples 2, this tendency even holds independent of whether the summary has been generated with or without the penalty since the two are almost identical.

Exactly the reverse picture unveils for the reverse case, shown in Table 4.7. Here, the three examples have identical article beginnings and reference summaries with the only exception as to the length. The model is not able to attain this. Instead, the summaries exhibit similar issues as in the previous tables.

4.4 Experiments

As described in Section 3.3.2 besides a penalty factor inspired by Keskar et al. (2019), an alternative penalty scheme has been tested according to Eq. 3.7. The difference between the two schemes consists in how it scales the scores. For the multiplicative penalty (XLNetSUMPen), the reduction in a score for a particular token depends on the size of the score before penalization. The larger the score is, the larger is its reduction in the multiplicative case. For the additive penalty scheme, further referred to as XLNetSUMPen_Alt, each score is reduced by the same amount, independent of its size. An interesting edge case arises when considering a logit score of 0. In the multiplicative case, the penalty would have no effect. In the additive case, the score would change as a result of applying the penalty.

Table 4.8 displays the results on the development set for both penalties in parallel. The setting without penalty is provided as point of comparison. Clearly, the multiplicative penalty trumps the additive penalty. The additive penalty still impacts the results compared to the setting without any penalty. Its effect is, however, significantly smaller than for the multiplicative penalty. The larger success of the multiplicative penalty could be explained with the following numerical example. Consider two tokens with scores $s_1 = 1.8$ and $s_2 = 1.55$ and let these two scores be the top scores in the distribution. Let x_1 be a repetitive token. Then applying the multiplicative penalty will result in a reduced score $s_1^{\text{Pen}} = 1.5$, while the additive penalty will shrink the score to $s_1^{\text{Alt}} = 1.6$. Thus, in the multiplicative case, token x_2 would be selected as the prediction. In the additive case, however, the repetitive token x_1 would again be chosen as the candidate.

Furthermore, the inclusion of a prompt for generation has been explored on the development set. It is tested on the setting with a multiplicative penalty factor. For prompting, the first 7 tokens of the story, identified after encoding, are used. This is equivalent to about 10% of the generated summary. Results for this setting, further referred to as XLNetSUMPen_Prompt, are displayed in Table 4.9. Surprisingly, adding the prompt lowered the ROUGE-1 and ROUGE-L scores slightly. At the same time, adding the prompt has led to an increase in bigram overlap as measured by ROUGE-2. At first glance, there does not appear a clear explanation for this behavior. This should be further explored by repeating the setting with different number of prompt tokens.

By skimming through the generated summaries, it is however clearly noticeable that the constant generation of the full stop at the beginning of the summary, criticized in Section 4.3, has been impacted by the prompt. Only a portion of summaries has a full stop

between the prompt and the generated section.

Furthermore, repetition at segment level seems to have been delayed by the prompt. Random samples show that the prompt helps the model to spin the summary further at the beginning. The repetition then starts again towards the end. For an example, consider these two summaries:

> **XLNetSUMPen 4.** . The Guardian reported that a South Korean woman got a rude awakening when she left her robot vacuum to do the cleaning while she took a nap. Scroll down for video. The Guardian reported that a South Korean woman got a rude awakening when she left her robot vacuum to do the cleaning while she took a nap.

> **XLNetSUMPen_Prompt 1.** A South Korean woman got a rudeawakening when she left her robot vacuum to do the cleaning while she took a nap. Scroll down for video. The woman's hair was not clean and had been sucked up by the vacuum cleaner. The woman was later rescued by the fire department. Scroll down for video. The woman's hair was not clean

These observations have been made on a small sample of the data. A more detailed statistical analysis is needed to adequately assess the usefulness of a prompt. These observations only make tendencies apparent.

Nr	setting	**best cases**
1	reference	A 13-year-old boy fell from slippery rocks to into a creek at Wildwood Falls and his 16-year-old brother died saving him. The 16-year-old boy lowered himself to the rocks to try to hold onto. his brother who was nearly washed away by the rapid moving water. Both brothers fell down a narrow chute of water and only the 13-year-old emerged alive.
	lead67	A 16-year-old boy died trying to save his 13-year-old brother after he fell from slippery rocks into a creek on Tuesday. The 13-year old boy whose identity has not been released fell into the water at Lasells Stewart Picnic Area in Wildwood Falls on the Row River in Oregon. The 16
	XLNetSUMPen	. The "Six-Year-Old Boy died trying to save his 13-year-old brother after he fell from slippery rocks into a creek on Tuesday. The 16-year-old boy died trying to save his 13-year-old brother after he fell from slippery rocks into a creek on Tuesday. A
	XLNetSUM	. The "Six" "Six" "Six" "Six" "Six" "Six" "Six" "Six" "Six" "Six" "Six" "Six" "
2	reference	Gate guards at an Air Force base in central Georgia have been ordered to stop telling visitors to 'have a blessed day' A man who identified himself as an active duty military member posted on the Military Religious Freedom Foundation website. He said that Robins Air Force Base personnel have told him to have a blessed day more than a dozen times in the past two weeks.
	lead67	Gate guards at an Air Force base in central Georgia have been ordered to stop telling visitors to 'have a blessed day.' A man who identified himself as an active duty military member posted on the Military Religious Freedom Foundation website that Robins personnel have told him to have a blessed day more than a dozen times in the past two weeks
	XLNetSUMPen	. The "The American Family" website posted a message to the military religious freedom foundation website that the military has been ordered to stop telling visitors to have a blessed day.' The statement was made on the website by a man who identified himself as an active duty military member andemployee. Changes: Change:
	XLNetSUM	. The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The "The
3	reference	Kywan Fisher must pay damages of $150,000 per film to adult entertainment company Flava Works.
	lead67	By. Graham Smith. PUBLISHED:. 08:15 EST, 2 November 2012. —. UPDATED:. 08:17 EST, 2 November 2012. An illegal file-sharer has been fined a record $1.5million for pirating ten gay porn films online. Ky
	XLNetSUMPen	. "The World of Poker has been awarded a record $1.5million for illegally sharing ten gay porn films online." —. UPDATED:. 08:17 EST, 2 November 2012. Fine: Kywan Fisher must pay damages of $150,000 per film to adult entertainment company Flava Works
	XLNetSUM	. "The "The "" is a new movie. It is a new series of movies. It is a new series of movies. It is a new series of movies. It is a new series of movies. It is a new series of movies. It is a new series of movies."

Table 4.4: Top 3 examples of articles whose summary benefits the most from the penalty factor.

Nr.	setting	**worst cases**
1	reference	Pennsylvania boy pretended to be woman named Jessica Carabello and promised nude photos in return. Threatened to post the boys' images online if they did not send more. He also sent messages to 100 additional boys who did not respond. Was arrested and charged with theft by extortion, solicitation and transmission of sexually explicit images.
	lead67	A 14-year-old Pennsylvania boy was arrested after he posed as a woman on Facebook and tricked 48 boys into sending him nude photos and gift cards. The teen created a profile under the name of Jessica Carabello and promised to provide naked photographs of 'her' if they sent images of themselves in return. The unidentified
	XLNetSUMPen	. The "Six"–The "Six"–The "Six"–The "Six"–The "Six"–The "Six"–The "Six"–The "Six"–
	XLNetSUM	. The "Six" (pictured) was arrested after he posed as a woman on Facebook and tricked 48 boys into sending him nude photos and gift cards. The teen was charged with theft by extortion, solicitation and transmission of sexually explicit images. In 2013, a 14-year-old Pennsylvania boy
2	reference	Tony Sinclair won best groundsman trophy for his work at Old Trafford. Manchester United have a 30-strong team to tend to their pitches. Sir Alex Ferguson presented the award before the game against Newcastle.
	lead67	Sir Alex Ferguson was on the pitch at Old Trafford ahead of Manchester United's match against Newcastle to present groundsman Tony Sinclair with an award for his fine work. Sinclair won the best groundsman trophy and was presented the accolade by the club's legendary former manager. The Red Devils have a 30-strong team to ensure their pitch
	XLNetSUMPen	the United's academy side. The United Football Grounds Team of the Year The United Football Grounds Team of the Year The United Football Grounds Team of the Year The United Football Grounds Team of the Year The United Football Grounds Team of the Year The United Football Ground of the Year
	XLNetSUM	the United's academy side. United's academy side is on the pitch at Old Trafford ahead of Manchester United's match against Newcastle. Sir Alex Ferguson was presenting groundsman Tony Sinclair with an award for his fine work ahead of their match. Sir Alex Ferguson was presenting groundsman Tony Sinclair with an award.
3	reference	Low-cut 'Forever' dress by London-based label Issa is a popular choice in the Middleton family.
	lead67	Low-cut 'Forever' dress by London-based label Issa is a popular choice in the Middleton family. By. Lara Gould and Tom Goodenough. PUBLISHED:. 08:31 EST, 20 May 2012. —. UPDATED:. 08:39
	XLNetSUMPen	. "The Duchess of Cambridge's wedding was a very special occasion for her sister Pippa Middleton." —. UPDATED:. 08:29 EST, 20 May 2012. —. UPDATED:. 08:30 EST, 20 May 2012. —. The classic
	XLNetSUM	. "The "Forever" dress by Issa is a popular choice in the Middleton family. —. UPDATED:. 08:29 EST, 20 May 2012. —. UPDATED:. 08:29 EST, 20 May 2012. —. The classic

Table 4.5: Bottom 3 examples of articles whose summary is harmed the most by the penalty factor.

Nr.	setting	**best cases**
1	reference	Kywan Fisher must pay damages of $150,000 per film to adult entertainment company Flava Works.
	lead67	By. Graham Smith. PUBLISHED:. 08:15 EST, 2 November 2012. —. UPDATED:. 08:17 EST, 2 November 2012. An illegal file-sharer has been fined a record $1.5million for pirating ten gay porn films online. Ky
	XLNetSUMPen	. "The World of Poker has been awarded a record $1.5million for illegally sharing ten gay porn films online." —. UPDATED:. 08:17 EST, 2 November 2012. Fine: Kywan Fisher must pay damages of $150,000 per film to adult entertainment company Flava Works
	XLNetSUM	. "The "The "" is a new movie. It is a new series of movies. It is a new series of movies. It is a new series of movies. It is a new series of movies. It is a new series of movies."
2	reference	SoHostel is located on Dean Street, W1, just minutes from Oxford Street. It has room for 150 guests and also offers double rooms for £55-a-night. Hostel has onsite bar, big-screen TV and 'Hollywood-style pamper room' Run by One Housing, a not-for-profit organisation helping London's homeless.
	lead67	As Londoners are no doubt all too aware, £25 won't get you very far in the city these days. In fact, it's barely enough to cover a round of drinks. But thanks to a new budget hostel, it can now buy you a bed for the night in the heart of the West End.
	XLNetSUMPen	. The hostel is located on Dean Street, W1, which is minutes away from Oxford Street and Piccadilly Circus. The hostel has room for 150 guests and also offers double rooms for £55-a-night. The hostel has room for 150 guests and also offers double rooms for £55-night
	XLNetSUM	. The hostel is located on Dean Street, W1, which is minutes away from Oxford Street, Piccadilly Circus and Leicester Square. The hostel has room for 150 guests and also offers double rooms for £55-a-night. The hostel has room for 150 guests and also offers double rooms.
3	reference	Stunning embellished hearing aids and walking aids part of project 'Tools of Life' Work of artist Elana Langer and photographer Hanna Agar. Aims to challenge the stigma and social discomfort of disability aids.
	lead67	By. Katy Winter. At first glance a high glamour glossy fashion shoot, these pictures soon reveal themselves to be so much more. The opulent jewellery dripping off the model's ears are in fact diamond encrusted hearing aids, and several of the shots also show the model leaning on elaborately decorated walking
	XLNetSUMPen	. The project, called Tools of Life and aims to challenge the stigma and social discomfort of disability. The project, called Tools of Life and aims to challenge the stigma and social discomfort of disability is a stunning and extravagant fashion statement. The photos are taken in the studio of artist Elana Langer and photographer Hanna Agar with
	XLNetSUM	. The project, called Tools of Life, is the work of ethical artist Elana Langer and photographer Hanna Agar. The project, called Tools of Life, is the work of ethical artist Elana Langer and photographer Hanna Agar. The project, called Tools of Life, is the work of ethical artist and photographer

Table 4.6: Top 3 examples of articles whose show greatest difference in ROUGE-1 score between summary generated with penalty and baseline.

Nr.	setting	**worst cases**
1	reference	Margot Robbie and Jonah Hill fall foul of autocue error. Elisabeth Moss blurts out 'oh sh**' Emma Thompson sends up stage frights by carrying drink on stage. Leonardo DiCaprio's mispronunciation of Philomena as Phil-o-MANIA starts trending on Twitter.
	lead67	Margot Robbie and Jonah Hill fall foul of autocue error. Elisabeth Moss blurts out 'oh sh**' Emma Thompson sends up stage frights by carrying drink on stage. Leonardo DiCaprio's mispronunciation of Philomena as Phil-o-MANIA starts trending on Twitter.
	XLNetSUMPen	.' "Bonkers!' "Sic." "The best speech in awards history." "Bonkers!" "Bonkers!" "Bonkers!" "Bonkers!" "Bonkers!" "Bonkers
	XLNetSUM	.' 'Bonkers' Bisset's speech was a blooper. #GoldenGlobes' 'Maddest acceptance speech in awards history'. #GoldenGlobes' #Maddest acceptance speech in awards history'. #GoldenGlobes' stars
2	reference	Myleene's fuchsia dress becomes Littlewoods' first ever social media sell-out after autumn ad launch. Expected to hit sales of 2,000 by Christmas. Traffic to the £75 fuchsia dress page of Littlewoods.com up by more than 1,200% week on week.
	lead67	Myleene's fuchsia dress becomes Littlewoods' first ever social media sell-out after autumn ad launch. Expected to hit sales of 2,000 by Christmas. Traffic to the £75 fuchsia dress page of Littlewoods.com up by more than 1,200% week on week. By. B
	XLNetSUMPen	. —. UPDATED:. 11:22 EST, 9 October 2012. —. UPDATED:. 11:22 EST, 9 October 2012. —. UPDATED:. 11:22 EST, 9 October 2012. —. UPDATED:. 11:22 EST!
	XLNetSUM	. —. UPDATED:. 11:22 EST, 9 October 2012. —. UPDATED:. 11:22 EST, 9 October 2012. —. UPDATED:. 11:22 EST, 9 October 2012. —. UPDATED:. 11:22 EST.
3	reference	'I still speak to Arg and he will. definitely be getting one of my perfumes for Christmas but the other. cast members weren't my friends, they were just co-workers' Hand-picked model for shoot. Signed for another season on Take Me Out: The Gossip.
	lead67	'I still speak to Arg and he will. definitely be getting one of my perfumes for Christmas but the other. cast members weren't my friends, they were just co-workers' Hand-picked model for shoot. Signed for another season on Take Me Out: The Gossip. By. Bian
	XLNetSUMPen	. Mark Wright's first fragrance was launched in 2011. The perfume was designed by Mark and his parents Carol and Mark Snr. He is 25-year-old Mark Wright. —. UPDATED:. 05:55 EST, 23 November 2012. Mark Wright's second fragrance launched in 2011
	XLNetSUM	. Mark Wright's first fragrance was launched in 2011. The first fragrance was launched in 2012. By. Lauren Goodger. —. UPDATED:. 05:55 EST, 23 November 2012. —. UPDATED:. 05:55 EST, 23 November 2012. Mark says:

Table 4.7: Top 3 examples of articles whose show greatest difference in ROUGE-1 score between baseline and summary generated with penalty

setting	ROUGE-1	ROUGE-2	ROUGE-L
XLNetSUM	15.87	3.59	12.15
XLNetSUMPen	20.96	5.02	14.07
XLNetSUMPen_Alt	16.87	3.88	12.62

Table 4.8: ROUGE F-scores for baseline, XLNet evaluated without and with the repetition penalty on the development dataset

setting	ROUGE-1	ROUGE-2	ROUGE-L
XLNetSUM	15.87	3.59	12.15
XLNetSUMPen	20.96	5.02	14.07
XLNetSUMPen_Prompt	20.35	5.89 0	13.61

Table 4.9: ROUGE F-scores for baseline, XLNet evaluated without and with the repetition penalty on the development dataset

5 Conclusion and Future Work

In recent years, a new approach to training models for different tasks has emerged. Instead of training individual models from scratch for individual tasks, large language models have been trained on an immense amount of data. Such models are then finetuned to specific down-stream tasks. One such tasks is abstractive summarization. Previous work has either finetuned pretrained Transformer-based models such as GPT-2 or BERT (Radford et al., 2019; Khandelwal et al., 2019) for summarization or built a sequence-to-sequence model using these models (Rothe et al., 2019; Liu and Lapata, 2019). This thesis asks whether the novel language model, XLNet, is capable of abstractively summarizing text without finetuning it for the task.

The results of this thesis indicate that XLNet does not lend itself well to unsupervised summarization as the obtained scores stay behind results for BERT or GPT-2 in the past. Its performance can, however, be clearly improved if a repetition penalty is added to let the model break from generating the same token and over again. Experiments show that a multiplicative penalty is preferred to an additive one. This penalty can however transfer the repetition from a word level to a sentence level. Repetition can be mitigated to a certain extent if the model is given several tokens of the story as a prompt. The model seems to be thus put on the right 'track' at the beginning of the generation process. A more detailed, statistical analysis on the effect of the prompt is however needed.

Evidently, the next step would be to finetune the model using the reference summaries. Results from previous work (compare Section 4.2) suggest that ROUGE scores could thus be improved by a factor of 2. It would also be interesting to compare a finetuned XLNet to a sequence-to-sequence model with both encoder and decoder based on XLNet. The fine-tuning routine has been prepared for this thesis.[1] Although evaluating the model is possible on an 11GB GPU with a batchsize of 8 and sequence length of 1024, this is not sufficient for finetuning. Originally, XLNet was trained on 512 TPU chips. Unfortunately, even the TPU-compatible training program did not run successfully on the Google Cloud TPUs. The related errors will be examined in the future.

[1] see `https://github.com/anbestCL/XLNetforSummarisation`

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Edunov, S., Baevski, A., and Auli, M. (2019). Pre-trained Language Model Representations for Language Generation. In *NAACL-HLT*.

Gehrmann, S., Deng, Y., and Rush, A. (2018). Bottom-Up Abstractive Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Hermann, K. M., áš Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). CTRL: A Conditional Transformer Language Model for Controllable Generation. *ArXiv*, abs/1909.05858.

Khandelwal, U., Clark, K., Jurafsky, D., and Kaiser, L. (2019). Sample efficient text summarization using a single pre-trained transformer. *CoRR*, abs/1905.08836.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. pages 66–71.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *arXiv:1801.10198 [cs]*.

Liu, Y. and Lapata, M. (2019). Text Summarization with Pretrained Encoders. In *EMNLP/IJCNLP*.

Nallapati, R., Zhou, B., dos Santos, C., GuÌçehre, Ç., and Xiang, B. (2016). Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Radford, A. (2018). Improving Language Understanding by Generative Pre-Training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Rothe, S., Narayan, S., and Severyn, A. (2019). *Leveraging Pre-trained Checkpoints for Sequence Generation Tasks*.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *EMNLP*.

See, A., Liu, P. J., and Manning, C. D. (2017). Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

Subramanian, S., Li, R., Pilault, J., and Pal, C. (2019). *On Extractive and Abstractive Neural Document Summarization with Transformer Language Models*.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. (2016). Neural autoregressive distribution estimation. *J. Mach. Learn. Res.*, 17:205:1–205:37.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *ArXiv*, abs/1906.08237.