POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Automatic Video Lecture Summarization with Injection of Multimodal Information: Two Novel Datasets and a New Approach

Supervisors

Prof. Luca CAGLIERO

Dr. Moreno LA QUATRA

Candidate

Enrico CASTELLI

April 2023

Abstract

With the growing diffusion of online courses with video lectures, both from universities such as PoliTo and from MOOC[1] platforms, the ability to distill key information is becoming more and more quintessential to the life of a student. Video lectures provide their contents in a multimodal way, not only with the voice of the speaker, which can be transcribed, but also with visual information such as writings on a blackboard or projected slides. The aim of this work is to offer a new tool to learners and teachers that will allow them to supply one of the proposed models with the transcript of a video lecture and obtain its short summary in return in a fully automatic way. To train our Transformer-based models, we build two datasets: OpenULTD, a university lecture and public talk transcripts dataset that expands on the previously released VT-SSum, and UniSum, an original transcript-summary dataset of university lectures from sixtyseven courses offered at MIT and Yale, which we also extend leveraging the lectures' visual information.

[1]Massive Open Online Course.

Ringraziamenti

Ai miei genitori.
Alle mie nonne e ai miei nonni.
A Carolina.
Ai miei amici.
A tutti coloro con cui ho avuto uno scambio di idee.

Ringrazio il prof. Cagliero per avermi permesso di sviluppare un progetto di tesi al punto di intersezione tra i miei interessi e una tecnologia che si sta rivelando sempre più rivoluzionaria.

Table of Contents

List of Figures

List of Tables

List of Code Snippets

Acronyms

AI

Artificial Intelligence

API

Application Programming Interface

ASCII

American Standard Code for Information Interchange

ASR

Automatic Speech Recognition

BART

Bidirectional and Auto-Regressive Transformers

BERT

Bidirectional Encoder Representations from Transformers

BLEU

BiLingual Evaluation Understudy

BPE

Byte Pair Encoding

BPTT

Back-Propagation Through Time

C4

Colossal Clean Crawled Corpus

CLI

Command Line Interface

CPU

Central Processing Unit

CSS

Cascading Style Sheets

CUDA

Compute Unified Device Architecture

CV

Computer Vision

DA

Domain Adaptation

DL

Deep Learning

DOM

Document Object Model

FLOPS

FLoating point Operations Per Second

FRNN

Fully Recurrent Neural Network

GELU

Gaussian Error Linear Unit

GLUE

General Language Understanding Evaluation

GNU

GNU is Not UNIX

GPGPU

General-Purpose computing on Graphics Processing Units

GPU

Graphics Processing Unit

GRU

Gater Recurrent Unit

GSG

Gap Sentences Generation

HTML

HyperText Markup Language

IDE

Integrated Development Environment

JSON

JavaScript Object Notation

LCS

Longest Common Subsequence

LED

Longformer-Encoder-Decoder

LLM

Large Language Model

LSG

Local Sparse Global attention

LSTM

Long Short-Term Memory

MIT

Massachusetts Institute of Technology

ML

Machine Learning

MLM

Masked Language Modeling

MNLI

Multi-genre Natural Language Inference corpus

MOOC

Massive Open Online Course

MT

Machine Translation

NLI

Natural Language Inference

NLP

Natural Language Processing

NLTK

Natural Language ToolKit

NLU

Natural Language Understanding

NN

Neural Network

NSP

Next Sentence Prediction

OS

Operating System

OpenULTD

Open University Lecture Transcripts Dataset

PDP

Parallel Distributed Processing

PEGASUS

Pre-training with Extracted Gap-sentences for Abstractive Summarization

PyPI

Python Package Index

QA

Question Answering

RAM

Random-Access Memory

ReLU

Rectified Linear Unit

RNN

Recurrent Neural Network

ROUGE

Recall-Oriented Understudy for Gisting Evaluation

SCROLLS

Standardized CompaRison Over Long Language Sequences

SGD

Stochastic Gradient Descent

SOTA

State Of The Art

SQuAD

Stanford Question Answering Dataset

SSD

Solid-State Drive

T5

Text-To-Text Transfer Transformer

TPU

Tensor Processing Unit

UniSum

University Summarization dataset

URL

Uniform Resource Locator

VRAM

Video Random-Access Memory

VT-SSum

Video Transcript Segmentation and Summarization

WebVTT

Web Video Text Tracks

X-CLIP

eXpanded Contrastive Language-Image Pretraining

Chapter 1

Introduction

Online courses are growing exponentially in availability. Universities of the whole world, among which PoliTo, have begun offering online lecture recordings and study material years ago, and with the recent pandemic this effort has dramatically increased. Massive open online courses platforms are also seeing a steady rise in popularity.
This tremendous amount of information, made possible by this so diverse offering of e-learning content, may sometimes seem too onerous to delve into at once. The aim of the present master's thesis is thus to improve on existing ways of distilling the most crucial information from very large quantities of text, providing high-quality summaries of entire lecture transcripts, while also making use of visual information contained in the corresponding video files.
Our focus is in the abstractive branch of the field of summarization, which means we are not only interested in the extraction of the underlying concise meaning of an extended body of text, but also in re-elaborating this information in a way that is coherent and pleasant to read.

Our objective carries with it a number of different aspects to solve: first, we need to identify what is the state-of-the-art technology to create text summaries. Even though the field of natural language processing (NLP for short) has existed for at least seven decades, an important breakthrough has been achieved only recently in 2017. This consisted of a novel at the time deep neural network architecture, known as the Transformer [1], which was the first to be purely based on an associative memory mechanism similar to part of the human brain's behavior (the same architecture is what powers the now extremely popular ChatGPT large language model). What also contributed very significantly to the rapid adoption of this particular architecture was the introduction, a year later in 2018, of the pretraining-finetuning paradigm for language models [2]: this is the idea of training a model on large general-purpose text datasets, *e.g.* the English Wikipedia

[3], to let it learn the underlying rules and representations of the language in question, to then release the "pretrained" model (this has been usually done by research groups at big tech companies such as Google, Meta, and Microsoft); at this point, any practitioner with a much smaller dataset at hand and a specific task in mind can leverage the pre-made model to accomplish their objective at a much lower cost than what would have been required to train a model from scratch.

Secondly, we have to find a big enough source of data to train our neural network on, in order for it to learn how to perform our task in the best possible way. Two requirements for this data source are, naturally, that its use is permitted, and that the transcripts it contains are made by a human expert of the subject, instead of an automated system, which would be more error-prone in domain-specific contexts. We find that the MIT OpenCourseWare platform and the YouTube channel of Yale University offer the largest quantity of quality text and video data with an open license, so we scrape and thoroughly clean this information to create our proposed summarization dataset, UniSum, which contains 1,583 lecture transcript-summary pairs from 67 courses, of which 31 offered at MIT and 36 at Yale. We also propose a variant of this dataset extended with the additional information, derived from the video files with a separate video classifier model [4], of whether the lecture speaker was writing (*e.g.* on a blackboard) while pronouncing a sentence, for each sentence of every transcript. The extended version of UniSum can thus be considered multimodal.
We hypothesize that finetuning a model, which was pretrained on written text, on text transcribed from natural speech would not allow it to reach its maximum performance. To address this domain incompatibility, we propose another dataset, OpenULTD, with 14,677 transcripts from the MIT OpenCourseWare platform (3,397), the Yale Courses YouTube channel (1,094), the German OpenHPI platform (957), and a previously released spoken text summarization dataset, VT-SSum [5] (9,229).

Finally, since lecture transcripts are quite long, we need to overcome the inherent maximum input length limitation of Transformer-based models. To do this, we adopt two different but similar methods that have been proposed in 2020 (Longformer [6]) and in 2022 (local-sparse-global attention [7]). Both of these approaches reduce the amount of memory required to use a model on an input sequence of length N from a quadratic dependence $\mathcal{O}(N^2)$ to a linear one $\mathcal{O}(N)$ by implementing a sliding window plus global attention mechanism that allows the model to mainly refer to the surrounding context of the word in question, while still being able to reference the contents of the entire document. In this way, if a professor mentions the beginning of the lecture in even one of the final sentences, the context can still be retrieved.

On top of the results that we obtain with a model trained on our standard dataset, we investigate the advantages given by pretraining on OpenULTD and by using the extended UniSum variant. We measure the performance of our models with the ROUGE [8] and BERTScore [9] metrics, which respectively reflect their recall and semantic similarity abilities by comparing their generated summaries with the corresponding human-made originals.
To perform our summarization experiments we choose BART [10], which is a Transformer-based model pretrained by Meta AI Research with a text denoising task, that is to say trained to restore corrupted text, which its authors find improves the network's abstractive summarization capabilities. We continue its pretraining on our transcript-only dataset, OpenULTD, for the process known as domain adaptation. We then modify the model with one of the previously mentioned more efficient attention mechanisms to make it able to process long input sequences. We finetune different instances of the resulting model on the standard and extended variants of the UniSum dataset, to obtain a final model that generates qualitatively good summaries.
We notice that the model variant finetuned on extended UniSum tends to improve upon the performance of its counterpart finetuned on standard UniSum when generating summaries of course lectures where the speaker tends to only write a few things (*i.e.* mostly in humanities courses in our dataset, such as literature or psychology), which are likely to be very important. We think that this positive effect is negatively correlated with the amount of writing that a professor produces while teaching a lecture. That is, put more clearly: if most sentences of a lecture are classified as "writing", the model will not be able to capture any new information from this augmented input; but if, instead, most sentences are classified as "not writing" and only a few as "writing", then those few can be deemed more relevant for the summary by the model.
We find that, although not to the same extent and in every case, the domain adaptation step performed with OpenULTD and the additional information derived from the video modality contained in the extended version of our UniSum dataset contribute to measurably better performance.

The curious reader will find some example summaries generated by our best model in table A.1 and our released code, datasets, and models in our repository available at `https://github.com/e-caste/masters-thesis`.

Chapter 2

Background

In this chapter we are going to present the most relevant historical results that have allowed for the birth of the field of natural language processing (§ 2.1), which this master's thesis is part of, and its subsequent thriving in today's deeply interconnected world (§ 2.2 and § 2.3).

2.1 Before the Transformer

The father of all the models upon which the present work is based on is the Transformer, a «revolutionary» (LeCun [11]) neural network architecture authored by Vaswani *et al.* [1] in 2017, which first introduced models purely built on the attention mechanism, and is currently «taking the world by storm» [11], so much so that researchers are replacing their whole neural networks with Transformers.

In order to illustrate why and how the Transformer has revolutionized the field of natural language processing (*e.g.* ELMo, Peters *et al.* 2018 [12]; BERT, Devlin *et al.* 2018 [2]; GPT, Radford *et al.* 2018 [13]; RoBERTa, Liu *et al.* 2019 [14]; XLNet, Yang *et al.* 2019 [15]; BART, Lewis *et al.* 2019 [10]; GPT-2, Radford *et al.* 2019 [16]; T5, Raffel *et al.* 2019 [17]; PEGASUS, Zhang *et al.* 2019 [18]; GPT-3, Brown *et al.* 2020 [19]; BigBird, Zaheer *et al.* 2020 [20]; Codex, Chen *et al.* 2021 [21], powers GitHub Copilot; InstructGPT, Ouyang *et al.* 2022 [22], is the basis for ChatGPT; BlenderBot, Shuster *et al.* 2022 [23]) and is currently making its way into other fields, such as computer vision (*e.g.* DETR, Carion *et al.* 2020 [24]; ViT, Dosovitskiy *et al.* 2020 [25]; CLIP, Radford *et al.* 2021 [26]; DALL-E and DALL-E 2, Ramesh *et al.* 2021 [27] and 2022 [28]; Stable Diffusion, Rombach *et al.* 2022 [29]) and automatic speech recognition (*e.g.* HuBERT, Hsu *et al.* 2021 [30]; Whisper, Radford *et al.* 2022 [31]), we must first review what came before it.

As the human language, in its spoken and written forms, is the principal way of communication on the planet, researchers and innovators have thought of NLP[1] and NLU[2], which are the ability of a machine to work with language as is created by humans to obtain a certain outcome, and to have a formal understanding of the underlying rules of language, such as syntax, grammar, and context, since the birth of machines themselves.

Though an attempt to make a mechanical computer was carried out by Charles Babbage already in the 1830s with partial success [32], Alan Turing was the first to propose, in 1950 [33], a way to evaluate if a machine «could think», where a machine is more specifically defined as an «'electronic computer' or 'digital computer'», which he called «the imitation game» and is now known to the general public as the Turing test. The game is posed as such: there are three participants, A, B, and C, in different rooms, communicating via typewritten text only; A and B are a man and a woman; C is the interrogator, who can be of either sex, and their objective is to correctly guess the gender of the other participants by asking them any kind of question and evaluating their replies. Turing argues that this same game, after replacing A or B with a machine[3], can be used as a test of the quality of the machine: if the interrogator is not only unable to distinguish A from B, but is fooled by the machine to choose it instead of the human competitor at least in some instances, then the machine has done well and has won the imitation game. The attentive reader will have noticed that the question that the imitation game answers has changed from the initial formulation "Can machines think?" to "How well does the machine perform?". This is intended, and follows Turing's article [33]: first, he proposes the original question; then, since there is no way to measure thinking («The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion.»), he suggests to replace it with the question given by the imitation game: «Are there imaginable digital computers which would do well in the imitation game?»; and finally, after defining that digital computers are, in fact, discrete state machines with a very high number of possible states, he updates the question to: «Are there discrete state machines which would do well?» (in the imitation game).
In the same article [33], Turing also attempts to predict the distant future:

[1]Natural Language Processing.

[2]Natural Language Understanding.

[3]He actually specifies that the machine would replace the woman and not the man, likely due to other people's theological arguments at the time, possibly more felt in the UK in the 1950s than today, about only the men certainly having a soul given by God; in this way, the machine would beat a soul-given human.

> «I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent chance of making the right identification after five minutes of questioning.»[4] [1]

and

> «I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.» [2]

While in current times many still find the latter [2] quite debatable [34], particularly with respect to the definition of "sentience" more than thinking in general (which can be seen as processing information), the former [1] seems to have come true.

The recent article "Human or machine? Turing tests for vision and language" by Zhang *et al.* [35] released in November 2022 shows (see figure 2.1) that, within the scope of conversation, some of the latest Transformer-based AI chatbots such as BlenderBot by Meta AI Research [23] and GPT-3 by OpenAI [19] are classified as humans by human judges more than 50% of the times, with this score reaching the proximity of 60% in shorter conversations.

[4]The storage is, for Turing, the union of the table of instructions and the active variables. In current terms, these would be the CPU registers and memory. The storage capacity he refers to is the base-2 logarithm of the number of binary states: a capacity of 10^9 bits (= 1 gigabit $\approx$ 100 megabytes) corresponds to 2^{10^9} states.

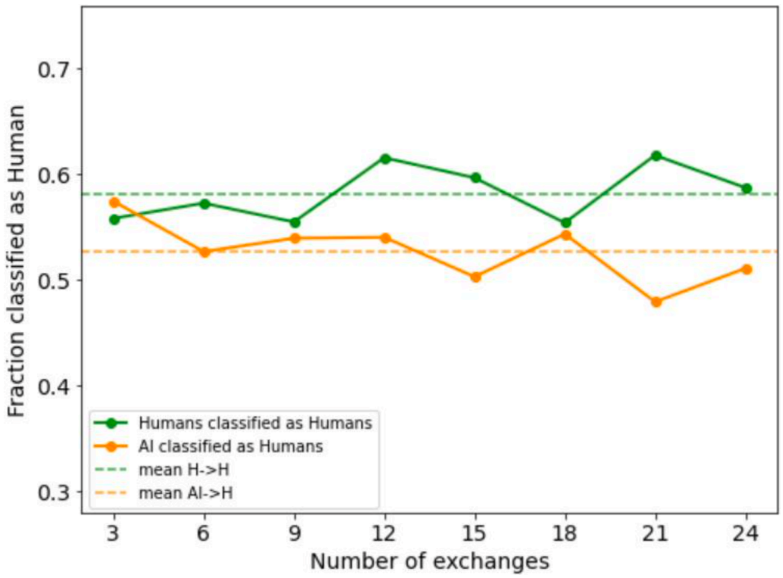


Figure 2.1: Length dependence of Turing test results for different settings and models, average curves over all settings and models. Figure S88a in [35].

Considering that the length of these AI-human judge or person-human judge conversations are reported to not exceed a duration of ten minutes, and that the number of exchanges is twentyfour (twelve texts by the human judge and twelve texts by the receiving part under the Turing test, either an AI or a person), then Turing's proposed five minute mark $\boxed{1}$ corresponds to the twelfth exchange on the graph, where it is reported that AIs are classified as humans by human judges in ~55% of the instances, while people are classified correctly as humans by human judges slightly more than 60% of the times. Thus, given that the probability of the text being written by a human or by an AI is the same («in total we collected 40 conversations for each category» [35], where the categories are AI-AI, AI-human, and human-human):

$$\begin{aligned} p(AI\ as\ H) &= \frac{p(AI\ as\ H \mid text\ written\ by\ H) + p(AI\ as\ H \mid text\ written\ by\ AI)}{2} \\ &= \frac{(1 - 0.65) + 0.55}{2} \\ &= \frac{0.35 + 0.55}{2} \\ &= \frac{0.9}{2} \\ &= 0.45 \\ &= 45\% \end{aligned} \tag{2.1}$$

where $p(AI\ as\ H)$ is the probability of an AI being classified as a human.
Equation 2.1 shows that Turing's conjecture of human judges being unable to distinguish a machine from a human in at least 30% of tests after five minutes of textual conversation within the end of the century [1] has been proven correct, even though with a delay of twentytwo years.

This preamble allows us to highlight the fact that natural language has been regarded as the primary desired form of human-machine interaction for at least seven decades, therefore an excellent way to perform its processing and understanding has been in the past and is being now constantly researched and improved upon. The extensively detailed report of the Georgetown-IBM system demonstration of 1954 published by Hutchins in 2005 [36] explores the first publicly known seq2seq[5] model running on a computer. Although the idea of using computers for machine translation had been alredy suggested in the late 1940s, this was the first instance of a computer performing it in practice. The hardware that IBM had provided to the Georgetown university was a 701-type machine initially «developed for military applications» [36] and «was at that time only one of about one hundred general-purpose computers in existence». The IBM 701 was made of eleven distinct electronic units [36], had a memory that could store up to 4,096 words of 36 bits each (for a total of 18 kilobytes), and could process 14,000 instructions per second [37].
Due to the textual nature of the data to process in a typical MT[6] task, and that as we have seen the IBM 701 was one of the very first general-purpose digital computers available in those days, every action to be performed on non-numerical data had to be programmed from scratch, including the text encoding [36]. The two most relevant aspects to the present thesis of that experiment are however the model and the data that it was to process. In 1953, the most advanced technology for MT consisted of rule-based models; more specifically, the model that they used to translate more than sixty sentences, for the great part about chemical processes and including some of general interest, from Russian to English, had only six basic rules to follow [36]. These rules would be transcribed in very few lines of code of a modern programming language rather than in machine code on punch cards, but then again, that would not mean pioneering neither machine translation nor digital general-purpose computing, the both of which Georgetown University and IBM's researchers were in fact doing at the time. Furthermore, the model's

[5]Sequence-to-sequence: a class of models that receives a sequence as input, applies some transformations to it, and outputs the transformed sequence. A sequence is defined as a set where the order of its composing elements is crucial to the set's identity. In NLP the elements could be words or word stems, thus an example of a sequence would be a sentence.

[6]Machine Translation.

entire vocabulary only encompassed two hundred and fifty between stems and word endings[7] in Russian, with either one or two corresponding English translations. This publicity stunt worked very well in favor of Georgetown University, which received close to M$ 2 by the CIA in research funds to continue working on the matter.
But the post-war optimism clouded the judgment of both the leaders and the viewers about what was, in fact, a very simplistic experiment, which abundantly underestimated all that is needed to effectively understand a language, or even two languages in the case of translation. In the IBM press release which followed the demonstration, Dostert, the Georgetown professor heading the project, said [36]:

> «Those in charge of this experiment now consider it to be definitely established that meaning conversion through electronic language translation is feasible.» [3]

And, most notably:

> «Five, perhaps three, years hence, interlingual meaning conversion by electronic process in important functional areas of several languages may well be an accomplished fact.» [4]

This premature conclusion has proven to be quite incorrect: after having implemented a major update into Google Translate [38], the nature of which will be discussed later in more detail, Le and Schuster write: «machine translation is by no means solved» [39]. This was in 2016, sixtytwo years after Dostert's prediction of MT being solved within three to five years at most [4].

In the decades following the 1950s, the progress in the field of NLP, except for the appearance of the first rule-based chatbots[8] such as ELIZA in 1966 [40] (see figure 2.2), was very slow. ELIZA is a perfect example of the technology available at the time: using simple pattern matching and keyword ranking, *e.g.* "you" followed by "me", or "you" followed by "are", the latter being interpreted as an assertion, the chatbot would reply in a way that prompted excitement and bedazzlement in «even the most experienced observer» [40], by making them believe that it had understood the prompt. But, as Weizenbaum [40] poetically puts it:

> «once a particular program is unmasked, once its inner workings are explained in language sufficiently plain to induce understanding, its magic

[7] To facilitate NLP models to process variants of the same original word, words are usually split into stems and endings. As an example: given the word "playing", the stem would be "play-" and "-ing" would be the ending; "played" shares the same stem, but its ending is "-ed".

[8] A portmanteau of "chatter" and "robot".

crumbles away; it stands revealed as a mere collection of procedures, each quite comprehensible.» [5]

We will see that this [5] is less, although still in some sense, true today.

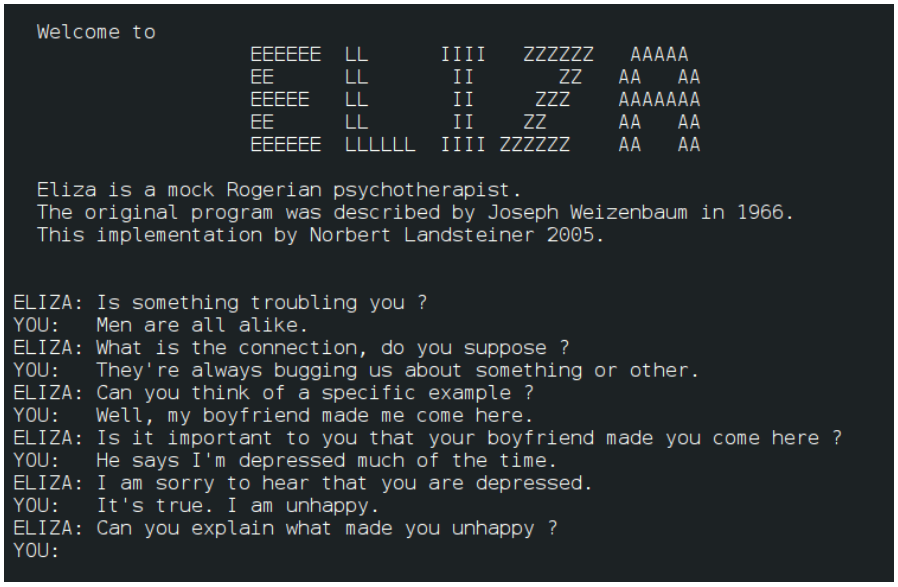


Figure 2.2: An example conversation with a reimplementation of the DOCTOR script of ELIZA, originally created by Joseph Weizenbaum in 1966 at MIT.

Starting in the 1970s and continuing the efforts in the 1980s, great discoveries were made in the dawning field of deep learning, some of which are still at the core of the current learning processes with little to no modification. Let us be clear, the hardware available at the time is in no way comparable to what we have now[9], hence few NLP practical applications were available. In 1986, Guida and Mauri [43] write:

> «Research on natural language processing (NLP) has received increasing attention over the last thirty years and, recently, it has begun to interest

[9] ...or, actually, it is comparable but the hardware has so hugely improved in various aspects that it makes little sense. But, since we are engineers, here is a number comparison of floating point operations per second of a 1982 supercomputer [41] versus a current generally available graphics card [42] (the one used for the present thesis):

Name	**Cost in 2021 USD**	**Performance**
Cray X-MP/48	39,120,000	0.8 GFLOPS (1x)
NVIDIA RTX 3090	1,000	35.6 TFLOPS (44,500x)

> concrete applications in business, services, and industry. While a lot of effort has been devoted in this period to the study of computational models of language and to their implementation into experimental systems, surprisingly enough only little attention has been reserved for the issue of evaluating their power and performance.» [6]

This lack of interest towards measuring the actual performance of NLP models was most likely due to the fact that the field had not yet flourished, and could not offer great results, at least for a couple of decades.
But, despite the commercial and general unavailability of efficient NLP systems, those years were crucial for laying the foundation of contemporary deep learning models [44]: the concepts introduced with PDP[10] networks (McClelland, Rumelhart, Hinton 1986 [45]), applied to NLP (Miikkulainen and Dyer 1991 [46]) in conjunction with deep networks (Hopfield 1982 [47]; Fukushima *et al.* 1983 [48]; Ackley, Hinton, Sejnowski 1985 [49]; LeCun *et al.* 1989 [50]) trained by means of backpropagation (Werbos 1974 [51]; Parker 1985 [52]; LeCun 1985 [53]; LeCun 1986 [54]; Rumelhart, Hinton, Williams 1986 [55]), would set the course of the field of NLP for decades to come.

The idea of machine learning was not conceived in the 1980s; it was not even conceived in the twentieth century. In fact, it was Ada Byron, Countess of Lovelace who probably expressed the first ever opinion on the matter, albeit negatively, in the 1840s [56], when writing about Charles Babbage's (unfinished) Analytical Engine:

> «The Analytical Engine has no pretensions whatever to *originate* anything. It can do whatever *we know how to order it* to perform.» [7]

In reference to this same quote [7], Hartree [57] adds:

> «This does not imply that it may not be possible to construct electronic equipment which will "think for itself", or in which, in biological terms, one could set up a conditioned reflex, which would serve as a basis for "learning". Whether this is possible in principle or not is a stimulating and exciting question suggested by some of these recent developments. But it did not seem that the machines constructed or projected at the time had this property.» [8]

[10] Parallel Distributed Processing.

Turing found himself [33] in strong agreement with Hartree [8], and clarified that he believed that Lady Lovelace, while working for the betterment of Babbage's Analytical Engine, simply had not had enough inputs as to which suppose that such a machine could "think" or "learn". Further, Turing seems to have correctly predicted the underlying idea of backpropagation [33]:

> «If, for instance, the machine was trying to find a solution of the equation $x^2 - 40x - 11 = 0$ one would be tempted to describe this equation as part of the machine's subject matter at that moment. In this sort of sense a machine undoubtedly can be its own subject matter. It may be used to help in making up its own programmes, or to predict the effect of alterations in its own structure. *By observing the results of its own behaviour it can modify its own programmes so as to achieve some purpose more effectively.* These are possibilities of the near future, rather than Utopian dreams.»[11] [9]

[11] Emphasis added to highlight the statement most closely related to the concept of backpropagation.

Before explaining in more detail the most significant precursor to the Transformer, that is to say the recurrent neural network, or RNN for short, it would be unwise to gloss over the main general principles of deep learning.

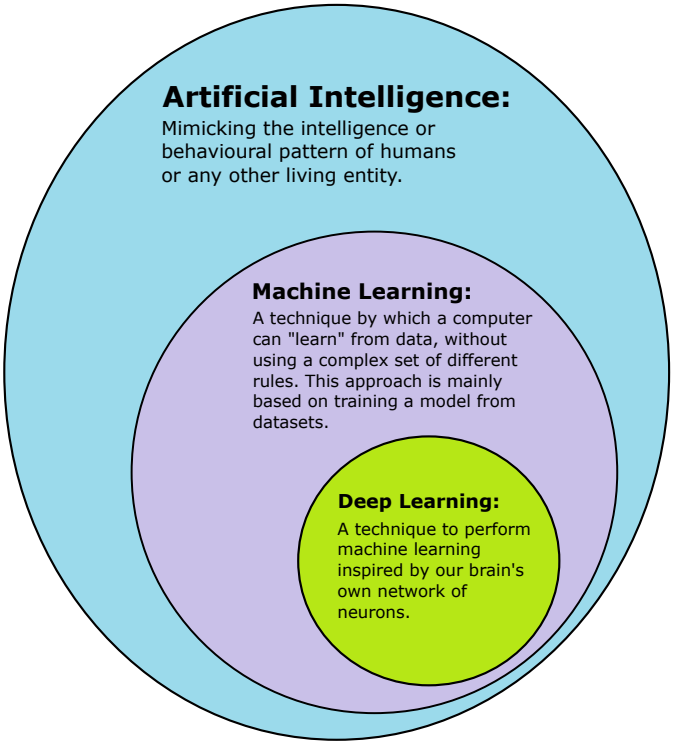


Figure 2.3: Venn diagram of AI including ML including, in turn, DL. By Wikipedia user Tukijaaliwa, shared under the CC BY-SA 4.0 license.

Deep learning is a subset of machine learning methods (as shown in figure 2.3) that make use of deep neural networks. Although single layer neural networks (see figure 2.4) exist and have been the basis for multiple layer networks (see figure 2.5), a deep neural network is characterized by the presence of hidden layers among its constituents. These hidden layers are so called due to the fact that they are neither the input layer, which is where the data is fed into, nor the output layer, which is where the network manifests the outcome of its internal processing.

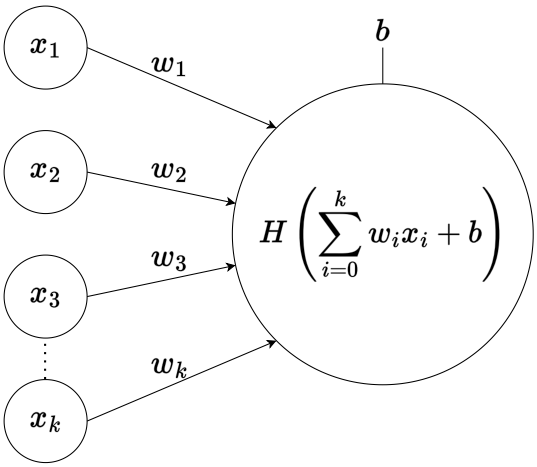


Figure 2.4: A single-layer perceptron, first introduced by Rosenblatt in 1957 [58] based on the 1943 results of McCulloch and Pitts [59]. The H function is known as the Heaviside step function, which outputs 1 if its input is greater than 0 and 0 if its input is less than or equal to 0 (see figure 2.6f). The x_i values are the inputs of the network, the w_i are the weights (one for each input), and b is the bias of the neuron, which in this case behaves as a binary classifier due to its activation function.

The hidden layers are essential to learn the inner hierarchical representations of the data that is input to the network, without having to «design a feature extractor that transforms the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, can detect or classify patterns in the input» [44].
LeCun, Bengio, and Hinton, often referred to as "the fathers of deep learning" [60], have provided an overview of why deep networks are so efficient at solving complex problems in 2015 [44]:

> «Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. [...] The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure. [...] [Deep learning] has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government.» [10]

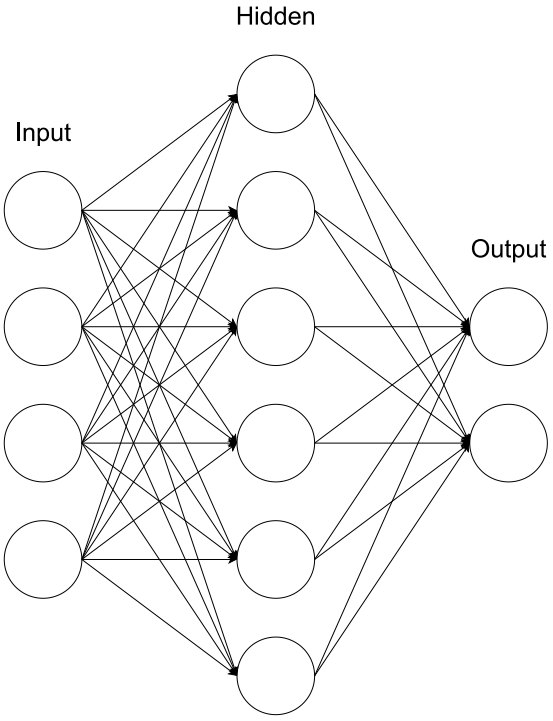
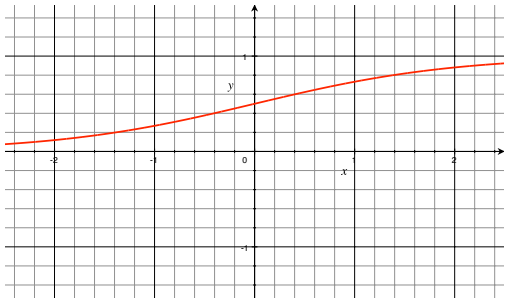
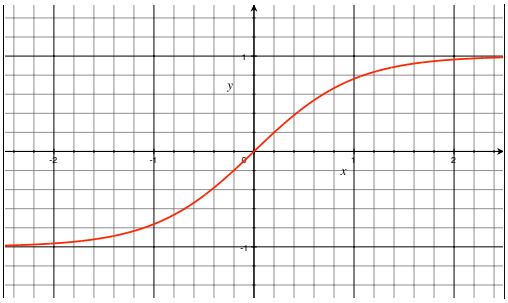


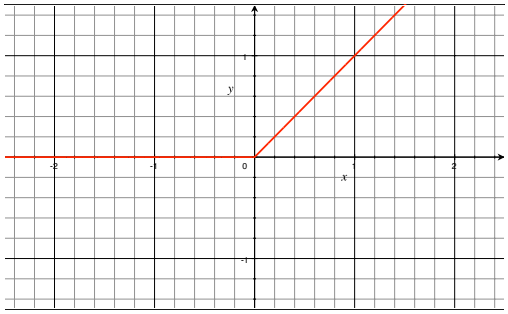
Figure 2.5: A deep neural network with one fully connected hidden layer. In this case, the Heaviside step function could be replaced by the positive part f^+, also known as the ReLU (Rectified Linear Unit) activation function (see figure 2.6c), commonly used in today's deep networks [61]. It can be implemented in Python as `max(0, value)`.



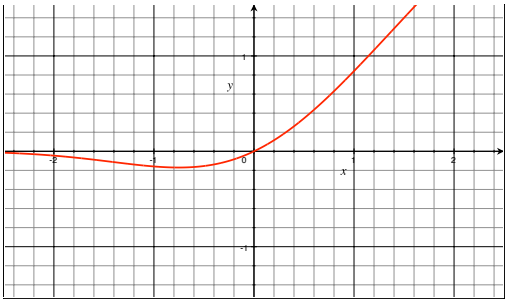
(a) $\sigma(x) = \frac{e^x}{e^x+1}$



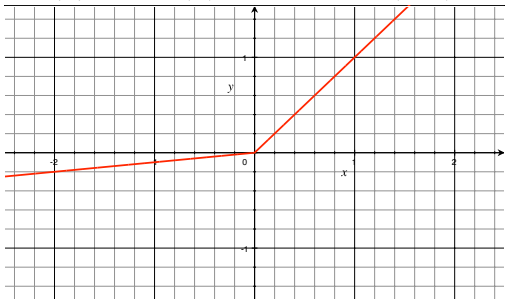
(b) $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$



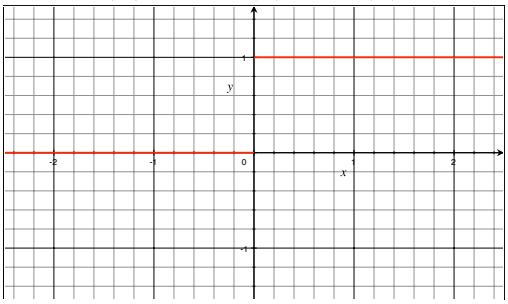
(c) $ReLU(x) = x^+ = max(0, x)$



(d) $GELU(x) = x\Phi(x)$



(e) $LeakyReLU(x) = \begin{cases} 0.1x & x < 0 \\ x & x \geq 0 \end{cases}$



(f) $H(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$

Figure 2.6: Five commonly used activation functions in today's deep networks and the Heaviside step function used in the Perceptron (2.6f). The most notable are the rectified linear unit (2.6c) and its derivatives: the GELU [62] (2.6d) and the leaky ReLU (2.6e, shown with an arbitrary 0.1 slope factor in the negative part). The sigmoid (2.6a) and the hyperbolic tangent (2.6b) were more widely used in the past.

To "train" these deep networks to accomplish the task they are designed for, researchers and practitioners use stochastic gradient descent [63], SGD for short, enabled by the backpropagation algorithm [53], [55]. The idea behind it is not overly complicated: given that a deep network is a non-linear function with a big number

of adjustable weights and biases, also known altogether as parameters, we need to know how to adjust them in a way that the model performs the task obtaining the best possible result. To do this, we iteratively show the model a sample of the available training data, known as mini-batch or simply batch, we compute with a previously established loss or cost function how big the error of the network's prediction is with respect to the known answer, also named label or labels in supervised learning, and at this point we compute how much each parameter is responsible for the error, which translates mathematically to the derivative of the parameter with respect to the loss function of the result, propagated through the network's layers using the chain rule. Finally, we adjust the parameter by that result. Since we usually have many parameters, instead of a single derivative it is more correct to refer to this variation as the gradient. This step is known as backpropagation, and once it is applied to all parameters, the effect is for the network to take a step towards the minimum of the loss function, which the lower value it assumes, the more it reflects that the model has learned well to perform the specified task given a dataset.

A core step forwards in terms of training performance of deep networks was made in the 2000s: drawing on the idea that DL models require a hefty quantity of matrix additions and multiplications (an in-depth example can be found in section 2.2), GPUs[12] could be used for the training process of deep networks instead of CPUs[13], leveraging their much higher core count[14]. NVIDIA was the first to release a framework for GPGPU[15] programming with their CUDA[16] API[17] in 2008 [64].
The advantage of using GPUs was apparent from the beginning: just the year after the first CUDA release, Raina, Madhavan, and Ng demonstrated a 70× decrease in training time on an NVIDIA GTX 280 with respect to a contemporary dual-core CPU [65], which drove researchers to networks with more parameters. The decisive moment that the world accepted GPUs as the best platform for training deep networks was a few years later in 2012, when Krizhevsky, Sutskever, and Hinton [66] proposed AlexNet, a CNN[18] trained on two NVIDIA GTX 580 which improved the state of the art in image classification by a very significant margin.

[12]Graphics Processing Unit.

[13]Central Processing Unit.

[14]In 2023, a typical CPU core count is ~10, while a common GPU core count is ~10,000.

[15]General-Purpose computing on Graphics Processing Units.

[16]Compute Unified Device Architecture.

[17]Application Programming Interface.

[18]Convolutional Neural Network.

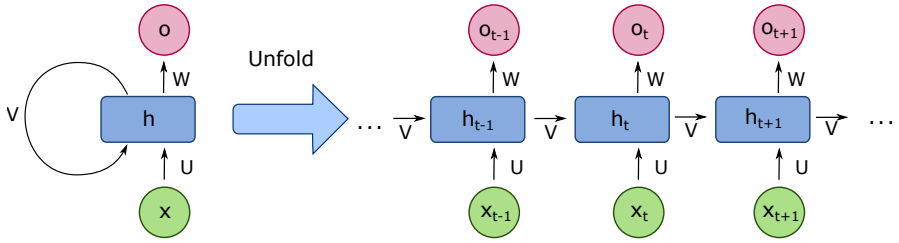


Figure 2.7: A diagram for a one-unit recurrent neural network (RNN). From bottom to top: input state x, hidden state h, output state o. U, V, and W are the weights of the network. Looping diagram on the left and unfolded version on the right. The t subscript indicates the time instant. By Wikipedia user fdeloche, shared under the CC BY-SA 4.0 license.

Among the various types of notable neural networks, the one that was the most applied to sequence-to-sequence tasks before the birth of the Transformer was the RNN, shown in figure 2.7. First introduced by Hopfield in 1982 [47], then improved by Rumelhart, Hinton, and Williams in 1986 [55], recurrent neural networks are a class of deep neural networks that can handle a variable-length input sequence. This approach has two critical consequences:

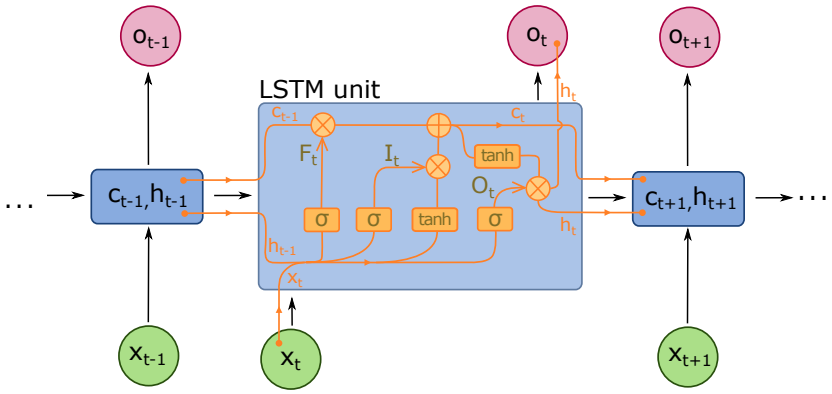
1. since the network cannot know the number of inputs a priori, all the layers share the same weights;

2. the looping network needs to be "unrolled" in all its intermediate states, the number of which depends on the number of inputs, before being able to apply the backpropagation algorithm to perform SGD. This variant of backpropagation is known as backpropagation through time [55], or BPTT for short.

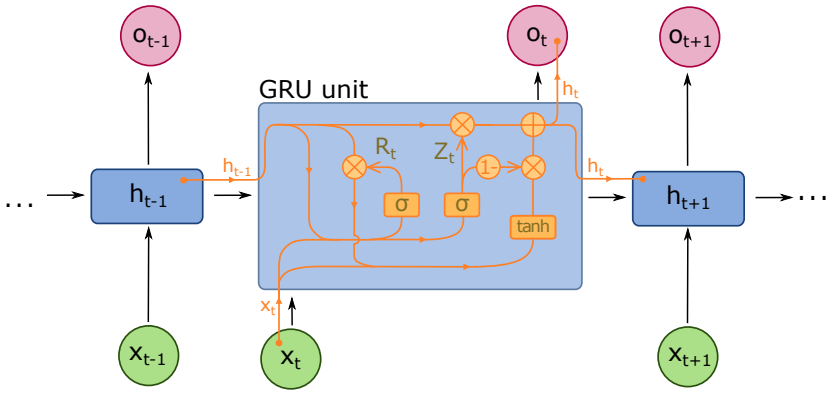
Point 2 is especially the reason why a simple, or fully, recurrent neural network (FRNN) will never be able to learn a task successfully with a long input sequence. This is known as the vanishing and exploding gradients problem, studied by Bengio *et al.* in 1994 [67]. A few attempts at solving these issues have been proposed over the years: in 1995, El Hihi and Bengio [68] studied hierarchical RNNs; in 1997, Hochreiter and Schmidhuber achieved a breakthrough with the LSTM[19] [69], shown in figure 2.8a, which specifically tackles the issue of long-distance memory; some optimization methods for training RNNs were proposed by Pascanu, Mikolov,

[19]Long Short-Term Memory.

and Bengio in 2012 [70] and by Sutskever in 2013 [71]; finally, in 2014, Cho *et al.* introduced the GRU[20] [72], shown in figure 2.8b, a further optimization of the LSTM.



(a) A diagram for a one-unit Long Short-Term Memory (LSTM).



(b) A diagram for a one-unit Gated Recurrent Unit (GRU).

Figure 2.8: From bottom to top: input state x_t, hidden state h_t, cell state c_t (in the LSTM), and output state o_t. Gates are sigmoids (σ) or hyperbolic tangents ($tanh$), both shown in figure 2.6. Other operators: element-wise sum and multiplication. Weights are not displayed. By Wikipedia user fdeloche, shared under the CC BY-SA 4.0 license.

[20] Gated Recurrent Unit.

2.2 The Transformer

The Transformer is, as mentioned at the beginning of section 2.1, the father of all the models employed in the present thesis. Its release in 2017 by Vaswani *et al.* [1] was a watershed moment for the world of NLP and is having an impact on other DL branches such as computer vision and speech recognition.

This neural network architecture (shown in figure 2.9) had when it was published and still has today a big significance in the deep learning field, specifically because it is «based solely on attention mechanisms, dispensing with recurrence and convolutions entirely» [1], where CNNs[21] and RNNs were the most widely used types of models until that moment.
This change of architecture has three main consequences with respect to the previous approach to sequences of recurrent networks:

1. since the Transformer does not contain any loops, the BPTT[22] technique used in RNNs can be replaced by simple backpropagation. This translates to a significant speed-up in training and inference times [1], since what was a sequential approach is now heavily parallelizable, especially on current hardware, *i.e.* GPUs and TPUs;

2. the downside of the previous point is that this comes at the cost of trading off variable-length input sequences for fixed-size inputs;

3. the Attention mechanism can be exploited for better associative memories than those found in RNNs, be they FRNNs, LSTMs, or GRUs, with long-range dependencies developed in such a way that the final input in the sequence can attend to the initial one (and viceversa) in a constant number of steps[23].

Let us now delve into some implementation details of the Transformer in order to gain some insights on the reason why it was a breakthrough in NLP. First of all, it is essential to have a high-level intuition of the Attention mechanism. This mechanism, apart from the addition of the scaling factor [1], existed before the Transformer under several names: multiplicative module, quadratic layer, product unit, Sigma-Pi unit [11]. These are all variations of a matrix-based software implementation of

[21] Convolutional Neural Networks, mainly used for computer vision tasks such as object recognition and semantic segmentation.

[22] Back-Propagation Through Time.

[23] This number of steps is exactly 1 when using the original paper's [1] implementation of Attention, whose memory requirement linearly depends on the fixed input size N with $\mathcal{O}(N)$, where *e.g.* $N = 512$. Methods to reduce the amount of required memory at the cost of extra steps are addressed in chapter 4.

a dimmer switch with multiple inputs: their purpose is to brighten or dim each of their inputs, with a shared maximum level of brightness. To this end, it is desirable to work with a weight distribution that sums to 1, which can be obtained by utilizing the softmax function, who some prefer to call softargmax, here applied to $\boldsymbol{w} \in \mathbb{R}^K$:

$$softmax(\boldsymbol{w})_i = \frac{e^{w_i}}{\sum_{j=1}^{K} e^{w_j}} \tag{2.2}$$

Despite its daunting look, it is a simple function; let us see an example with $\boldsymbol{w} = \{-1.23, 4.56, 3.14\}$ (hence $\boldsymbol{w} \in \mathbb{R}^3$ and $K = 3$):

$$\begin{aligned} sm(\boldsymbol{w}) &= \left\{ \frac{e^{-1.23}}{\sum_{j=1}^{3} e^{w_j}}, \frac{e^{4.56}}{\sum_{j=1}^{3} e^{w_j}}, \frac{e^{3.14}}{\sum_{j=1}^{3} e^{w_j}} \right\} \\ &= \left\{ \frac{0.29}{0.29 + 95.58 + 23.10}, \frac{95.58}{0.29 + 95.58 + 23.10}, \frac{23.10}{0.29 + 95.58 + 23.10} \right\} \\ &= \{0.002, 0.803, 0.195\} \end{aligned} \tag{2.3}$$

where sm is the $softmax$ function and $0.002 + 0.803 + 0.195 = 1$.
It is apparent that the $softmax$ function turns the biggest value of $\boldsymbol{w}$, in this case 4.56, into the highest weight, or probability, in the output distribution.
These weights $\boldsymbol{w}$ can then be applied to an input vector, say $\boldsymbol{x}$, to get in return the same input vector $\boldsymbol{x_w}$, now weighted by level of importance given to each x_i by its corresponding weight w_i.
This is the essence of the Attention mechanism: learning how important a certain input is with respect to the desired outcome in the given context, and focus more on the relevant parts while setting aside the unimportant details.

The main actors in the Transformer architecture are thus what we have called the input vector $\boldsymbol{x}$, which in practice translates to the three matrices Q, K, and V, for queries, keys, and values. These, along with the learnable parameters contained in the token embeddings, are what the Transformer optimizes when it is subject to training on a task with a given dataset. The whole process can be seen in figure 2.9.

Transformers are canonically composed of two main sections: the encoder (shown in the left half of figure 2.9) and the decoder (in the right half of the same figure). The encoder has the task of "converting" the input sequence, which in the context of NLP is usually a sentence string, to its internal representations. The decoder's objective is instead to provide a new sequence, taking into account the representations learned by the encoder, its own weights, and the previous sub-sequence.
Here is an example: the string `"Hello World!"` is provided to a Transformer. Before it can be processed into input or output embeddings (see the bottom of

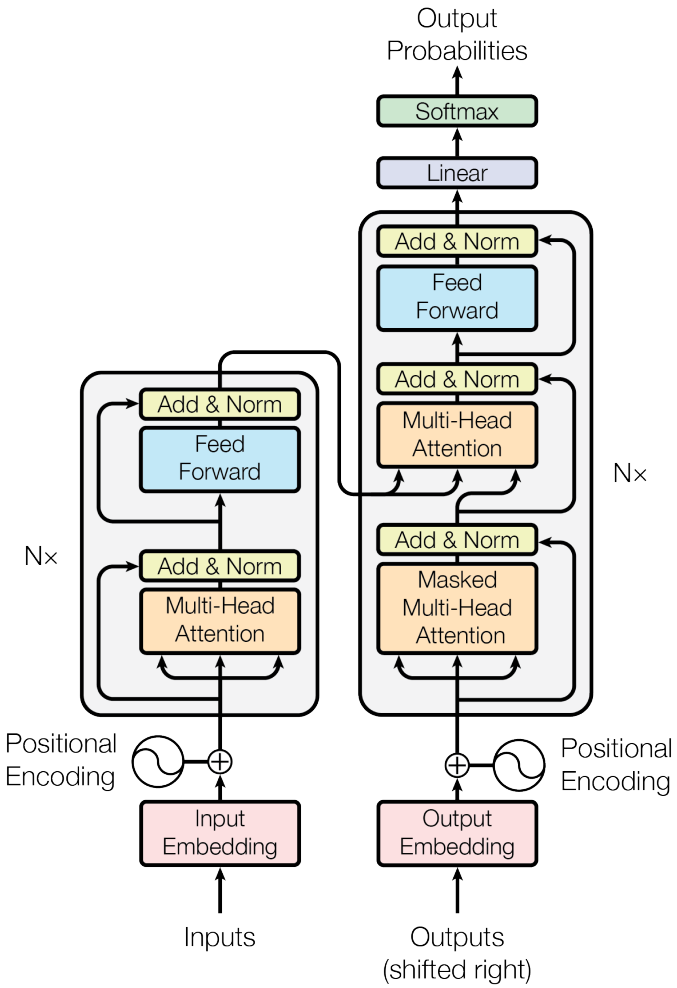


Figure 2.9: The Transformer architecture as presented in the original paper [1], where this is figure 1.

figure 2.9), it first needs to be translated into numerical tokens, *e.g.* `[72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 33]`. This toy example makes use of a character-level tokenizer, which simply returns the index in the ASCII[24] table given a character[25], but real-world tokenizers use more sophisticated algorithms like BPE[26] with subword units [73] or SentencePiece Unigram [74], [75]. The tokens can now be embedded into a matrix $E \in \mathbb{R}^{d_{vocab} \times d_{model}}$, where $d_{vocab} \approx 50{,}000$ and

[24] American Standard Code for Information Interchange.

[25] Easily implemented in Python: `str2tok = lambda s: [ord(char) for char in s]`.

[26] Byte Pair Encoding.

$d_{model} = 512$. This matrix is summed with another one of the same size, containing the embeddings for the absolute and relative positions between the tokens, which results in the actual input of both the encoder and the decoder. At this point the input embedding follows a different path depending on which "half" of the Transformer it is fed into:

- in the encoder[27], the embedding matrix E_{enc} is first processed with Self-Attention. Self-Attention is one of the possible Attention mechanism, and involves using the same knowledge source, here a numerical matrix, for queries, keys, and values. Specifically, the Transformer uses three matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, and $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, for queries, keys, and values respectively, where i is the index of the head to which the ith matrix belongs. The head is one of h[28] parallel attention layers, which the authors [1] demonstrate is beneficial to include in the architecture as each head learns different linguistic aspects as shown in figure 2.10. The Multi-Head Attention block shown in figure 2.9 is nothing more than the concatenation of the results of the different heads:

$$\begin{aligned} head_i :=& Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \\ head_{i_{enc}} =& Attention\left(E_{enc}W_{i_{enc}}^Q, E_{enc}W_{i_{enc}}^K, E_{enc}W_{i_{enc}}^V\right) \end{aligned} \tag{2.4}$$

  where $E_{enc} \equiv Q \equiv K \equiv V$ is applied since these heads perform Self-Attention, and:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.5}$$

  The intuition behind this last equation is the following: the dot product between the queries matrix Q and the transpose of the keys matrix K is, at its core, a projection which allows to determine how aligned each of their composing vectors is with each vector in the other matrix. The scaling factor $\frac{1}{\sqrt{d_k}}$ was introduced by the Transformer's authors [1] to avoid too polarizing attention vectors similar to one-hot encodings (*e.g.* $\{0.0005, 0.001, 0.998, 0.0005\}$), which would not allow a token to attend to other tokens with a soft distribution, and would instead force its focus on a limited number of other tokens, or even just a single one. Finally, the dot product with the values matrix V acts as a soft indexing inside what can be thought of as a key-value store or a hash

[27]The 2021 NYU Deep Learning course by LeCun and Canziani [11] and the 2023 GPT from scratch video and repository by Karpathy [76] have proven to be quite phenomenal resources for the understanding of the inner workings of the Transformer.

[28]The number used in the paper is $h = 8$, which as a consequence, due to the proposed formula $d_k = d_v = \frac{d_{model}}{h}$, causes $d_k = d_v = \frac{512}{8} = 64$.

table (a depiction of this kind of soft indexing is presented in figure 2.10). The line in figure 2.9 connecting the input embedding matrix directly to the "Add & Norm" block represents a residual connection [77], also employed in every other sublayer, which helps with optimization issues of deep neural networks. Further, the block in question also implements layer normalization [78] for the same reason. The final block, or sublayer, of the encoder is the "Feed Forward": this is a simple multi-layer perceptron (see figures 2.4 and 2.5), which is implemented as a fully connected layer that maps from input size d_{model} to $4 \times d_{model}$, then applies a non-linearity such as the ReLU (figure 2.6c), and maps back from $4 \times d_{model}$ to the original length d_{model}. The feedforward block allows the Attention head to store higher quality information about the Self-Attention communication that happened in the preceding block.

- Most of the things that happen in the encoder also happen in the decoder, with two notable differences:

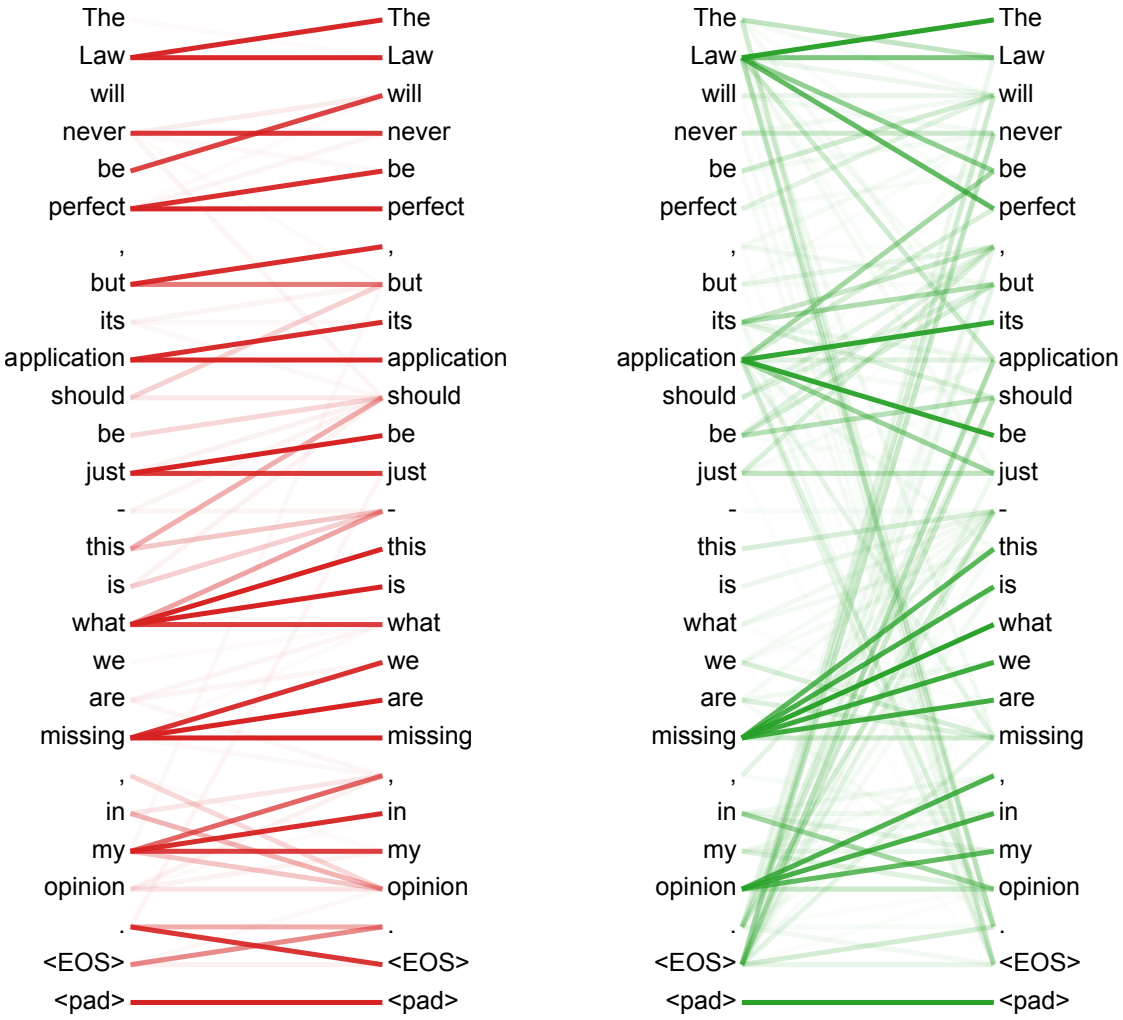
  1. the lower "Multi-Head Attention" block is now specifically "Masked": this means that the future tokens in the sequence are masked by multiplying the E_{dec} embedding matrix with a lower triangular matrix of ones, where its zeros are in turn replaced with $-\infty$, in such a way that illegal (future) tokens are ignored by the $softmax$ operation. This additional step is needed for the decoder to "unroll" the output sequence one token at a time, without being able to reference what cannot possibly already have been written by the model itself;
  2. the "Multi-Head Attention" block (third from the bottom in the decoder frame in figure 2.9) now replaces the Self-Attention operation with Cross Attention. This is the crucial modification that allows the decoder to query the language representations learned by the encoder, which may contain semantic, grammatical, and syntactical information. Equation 2.4 can now be rewritten as:

$$\begin{aligned} head_{i_{dec}} &= Attention\left(Q_{dec}W^{Q}_{i_{dec}}, K_{enc}W^{K}_{i_{dec}}, V_{enc}W^{V}_{i_{dec}}\right) \\ &= Attention\left(E_{dec}W^{Q}_{i_{dec}}, E_{enc}W^{K}_{i_{dec}}, E_{enc}W^{V}_{i_{dec}}\right) \end{aligned} \tag{2.6}$$

     Note the different *enc* and *dec* subscripts for different origins of the queries, keys, and values matrices.

The final "Linear" and "Softmax" layers in figure 2.9 are used to transform the embeddings back into token probabilities, from which the best one is selected to continue the sequence generation conditioned on both the encoder's knowledge and the decoder. As an example, the Transformer's authors apply the original model to the 2014 WMT English-German dataset [79]: while the encoder provides the

representations for the full German sentence “Wir wissen nicht, was passiert.”, the decoder generates each subword unit token for the English sentence “We do not know what is happening.” one at a time, conditioning itself with its own output during generation (“We”, then “We do”, then “We do not” and so on).



(a) One head of a trained Transformer.

(b) Another head of the same Transformer.

Figure 2.10: Figure 5 in the original Transformer paper [1]: «many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.»

2.3 BERT

Even though the Transformer from 2017 [1] was the neural network architecture breakthrough that revolutionized the field of NLP in recent years, it was not until the publication of the BERT paper in 2018 [2] that the adoption of Transformers became widespread. This was due to two main reasons: self-supervised training tasks and transfer learning.

BERT[29], or Bidirectional Encoder Representations from Transformers, is architecturally the same as the Transformer explained in section 2.2 and shown in figure 2.9 («BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani *et al.* [1] and released in the `tensor2tensor` library. Because the use of Transformers has become common and our implementation is almost identical to the original, we will omit an exhaustive background description of the model architecture and refer readers to Vaswani *et al.* [1] as well as excellent guides such as "The Annotated Transformer" [80]» [2]); the innovation proposed in the paper by Devlin *et al.* [2] consists, instead, of the training framework for the Transformer.

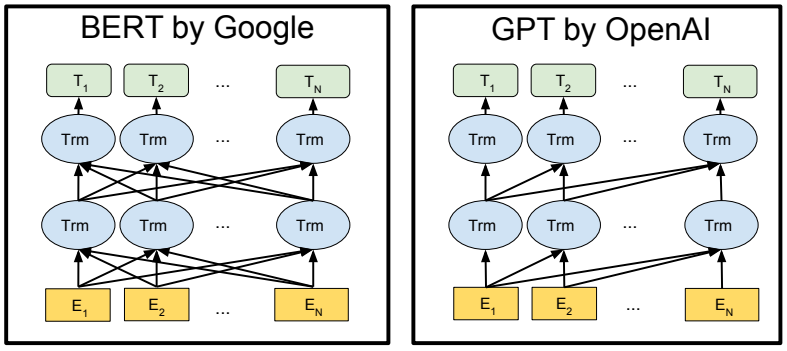


Figure 2.11: A diagram highlighting the bidirectional Transformer architecture of BERT [2] versus the left-to-right GPT [13]. BERT's representations are conditioned on left and right context in every layer. From figure 3 in [2].

It had already been demonstrated in 2014 [81], especially in the field of computer vision, how pretraining a network (*e.g.* a CNN) on a big dataset such as ImageNet [82] and then finetuning it for a downstream task (*e.g.* image classification, semantic

[29]There has been a fun tradition to call novel NLP models or frameworks with an acronym that forms the name of a Muppet from Sesame Street in recent years. Examples are ELMo [12], BERT [2], and BigBird [20].

segmentation) was an «effective recipe» [2]. While other researchers had previously attempted to apply the same approach to NLP models with varying degrees of success, the most notable in 2018 being OpenAI's GPT[30] [13], BERT's authors argue that a left-to-right pretraining procedure, where the token at the ith step t_i can only attend to the tokens in the preceding sequence $\{t_0, ..., t_{i-1}\}$, is not optimal for sentence- and token-level tasks. In its place, they propose a combined masked language modeling (MLM) and next sentence prediction (NSP) pretraining task, shown in figure 2.12.

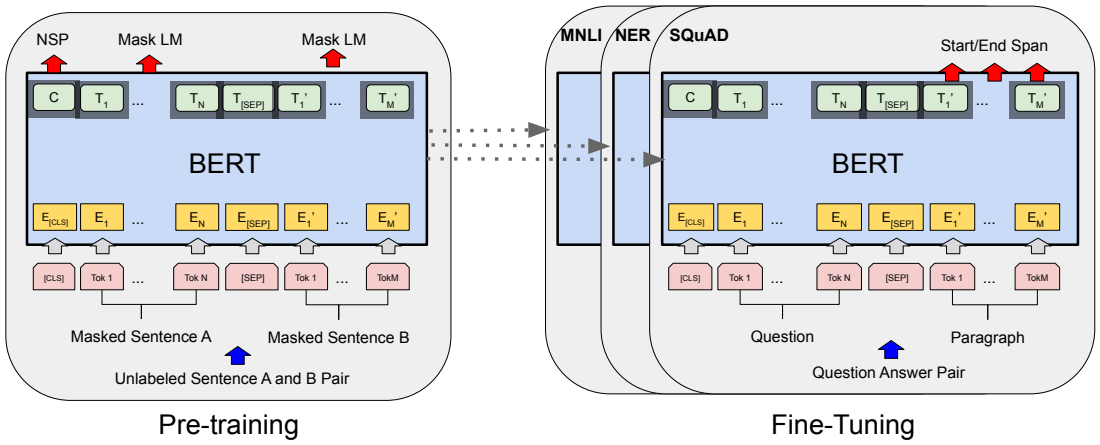


Figure 2.12: «Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (*e.g.* separating questions/answers).» Figure 1 in [2].

Here is a breakdown of the two pretraining tasks employed by BERT:

MLM Masked language modeling is Devlin's *et al.* technique to pretrain a language model's internal linguistic representations bidirectionally. It is inspired by the Cloze procedure, proposed by Taylor in 1953 [83], where the objective is to correctly guess a masked word given the surrounding context of a sentence (*e.g.* fill `[MASK]` in the sentence: "The `[MASK]` is chasing the mouse in the kitchen." The context to the left of the `[MASK]` token is almost useless for an accurate prediction, while being able to attend to the context to its right we can predict that the masked token is most likely "cat"). MLM is similar in

[30]GPT corresponds to the decoder part of the Transformer, while BERT corresponds to the encoder. Reference in figure 2.9.

principle to the denoising autoencoder [84], but instead of reconstructing the entire input its objective is to only predict the missing token.

NSP Next sentence prediction is used to jointly pretrain text-pair representations, such as a question followed by an answer or a name and its description. This was added because «many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling» [2]. To include this information in the training data, the segment to which the token belongs is added to its embedding vector, as can be seen in figure 2.13.

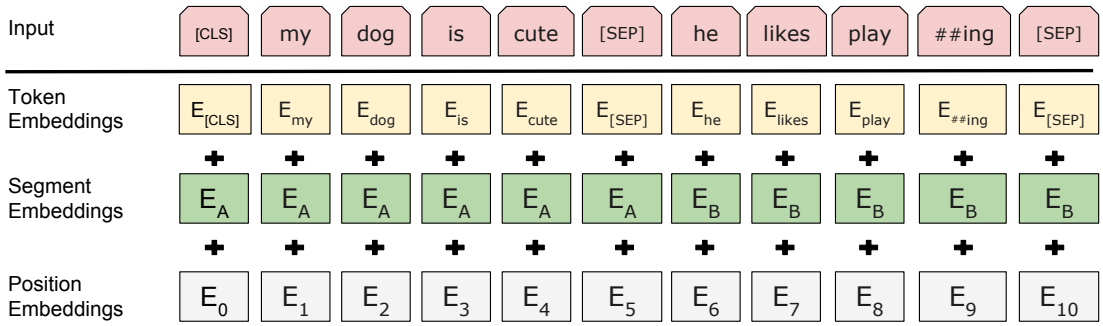


Figure 2.13: BERT's input representation. It is shown that, on top of the now familiar token and position embeddings (see § 2.2), BERT makes use of additional segment embeddings to encode information useful for its NSP pretraining task. Figure 2 in [2].

All in all, what the release of BERT caused has been a revolution in the ease of adoption of an NLP model for specific tasks. Platforms and libraries such as `https://huggingface.co` and HuggingFace Transformers [85] have popularized the application of Transformers to many downstream tasks, *e.g.* sentiment analysis, translation, summarization, question answering, conversation, etc.
This has been in large part due to the effective pretraining framework proposed by Devlin *et al.* [2] and by the release of language model checkpoints pretrained on large corpora (*e.g.* Wikipedia [3], BookCorpus [86]), whose learned language representations can then easily be transferred to a task-specific model without the need for an expensive training procedure from scratch.

Chapter 3

Related Works

In this chapter we are going to explore all the relevant previous works on which this master's thesis is based on. These can be thought of as falling under two main categories: models (§ 3.1), which include novel neural network architectures and underlying mechanisms, and datasets (§ 3.2), which either propose a new task or aim to put in the hands of researchers a more complete benchmark to test their models with than the pre-existing ones on an established task. Finally, we discuss some previous works in the specific field of video lecture summarization (§ 3.3).

Name	Type	Usage
BART [10]	Model	BART is the foundational model used throughout this work.
Longformer [6]	Long Attention	The Longformer efficient long attention mechanism will be applied to BART to allow us to work with very long input sequences.
LSG [7]	Long Attention	The LSG long attention mechanism is an alternative to the above.
X-CLIP [4]	Model	X-CLIP is a previously trained model that we will use to introduce information from the video modality into the transcripts of our novel lecture transcript-summary dataset, UniSum (§ 4.2.2).
Kinetics [87]	Dataset	The Kinetics dataset was used for the training of X-CLIP by its authors.

Table 3.1: A brief explanation of each of the components mentioned in this chapter. Their usage will be thoroughly illustrated in chapter 4.

3.1 Models

3.1.1 Summarization Models

Summarization is the task of reducing a long sequence to its core elements. It can be interpreted as a form of deliberately lossy compression, as, per its goal, it does not claim to maintain all the initial information.

While video summarization is the branch of automatic summarization that aims to extract key-frames or key-fragments from videos [88], [89] (*e.g.* YouTube's auto-generated thumbnails after uploading a video), in this thesis we will focus on automatic text summarization[1]. There are two kinds of automatic summarization:

- extractive summarization is the simpler one, where the summary does not include any rephrased part of the input text. It can be reduced to a sentence or sub-sentence tokenization[2] step followed by a binary classification into two "important" and "unimportant" classes for each token. At the end, tokens classified as "important" are merged together to form the final summary;
- abstractive summarization is the more complex kind of automatic summarization, which involves understanding the whole input text, implicitly categorizing what are the important notions (instead of tokens), and then reassembling them into the final summary in an original form.

Many automatic summarization techniques have been devised over the years [90] since interest for the field sprung in the 1950s [91]. What we will focus on in this section will be the most recent developments in deep-learning-oriented methods. The three most prominent Transformer-based models at the time of writing are T5[3] [17], PEGASUS[4] [18], and BART[5] [10].

T5 is not a summarization-specific model: it is a unified text-to-text model that can perform multiple tasks given a specific pre-prompt (see figure 3.1), among which there is also input text summarization. The authors specify that their «goal is not to propose new methods but instead to provide a comprehensive perspective

[1]For the sake of brevity, the term "automatic summarization" will always refer to "automatic text summarization" from this point onwards.

[2]Read: split. We use the word "token" since it may indicate any amount of text: a letter, a syllable or sub-word, a word, a clause or sub-sentence, a sentence, etc.

[3]Text-To-Text Transfer Transformer.

[4]Pre-training with Extracted Gap-sentences for Abstractive Summarization.

[5]Bidirectional and Auto-Regressive Transformers.

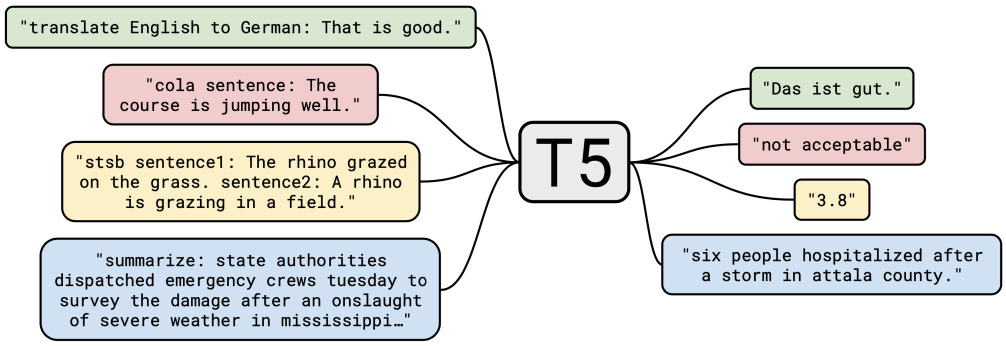


Figure 3.1: A diagram showing T5's ability to perform different text-to-text tasks without a specific finetuning for each downstream task. Figure 1 in [17].

on where the field stands» [17]. Nonetheless: «on CNN/Daily Mail [92] we attain state-of-the-art performance, though only by a significant amount on the ROUGE-2-F score» [17], which means that in some aspects T5 achieved SOTA summarization results for 2019. T5 also showed the importance of scaling a Transformer-based model to a large number of parameters to improve performance (the authors scale up to 11B parameters, but present-day LLMs[6] much surpass this threshold, with *e.g.* BLOOM[7] reaching 176B internal parameters [93]).

PEGASUS and BART introduce, instead, new pretraining objectives for the Transformer architecture [1] seen in section 2.2, both expanding upon the masked language modeling technique proposed for the training of BERT [2].

PEGASUS is specifically trained for abstractive summarization applications and uses a Gap Sentences Generation (or GSG for short) pretraining objective. The idea is similar to MLM, but instead of masking tokens, it is implemented by masking entire sentences. The authors find a positive correlation between masking «putatively important sentences» [18], instead of randomly selected ones, and an increase in performance. These important gap sentences, that are what PEGASUS has to predict during its training phase, are selected by ranking the document's sentences by decreasing ROUGE1-F1 (see § 5.1.1 for an explanation of the metric) with the rest of the document.
PEGASUS was trained on the C4[8] [17] and the HugeNews (currently unreleased by Google) datasets, and in 2019 achieved «state-of-the-art results on all 12 diverse

[6]Large Language Models.

[7]BigScience Large Open-science Open-access Multilingual Language Model.

[8]Colossal Clean Crawled Corpus.

downstream datasets considered» [18], which include: XSum [94], CNN/Daily Mail [92], NEWSROOM [95], Multi-News [96], Gigaword [97], [98], WikiHow [99], Reddit TIFU [100], BIGPATENT [101], arXiv [102], PubMed [103], AESLC [104], and BillSum [105].
Google is currently using PEGASUS as the underlying model for some of its products: in 2022, they published a blog post [106] where they explained how they offer automatic document summarization in Google Docs. As a note, to optimize inference times, they replaced PEGASUS's decoder with an RNN [107] without performance loss.

BART [10] takes a different approach to tackle the same issue, with a twist: the authors use the (well-tried in CV) technique of the denoising autoencoder, thus the model instead of having to predict a part of the original input that has been masked, it has to reconstruct it entirely. This pretraining objective is seen by Lewis *et al.* as a generalization of BERT's MLM plus NSP objective, and even though they did not intend originally to specifically address the summarization downstream task, they show that «BART is particularly effective when finetuned for text generation but also works well for comprehension tasks. It matches the performance of RoBERTa [14] with comparable training resources on GLUE[9] [108] and SQuAD[10] [109], and achieves new state-of-the-art results on a range of abstractive dialogue, question answering, and summarization tasks. For example, it improves performance by 6 ROUGE over previous work on XSum [94]» [10].

BART's architecture only slightly differs from BERT's: instead of using ReLU [61], BART's authors use its slight variant GELU [62] (see figure 2.6d), and they remove the feed-forward network from the encoder.

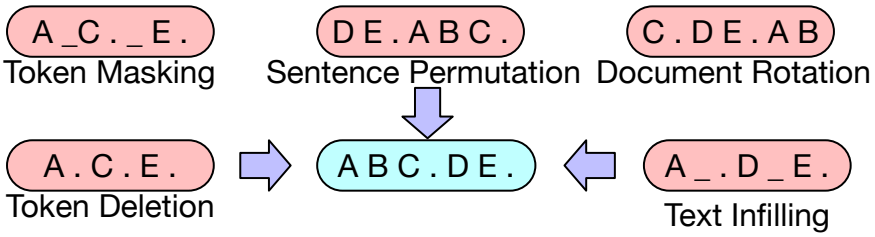


Figure 3.2: The five composable transformations that are used to add noise to text fed to BART during pretraining. Dots "." denote the end of a sentence, underscores "_" indicate the insertion of a `[MASK]` token. Figure 2 in [10].

[9]General Language Understanding Evaluation benchmark.

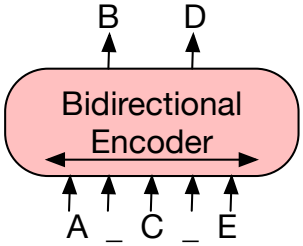
[10]Stanford Question Answering Dataset.

As is shown in figure 3.2, the text submitted to BART during pretraining can be transformed in different ways:

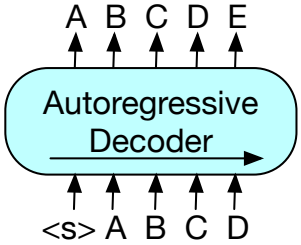
1. token masking is the same as MLM used in BERT [2];
2. token deletion involves completely removing a token without it leaving any trace in the input text;
3. sentence permutation is the random shuffling of the sentences that compose the input document;
4. document rotation is the random choice of a token with which the input text will begin, "rotating" the document preserving the relative positions of the tokens;
5. text infilling is the most interesting transformation: a number is extracted from a Poisson distribution with $\lambda = 3$; this (which could also be 0) becomes the number of tokens that will be replaced with a single `[MASK]` token. This was inspired by SpanBERT [110], though this approach applies a heavier transformation.

These noising functions teach BART different aspects about the input reconstruction process, such as which positions have a missing token, how many tokens are missing from a span, and where is the start of the document [10].

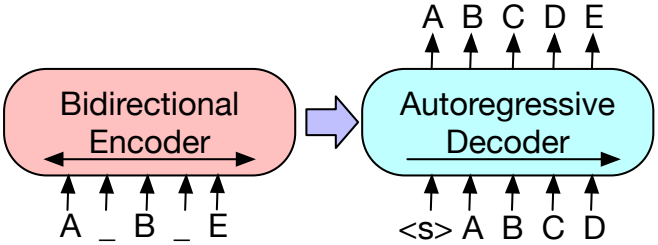
Lewis *et al.* provided the test results for BART on different tasks, among which we focus on summarization with the CNN/Daily Mail [92] and XSum [94] datasets. Even though the CNN/Daily Mail dataset contains summaries that often resemble sentences picked from the story, making it a dataset prone to give good results to extractive models, the authors report that BART outperforms all existing works. XSum is, instead, a highly abstractive dataset, and on it BART improves on the previous top works by ~6 ROUGE points on all metrics, maintaining high output quality. We note that PEGASUS_{LARGE} trained on the HugeNews dataset outperforms BART_{LARGE} by up to 2.0 points in different ROUGE metrics, when testing on both the XSum and the CNN/Daily Mail datasets.



(a) With BERT, tokens are randomly replaced with masks and the input sequence is encoded bidirectionally. The masked tokens are predicted independently of each other, which does not allow for text generation.



(b) With GPT, the tokens are predicted in an autoregressive fashion, which works well for sequence generation. Its main limitation is the left-to-right Transformer directionality, which precludes GPT from learning bidirectional interactions.



(c) With BART, the input document is subjected to arbitrary noise transformations and fed into a bidirectional encoder (on the left), then the likelihood of the original input sequence is computed with the autoregressive decoder (on the right). To finetune BART, an uncorrupted input is given to both encoder and decoder, then the representations from the final hidden state of the decoder are used.

Figure 3.3: A comparison of the different pretraining objectives of BERT, GPT, and BART and how they influence their applications. Figure 1 in [10].

3.1.2 Long-sequence Attention

As we have seen in section 2.1, the RNN had the advantage of theoretically being able to process up to infinite-length input sequences, though with many difficulties, such as the vanishing and exploding gradients problem [67], that were progressively solved during the years, with the LSTM [69], the GRU [72], and other clever modifications of the original architecture. But when the Transformer was published in 2017 [1], the performance of NLP tasks leapt forward in a way that RNNs could not compete with. This came at the cost, anticipated in section 2.2, of alleviating the constraint of indefinite-length input sequences. This was not only due to the internals of the Transformer's encoder and decoder, which had been designed to work with matrices of predefined dimensions from the start, but also to the implementation of the Self-Attention mechanism (see equation 2.4), which relies on the assumption that each token can attend to every other token in the sequence, in such a way that having an N-tokens-long input sequence, the Self-Attention operation will need to work on a matrix of dimensions $N \times N$. This made the memory dependence quadratic, or $\mathcal{O}(N^2)$ (shown by the blue line in figure 3.4), and as a consequence made the maximum input lengths of popular Transformer-based models, such as BERT, GPT, and BART quite limited[11] on current hardware[12].

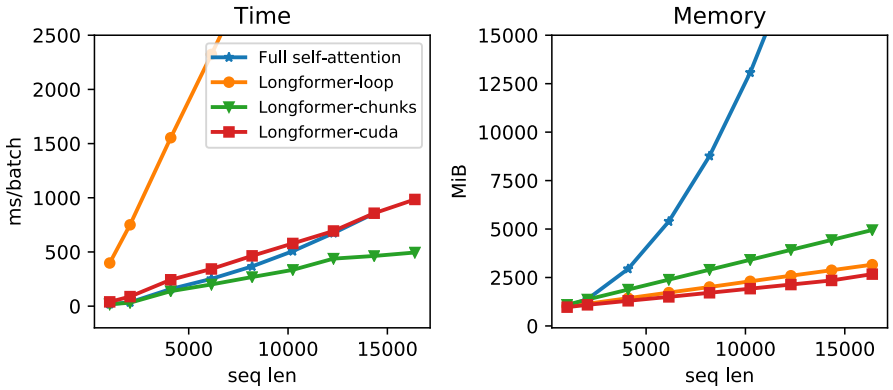


Figure 3.4: Runtime and memory usage dependence from the input sequence length in tokens. Comparison of full Self-Attention with respect to different implementations of the Longformer attention. Figure 1 in [6].

For this reason, in recent years, many articles have been published on research

[11]BERT has a default maximum input length of 512 tokens; GPT-2 and BART have a default maximum input length of 1,024 tokens.

[12]Though they can be clustered for more distributed compute and memory, single NVIDIA GPUs currently integrate from 8 to 80 GB of on-board memory [111].

to efficiently extend these limits. In 2020, Beltagy *et al.* published Longformer [6], and Zaheer *et al.* released BigBird [20]. In 2022 Condevaux and Harispe proposed a similar kind of long-sequence attention to apply to many existing models, LSG [7], and Phang *et al.* looked into extending the original PEGASUS model for long sequences [112].
All these slightly different approaches (see figure 3.5) share the underlying idea of a sparse full Self-Attention matrix with a sliding window and global tokens. The sliding window improves the memory dependence of an N-tokens-long input sequence from $\mathcal{O}(N^2)$ (quadratic) to $\mathcal{O}(N)$ (linear), and is justified by the data locality principle [113]: in essence, the most important information for a token is contained in its surrounding context. On the other hand, global tokens favor the flow of information between all tokens, which is beneficial for long-range dependencies (*e.g.* the conclusion of the input sequence references the introduction).

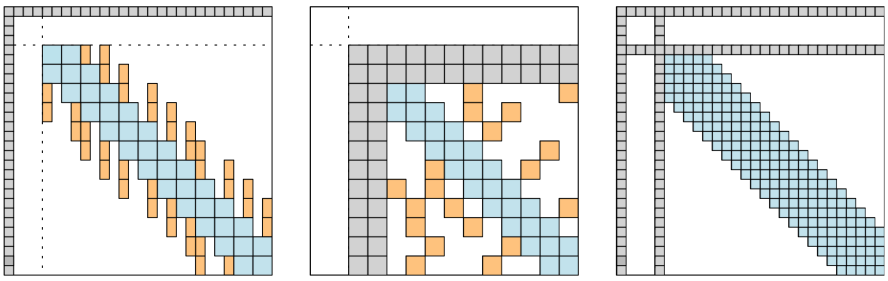


Figure 3.5: A comparison of different long attention patterns. From left to right: LSG [7], BigBird [20], and Longformer [6]. Figure 1 in [7].

Here is a list of peculiarities for each of the aforementioned models:

Longformer Beltagy's *et al.* main proposition was an encoder-only RoBERTa-based[13] [14] Transformer that, with some MLM pretraining continued from a RoBERTa checkpoint, could improve the SOTA results on the WikiHop [114] and the TriviaQA [115] datasets. But they also proposed an encoder-decoder variant, known as LED[14], that was instead based on BART [10], and without additional pretraining or finetuning, marginally improved the SOTA (BigBird at the time) results on the arXiv summarization dataset [102]. This was an

[13]Longformer and LED models based on a previously trained model with maximum input length $L = 1024$ means that the authors increased L to 16,384 and copied the position embedding matrix 16 times into the new matrix [6].

[14]Longformer-Encoder-Decoder.

important piece of the puzzle when coming up with the methodology for the present thesis.

BigBird Zaheer's *et al.* work, other than including some theoretical proofs of the reasons why sparse Self-Attention, when replacing full Self-Attention, works in long contexts, is very similar to Beltagy's *et al.* paper [6], with the exception of the additional use of random token attention on top of sliding window and global attention. The most significant difference is that the base model for BigBird for seq2seq tasks like summarization was Google's own PEGASUS [18]. With this extension, the authors achieved SOTA results on three summarization datasets: arXiv [102], PubMed [103], and BIGPATENT [101].

LSG Attention The contribution by Condevaux and Harispe is twofold: first, they propose a different approach with respect to BigBird's random attention that they call "sparse"; second, they publish an open source Python library[15] to convert existing checkpoints of many popular models with their LSG attention. There are several ways to compute the tokens in the sparse matrix that each attention head independently considers; an example is "max norm", which is their library's default. Its objective is to select the most highly rated key tokens in the score matrix by computing the alignment between the key and query tokens $\boldsymbol{q}, \boldsymbol{k} \in \mathbb{R}^d$ with $\boldsymbol{q}\boldsymbol{k}^T = \cos(\theta)\,||\boldsymbol{q}||\,||\boldsymbol{k}||$, where the sign of $\cos(\theta)$ is unknown, but the authors write that in most situations the correct key should dominate the softmax regardless of the query. The key tokens are thus chosen inside each sequence block and each attention head.

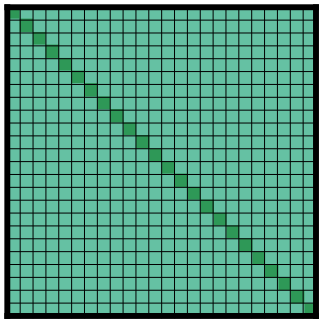
PEGASUS-X The work of Phang *et al.* focuses on the study of the possible variations on a Transformer-based summarization model that accepts long input sequences. They show that a global-local architecture (similar to the work above) with block staggering, a large number of global tokens, and a large block size gets the best results, along with a relatively brief pretraining continuation of the elongated model before its finetuning. Their large variant of the PEGASUS-X model achieves SOTA results on the BIGPATENT [101] and GovReport [116] datasets.

Along these and other long-input Transformer models, some benchmarks to reliably compare them have been proposed. Long-Range Arena[16] (by Tay *et al.* [117]) offers a unified benchmark for evaluating different aspects of Transformers

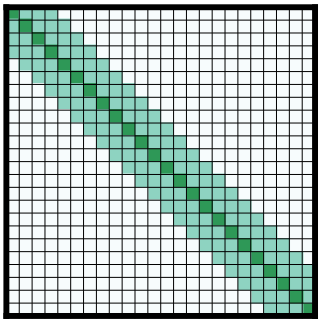
[15]Available at `https://github.com/ccdv-ai/convert_checkpoint_to_lsg`.

[16]Benchmark available at `https://github.com/google-research/long-range-arena`.

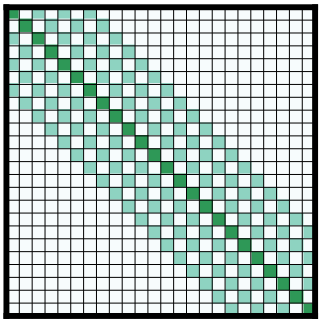
with input sequences ranging from 1,024 to 16,384 tokens. SCROLLS[17] [118] is an answer to Long-Range Arena that is entirely NLP-oriented.



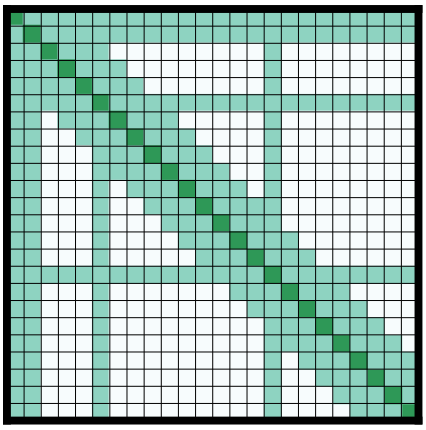
(a) Full Self-Attention. Every token attends to every other token in the embeddings matrix.



(b) Sliding window attention. If the sliding window's width is w, every ith token attends to the tokens in the sliding window: $\{t_{i-\frac{w}{2}}, ..., t_i, ..., t_{i+\frac{w}{2}}\}$.



(c) Dilated sliding window attention. Same as sliding window attention, but an additional dilation factor is considered when computing the token indices in the window.



(d) The complete attention pattern proposed in the Longformer paper [6]. Each token attends to the other tokens in the sliding window, plus some "global" tokens can attend to every other one like in full Self-Attention.

Figure 3.6: A comparison of the different composing subpatterns of the Longformer attention mechanism. Figure 2 in [6].

[17]Standardized CompaRison Over Long Language Sequences. Benchmark available at `https://www.scrolls-benchmark.com`.

3.1.3 X-CLIP

X-CLIP[18] [4] is a Transformer-based video-language model that builds upon CLIP [26], a previously released model by OpenAI that learns joint visual-textual representations.

X-CLIP's innovation is threefold: first, instead of training on a closed-set of possible textual labels that correspond to a set of video clips (like previous work such as Swin [119] and ViViT [120]), it uses the joint internal language-image representations learned with a method based on CLIP; second, it does so with much less computation (the authors report 12× fewer FLOPS); third, it improves the zero-shot[19] video recognition SOTA results on the Kinetics-400 dataset [87] and obtains improved few-shot[20] performance when the amount of labeled data is extremely limited.

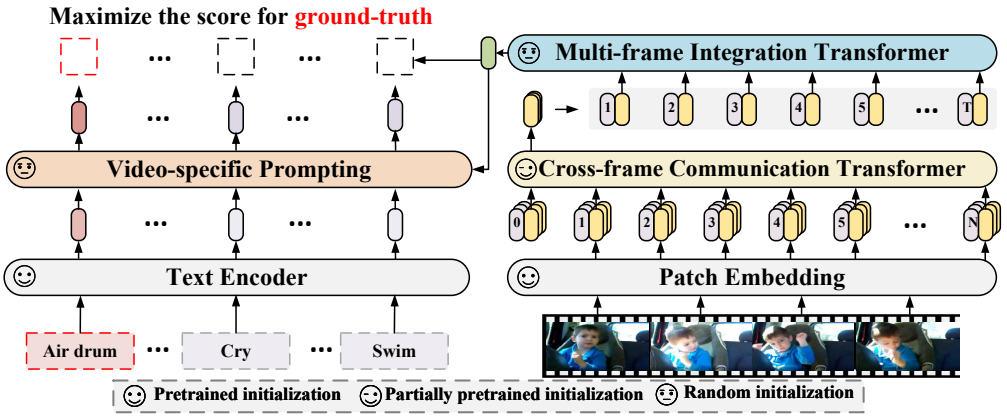


Figure 3.7: The X-CLIP framework, which allows for freely chosen natural language text labels and jointly processes them with a video clip using a cross-frame communication Transformer in combination with a multi-frame integration Transformer. Figure 2 in [4].

The main application of X-CLIP is video classification on natural language textual classes. We will explain how this has been used in the present work in chapter 4.

[18] eXpanded Contrastive Language-Image Pretraining.

[19] Zero-shot learning is a type of ML problem where the classification model at test time has to predict classes unseen during training.

[20] Similar to zero-shot learning, with the difference that the model has access to a few examples per class during training.

3.2 Datasets

The work contained in this thesis, which will be explored in detail in chapter 4, makes use of two pre-existing datasets. A brief overview of these is set out below.

3.2.1 VT-SSum

VT-SSum[21] [5] is an extractive text summarization dataset that focuses on spoken language, with data extracted from `http://videolectures.net`, «which includes 125K transcript-summary pairs from 9,616 videos» [5].

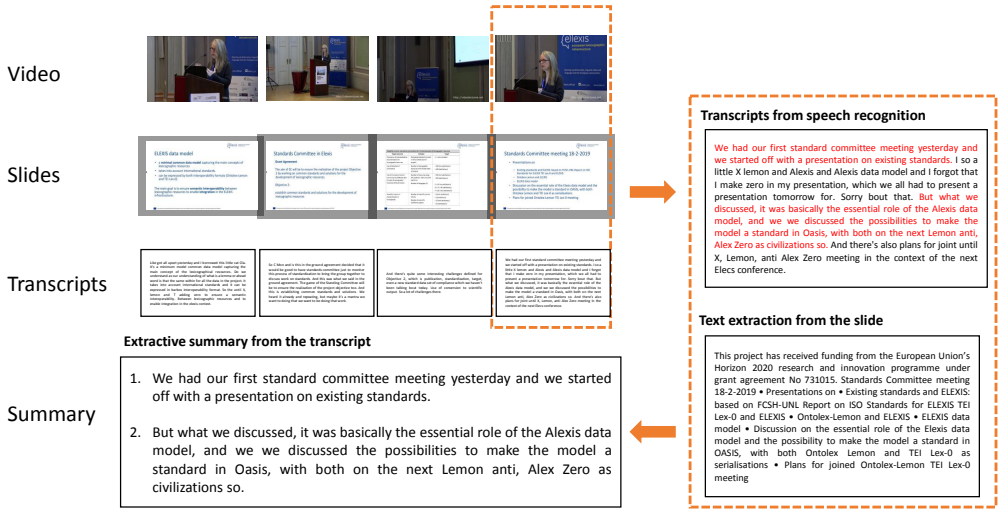


Figure 3.8: An illustration of the process VT-SSum's [5] authors used to extract the summaries from the full transcripts with weak supervision from the corresponding slide. Figure 1 in [5].

The process to make the VT-SSum dataset follows these steps:

Data Collection The videos and their corresponding slides are downloaded from `http://videolectures.net`. This platform also provides the timestamps of start and end of each slide's projection.

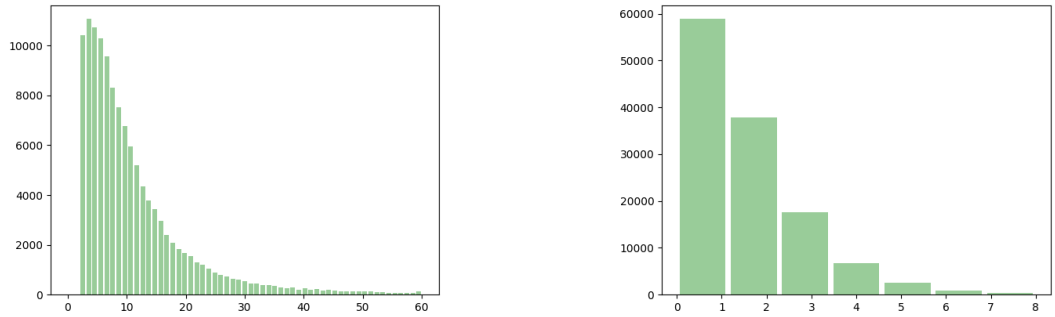
Transcript Extraction The authors extract the textual transcript from the audio track of the video lecture with Microsoft Speech to Text.

[21] Video Transcript Segmentation and Summarization. Dataset available at `https://github.com/Dod-o/VT-SSum`.

Slide-Text Block Alignment The full transcript can then be separated into blocks of text spoken during the projection of a slide based on the aforementioned slide timestamps.

Transcript Segmentation Each block of text is segmented into single sentences, which are then binarily classified into "sentences to consider for the summary" and its opposite class by computing if the ROUGE1-F1 score between the sentence and the contents of the slide crosses a fixed threshold. The concatenation of these selected sentences is considered the gold summary.

The distribution of the data contained in VT-SSum is shown in figure 3.9. The VT-SSum dataset is under the Creative Commons Attribution-NonCommercial-NoDerivatives International 4.0 license.



(a) **(b)**

Figure 3.9: The lengths distributions of the transcripts (3.9a) and the summaries (3.9b). The x-axis denotes the number of sentences and the y-axis denotes the number of samples. Figure 2 in [5].

3.2.2 Kinetics

Google DeepMind offers «a collection of large-scale, high-quality datasets[22] of URL links of up to 650,000 video clips that cover 400/600/700 human action classes, depending on the dataset version. The videos include human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging. Each action class has at least 400/600/700 video clips. Each clip is human annotated with a single action class and lasts around 10 seconds.»

Of the three datasets (Kinetics-400 [87], Kinetics-600 [121], Kinetics-700 [122] then updated in 2020 [123]), we especially mention Kinetics-400 and Kinetics-600 for their benchmark use in the X-CLIP paper [4] (see § 3.1.3), which we will see applied in the scope of this thesis in chapter 4.

(a) "dunking basketball" **(b)** "playing violin"

(c) "robot dancing" **(d)** "riding a bike"

Figure 3.10: A few video clip examples from the Kinetics-400 dataset. Each figure's caption corresponds to the video clip label. From figure 1 in [87].

The Kinetics datasets are released under the Creative Commons Attribution 4.0 International license.

[22] Official website at `https://www.deepmind.com/open-source/kinetics`.

3.3 Video Lecture Summarization

Before proceeding onto our proposed methodology, it is appropriate to review some of the previous works in our particular chosen field, which is that of automatic summarization of educational content, and, more specifically, of video lectures.

Name	Type	Description
BERT [2]	Model	BERT's embeddings can be clustered and used for extractive summarization [124].
PreSumm [125]	Model	PreSumm is a set of BERT-based extractive and abstractive summarization models.
VT-SSum [5]	Dataset	The VT-SSum dataset is a part of our novel long lectures transcripts dataset, OpenULTD (§ 4.2.1).
Lecture2Note [126]	Framework	Lecture2Note is an unreleased note-generation tool that transforms slide-based video lectures into schematic notes.
Lecture2Notes [127]	Framework	Lecture2Notes is an open source service that summarizes entire video lecture transcripts. It also provides a slide-by-slide summary.

Table 3.2: A brief overview of related works in the field of video lecture summarization.

Video lectures contain two modalities, namely video and audio, from which other modalities can be derived, such as textual, where the transcript can be generated from the audio by an ASR model or a human, and visual, where the main figures, diagrams, formulas, etc. can be extrapolated from the video in different ways. As such, many different approaches have been taken during the years by various researchers; of these many approaches we focus on the ones leveraging the Transformer neural architecture, since that is what allows to reach state-of-the-art results at the time of writing.
Though a notable mention must be given to Xu *et al.* [126], who proposed a method to provide students with notes-like documents containing images and text extracted from a slide-based video lecture (see figure 3.11), we need to narrow our focus down to purely textual summarization for a relevant and fair comparison with our work.

Most of the available approaches to summarization of long spoken text follow the extractive pattern. We have seen in section 3.2.1 that VT-SSum's [5] authors proposed a weak supervision method based on the slide contents to extract what

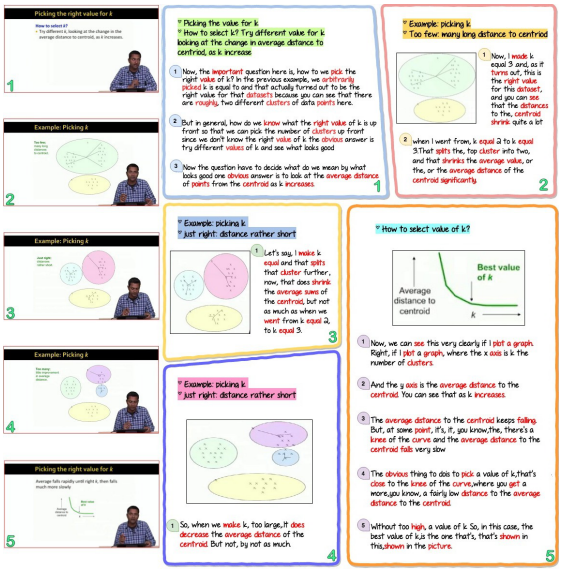
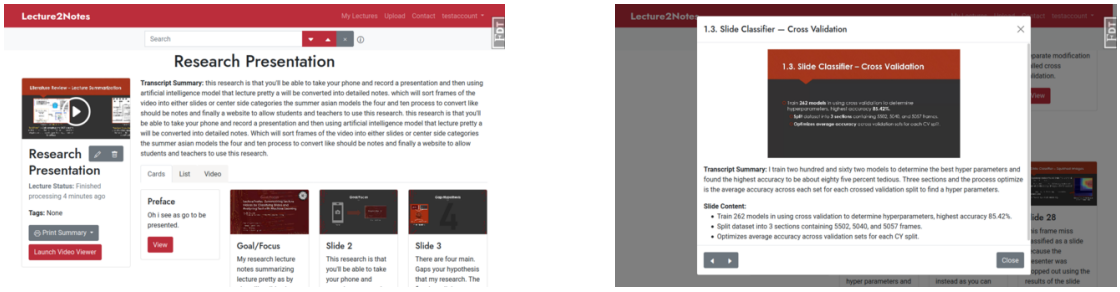


Figure 3.11: The neatly formatted notes output by the Lecture2Note system. Figure 3 in [126].

they consider the most relevant sentences of the speech. To give a numerical baseline, they employ a variant of PreSumm [125], which is a set of extractive and abstractive BERT-based models for summarization.
A previous work by Miller [124] used the embeddings produced by the original pretrained BERT directly to perform the selection of the most significant sentences via K-Means clustering [128].

The first of the two works most similar to the present master's thesis, mainly due to the use of abstractive summarization applied to educational videos, is the one by Housen [127], who proposed and deployed an end-to-end system to generate a textual summarization of the speech corresponding to each projected slide (see figure 3.12) and to the whole lecture. At the core of the summarization system, he used, in part like us, a BART model extended with the long attention mechanism proposed with Longformer [6]. His research was more integration-oriented, and from his paper it does not appear he concentrated on the specific aspect of summarization, though he released the TransformerSum library[23], but instead he presented a ready-to-use system accessible through a dedicated working website, with many underlying components made to fit together.

[23]Available at `https://github.com/HHousen/TransformerSum`.



(a) **(b)**

Figure 3.12: The summary of the complete transcript (3.12a) and of the transcript corresponding to a single slide (3.12b) made with the Lecture2Notes system. From figure 5 in [127].

The second work akin to the current one was made by Benedetto *et al.* at PoliTo in 2021 [129]. Their contributions were a novel transcript-summary dataset based on courses offered at MIT called EduSum (which comprises 7 courses; we propose a ~10× larger educational summarization dataset, UniSum § 4.2.2, of 67 courses), a method for the restoration of missing punctuation signs applicable to transcripts generated by an ASR model, and an extractive-abstractive approach to long input sequence summarization, which does not make use of linear memory dependent attention mechanisms.

As a final note, it is key that we investigate the effect that applying our proposed methods in schools and universities would have on the students' preparation. Our aim is, hopefully, to put in the hands of the students a tool to facilitate their study and revision of course materials, with two upsides with respect to a lecture summary made by a human, be they the professor or a teaching assistant: first, the process can be automated, which should grant more time to the human to dedicate to other work; second, the summary content is derived directly from what was said during the lecture, in a way that allows the most relevant topics to emerge in the summary.
It has been demonstrated in a recent study [130], where 58 graduate students of health informatics were asked to classify annotated bibliographies (which require expertise in a certain field to produce) between AI-made and human-made, that AI-generated content in an educational context can yield human-level-quality summaries. This can be interpreted both as a sign of the fact that we can already apply these deep models to real world tasks, and, as the authors note, that the task of summarization is, while complex, not high-level enough to judge a student's creative abilities since an AI might have been used for cheating.

Given the scarcity of previous works in this specific field, we conclude that ours would be a novel contribution.

Chapter 4

Methodology

Now that we have a general understanding of all the underlying technology, including some historical background, it is time to present the purpose that the present master's thesis intends to achieve.
We will first cover the broad task and its main challenges (§ 4.1), then we will propose two novel datasets especially designed to solve said task (§ 4.2), and finally we will see some of the techniques that can be adopted to train a Transformer-based deep network on these datasets (§ 4.4).

4.1 Strategy

With the present work, we want to explore the automatic summarization of university video lectures. This task presents three main obstacles to overcome:

1. we have seen in section 2.3 that current successful language models follow the pretraining-finetuning transfer learning pattern to achieve downstream task flexibility and reusability of model checkpoints pretrained with general language tasks, such as MLM [2], GSG [18], and denoising [10]. Most of the corpora these models are trained on only contain written text; two examples would be Wikipedia [3] and BookCorpus [86]. But video lectures are given by professors who, as eloquent as they can be, are not speaking like one would write a book; their speeches contain repetitions, anacolutha, conversations with students, interruptions, etc. The first challenge is therefore that, while language models are pretrained on datasets in the written text domain, we need to apply them on text from another domain, which is that of spoken text (*i.e.* transcripts);

2. a typical university lecture tends to last from one to two hours. This translates to a large quantity of information that the professor is able to convey during

that time, which in turn, when the speech is converted into text, makes for a very long transcript. As we have discussed in depth in sections 2.2 and 3.1.2, a long input sequence requires some special handling when fed to a Transformer-based model;

3. finally, as a bonus challenge that is not dictated by the constraints of the data, but by our need to attempt to use as much information from the original videos as possible, we try to inject multimodal information into the lecture transcripts.

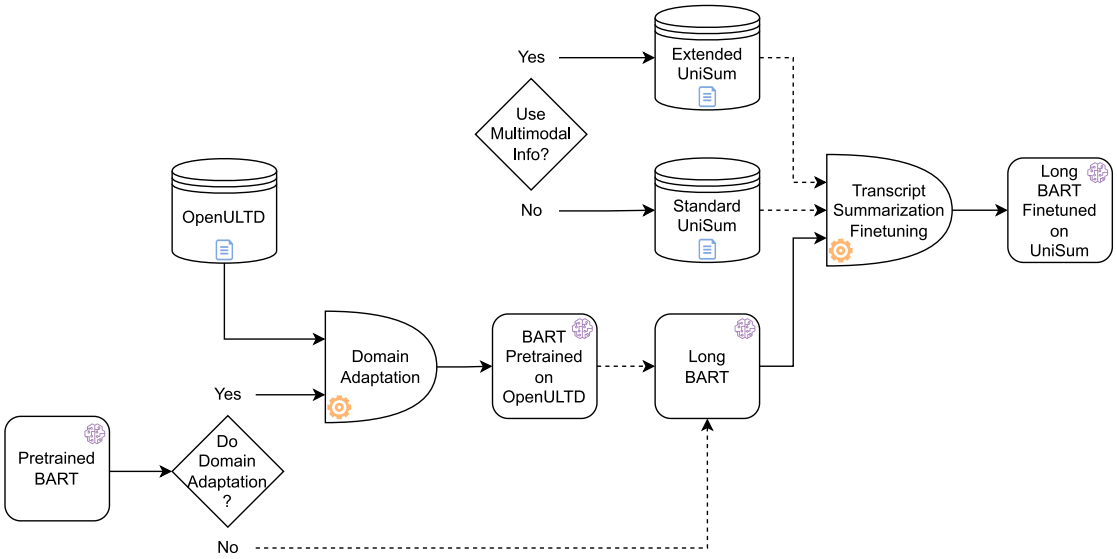


Figure 4.1: A diagram of the training process used to produce the models presented in this work. First we perform BART's [10] domain adaptation on OpenULTD (§ 4.2.1), then we convert BART with long attention to either LED [6] or LSG [7], and finally we finetune the resulting model on either the standard or the extended variant of the UniSum dataset (§ 4.2.2).

The strategy we use to solve all of these problems is the following:

1. to account for the difference in domain between the written text datasets used for model pretraining and the spoken text data used for model finetuning on the summarization task, we continue the pretraining of the model in this new domain, effectively performing the step of domain adaptation [131]. To accomplish this, we propose a novel dataset of 14,677 unlabeled university lecture transcripts, illustrated in section 4.2.1;

2. then, to enable the model to handle very long input sequences (up to 16,384

tokens), we extend the original attention mechanism of the model in question with quadratic memory dependency with a global plus sliding window mechanism (see § 3.1.2 for different approaches), which has a linear memory dependency. To finetune the resulting model on the summarization task, we propose a novel dataset of 1,583 lecture transcript-summary pairs, explored in detail in section 4.2.2;

3. to inject additional information into the transcripts of the summarization dataset, we use X-CLIP [4] (see § 3.1.3) to extract a binary classification from the video modality that reflects the moments during which the lecture speaker is writing.

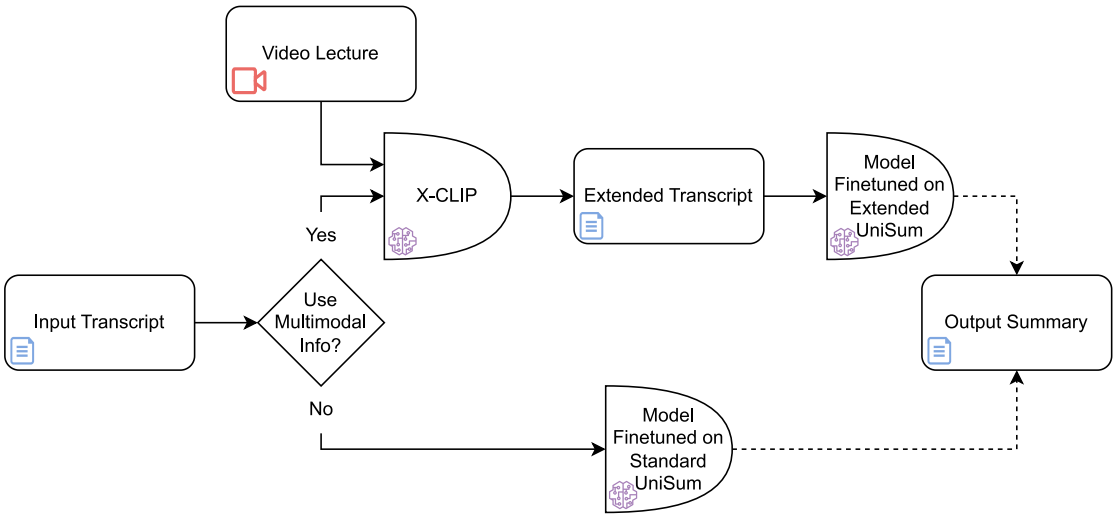


Figure 4.2: A diagram of the inference process proposed in the present thesis.

It is now appropriate to illustrate the proposed datasets.

4.2 Novel Datasets

4.2.1 OpenULTD

OpenULTD[1] is the first of the two novel datasets proposed with the present work. Its name is derived from the fact that it is made of four subdatasets, each from an open learning platform. More specifically, we include an existing dataset, VT-SSum [5] (see § 3.2.1), and we add data from the MIT OpenCourseWare platform[2] (OCW for short), the OpenHPI platform[3], and the Yale Open Courses platform[4], for a total of 14,677 samples.

Subset	Number of Samples	Number of Courses	License
MIT OCW	3,397	126	CC BY-NC-SA
OpenHPI	957	28	CC BY-NC-SA
VT-SSum	9,229	N/A	CC BY-NC-ND
Yale	1,094	43	CC BY-NC-SA

Table 4.1: The data composition of the OpenULTD with respect to its subsets. Note that the number of courses of the VT-SSum subset is not available due to each lecture being standalone (*e.g.* a seminar).

Some information about the composition of the dataset are shown in table 4.1. We note that, while the other three subsets have a license that allows derivative works, the VT-SSum dataset is licensed under the stricter Creative Commons Attribution-NonCommercial-NoDerivatives variant of the license. This means that «if you remix, transform, or build upon the material, you may not distribute the modified material» [132], therefore the published OpenULTD does not contain the VT-SSum subset, but it contains very simple instructions to follow to reconstruct the full dataset.

[1] Open University Lecture Transcripts Dataset.

[2] Available at `https://ocw.mit.edu`.

[3] Available at `https://open.hpi.de`.

[4] Available at `https://oyc.yale.edu`. Videos available at `https://www.youtube.com/@YaleCourses`.

Transcript
Professor Steven Smith: Okay, good morning. I'm going to show another movie today but not until a little bit later in the class. We'll get it. We'll get there. Don't worry! It doesn't fit until the last part of the class. But today, I want to talk about sovereignty. There are two great concepts that come out of Hobbes that you have to remember. One is the state of nature and the other is sovereignty. I spoke a bit about the first one yesterday or Monday rather. Today, I want to talk about Hobbes' theory of the sovereign state, the creation of the sovereign. Hobbes refers to the sovereign as a mortal god, as his answer to the problems of the state of nature, the state, the condition of life being solitary, poor, nasty, brutish and short. And it is only the creation of the sovereign for Hobbes, endowed or possessed with absolute power, that is sufficient to put an end to the condition of perpetual uncertainty, anxiety and unrest that is the case of the natural condition. [...]
Yeah hi, I'm Lucy. I'm in the University of Southampton and together with Pavlos who's also in Southampton and Hardy from the laboratory. Laboratoire Uber Korea. We worked on newer generation of multilingual Wikipedia summaries from wiki data for article placeholders. If you don't know those words, I'll explain them in a. Uh, while we have little content on Wikipedia in different languages, we have a lot of resources in wiki data. Wiki data is a knowledge base, maintain and edited by community of users and we have around 48 million items on wiki data and each item can be labeled and over 400 languages. And as we showed in a previous study, the varieties of languages covered by wiki data is pretty high. Exists, which is the article placeholder. Those article placeholders display triples on Wikipedia and a tabular form and they are dynamically generated, meaning if something up is updated on wiki data, it is automatically updated in Wikipedia as well. [...]
OK. So my name is Kasper. Listen and I'm a PhD student of Professor Bunker at the University of Bern in Switzerland. So this talk here considers the general graph matching paradigm of graph edit distance. So first of all I will talk about graph edit distance, so graph matching paradigm then. And I will briefly talk about standard procedure for computing the graph. Edit distance by means of a tree search algorithm, and then next I will talk about more Chris algorithm, which is in fact an algorithm for solving the assignment problem in polynomial time complexity, and then our contribution here is that we adapt this algorithm for solving the assignment problem to the problem of graph edit distance. So I will talk about this. Then of course we will provide several experimental results that we have achieved. With our new algorithm for computing graph, edit distance and finally I will draw conclusions. And that we apply. [...]

Table 4.2: A sample of the contents of the OpenULTD. The transcript is truncated to the end of the last full sentence within the first 1,000 characters.

After having taken a look at a sample of the data contained in the OpenULTD shown in table 4.2, albeit very limited in quantity and truncated in length, it is time to describe the process that was used to collect these transcripts. A breakdown of the steps for each subset follows.

VT-SSum Starting from the simplest subset, since it was pre-made for the most part, we first need to `git clone` the repository from GitHub (link in § 3.2.1), which already contains all the data in the dev, test, and train directories. Each of these directories, which represent the dataset split given by the authors, but which we will ignore for our purposes, contain thousands of files in JSON[5] format, with each of these files corresponding to one video lecture from `http://videolectures.net`. We are particularly interested in the `segmentation` object, which is the list of all transcript sentences segmented by projected slide. We re-aggregate these sentences to obtain the full transcript back, and with these we create a single JSON file in return, which represents a single list of objects, each composed by the `title`, `url`, and `text` data for each video lecture; a sample is shown in code snippet 4.1.

```
[
    {
        "title": "NMR of Demixing and Phase Separation in Liquid Crystals",
        "url": "http://videolectures.net/clc2010_zalar_dpsls/",
        "transcript": "Hello everybody. The engagement of Fabiana Group with investigations of
        ↪  phase separation in oriented liquid's to my knowledge basically started half a
        ↪  century ago. When I visit it with, then fill a 10 or Philadelphia it can state right?
        ↪  [...]"
    },
    ...
```

Code Snippet 4.1: The beginning of the transcripts file of the VT-SSum subset of the OpenULTD in JSON format.

MIT OpenCourseWare The MIT offers a multitude of open courses on its online platform. Unfortunately, what they do not offer is a public set of APIs to collect data on their courses, thus we need to scrape this information directly from their website. To do this, we use the excellent Selenium[6] library for Python, which allows us to interact with every webpage without restrictions, meaning we can inject custom JavaScript code, *e.g.* to scroll down, and read CSS[8]

[5] JavaScript Object Notation.

[6] Available at `https://www.selenium.dev`. Downloadable from PyPI[7] at `https://pypi.org/project/selenium`.

[7] Python Package Index.

[8] Cascading Style Sheets.

classes to select specific elements from the HTML[9] DOM[10]. The scraping code, provided in the repository associated with this master's thesis, is unfortunately quite complicated due to it having to account for the total freedom that MIT leaves to professors in organizing their online courses, including the ability to make custom webpages. Further, the code in question has been written in September 2022, but possibly due to some changes to the OpenCourseWare platform, it seems to not be working anymore; the collected data is instead available. After having tentatively scraped the data for 233 courses, including all the download URLs[11] (see code snippet 4.2) from the MIT OCW platform, the next step is to effectively download the linked data, including the videos and the transcripts, from said URLs. Of these initial 233 courses, we filter the duplicate courses, the duplicate lectures, and of these lectures we select only those with a transcript, to form the final 126-courses-long MIT OCW subset of the OpenULTD. The assembled JSON file containing the downloaded transcripts and fewer metadata is shown in code snippet 4.3.

OpenHPI The process of data collection from the OpenHPI platform is very similar to the one described above. Luckily, in this case the scraping is easier due to a very predictable interface of the website. Like above, we start by scraping the website (in this case we need to create an account to access all materials) with Selenium for URLs (a sample of these is shown in code snippet 4.4), then we download the transcript files and we aggregate them in a single JSON file for the whole subset, shown in code snippet 4.5, after having filtered the duplicate courses in this case as well.

Yale For the Yale subset, the procedure to apply is slightly different. Since their video lectures are offered on YouTube, the initial "scraping" of the video playlists has to be done by hand (and brain); the objective is to maximize the number of courses without including false positives, such as playlists of shorter videos (see sample of the playlist files in code snippet 4.6). Then, the data scraping for each video can be done with the handy YouTube API[12], which offers the ability to get not only details about the playlist given its URL, but also of all the individual videos it is made of; the metadata file with the URLs is shown in code snippet 4.7. For ease of use, to download the

[9]HyperText Markup Language.

[10]Document Object Model.

[11]Uniform Resource Locator.

[12]Documentation available at `https://developers.google.com/youtube/v3/getting-started`.

```
[
    {
        "title": "Introduction to Special Relativity",
        "url": "https://ocw.mit.edu/courses/8-20-introduction-to-special-relativity-january-iap-2
        ↪  021/",
        "description": "The theory of special relativity, originally proposed by Albert Einstein
        ↪  in his famous 1905 paper, has had profound consequences on our view of physics,
        ↪  space, and time. This course will introduce you to the concepts behind special
        ↪  relativity including, but not limited to, length contraction, time dilation, the
        ↪  Lorentz transformation, relativistic kinematics, Doppler shifts, and even so-called
        ↪  "paradoxes."",
        "professors": [
            "Prof. Markus Klute"
        ],
        "departments": [
            "Physics"
        ],
        "categories": [
            "Science",
            "Relativity",
            "Physics"
        ],
        "codes": [
            "8.20"
        ],
        "levels": [],
        "lectures": [
            {
                "title": "Week 1: Foundations of Special Relativity\nLECTURE 1.1: COURSE
                ↪  ORGANIZATION",
                "description": "Description: Discussion of the course outline and setup, grading
                ↪  scheme, and first introduction to the concept of relativity.
                ↪  (19:07)\nInstructor: Prof. Markus Klute",
                "video_url":
                ↪  "https://archive.org/download/MIT8.20IAP21/MIT8_20IAP21_lec01-1_300k.mp4",
                "transcript_url": "https://ocw.mit.edu/courses/8-20-introduction-to-special-relat
                ↪  ivity-january-iap-2021/8483288e0c9b5b24beb9c4d3639b4170_uMc-j5aQTH8.vtt",
                "duration_in_seconds": 1146.859
            },
            ...
```


Code Snippet 4.2: The beginning of the URLs file of the MIT OpenCourseWare subset of the OpenULTD in JSON format.

transcript files we use the popular CLI[13] utility `youtube-dl`[14], to finally get a JSON file in a format similar to the one of the other subsets, partially shown in code snippet 4.8.

For the MIT OCW, OpenHPI, and Yale subsets, we apply a few regular expressions to filter repetitive or unnecessary parts of the transcripts. The aggregated

[13]Command Line Interface.

[14]Available at `https://youtube-dl.org`.

```
[
    {
        "title": "Introduction to Special Relativity",
        "url": "https://ocw.mit.edu/courses/8-20-introduction-to-special-relativity-january-iap-2
        ↪  021/",
        "lectures": [
            {
                "title": "Week 1: Foundations of Special Relativity\nLECTURE 1.1: COURSE
                ↪  ORGANIZATION",
                "transcript": "MARKUS KLUTE: Welcome to 8.20. Welcome to Special Relativity. And
                ↪  let me start by wishing you all a happy New Year, happy New Year 2021. I'm
                ↪  pretty sure this is going to be an exciting year with a lot of changes ahead
                ↪  and a lot of exciting events. My name is Markus Klute, and I will guide you
                ↪  through this IAP lecture on special relativity. This is very likely my
                ↪  favorite class at MIT, A, because it's IAP, and we start a new year. There's
                ↪  a lot of excitement in the air. And we have a chance to focus for this one
                ↪  month of January on this specific subject. B. I have a chance to introduce a
                ↪  man, Albert Einstein, through a discussion of his physics, [...]"
            },
            ...
```

Code Snippet 4.3: The beginning of the transcripts file of the MIT OpenCourseWare subset of the OpenULTD in JSON format.

```
{
    "sustainablesoftware2022": {
        "Lecturer Introduction: Mathias Renner": {
            "lecture": "https://open.hpi.de/courses/sustainablesoftware2022/items/4syELkfTqiXoLbk
            ↪  Vqkv2vM",
            "material": [
                "https://open.hpi.de/streams/2bf2d932-54f7-4ff6-9c52-5b91ed13cf43/downloads/hd",
                "https://open.hpi.de/streams/2bf2d932-54f7-4ff6-9c52-5b91ed13cf43/downloads/sd",
                "https://openhpi-video.s3.openhpicloud.de/streams/1kVEqFO4YTp6lsdmDJjHWP/audio_v8
                ↪  .mp3"
            ],
            "transcript": "https://open.hpi.de/subtitles/4b27625c-798a-44e4-acfe-ff179da15ac7"
        },
        ...
```

Code Snippet 4.4: The beginning of the URLs file of the OpenHPI subset of the OpenULTD in JSON format.

JSON file is then processed one last time to remove 112 transcripts shorter than 500 characters and 552 duplicate transcripts with the help of the incredibly terse Pandas[15] data analysis library for Python.

[15]Available at `https://pandas.pydata.org`. Downloadable from PyPI at `https://pypi.org/project/pandas`.

```
[
    {
        "title": "sustainablesoftware2022",
        "url": "https://open.hpi.de/courses/sustainablesoftware2022",
        "lectures": [
            {
                "title": "Lecturer Introduction: Mathias Renner",
                "url": "https://open.hpi.de/courses/sustainablesoftware2022/items/4syELkfTqiXoLbk
                ↪  Vqkv2vM",
                "transcript": "Hi, a few words about myself. My name is Mathias. I hold a
                ↪  bachelor's and master's degree in Information Systems, in which I focus on
                ↪  two topics that I [...]"
            },
            ...
```

Code Snippet 4.5: The beginning of the transcripts file of the OpenHPI subset of the OpenULTD in JSON format.

```
[
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNeyViG2ar68jkgEi4y6doNZy",
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNez7ZuPRY3KNJ2ef16qebyZe",
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNewA25FVJ-lawQ-yr-alF58z",
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNeyuvTEbD-Ei0JdMUujXfyWi",
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNeyqnC6Gj5VCZERhhy9CC1S6",
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNezhx8YiGIV8I22ICSuzslja",
    "https://www.youtube.com/playlist?list=PLh9mgdi4rNewfxO7LhBoz_1Mx1MaO6sw_",
    "https://www.youtube.com/playlist?list=PL3F6BC200B2930084",
    "https://www.youtube.com/playlist?list=PL851F45079A91C3F2",
    ...
```

Code Snippet 4.6: The beginning of the playlist URLs file of the Yale subset of the OpenULTD in JSON format.

```
[
    {
        "title": "Power and Politics in Today's World",
        "url": "https://www.youtube.com/playlist?list=PLh9mgdi4rNeyViG2ar68jkgEi4y6doNZy",
        "description": "This course provides an examination of political dynamics and
        ↪  institutions over this past tumultuous quarter century, and the implications of these
        ↪  changes for what comes next. Among the topics covered are the decline of trade unions
        ↪  and enlarged role of business as political forces, changing attitudes towards parties
        ↪  and other political institutions amidst the growth of inequality and middle-class
        ↪  insecurity, the emergence of new forms of authoritarianism, and the character and
        ↪  durability of the unipolar international order that replaced the Cold War. To view
        ↪  the Office Hours of this course, visit
        ↪  https://www.youtube.com/playlist?list=PLh9mgdi4rNewXXh3Yej7icrFY35qDmGT5",
        "lectures": [
            {
                "title": "Lecture 1: Introduction to Power and Politics in Today's World",
                "description": "Professor Ian Shapiro introduces the class "Power and Politics in
                ↪  Today's World."  \nThis course provides an examination of political dynamics
                ↪  and institutions over this past tumultuous quarter century, and the
                ↪  implications of these changes for what comes next. Among the topics covered
                ↪  are the decline of trade unions and enlarged role of business as political
                ↪  forces, changing attitudes towards parties and other political institutions
                ↪  amidst the growth of inequality and middle-class insecurity, the emergence of
                ↪  new forms of authoritarianism, and the character and durability of the
                ↪  unipolar international order that replaced the Cold War.",
                "video_url": "https://www.youtube.com/watch?v=BDqvzFY72mg",
                "duration_in_seconds": 3375
            },
            ...
```


Code Snippet 4.7: The beginning of the URLs file of the Yale subset of the OpenULTD in JSON format.


```
[
    {
        "title": "Power and Politics in Today's World",
        "url": "https://www.youtube.com/playlist?list=PLh9mgdi4rNeyViG2ar68jkgEi4y6doNZy",
        "lectures": [
            {
                "title": "Lecture 1: Introduction to Power and Politics in Today's World",
                "video_url": "https://www.youtube.com/watch?v=BDqvzFY72mg",
                "transcript": "- Hello everybody and welcome. How is everybody today? Great.
                ↪  Well, I'm delighted to have the opportunity to be giving the DeVane Lectures.
                ↪  And the DeVane Lectures, as you can tell, from looking around you double as
                ↪  being a regular Yale course for credit that students can take for credit and
                ↪  lectures that are open to the general public. [...]"
            },
            ...
```


Code Snippet 4.8: The beginning of the transcripts file of the Yale subset of the OpenULTD in JSON format.

As is shown in figure 4.3, created like all the other plots in this chapter with the very powerful Seaborn[16] statistical data visualization library for Python, the transcript lengths in tokens (obtained with the BART [10] tokenizer available in the Transformers library [85]) are quite evenly distributed in the MIT OCW and Yale subsets, while OpenHPI has the shortest transcripts and VT-SSum contains many relatively short ones, though with a tail that touches the 16K tokens. As a reminder, the standard input token length of BERT [2] is 512, for BART 1,024, and attention mechanisms optimized for long sequences can reach 16,384.
We note that OpenULTD transcripts fall within the 16,384 token threshold up to the 98.9th percentile.

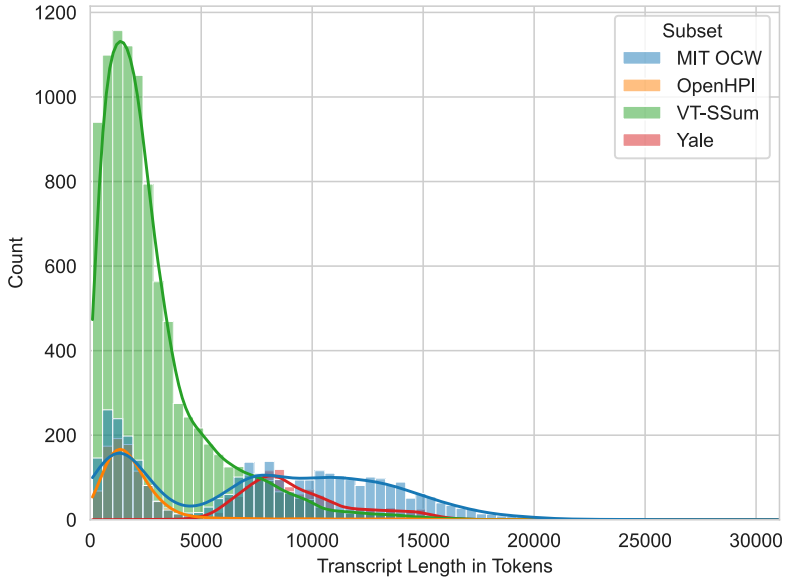


Figure 4.3: The overlaid distributions of the transcript lengths in tokens of the four OpenULTD subsets.

Looking instead at figure 4.4, it is apparent from the overlap of the blue and yellow lines that the transcript lengths in tokens are just a little larger than those in words, which means that most tokens correspond to one word exactly. Also, all four distributions approximately follow the same curve, which means there is a

[16]Available at `https://seaborn.pydata.org`. Downloadable from PyPI at `https://pypi.org/project/seaborn`.

direct proportionality between characters, tokens, words, and sentences.

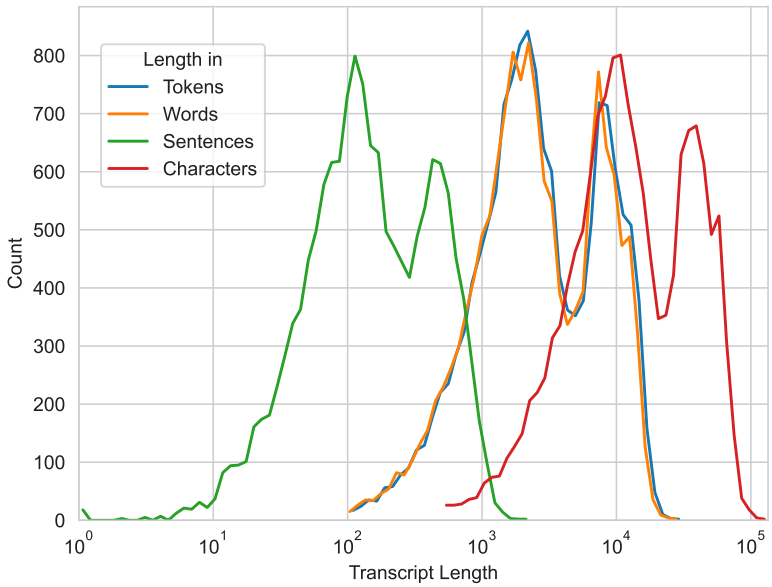


Figure 4.4: A comparison of the distributions of the transcript lengths of the whole OpenULTD, shown in tokens, words, sentences, and characters with the x axis in logarithmic scale.

We conclude our commentary of OpenULTD by interpreting figure 4.5, which represents the number of lectures per course divided by subset. It is clear how most courses contain a number of lectures between 20 and 40, which is what many would consider a normal amount for a higher education course. MIT OCW is the subset with the most outliers, with one course of even ~240 lectures, while Yale is the most consistent subset.

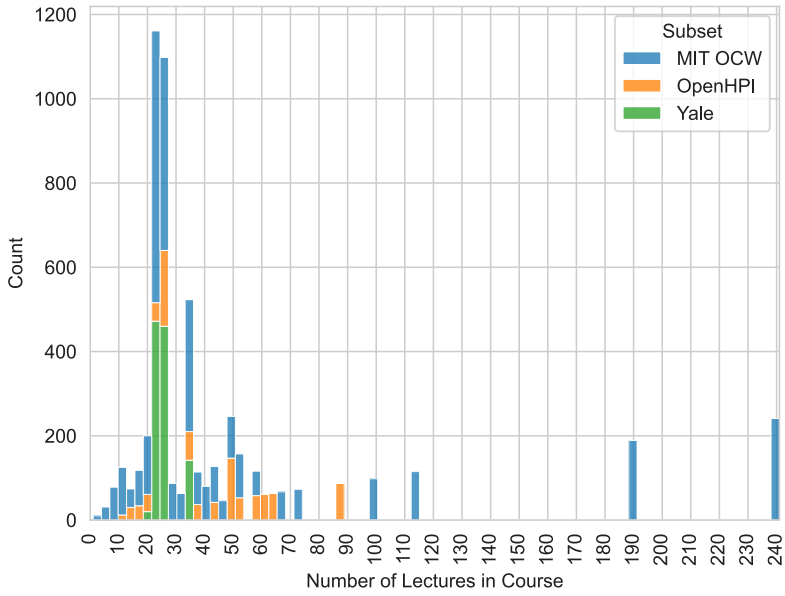


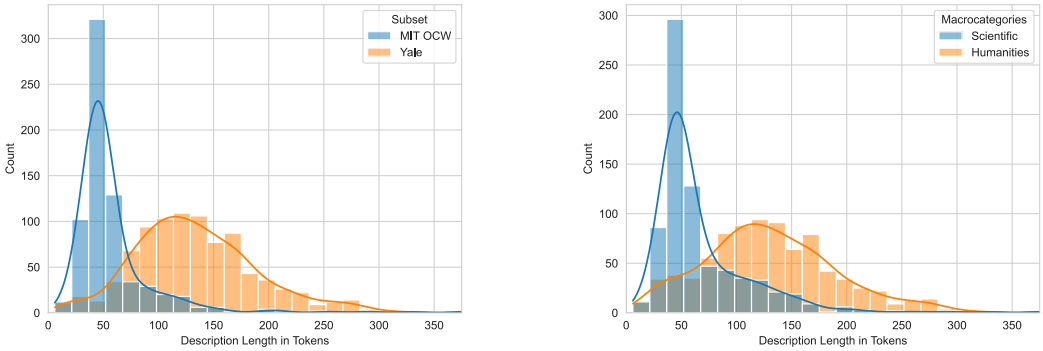
Figure 4.5: The distribution of the course lengths in terms of lectures of the MIT OCW, OpenHPI, and Yale subsets of OpenULTD. VT-SSum is not considered due to its course-less structure.

4.2.2 UniSum

UniSum[17] is the second of the two novel datasets proposed with this work. Its name takes inspiration from other popular summarization datasets, like XSum [94]. As shown in table 4.3, it is derived from the OpenULTD (see § 4.2.1) MIT OCW and Yale subsets, which we choose due to their human-made transcripts and video descriptions, useful for abstractive summarization. More specifically, from these two subsets we select only the lectures that contain both a transcript and a description (also respectively referred to as "text" and "target" or "summary"), and as a last step we remove 16 lectures with a video file unavailable at the previously scraped URL, 1 lecture with a duplicate transcript, and 18 lectures with a duplicate description, leaving us with a total of 1,583 samples.

Subset	Number of Samples	Number of Courses
MIT OCW	693	31
Yale	890	36

Table 4.3: The data composition of the UniSum dataset with respect to its subsets. For the subset licenses, see table 4.1.



(a) Divided by subset. **(b)** Divided by macrocategory.

Figure 4.6: The distribution of the description (or summary) lengths of the UniSum dataset in tokens.

It is immediately apparent from figure 4.6 that there is a high correlation between MIT courses being in the scientific macrocategory and Yale courses falling

[17]University Summarization dataset.

more in the humanities macrocategory, though there are a few exceptions.
The same figure also tells us an important fact about the UniSum dataset, that is that not all lecture descriptions follow the same length distribution: descriptions of lectures of scientific courses tend to be more succinct, with a maximum length of ~200 tokens, while lectures of courses in the humanities field also go up to ~350. The absolute maximum length is 375 tokens, which is manageable by a Transformer decoder (see § 2.2) with a standard Self-Attention mechanism.

In figure 4.7 we see how most MIT OCW outlier courses, by length in lectures, present in OpenULTD (see figure 4.5), have been pruned.
The UniSum dataset is therefore more consistent in terms of kind of courses it is composed of.

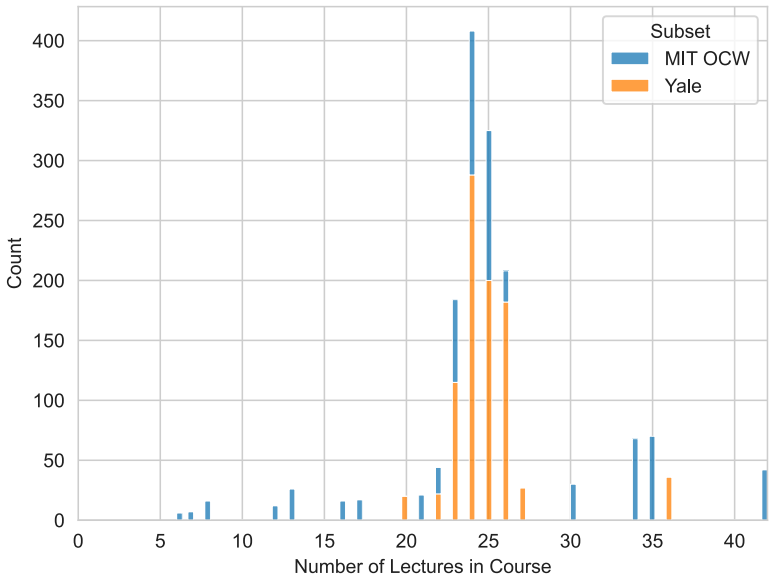


Figure 4.7: The distribution of the course lengths in terms of lectures in the UniSum dataset, divided by subset.

Table 4.4 shows some Unisum data samples, which are transcript-summary pairs. We note that, though unable to verify this information for each description, we strive to collect the most part of human-made summaries, avoiding auto-generated ones to exclude the chance of using these data to learn the internal language representations previously learned by another automatic summarization model.

Transcript	Summary
RICHARD SCHMALENSEE: So today, we're going to talk about non-renewable sources– boring, dull, fossil stuff. The main reason for talking about this stuff is it's really important. It's about 92% of US primary energy. And if you look global, and you look just at marketed energy, forgetting gathered wood and related stuff, it's also about 92%. And this is an EIA graph that basically shows the [...]	This lecture focuses on the state of non-renewable energy on the global market. Classic hotelling theory is covered in the beginning, and then oil, coal, and natural gas markets are analyzed over the across geographies and time.
Hi. This is the first lecture in MIT's course 18.06, linear algebra, and I'm Gilbert Strang. The text for the course is this book, Introduction to Linear Algebra. And the course web page, which has got a lot of exercises from the past, MatLab codes, the syllabus for the course, is web.mit.edu/18.06. And this is the first lecture, lecture one. So, and later we'll give the web address for viewing [...]	A major application of linear algebra is to solving systems of linear equations. This lecture presents three ways of thinking about these systems. The "row method" focuses on the individual equations, the "column method" focuses on combining the columns, and the "matrix method" is an even more compact and powerful way of describing systems of linear equations.
Well, in 1921 a group of workmen working on the highway near the village of St. Osyth, which is here in Essex, in East Anglia, discovered a skeleton. And at first they thought they'd uncovered a modern crime, but it was soon established that it was very old. And subsequently, on the basis of both documentary evidence and forensic evidence, they identified it as being probably the remains of a [...]	In this lecture, Professor Wrightson discusses witchcraft and magic. He begins with the context of magic beliefs in this period, introducing the 'cunning folk' who had reputations as healers and were often consulted. He then considers the specific problem of witchcraft, the use of magic to do harm, and its identification by the late medieval church as a form of anti-Christian cult. [...]

Table 4.4: A sample of the contents of the UniSum dataset. Transcripts and summaries are truncated (indicated by [...]) to the first 400 characters.

At this point, it is necessary to add that we provide two flavors, or variants, of the UniSum dataset, which we refer to as "standard" and "extended". A sample of the standard variant is what is shown in table 4.4.
We aim to "extend" this initial version of the dataset in a multimodal fashion, where the first modality is text and the second modality is video. To decide what information to include from the video files[18] of the lectures, we guess that an interesting additional fact to feed the summarization model would be, for each sentence of the transcript, the binary classification of whether the lecture speaker was writing while saying the sentence or not, based on the hypothesis that if a professor writes down a concept while they are explaining it, what they are saying at the same time is likely worth of highlighting.
To extract this information from the video lectures, we proceed in three steps:

1. the first thing we do is finding the correspondence between each sentence and their beginning and end in terms of seconds from the start of the video. To do so, we implement our own function to read and split into captions the transcript files in WebVTT[19] format. We keep the captions, separated by initial and end timecodes, and we separate again the full transcript, this time into sentences, using the NLTK[20] library for Python. Using what revealed itself to be not so trivial code, we obtain the initial and end timecodes for each full sentence;

2. the second step of our plan involves effectively performing the binary classification of each of the composing video clips, corresponding to one full sentence, of each video lecture. We use a Microsoft X-CLIP [4] model from the Transformers [85] library, specifically its `"microsoft/xclip-large-patch14-kinetics-600"` checkpoint, chosen after a few empirical trials. Since this model, as seen in section 3.1.3, supports natural language labels, we experiment until we find that `labels = ["not a person who is writing", "a person who is writing"]` are what works best for our use case. Specifically, we feed X-CLIP video clips of 8 linearly spaced frames at a 224×224 resolution;

3. once we have collected the binary classification list for each lecture, we join the transcript sentences wrapped between two tokens, to get `f"<w>{sentence}</w>"` if the classification for the video clip corresponding to the sentence was `"a person who is writing"`, and `f"<nw>{sentence}</nw>"` otherwise, where

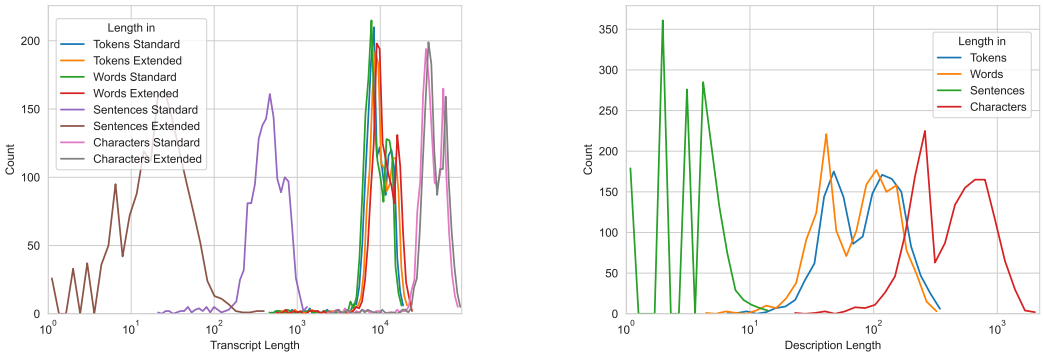
[18] We make the 1,603 videos downloaded in early January 2023 available at `https://cloud.caste.dev/s/7zkgBJ4KHrmDFPD`.

[19] Web Video Text Tracks.

[20] Natural Language ToolKit. Available at `https://www.nltk.org`. Downloadable from PyPI at `https://pypi.org/project/nltk`.

"w" stands for "writing" and "nw" for "not writing". These HTML-style tags are inspired by BART's [10] standard tokens and represent a special kind of token that the summarization model will learn to recognize and assign a meaning to autonomously. We do so by leveraging the feature of the tokenizer classes offered by the Transformers library [85] that allows to easily add a list of custom special tokens, which for us would be `["<w>", "</w>", "<nw>", "</nw>"]`.

Figure 4.8 shows the length distributions of the transcripts and the descriptions of the UniSum dataset. We note that the additional information extracted from the videos is not applicable to the descriptions, thus it is only in figure 4.8a that we can see a comparison of the transcripts from the standard and extended UniSum dataset variants. More specifically, we see that the extended transcripts are only marginally longer than the standard ones. Additionally, the NLTK library seems to have been thrown off by the HTML-like tokens in determining the number of sentences in the extended UniSum version (shown by the brown line).
We note that, while the standard transcripts are within the 16,384 tokens mark until the 98th percentile, the extended ones are so until the 93.8th percentile, which translates to a not irrelevant loss of information by the summarization model working with the extended dataset, which truncates any token in the input sequence after the 16,384th, but still preserves a large enough part of information. Therefore, we deem the extended dataset usable for the experiments that we will see in section 4.4.



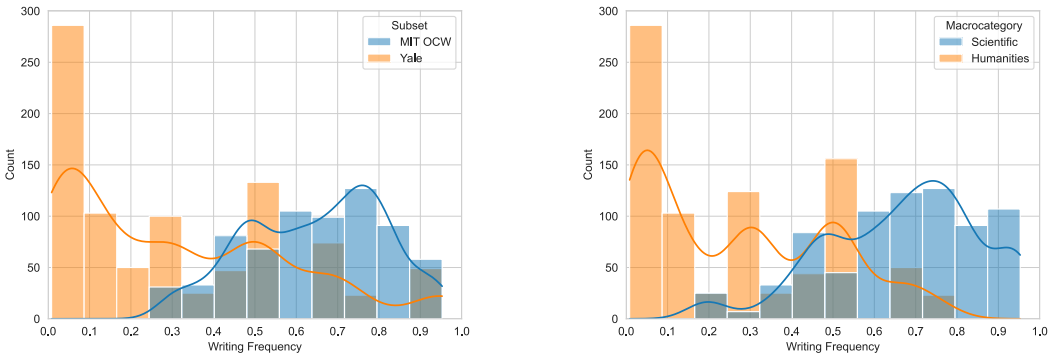
(a) **(b)**

Figure 4.8: A comparison of the UniSum dataset transcript and description lengths measured in characters, tokens, words, and sentences.

Standard Transcript	Extended Transcript
RICHARD SCHMALENSEE: So today, we're going to talk about non-renewable sources– boring, dull, fossil stuff. The main reason for talking about this stuff is it's really important. It's about 92% of US primary energy. And if you look global, and you look just at marketed energy, forgetting gathered wood and related stuff, it's also about 92%. And this is an EIA graph that basically shows the [...]	<nw>RICHARD SCHMALENSEE: So today, we're going to talk about non-renewable sources– boring, dull, fossil stuff.</nw><nw>The main reason for talking about this stuff is it's really important.</nw><nw>It's about 92% of US primary energy.</nw><nw>And if you look global, and you look just at marketed energy, forgetting gathered wood and related stuff, it's also about 92%.</nw><w>And this is an [...]
Hi. This is the first lecture in MIT's course 18.06, linear algebra, and I'm Gilbert Strang. The text for the course is this book, Introduction to Linear Algebra. And the course web page, which has got a lot of exercises from the past, MatLab codes, the syllabus for the course, is web.mit.edu/18.06. And this is the first lecture, lecture one. So, and later we'll give the web address for viewing [...]	<w>Hi.</w><w>This is the first lecture in MIT's course 18.06, linear algebra, and I'm Gilbert Strang.</w><w>The text for the course is this book, Introduction to Linear Algebra.</w><w>And the course web page, which has got a lot of exercises from the past, MatLab codes, the syllabus for the course, is web.mit.edu/18.06.</w><w>And this is the first lecture, lecture one.</w><w>So, and later we'll [...]
Well, in 1921 a group of workmen working on the highway near the village of St. Osyth, which is here in Essex, in East Anglia, discovered a skeleton. And at first they thought they'd uncovered a modern crime, but it was soon established that it was very old. And subsequently, on the basis of both documentary evidence and forensic evidence, they identified it as being probably the remains of a [...]	<w>Well, in 1921 a group of workmen working on the highway near the village of St. Osyth, which is here in Essex, in East Anglia, discovered a skeleton.</w><w>And at first they thought they'd uncovered a modern crime, but it was soon established that it was very old.</w><w>And subsequently, on the basis of both documentary evidence and forensic evidence, they identified it as [...]

Table 4.5: A comparison of the standard transcripts with the transcripts extended with the "writing" video classification in the UniSum dataset. Transcripts are truncated (indicated by [...]) to the first 400 characters.

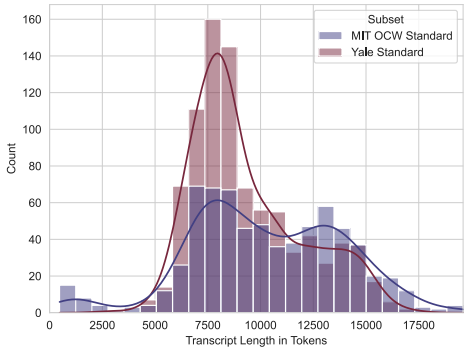
We corroborate our hypothesis of subset-macrocategory positive correlation, previously made in reference to figure 4.6, by looking at figure 4.9. This figure shows an aspect of the UniSum dataset directly derived from the binary "writing"/"not writing" classification obtained with X-CLIP [4], which is how frequently the sentences in the lectures of a course are classified as "writing".
A writing frequency of 0 means that no sentences in none of the lectures in a course have been classified as being spoken while the lecture speaker was writing, and a writing frequency of 1 means that the speaker was writing while saying every sentence in every lecture of the course.
We note that MIT and scientific courses have a higher writing frequency than Yale and humanities courses, which is expected. The majority of the former have a writing frequency greater than 50%, while the exact opposite can be said for the latter.



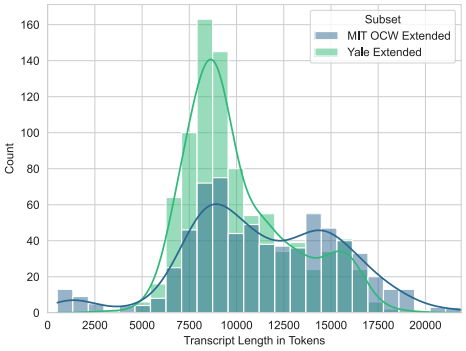
(a) Divided by subset. **(b)** Divided by macrocategory.

Figure 4.9: The distribution of the course writing frequency in the UniSum dataset.

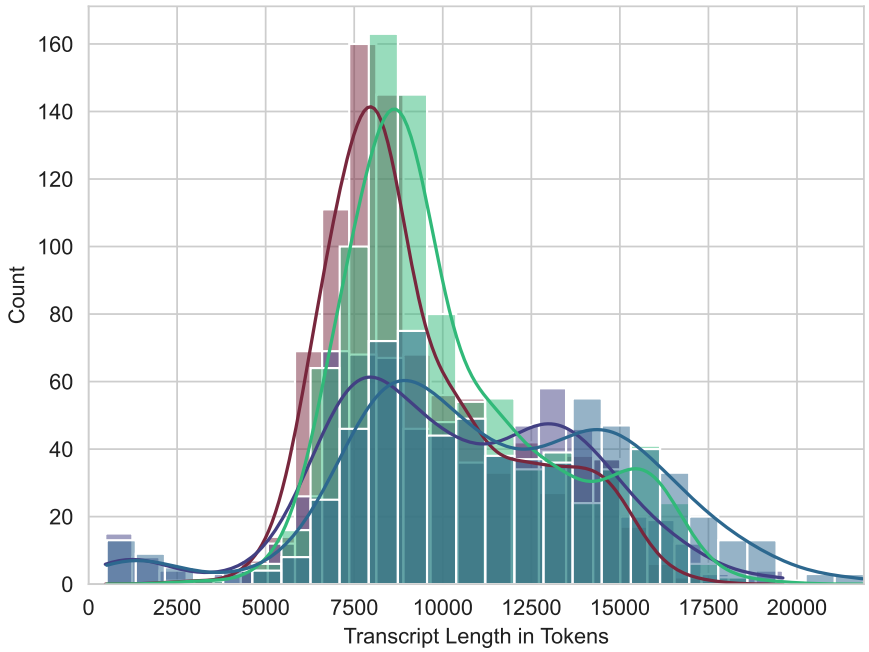
We conclude our presentation of the UniSum dataset by looking at figures 4.10 and 4.11, which differ only by the division of their composing histograms, by subset in the first case and by macrocategory in the second.
We note that the transcript lengths distributions follow very similar curves for both the standard and extended variants of the dataset, the extended transcripts being derived from the standard.



(a) From the standard variant.

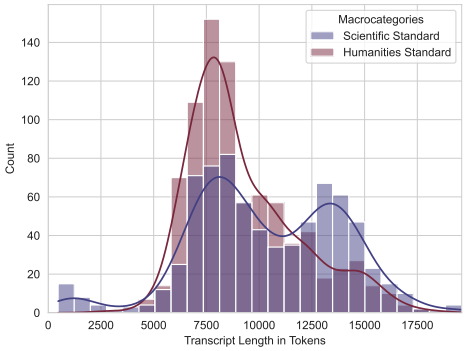


(b) From the extended variant.

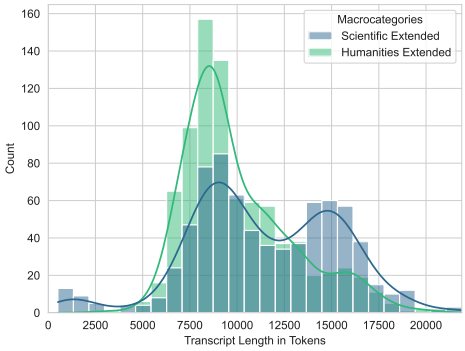


(c) Overlaid standard and extended variants.

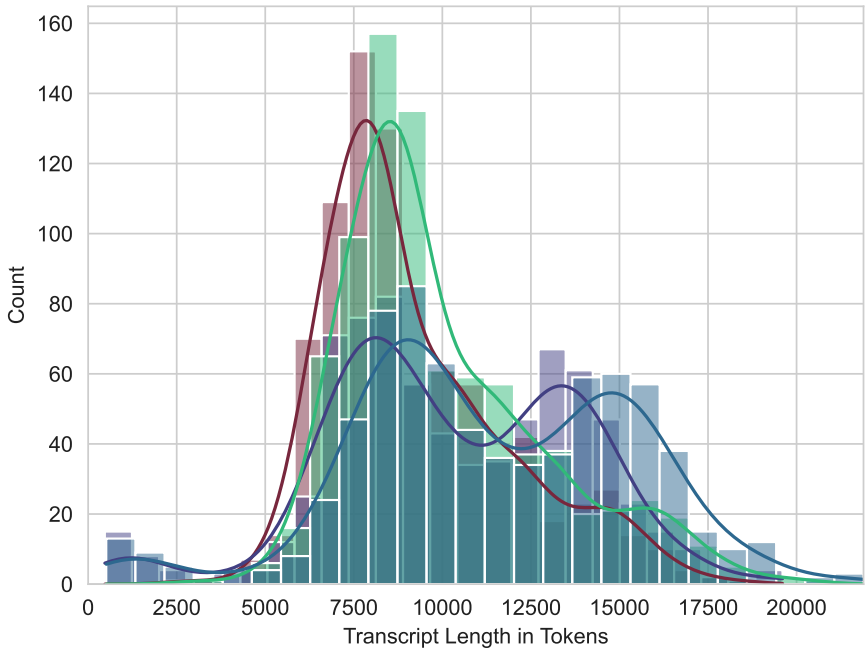
Figure 4.10: A comparison of the transcript lengths distributions for the standard and extended variants of the UniSum dataset, divided by subset.



(a) From the standard variant.



(b) From the extended variant.



(c) Overlaid standard and extended variants.

Figure 4.11: A comparison of the transcript lengths distributions for the standard and extended variants of the UniSum dataset, divided by macrocategory.

4.3 Reproducibility

In this brief section, we aim to provide the reader with all the possible information needed to reproduce our results.

4.3.1 Hardware Specifications

Before discussing the training process that was used for the present thesis, since we will give some information about computation times, batch sizes, and so on, which are all hardware-dependent information, we must first specify on what kind of hardware our experiments were run.
Though some experimentation was conducted in the beginning on Google Colab[21] and on a different local machine, all experiments have been run on the following ad-hoc custom-built configuration:

Component	Model	Specifications
CPU	AMD Ryzen 5 5600X	6 cores, 12 threads, 3.7-4.6 GHz
RAM[22]	G.SKILL Aegis	DDR4, 3200 MHz, 48 GB
GPU	NVIDIA RTX 3090	10,496 CUDA cores, 24 GB of GDDR6X VRAM[23]
SSD[24]	Kingston NV1	NVMe, 500 GB
Motherboard	MSI B550M PRO-VDH	N/A

Table 4.6: The hardware specifications of the machine purpose-built to run the experiments required by the present thesis.

4.3.2 Software Specifications

As with the hardware specifications, it is no less important to show the specific software that was used to conduct all the experiments presented in the next sections.

We use a Debian-based GNU/Linux[25] distribution as the operating system and Python as the programming language, since it currently is the language of

[21] Available at `https://colab.research.google.com`.

[22] Random-Access Memory.

[23] Video Random-Access Memory.

[24] Solid-State Drive.

[25] GNU is Not UNIX.

choice of data scientists and ML practitioners, and thus has the most flourishing developer community among other languages. Most of the code is written within the browser-based Jupyter Lab IDE[26] for easier interaction, but some scripts, such as those used for web crawling and data scraping, have been written using the JetBrains PyCharm Professional IDE.
The main Python libraries used to interact with the models, be it for training or for inference, are the ones made by Hugging Face [85]. The Weights & Biases library [133] is used to track the experiments, version the models, and make graphs related to the training process.
In table 4.7 we show the specific versions of the most relevant software used in this work, whereas the full list is published in the code repository.

Type	**Name**	**Version**
OS[27]	System76 Pop!_OS	22.04 LTS
Kernel	Linux	6.0.12
GPU Driver	NVIDIA	525.85.05
CUDA Toolkit	NVIDIA	11.2
Language	Python	3.10.6
Python Library	PyTorch	1.13.0+cu117
Python Library	Hugging Face Transformers	4.25.1
Python Library	Hugging Face Datasets	2.7.1
Python Library	Hugging Face Evaluate	0.4.0
Python Library	LSG Converter	0.0.4
Python Library	Pandas	1.5.2
Python Library	NLTK	3.8
Python Library	Seaborn	0.12.1
Python Library	NumPy	1.23.5
Python Library	Weights & Biases	0.13.7
Python Library	Jupyter Lab	3.5.1

Table 4.7: The software specifications used to run the experiments required by the present thesis.

[26]Integrated Development Environment.

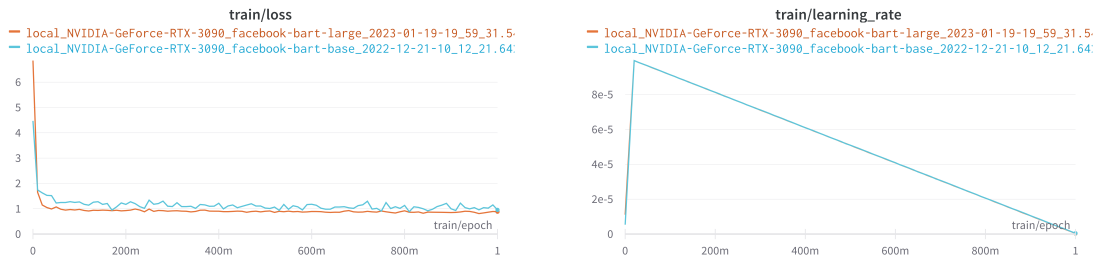
[27]Operating System.

4.4 Model Training

In this section we will describe in detail the process we followed to train the deep models that we used to obtain the results presented in chapter 5. A diagram of the complete process is shown in figure 4.1.

4.4.1 Domain Adaptation

The first step in training our models was adapting BART, already pretrained by Meta AI Research, on the domain of video lecture transcripts. The main idea behind this procedure is to perform a short continuation of BART's pretraining, with the same denoising objective (described in § 3.1.1), on the whole OpenULTD. This is useful for focusing the model on a certain data distribution, which in the case of OpenULTD is that of video lectures, presentations, and public talks.
We train both the base and large variants of BART for 1 epoch, following recent results that advise against data repetition during pretraining [134], with a learning rate of 1×10^{-4}, a linear scheduler, a number of warm-up steps equal to 1% of the total steps (as shown in figure 4.12b), and an effective batch size[28] of 64 when training BART_{LARGE} and 4 when training BART_{BASE}. Since most transcripts present in OpenULTD are longer than BART's maximum input length of 1,024 tokens (see figure 4.3), we split them into multiple 1,024-tokens-long sequences, converting the original 14,677 long OpenULTD transcripts into 65,930 shorter samples.



(a) **(b)**

Figure 4.12: The training loss and the linearly scheduled learning rate of the two BART models on which we perform the domain adaptation with one epoch on OpenULTD. Graphs made with Weights & Biases [133].

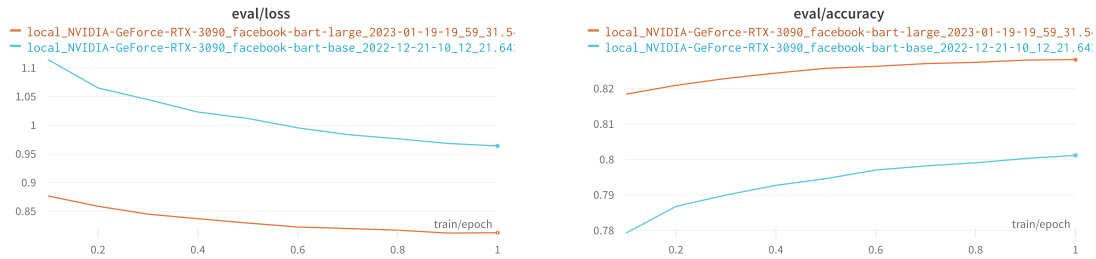
[28] $ebs = bs \times gas$, where ebs is the effective batch size, bs is the training batch size (actual number of samples in the GPU memory at each step), and gas is the number of gradient accumulation steps.

Among the eleven experiments that we run, for a total compute time of ~24 hours spent in training, we select one for each size of BART, which we compare to the original pretrained models in table 4.8.

Model	Validation Loss ↓	Perplexity[29]↓	Accuracy ↑
BART_{BASE}	3.6060	36.8185	N/A
$\text{BART}^{\star}_{BASE}$	0.9645	2.6234	0.8008
BART_{LARGE}	5.2540	191.3301	N/A
$\text{BART}^{\star}_{LARGE}$	0.8125	2.2535	0.8282

Table 4.8: The numerical results of the domain adaptation of the base and large variants of BART on OpenULTD. The models accompanied by the $\star$ symbol have been subjected to domain adaptation.

The graphs of validation loss and accuracy on the validation split of OpenULTD during training are shown in figure 4.13.



(a) **(b)**

Figure 4.13: The validation loss and accuracy of the two BART models on which we perform the domain adaptation with one epoch on OpenULTD. Graphs made with Weights & Biases [133].

[29] $p = e^l$, where p is the perplexity and l is the loss.

4.4.2 Summarization Finetuning

The second and final step of our training process is the finetuning of our initial models on the two flavors of the UniSum dataset, standard and extended.
This involves finetuning many model variants, whose test results are described and compared in section 5.2.

As a first step, we determine the best learning rate values for the base and large versions of our models. We conduct 10 experiments, varying the learning rate and the number of warmup steps before settling on 10% warmup steps of the total training steps for all models, a 7.5×10^{-5} learning rate for the BART_{BASE}-derived models, and a 5×10^{-5} learning rate for the BART_{LARGE}-derived models. Though we experiment with a pure cosinusoidal scheduler and a cosinusoidal scheduler with hard restarts, we find that a simpler linear scheduler works best. All of our models are trained with an effective batch size of 32 (with a training batch size of 2 and 16 gradient accumulation steps for base models and a training batch size of 1 and 32 gradient accumulation steps for large models) for 10 epochs with 16-bit floating point precision and the AdamW optimizer [135] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\times10^{-8}$, and a weight decay of 0.01. We perform one evaluation per epoch, and we choose the best checkpoint (that we use for testing) as the one that obtains the highest ROUGE-2 metric result on the validation set during training. We always use, for validation, testing, and inference, a length penalty of 2.0, which we find favors a mean generation length similar to the mean overall length of the summaries contained in our dataset, and a number of beams for beam search equal to 5, which is a larger-than-default number that does not yet cause memory or performance issues while allowing for slightly higher-quality results.
Having identified these hyperparameters, we finetune 22 other models for a total of more than 450 hours, or 19 days, of compute time. We show the different possible model variations that we train in table 4.9.

Variant	**Variants Number**	**Variant Names**
Model Size	2	Base, Large
Model Type	3	BART, LED, LSG
Domain Adaptation	2	None, Short (done on BART)
Dataset Flavor	2	Standard, Extended

Table 4.9: The different model variants that we obtain at the end of the finetuning step.

In the following pages we show the differences in loss and metric results for each of the model variations of table 4.9 with 16 of our trained models: in figure 4.14 between base and large models (8 and 8); in figure 4.15 between models with different types of long attention (8 with a Longformer and 8 with an LSG type of Self-Attention); in figure 4.16 between models trained with (8) and without (8) domain adaptation; and in figure 4.17 between models finetuned on the standard and extended variants of the UniSum dataset (8 on standard and 8 on extended).

Figure 4.14 tells us a few things: first, large models learn faster (figure 4.14a), even at a lower learning rate with respect to base models (figure 4.14b), though for the same reason they tend to also overfit the training data at epochs closer to the beginning (figure 4.14c). As for the metrics, where all ROUGE-N, ROUGE-L and BERTScore higher values mean a better result (the generation length is simply an indication of what is the average length of the generated summaries across all the validation samples), the mean lines seem to indicate that base models perform better than large ones in general. This counterintuitive behavior likely stems from the fact that some of the larger models are heavily penalized by their quicker learning towards the end of the finetuning process (in the right-hand side of the graphs), especially models with the Longformer type of Self-Attention as can be seen in figure 4.15b.
From all metrics in figure 4.15, it is apparent how the LSG long attention type is the one that better preserves the performance of the original quadratically memory-dependent Self-Attention mechanism, given both its higher mean lines and smaller value range.
Figure 4.16 shows how having previously performed the domain adaptation step is helpful for the general performance of the model.
Finally, figure 4.17 shows that there is no appreciable difference in training and validation loss between models trained on different dataset versions, though the models finetuned on the extended version suffer from a general small loss in performance across all metrics likely due to the slightly smaller amount of text available for training, since it gets truncated at the 16,384th token and the additional special tokens may cause a “data overflow” in some cases.

The learning rate is chosen based on the model size as seen in figure 4.14a, thus it is omitted from the figures following the first.

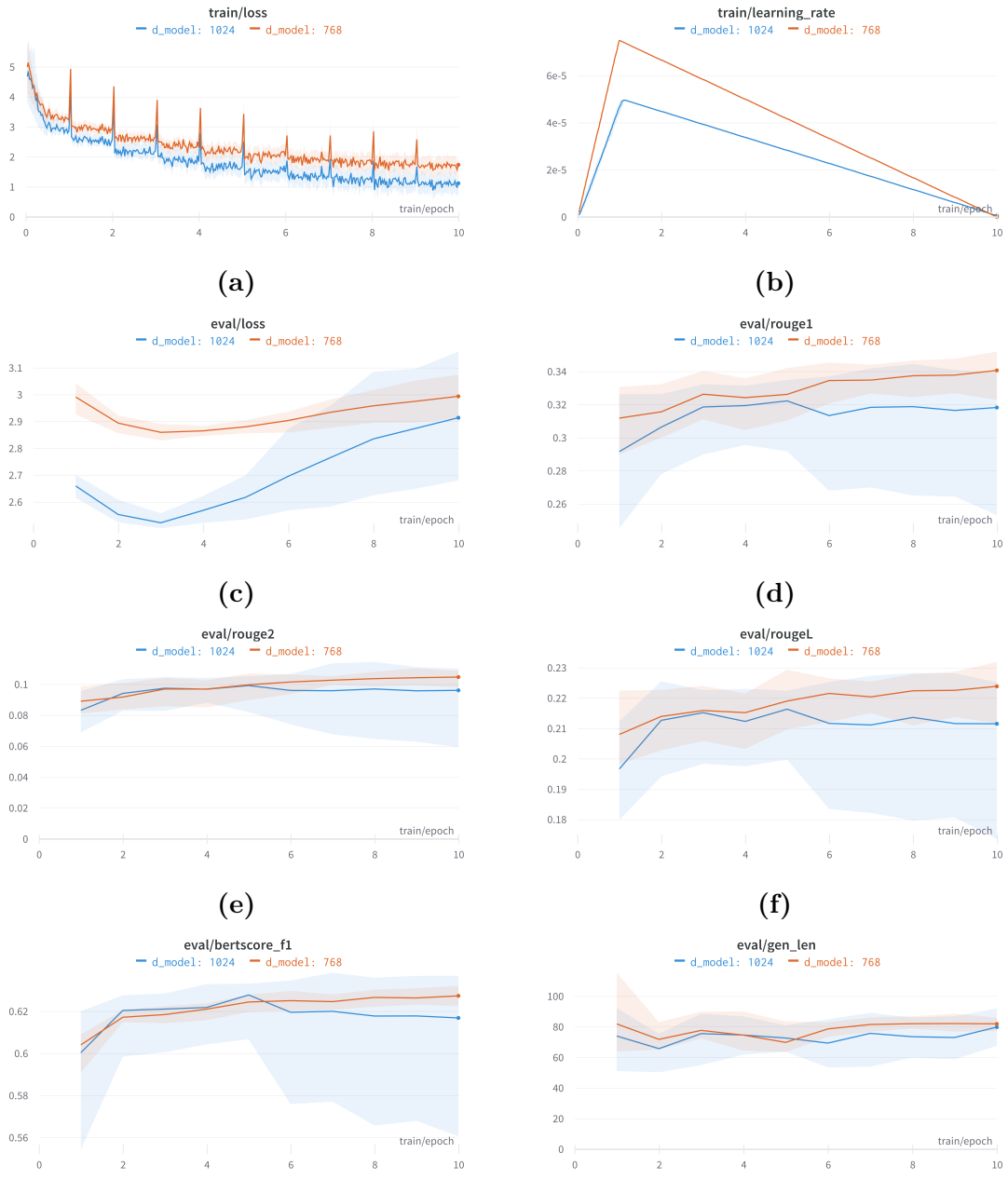


Figure 4.14: The mean values and their minimum-to-maximum range of the training and evaluation metrics of all the trained models with long attention grouped by d_{model} (where $d_{model} = 768$ indicates a base model and $d_{model} = 1{,}024$ indicates a large model; d_{model} is the internal dimension of the network as seen in § 2.2). Graphs made with Weights & Biases [133].

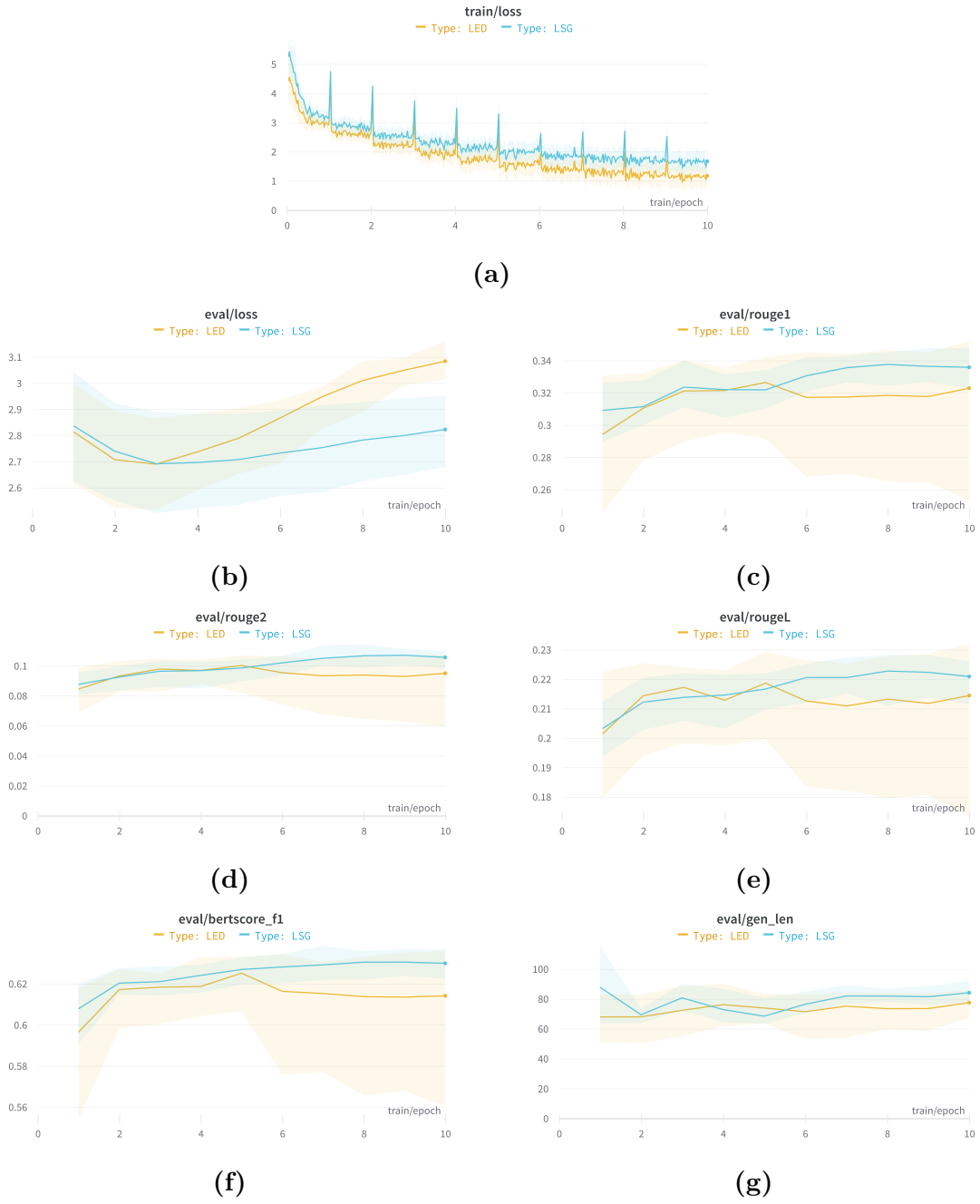


Figure 4.15: The mean values and their minimum-to-maximum range of the training and evaluation metrics of all the trained models with long attention grouped by type of long attention (LED indicates a Longformer type of Self-Attention, as seen in § 3.1.2). Graphs made with Weights & Biases [133].

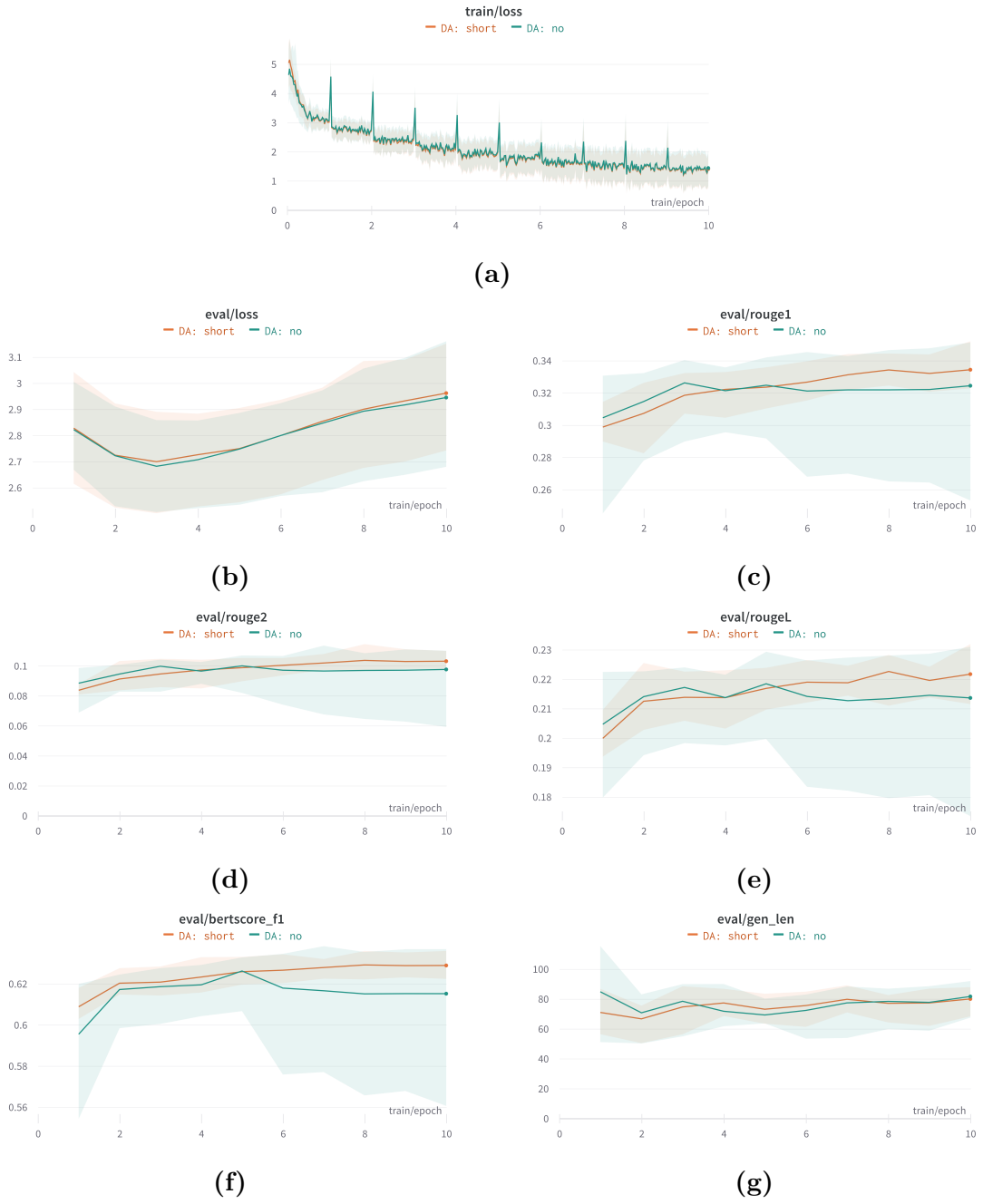


Figure 4.16: The mean values and their minimum-to-maximum range of the training and evaluation metrics of all the trained models with long attention grouped by type of domain adaptation (DA). Graphs made with Weights & Biases [133].

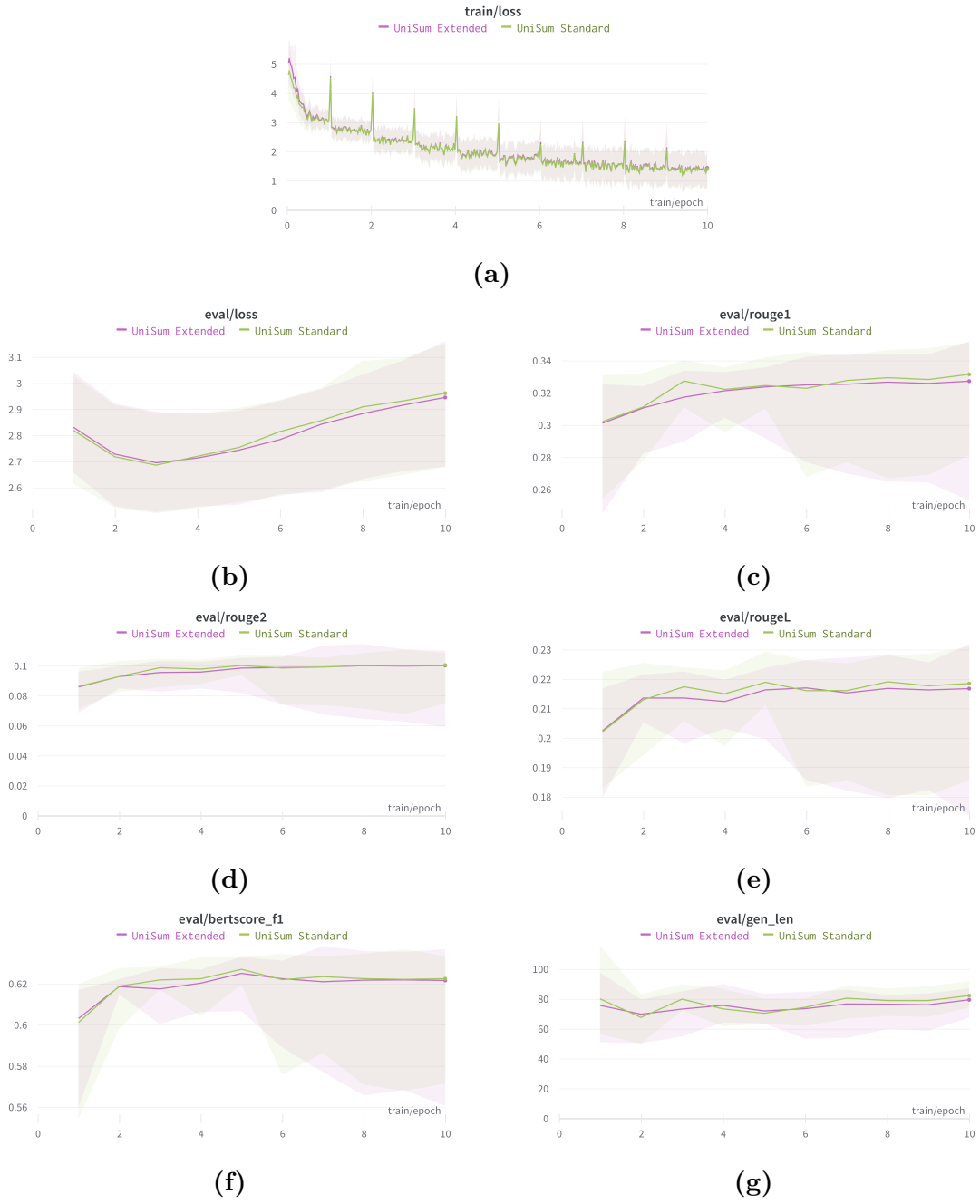


Figure 4.17: The mean values and their minimum-to-maximum range of the training and evaluation metrics of all the trained models with long attention grouped by version of the UniSum dataset they have been trained on. Graphs made with Weights & Biases [133].

Chapter 5

Experimental Evaluation

Since some experimental results will be presented in this chapter (§ 5.2), a brief overview of the metrics (§ 5.1) used to measure the performance of the involved models is the first thing that we must cover in order to have a general understanding of the numerical results we obtain.

5.1 Metrics

Even though the loss function of a deep network is what actually defines how well it is performing, it is numerically incomparable between different models and even between different sizes of the same model.
Since it is fundamental for researchers to be able to compare the results of different networks, some task-specific metrics have been designed during the years.

5.1.1 ROUGE

ROUGE[1] is a set of four metrics (ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S) proposed in 2004 by Lin [8] for the automatic evaluation of summaries. Its idea stems from the previous work of Papineni *et al*, who in 2002 proposed a similar metric, BLEU[2] [136], for measuring the quality of machine-generated translations.

[1]Recall-Oriented Understudy for Gisting Evaluation. A Python implementation of the original ROUGE Perl package is available at `https://github.com/google-research/google-research/tree/master/rouge`.

[2]BiLingual Evaluation Understudy.

Here is an explanation of the two most popular variants of ROUGE:

ROUGE-N This metric «is an n-gram recall between a candidate summary and a set of reference summaries» [8], where an n-gram is an ordered sequence of n tokens, such as words or sub-words. It is computed as:

$$ROUGE_N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} count_{match}\left(gram_n\right)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} count\left(gram_n\right)} \tag{5.1}$$

This equation can be simplified in the case of a single reference summary to:

$$ROUGE_N = \frac{\sum_{gram_n \in GeneratedSummary} count_{match}\left(gram_n\right)}{\sum_{gram_n \in ReferenceSummary} count\left(gram_n\right)} \tag{5.2}$$

where, in both equations 5.1 and 5.2, the *count* function simply counts the number of instances of an n-gram in the sequence, and $count_{match}$ counts the number of overlapping n-grams from the generated summary with the reference summary.

ROUGE-L This variant of ROUGE measures the LCS[3] given a generated summary and its reference. The length of the LCS is to be intended as a measure of similarity between two sequences: the higher the ROUGE-L is, the more similar the generated summary is to the reference. Given two sequences, X of length m and Y of length n, the LCS function which returns the longest common subsequence of two sequences, the recall $R_{LCS} = \frac{LCS(X,Y)}{m}$, and the precision $P_{LCS} = \frac{LCS(X,Y)}{n}$, with β set to a big number:

$$ROUGE_L = \frac{(1+\beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} \tag{5.3}$$

It is important to note that in this context the elements of each sequence occupy consecutive positions within the sequence, that is to say, given the generated sentence “the cat is a table” and its reference “the cat is on the table”, the LCS between the two would be “the cat is”, and not “the cat is table”.

[3]Longest Common Subsequence.

5.1.2 BERTScore

The BERTScore[4] is a metric introduced by Zhang *et al.* in 2019 [9] that measures the similarity between two sequences of tokens using the learned contextual embeddings of a pretrained BERT-based model, which improves the Pearson correlation with human judgment with respect to other metrics such as BLEU [136]. «BERTScore computes the similarity of two sentences as a sum of cosine similarities between their tokens' embeddings» [9].

An example of why BERTScore works better than other metrics in this regard is the following: given the reference sentence "people like foreign cars", BLUE would rate the generated sentence "people like visiting places abroad" higher than "consumers prefer imported cars" in similarity, but the meaning of the second sentence is actually more similar to the reference sentence than that of the first one.

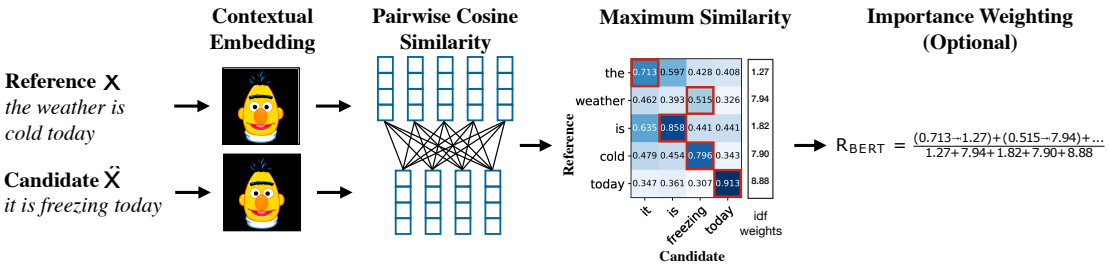


Figure 5.1: A diagram of the computation of the recall measure in BERTScore. Figure 1 in [9].

The human judgment correlation changes with different models: while the authors at the time of publication [9] advised to use the 24-layer variant of RoBERTa$_{LARGE}$ [14], they have investigated other models since and, at the time of writing, the model with the highest correlation, which we use for this reason, is DeBERTa$_{XLARGE}$ [137] finetuned on the MNLI[5] dataset contained in the GLUE benchmark dataset [108].

[4]Python implementation available at `https://github.com/Tiiiger/bert_score`.

[5]Multi-genre Natural Language Inference corpus.

5.2 Results

In this section, we will present our results obtained with all the different model variants seen in section 4.4. We note that, like for the training and validation splits of the UniSum dataset, there are also two different test splits depending on the dataset version, either standard or extended. We remind the reader that this only affects the input and not the output sequences of the model, since all summaries are free of the additional tokens mentioned in section 4.2.2.

5.2.1 Interpretation Notes

For all tables:

- each row represents one test, which is done with one model on one test set;
- if the test set is not specified, then the use of the complete UniSum test set ($N = 159$) is implied;
- the tables are split by double vertical lines: the left part indicates the test configuration, the right part shows the corresponding results;
- the table headers are abbreviations: size stands for model size, type stands for long attention type (or lack thereof), DA means domain adaptation, train stands for training set (employed in model finetuning), R1, R2, and RL are different types of the ROUGE metric (respectively, ROUGE-1, ROUGE-2, and ROUGE-L, as seen in § 5.1.1), BS means BERTScore (see § 5.1.2);
- all metrics correlate higher values with better performance;
- we multiply all metric values by 100, to avoid redundant leading zeros (*e.g.* we show 12.34 instead of 0.1234), keeping in mind that ROUGE values range from 0 to 1, so in our case from 0 to 100, and BERTScore values go from -1 to 1, in our case from -100 to 100;
- both ROUGE and BERTScore metrics are obtained from precision and recall values. We always use the F_1 score, calculated as their harmonic mean:

$$F_1 = 2\frac{PR}{P+R} = \frac{2tp}{2tp+fp+fn} \tag{5.4}$$

  where P is the precision, R is the recall, tp represents the number of true positives, fp the number of false positives, and fn the number of false negatives;

- close rows that are not separated by a horizontal line are directly comparable (only one configuration parameter changes between them), and the best value between them is underlined;
- the **bold** numbers are the best values for the column they belong to for each test set;
- the boxed numbers are the best overall value in their column, across test sets and different configurations. The boxing is omitted when only one test set is used, in which case it is sufficient to refer to the bold values.

Before proceeding with the presentation of our results, a few further notes must be made:

- in most tests, the BERTScore value is anti-correlated with better ROUGE values. This metric has been included for completeness, but it is important to remember that the BERT model used to measure the quality of the text produced by our finetuned models has been trained in a general domain, while our work is focused on spoken language used in higher education settings;
- in many tests, large models perform worse than their base counterpart, especially those with a Longformer type of Self-Attention. The reader must keep in mind that, as described in section 4.4.2, we always select the model checkpoint which obtains the best ROUGE-2 value during training, regardless of any other variable. Due to this method, we find ourselves picking the best checkpoint at earlier finetuning steps for large models than for base models, which as on one hand prevents them from overfitting the training data too heavily, on the other hand it likely stops them from learning as well as the base models.

With our setup, the testing process took ~150 hours, or about 6 days. Further statistics about this project are shown in table 6.1.

5.2.2 Results Analysis

With each of the tables present in this chapter we aim not only to illustrate our general findings, but also to compare different variations of the models, with a special interest towards the effectiveness of the domain adaptation step and the usefulness of the "writing"/"not writing" classification added in the extended UniSum flavor.

Size	Type	DA	Train	R1	R2	RL	BS
base	BART	no	none	17.03	3.26	10.31	49.67
base	BART	yes	none	17.34	3.07	9.90	50.94
large	BART	no	none	**17.61**	**3.44**	**10.40**	51.92
large	BART	yes	none	17.27	3.11	10.11	50.70
base	LSG	no	none	14.72	2.40	9.43	44.58
base	LSG	yes	none	17.39	2.75	9.79	51.22
base	LED	no	none	15.32	2.46	9.29	47.59
base	LED	yes	none	17.42	2.92	9.80	**51.97**
large	LSG	no	none	16.34	2.82	9.91	49.97
large	LSG	yes	none	16.99	2.93	9.96	51.23
large	LED	no	none	15.89	2.77	9.88	48.41
large	LED	yes	none	17.40	2.75	9.78	51.66
base	LSG	no	std	35.12	11.13	22.84	63.03
base	LSG	yes	std	33.51	10.58	21.85	62.65
base	LED	no	std	**35.25**	11.07	22.81	63.41
base	LED	yes	std	34.93	11.49	22.69	62.90
large	LSG	no	std	34.83	11.48	23.12	64.22
large	LSG	yes	std	34.55	**11.61**	22.61	63.84
large	LED	no	std	34.32	11.08	22.58	63.73
large	LED	yes	std	34.49	11.42	**23.23**	**64.31**

Table 5.1: The effect of domain adaptation on models before finetuning (upper part of the table) and after finetuning on the standard variant of UniSum (lower part of the table).

Table 5.1 shows what effect performing the domain adaptation step has on the final summarization results of the models. While the test results of the models lacking finetuning, shown for completeness in the upper sub-table, are not particularly intriguing, we can see from the results of the finetuned models that the domain adaptation does not always have an overall positive effect, which is somewhat unexpected. We hypothesize that this behavior is due to possibly two factors: first, the data from the VT-SSum subset, which was obtained with an

ASR model, mostly accurate but still imperfect, particularly with complex words, constitutes 62.8% of OpenULTD; second, despite our extensive checks, we might have missed the absolute best way to configure BART's pretraining continuation, thus resulting in sub-par domain adaptation performance.
That said, we wish to highlight the fact that three out of four metrics' best results are obtained by a model with domain adaptation, specifically the ROUGE-2, ROUGE-L, and BERTScore.

Size	Type	DA	Train	R1	R2	RL	BS
base	BART	no	std	31.71	9.71	21.59	61.60
base	LED	no	std	**35.24**	11.05	22.79	63.41
base	LSG	no	std	35.06	11.11	22.77	63.03
large	BART	no	std	31.38	9.57	21.31	61.51
large	LED	no	std	34.23	11.06	22.56	63.73
large	LSG	no	std	34.84	11.46	23.10	64.22
base	BART	yes	std	31.61	9.28	21.30	60.92
base	LED	yes	std	34.89	11.50	22.67	62.90
base	LSG	yes	std	33.51	10.58	21.85	62.65
large	BART	yes	std	32.15	9.93	21.59	61.92
large	LED	yes	std	34.48	11.44	**23.22**	**64.31**
large	LSG	yes	std	34.55	**11.61**	22.61	63.84

Table 5.2: The effect of different types of long attention on the summarization performance of the models.

In table 5.2 we compare the different kinds of Self-Attention used in the model encoders: the original quadratic one from BART [10], and its two extended variants. While the Longformer Self-Attention [6] variant (LED in the table) seems to work best with base-sized models, LSG [7] gets the highest results with large-sized ones, especially without domain adaptation.
LED looks like the best performer overall, obtaining three out of four best metric results (ROUGE-1, ROUGE-L, BERTScore), with LED_{LARGE} requiring slightly more specific training to consistently outperform LSG.

Size	Type	DA	Train	R1	R2	RL	BS
base	LSG	no	std	35.06	11.11	22.77	63.03
base	LSG	no	ext	34.71	11.02	22.58	63.13
base	LED	no	std	35.24	11.05	22.79	63.41
base	LED	no	ext	35.18	11.11	22.54	63.42
large	LSG	no	std	34.84	11.46	23.10	64.22
large	LSG	no	ext	**35.74**	11.61	23.13	63.91
large	LED	no	std	34.23	11.06	22.56	63.73
large	LED	no	ext	33.67	10.54	22.28	62.87
base	LSG	yes	std	33.51	10.58	21.85	62.65
base	LSG	yes	ext	33.67	10.20	21.55	62.97
base	LED	yes	std	34.89	11.50	22.67	62.90
base	LED	yes	ext	35.46	**11.67**	**23.27**	63.45
large	LSG	yes	std	34.55	11.61	22.61	63.84
large	LSG	yes	ext	34.81	11.11	22.51	63.78
large	LED	yes	std	34.48	11.44	23.22	**64.31**
large	LED	yes	ext	34.96	11.57	23.16	63.81

Table 5.3: The effect of the training set version of UniSum used for finetuning on the summarization performance of the models.

In table 5.3 we are interested in understanding in what cases the extended UniSum flavor works best. Even though all four metrics are improved together by using this variant only in one case (LED_{BASE} with domain adaptation), while it is common for it to instead improve only some of a model's metrics, we note that all ROUGE metrics' best values are obtained with a model finetuned on UniSum extended.

To further investigate the effectiveness of extended UniSum, we split the dataset's test set by course macrocategory, course category, course, and writing frequency[6]. Table 5.4 particularly shows a remarkable improvement in the ROUGE metrics' results when using the extended dataset with courses in the humanities field, which as is shown in figure 4.9b, correlates positively with a lower mean course writing frequency. This is in agreement with the results shown in the "infrequent writing" section of table 5.5, where especially models with domain adaptation highlight a significant performance improvement when also making use of the

[6]While we also provide the writing frequency for each lecture individually, we split the dataset by mean writing frequency over all the lectures of each course. The writing frequency is computed as $f_w = \frac{sentences\,classified\,as\,writing}{total\,sentences}$.

writing classification.

Size	Type	DA	Train	Test	R1	R2	RL	BS
base	LSG	no	std	hum	36.74	11.19	22.29	61.87
base	LSG	no	ext	hum	36.80	11.65	22.79	62.06
base	LED	no	std	hum	36.88	11.41	22.39	62.45
base	LED	no	ext	hum	36.98	11.26	22.35	62.22
large	LSG	no	std	hum	34.80	11.39	22.09	62.55
large	LSG	no	ext	hum	35.81	11.58	22.31	62.64
large	LED	no	std	hum	35.98	11.54	22.46	62.73
large	LED	no	ext	hum	34.47	10.59	21.64	61.62
base	LSG	yes	std	hum	34.65	10.65	21.24	61.65
base	LSG	yes	ext	hum	35.40	10.49	21.39	62.07
base	LED	yes	std	hum	36.10	11.43	22.28	61.69
base	LED	yes	ext	hum	**37.47**	**11.76**	**22.82**	62.27
large	LSG	yes	std	hum	35.40	11.26	22.10	62.82
large	LSG	yes	ext	hum	35.33	11.03	21.85	62.79
large	LED	yes	std	hum	34.88	10.80	22.21	**63.14**
large	LED	yes	ext	hum	36.33	11.60	22.78	62.78
base	LSG	no	std	sci	33.29	11.09	23.27	64.27
base	LSG	no	ext	sci	32.48	10.34	22.39	64.28
base	LED	no	std	sci	33.47	10.70	23.18	64.43
base	LED	no	ext	sci	33.32	10.96	22.70	64.69
large	LSG	no	std	sci	34.99	11.37	24.24	**65.99**
large	LSG	no	ext	sci	**35.69**	11.70	24.07	65.27
large	LED	no	std	sci	32.57	10.51	22.65	64.79
large	LED	no	ext	sci	32.67	10.42	23.00	64.20
base	LSG	yes	std	sci	32.33	10.56	22.49	63.72
base	LSG	yes	ext	sci	31.92	9.83	21.77	63.92
base	LED	yes	std	sci	33.76	11.61	23.06	64.19
base	LED	yes	ext	sci	33.35	11.52	23.81	64.70
large	LSG	yes	std	sci	33.46	11.97	23.20	64.92
large	LSG	yes	ext	sci	34.35	11.21	23.15	64.83
large	LED	yes	std	sci	34.10	**12.11**	**24.39**	65.57
large	LED	yes	ext	sci	33.50	11.45	23.60	64.90

Table 5.4: The effect of the training set version of UniSum used for finetuning on the summarization performance of the models on the test set split by macrocategory ($N_{humanities} = 82$, $N_{scientific} = 77$).

We note that using the extended UniSum flavor for courses in the scientific macrocategory does not favor performance in our testing. The same can be said for courses with a medium writing frequency, while those with a higher frequency unexpectedly also see an improvement when treated by models trained on extended UniSum in a few cases, especially with LSG_{LARGE} with no domain adaptation.

Size	**Type**	**DA**	**Train**	**Test**	**R1**	**R2**	**RL**	**BS**
base	LSG	no	std	freq	32.72	10.24	22.21	63.04
base	LSG	no	ext	freq	31.62	9.70	21.74	63.01
base	LED	no	std	freq	33.21	10.82	22.58	63.50
base	LED	no	ext	freq	33.55	10.62	22.38	63.76
large	LSG	no	std	freq	34.63	11.40	23.50	**64.99**
large	LSG	no	ext	freq	**35.59**	11.89	**23.91**	64.47
large	LED	no	std	freq	33.46	10.86	22.97	64.27
large	LED	no	ext	freq	31.56	10.23	21.81	62.97
base	LSG	yes	std	freq	31.97	10.91	21.87	62.90
base	LSG	yes	ext	freq	32.16	9.61	21.64	62.82
base	LED	yes	std	freq	33.44	11.37	22.39	63.41
base	LED	yes	ext	freq	33.22	11.36	23.46	63.92
large	LSG	yes	std	freq	33.52	**12.23**	23.10	64.22
large	LSG	yes	ext	freq	35.16	11.14	23.44	64.10
large	LED	yes	std	freq	33.39	12.06	23.65	64.58
large	LED	yes	ext	freq	33.39	11.51	23.23	64.07
base	LSG	no	std	med	33.60	**10.88**	**23.13**	64.23
base	LSG	no	ext	med	**33.90**	10.27	22.36	64.28
base	LED	no	std	med	33.54	9.58	22.21	64.36
base	LED	no	ext	med	33.70	10.11	21.79	64.42
large	LSG	no	std	med	31.61	9.71	22.33	64.41
large	LSG	no	ext	med	33.16	9.78	22.21	64.49
large	LED	no	std	med	31.15	9.05	21.05	63.77
large	LED	no	ext	med	31.30	8.61	22.04	63.36
base	LSG	yes	std	med	31.83	9.39	21.81	63.46
base	LSG	yes	ext	med	31.26	9.36	20.85	63.75
base	LED	yes	std	med	32.69	10.53	22.01	63.36
base	LED	yes	ext	med	32.70	10.67	22.54	63.73
large	LSG	yes	std	med	32.68	9.91	21.45	64.53
large	LSG	yes	ext	med	31.70	10.24	21.28	64.18
large	LED	yes	std	med	32.64	10.18	22.66	**65.18**
large	LED	yes	ext	med	33.31	9.84	22.62	64.51

Continues →

base	LSG	no	std	infreq	38.45	12.11	23.18	62.07
base	LSG	no	ext	infreq	38.39	12.80	23.39	62.34
base	LED	no	std	infreq	38.58	12.46	23.45	62.57
base	LED	no	ext	infreq	38.01	12.31	23.20	62.29
large	LSG	no	std	infreq	37.54	12.77	23.42	63.33
large	LSG	no	ext	infreq	37.87	12.79	23.18	62.93
large	LED	no	std	infreq	37.64	12.75	23.45	63.18
large	LED	no	ext	infreq	37.31	12.25	22.79	62.38
base	LSG	yes	std	infreq	36.41	11.31	21.77	61.77
base	LSG	yes	ext	infreq	37.17	11.38	22.15	62.48
base	LED	yes	std	infreq	38.02	12.35	23.43	62.05
base	LED	yes	ext	infreq	**39.79**	12.68	**23.77**	62.78
large	LSG	yes	std	infreq	36.88	12.33	22.95	62.92
large	LSG	yes	ext	infreq	37.10	11.74	22.58	63.15
large	LED	yes	std	infreq	36.97	11.94	23.27	**63.37**
large	LED	yes	ext	infreq	37.75	**12.90**	23.59	62.99

Table 5.5: The effect of the training set version of UniSum used for finetuning on the summarization performance of the models on the test set split by writing frequency averaged across all course lectures ($N_{frequent} = 55$, $N_{medium} = 46$, $N_{infrequent} = 58$).

We deem the 976-rows-long table with 16 tests for each of the 61 courses present in the test set too detailed to include in this thesis[7]; we instead show a middle ground level of aggregation between it and the course macrocategories (table 5.4), in the form of courses combined by specific category, in table 5.6. At a high level, the course categories that most benefit from the use of the extended UniSum variant are business, economics, history, philosophy, physics, and psychology.
In table 5.6 we use the following abbreviations: bio ↔ biology, biz ↔business, cs ↔ computer science, econ ↔ economics, eng ↔ engineering, hist ↔ history, lit ↔ literature, math ↔ mathematics, phil ↔ philosophy, phys ↔ physics, pol ↔ politics, psy ↔ psychology, soc ↔ social studies.

Size	Type	DA	Train	Test	R1	R2	RL	BS
base	LSG	no	std	arts	34.84	10.90	23.72	65.12
base	LSG	no	ext	arts	35.20	10.88	24.53	64.48
base	LED	no	std	arts	33.68	10.85	22.92	65.06

Continues →

[7]We make this table available in our repository.

base	LED	no	ext	arts	34.76	10.79	20.20	64.08
large	LSG	no	std	arts	31.51	9.74	21.11	64.76
large	LSG	no	ext	arts	30.96	8.11	19.92	65.58
large	LED	no	std	arts	32.30	7.36	20.25	63.44
large	LED	no	ext	arts	29.98	6.67	19.80	61.66
base	LSG	yes	std	arts	33.60	**11.80**	**26.58**	63.96
base	LSG	yes	ext	arts	**35.33**	9.46	20.65	**65.94**
base	LED	yes	std	arts	30.78	10.88	22.08	63.57
base	LED	yes	ext	arts	33.98	10.76	22.86	62.50
large	LSG	yes	std	arts	33.35	8.32	20.03	65.83
large	LSG	yes	ext	arts	29.13	10.27	20.32	64.20
large	LED	yes	std	arts	31.52	7.53	20.51	64.79
large	LED	yes	ext	arts	31.46	6.61	19.53	63.41
base	LSG	no	std	bio	34.49	14.95	27.62	71.45
base	LSG	no	ext	bio	36.95	14.37	28.50	71.04
base	LED	no	std	bio	31.40	10.84	24.01	69.74
base	LED	no	ext	bio	37.23	14.00	26.69	71.57
large	LSG	no	std	bio	38.71	17.60	33.15	**72.92**
large	LSG	no	ext	bio	37.00	13.56	27.55	70.63
large	LED	no	std	bio	29.41	9.03	23.17	69.71
large	LED	no	ext	bio	33.20	11.77	27.09	69.33
base	LSG	yes	std	bio	34.21	13.32	26.65	70.88
base	LSG	yes	ext	bio	34.35	13.77	24.40	70.91
base	LED	yes	std	bio	39.51	17.28	30.04	70.81
base	LED	yes	ext	bio	34.99	14.40	26.36	71.53
large	LSG	yes	std	bio	33.99	14.34	26.85	71.09
large	LSG	yes	ext	bio	35.29	14.06	26.16	69.54
large	LED	yes	std	bio	**41.71**	**21.82**	**34.75**	72.79
large	LED	yes	ext	bio	35.58	13.55	27.39	70.70
base	LSG	no	std	biz	28.29	6.85	20.72	63.09
base	LSG	no	ext	biz	25.14	6.07	20.86	62.01
base	LED	no	std	biz	29.46	6.35	21.46	63.98
base	LED	no	ext	biz	28.22	7.00	21.38	64.06
large	LSG	no	std	biz	28.49	9.44	23.11	66.79
large	LSG	no	ext	biz	**34.73**	**13.42**	**28.77**	**68.50**
large	LED	no	std	biz	31.96	9.16	23.73	66.31
large	LED	no	ext	biz	28.89	8.61	24.19	64.03
base	LSG	yes	std	biz	30.28	11.97	21.54	63.20
base	LSG	yes	ext	biz	30.67	9.74	23.60	63.31

Continues →

base	LED	yes	std	biz	29.27	10.73	21.77	63.71
base	LED	yes	ext	biz	21.78	5.83	18.81	62.45
large	LSG	yes	std	biz	30.59	10.04	22.81	65.38
large	LSG	yes	ext	biz	30.85	9.93	23.73	65.64
large	LED	yes	std	biz	29.56	10.80	23.33	65.32
large	LED	yes	ext	biz	31.88	8.88	26.47	66.25
base	LSG	no	std	cs	32.20	11.00	24.03	64.69
base	LSG	no	ext	cs	31.63	10.36	22.77	64.91
base	LED	no	std	cs	32.30	9.79	22.84	64.73
base	LED	no	ext	cs	31.34	9.35	22.16	64.39
large	LSG	no	std	cs	31.54	9.68	22.62	**66.20**
large	LSG	no	ext	cs	33.47	9.91	23.40	65.42
large	LED	no	std	cs	31.43	9.74	22.47	65.65
large	LED	no	ext	cs	31.04	8.56	22.19	64.51
base	LSG	yes	std	cs	30.87	8.92	21.40	63.74
base	LSG	yes	ext	cs	31.53	9.18	21.98	64.57
base	LED	yes	std	cs	**33.49**	**11.04**	23.50	64.10
base	LED	yes	ext	cs	31.71	10.13	23.53	64.53
large	LSG	yes	std	cs	32.04	10.28	22.34	65.07
large	LSG	yes	ext	cs	32.07	9.73	21.76	65.30
large	LED	yes	std	cs	33.16	9.96	**24.08**	65.71
large	LED	yes	ext	cs	32.48	10.22	23.12	64.98
base	LSG	no	std	econ	28.91	8.05	17.00	59.41
base	LSG	no	ext	econ	28.39	7.58	17.62	60.50
base	LED	no	std	econ	30.72	9.01	19.49	61.43
base	LED	no	ext	econ	30.54	8.58	18.52	**62.20**
large	LSG	no	std	econ	27.91	6.77	16.45	60.66
large	LSG	no	ext	econ	30.88	7.47	19.51	61.83
large	LED	no	std	econ	28.38	7.54	17.46	60.71
large	LED	no	ext	econ	27.41	6.71	17.59	61.37
base	LSG	yes	std	econ	28.55	9.44	18.46	59.68
base	LSG	yes	ext	econ	29.32	9.49	18.72	61.02
base	LED	yes	std	econ	29.63	8.66	17.87	59.76
base	LED	yes	ext	econ	30.92	**9.71**	19.64	61.60
large	LSG	yes	std	econ	29.64	8.87	18.45	62.18
large	LSG	yes	ext	econ	29.50	7.38	18.13	61.79
large	LED	yes	std	econ	26.87	8.48	17.58	61.87
large	LED	yes	ext	econ	**32.70**	8.80	**20.01**	61.24

Continues →

base	LSG	no	std	eng	27.76	4.15	16.95	60.05
base	LSG	no	ext	eng	28.61	5.38	17.35	62.43
base	LED	no	std	eng	30.75	5.22	20.07	63.76
base	LED	no	ext	eng	30.63	5.26	18.25	62.59
large	LSG	no	std	eng	28.63	4.75	17.96	62.90
large	LSG	no	ext	eng	30.83	5.11	18.82	62.35
large	LED	no	std	eng	25.42	3.89	16.52	61.46
large	LED	no	ext	eng	31.59	8.06	**21.70**	63.05
base	LSG	yes	std	eng	29.50	5.47	17.45	60.69
base	LSG	yes	ext	eng	23.57	3.62	15.29	59.53
base	LED	yes	std	eng	28.14	6.76	17.52	61.35
base	LED	yes	ext	eng	**32.47**	6.96	21.14	63.88
large	LSG	yes	std	eng	31.48	**9.98**	21.54	**65.08**
large	LSG	yes	ext	eng	30.18	4.86	18.16	62.46
large	LED	yes	std	eng	28.80	5.52	19.68	63.37
large	LED	yes	ext	eng	32.23	7.55	18.70	64.86
base	LSG	no	std	hist	37.90	10.82	22.68	61.41
base	LSG	no	ext	hist	**38.35**	11.47	**23.40**	62.24
base	LED	no	std	hist	37.22	10.94	22.45	61.82
base	LED	no	ext	hist	37.82	11.24	22.51	61.85
large	LSG	no	std	hist	35.95	11.61	22.42	62.52
large	LSG	no	ext	hist	37.66	11.74	23.30	**63.06**
large	LED	no	std	hist	37.13	**12.14**	23.03	63.00
large	LED	no	ext	hist	34.96	10.85	21.79	62.05
base	LSG	yes	std	hist	34.88	9.51	20.38	61.51
base	LSG	yes	ext	hist	36.22	9.88	21.02	61.92
base	LED	yes	std	hist	36.87	11.36	22.40	61.22
base	LED	yes	ext	hist	38.16	11.48	22.67	61.61
large	LSG	yes	std	hist	35.34	11.32	22.32	62.01
large	LSG	yes	ext	hist	35.42	11.13	22.27	62.86
large	LED	yes	std	hist	35.83	10.62	22.96	62.59
large	LED	yes	ext	hist	36.49	11.66	22.28	62.37
base	LSG	no	std	lit	42.46	14.43	26.02	62.87
base	LSG	no	ext	lit	42.52	**15.87**	26.77	63.02
base	LED	no	std	lit	42.76	14.97	26.13	62.83
base	LED	no	ext	lit	41.97	14.05	26.43	63.17
large	LSG	no	std	lit	40.49	14.87	26.18	63.26
large	LSG	no	ext	lit	40.27	13.72	24.30	62.63
large	LED	no	std	lit	42.52	15.14	**27.05**	63.83

Continues →

large	LED	no	ext	lit	40.79	12.67	24.77	61.45
base	LSG	yes	std	lit	40.25	14.20	23.88	62.45
base	LSG	yes	ext	lit	40.85	13.27	23.91	61.62
base	LED	yes	std	lit	42.53	14.49	26.11	62.50
base	LED	yes	ext	lit	**42.96**	14.69	26.50	62.91
large	LSG	yes	std	lit	40.84	13.43	24.82	63.29
large	LSG	yes	ext	lit	41.44	13.70	24.95	63.79
large	LED	yes	std	lit	39.49	13.79	24.97	**64.40**
large	LED	yes	ext	lit	41.27	14.40	26.53	64.30
base	LSG	no	std	math	34.93	12.23	22.32	63.73
base	LSG	no	ext	math	32.07	11.07	20.25	62.90
base	LED	no	std	math	32.40	13.39	23.35	63.58
base	LED	no	ext	math	33.45	12.37	21.91	63.87
large	LSG	no	std	math	**38.28**	13.21	**24.92**	**65.60**
large	LSG	no	ext	math	36.33	13.92	22.88	64.42
large	LED	no	std	math	34.80	12.73	23.48	64.08
large	LED	no	ext	math	33.51	12.24	21.12	62.93
base	LSG	yes	std	math	34.92	12.97	23.86	63.65
base	LSG	yes	ext	math	32.87	10.42	20.77	62.93
base	LED	yes	std	math	34.63	12.68	21.64	64.28
base	LED	yes	ext	math	33.33	13.37	23.39	63.74
large	LSG	yes	std	math	34.21	13.29	22.61	63.90
large	LSG	yes	ext	math	37.12	13.06	23.55	63.37
large	LED	yes	std	math	35.42	13.83	23.30	64.52
large	LED	yes	ext	math	36.18	**14.52**	24.67	64.29
base	LSG	no	std	phil	38.97	13.41	23.35	62.08
base	LSG	no	ext	phil	39.97	13.87	22.63	61.94
base	LED	no	std	phil	37.30	11.84	21.41	62.88
base	LED	no	ext	phil	38.70	13.26	22.18	62.45
large	LSG	no	std	phil	36.44	13.50	22.03	62.83
large	LSG	no	ext	phil	37.75	**15.24**	22.89	61.70
large	LED	no	std	phil	38.26	13.85	21.99	62.04
large	LED	no	ext	phil	37.56	13.08	21.41	61.65
base	LSG	yes	std	phil	35.98	11.54	21.30	61.50
base	LSG	yes	ext	phil	36.80	11.08	22.29	63.09
base	LED	yes	std	phil	37.85	13.24	23.30	61.92
base	LED	yes	ext	phil	**41.85**	14.39	24.29	63.45
large	LSG	yes	std	phil	36.47	13.63	23.56	63.03
large	LSG	yes	ext	phil	37.75	12.42	22.37	62.91

Continues →

large	LED	yes	std	phil	38.50	13.59	23.26	**63.47**
large	LED	yes	ext	phil	39.86	**15.24**	**24.63**	63.11
base	LSG	no	std	phys	35.60	11.93	24.46	63.72
base	LSG	no	ext	phys	34.92	10.98	23.91	63.68
base	LED	no	std	phys	38.04	11.91	24.38	63.50
base	LED	no	ext	phys	36.18	13.16	24.60	64.53
large	LSG	no	std	phys	37.73	12.43	24.99	64.91
large	LSG	no	ext	phys	**39.34**	**13.46**	**26.04**	64.76
large	LED	no	std	phys	35.51	12.51	23.91	63.82
large	LED	no	ext	phys	34.46	11.62	24.72	63.83
base	LSG	yes	std	phys	32.91	10.81	22.93	62.68
base	LSG	yes	ext	phys	34.19	11.26	23.62	63.55
base	LED	yes	std	phys	34.51	11.23	23.82	63.25
base	LED	yes	ext	phys	36.74	12.85	25.53	64.30
large	LSG	yes	std	phys	36.11	13.23	24.11	63.61
large	LSG	yes	ext	phys	36.36	12.93	25.51	64.86
large	LED	yes	std	phys	34.44	12.81	24.00	**64.94**
large	LED	yes	ext	phys	32.37	11.43	23.21	63.33
base	LSG	no	std	pol	37.65	10.99	22.58	65.33
base	LSG	no	ext	pol	35.82	13.43	23.86	65.15
base	LED	no	std	pol	**41.11**	13.51	24.62	65.10
base	LED	no	ext	pol	39.39	14.17	25.25	64.22
large	LSG	no	std	pol	38.04	12.65	24.48	65.37
large	LSG	no	ext	pol	38.50	14.42	23.48	64.61
large	LED	no	std	pol	36.82	11.72	24.26	65.37
large	LED	no	ext	pol	35.94	**15.02**	24.09	63.64
base	LSG	yes	std	pol	37.11	11.26	**25.42**	64.64
base	LSG	yes	ext	pol	33.66	12.17	23.42	63.91
base	LED	yes	std	pol	38.43	11.90	23.07	66.12
base	LED	yes	ext	pol	35.85	11.01	23.40	65.55
large	LSG	yes	std	pol	38.81	14.54	23.46	65.72
large	LSG	yes	ext	pol	34.37	13.01	21.94	63.37
large	LED	yes	std	pol	35.85	12.32	23.69	**66.34**
large	LED	yes	ext	pol	34.44	12.47	23.92	64.26
base	LSG	no	std	psy	24.41	7.69	18.99	65.80
base	LSG	no	ext	psy	22.50	2.85	15.20	61.90
base	LED	no	std	psy	30.08	**9.46**	20.39	67.50
base	LED	no	ext	psy	22.95	3.95	16.85	62.13

Continues →

large	LSG	no	std	psy	25.19	7.95	20.09	65.97
large	LSG	no	ext	psy	21.93	5.92	19.18	66.19
large	LED	no	std	psy	25.19	5.56	17.41	67.60
large	LED	no	ext	psy	27.12	8.30	20.33	66.41
base	LSG	yes	std	psy	22.88	7.32	18.03	64.39
base	LSG	yes	ext	psy	29.37	9.37	19.44	66.55
base	LED	yes	std	psy	25.39	8.30	20.94	66.50
base	LED	yes	ext	psy	31.11	7.99	18.61	67.73
large	LSG	yes	std	psy	26.41	8.55	21.07	67.54
large	LSG	yes	ext	psy	**34.06**	7.58	22.06	68.32
large	LED	yes	std	psy	31.54	8.67	23.26	68.29
large	LED	yes	ext	psy	31.23	8.94	**24.01**	**69.38**
base	LSG	no	std	soc	27.73	4.28	15.84	57.00
base	LSG	no	ext	soc	27.35	3.61	16.28	55.60
base	LED	no	std	soc	28.53	5.11	14.93	58.54
base	LED	no	ext	soc	**29.23**	3.60	15.86	57.15
large	LSG	no	std	soc	21.67	3.43	15.76	56.87
large	LSG	no	ext	soc	23.31	4.84	15.70	57.47
large	LED	no	std	soc	25.18	**5.54**	15.95	57.47
large	LED	no	ext	soc	24.80	4.31	**17.28**	56.09
base	LSG	yes	std	soc	25.38	5.06	14.04	57.34
base	LSG	yes	ext	soc	26.90	4.68	16.84	56.47
base	LED	yes	std	soc	24.48	3.82	14.45	56.50
base	LED	yes	ext	soc	27.69	4.99	15.27	57.02
large	LSG	yes	std	soc	28.95	4.30	16.68	**58.88**
large	LSG	yes	ext	soc	27.83	4.87	16.21	56.57
large	LED	yes	std	soc	23.73	2.62	14.42	56.86
large	LED	yes	ext	soc	27.14	4.48	15.37	57.65

Table 5.6: The effect of the training set version of UniSum used for finetuning on the summarization performance of the models on the test set split by course category ($N_{arts} = 5$, $N_{biology} = 6$, $N_{business} = 3$, $N_{chemistry} = 0$, $N_{computer\,science} = 24$, $N_{economics} = 9$, $N_{engineering} = 7$, $N_{history} = 29$, $N_{literature} = 16$, $N_{mathematics} = 18$, $N_{philosophy} = 11$, $N_{physics} = 19$, $N_{politics} = 5$, $N_{psychology} = 2$, $N_{social\,studies} = 5$).

Chapter 6

Conclusions

At last, it is time to draw the conclusions of this several-months-long master's thesis.

6.1 Project Review

We initially began building upon the connection between the ideas that there has recently been an upsurge in availability of university-level lectures, and that the quantity of information that higher-education students need to handle and assimilate is running more and more the risk of becoming overwhelming. Driven by the relatively recent revolution in the field of natural language processing that has been brought by the Transformer, after elaborating on these concepts, we found a contact point which had not received the quantity of attention we thought it deserved; we have thus devised a methodology to train a deep network (or in our case, as we have shown in section 4.4, more than twenty of them) to automate the process of extrapolating and re-elaborating the most essential information from a university lecture.

This has involved creating a dataset of transcript-summary pairs for each lecture with the largest quantity of high-quality, open-licensed, and manually curated university material. We call this the UniSum dataset. It contains 1,583 data points and we release it in two variants: standard, which does not include additional information, and extended, which adds a video-based binary classification for each sentence of the transcript of whether the lecture speaker was writing while saying it, and can be considered in some way multimodal (more details in § 4.2.2).

We have also created a larger transcript-only dataset, OpenULTD, to aid the BART models we start training from, previously trained on general written English

text, in their transition to the more complex spoken text domain the UniSum transcripts are part of. OpenULTD is made of the original 9,229 transcripts of the pre-existing VT-SSum dataset, plus our additional 5,448 transcripts from other reputable sources, for a total of 14,677 data samples (details in § 4.2.1).

6.2 On the Generated Summaries

Having chosen LED$_{BASE}$ with domain adaptation trained on UniSum extended as the best overall model, we analyze the summaries it generates given the UniSum test set as input[1].

We empirically notice that the model successfully gains a satisfying proficiency level in the usage of complex and context-specific words, and has the ability to abstract the concepts contained in the input transcript to convey the original meaning in an effective manner, although lacking the extensive prior knowledge of the subject typical of a professor, which allows them to provide higher-level abstractions and connections. An example of these skills would be: "In this lecture, Dr. Winston discusses the miracles of learning, including the Sussman-Yip machine. He begins with a discussion of phonological mechanisms and examples. He then moves on to talk about some naive biological mimicry".
We also want to highlight the flexibility of our model. As it is BART-derived, it shares its same 50,265-tokens-long vocabulary[2], which is a good amount of different textual expressions to be able to distinguish, but it is not infinite. Nonetheless, our model appears to understand the importance and correct usage of tokens that are not included in its vocabulary, such as in this instance, where it uses the unicode character for the Greek letter psi[3], which it encountered in the input transcript: "The allowed energy states of a particle in a ring are discussed. The allowed energy of a free particle is quantized and the wave function ψ(x) is explained. The quantization of energy is explained using the Schrödinger equation. Finally, the allowed wave functions are explained".

As expected, our model also has some limitations. We especially notice the following ones:

- duplicates in lists;

[1] A selection of these is available in table A.1.

[2] 50,269 if we include our additional tokens for sentence classification.

[3] We replace it in this document with its corresponding LaTeX character generated with `$\psi$`.

- name hallucinations or confusions, particularly with professor last names and character names from literary works which are the lecture's subject;
- errors with numbers and years, mostly when written in digits;
- confusion between beginning and end of a course or lecture. This does not affect the contents of the summary, just words such as "begins" and "concludes";
- unawareness of gender. This means that the model may confuse *e.g.* a male professor for a woman and viceversa.

Despite these, we feel the generated summaries may still be used in an informal setting, for example to give students the ability to quickly check what are the main topics of a lecture or of a course, without the expectations of an impeccable wording.

6.3 Possible Improvements

Our efforts have brought us to an acceptable result, but it is apparent by reading the previous section and the table in the appendix that more work can be done in different directions to perfect the summarization capabilities of our models.

First, to improve our very same models, an investigation on the underlying reasons why LED_{LARGE} has in many test cases a worse performance than LED_{BASE} would be in order. The researcher who does this would most likely obtain a model that outperforms all of ours.
A second line of work that we left uncharted due to time constraints is exploring the benefits of giving the model an input that contains contextual information on top of the transcript, such as the course description or the course and lecture titles.
A relatively simple exploration could also be done by replacing the BART model with different pretrained language models, such as PEGASUS [18], PEGASUS-X [112], T5 [17], or others. The performance could even be measured against one of the recent large language models to get an idea of the purely summarization-oriented models' efficiency (*e.g.* in terms of model parameters with respect to the obtained ROUGE scores).
Finally, the same approach could be expanded to languages other than English: for example, BART has recently been made available in Italian [138]; the main difficulty would be creating a dataset equivalent to UniSum in the target language.

We show a few statistics about our work in table 6.1.

Statistic		Value
SLOC[4]	Datasets | Python	3,009
	Experiments | Python	5,091
	Thesis | LaTeX	4,238
	Total	12,338
Compute Time	Domain Adaptation	24 hours | 1 day
	UniSum Extension	69 hours | 3 days
	Finetuning	450 hours | 19 days
	Testing	150 hours | 6 days
	Total	693 hours | 29 days

Table 6.1: A synthetic summary of our efforts.

We release our code, datasets, and models at `https://github.com/e-caste/masters-thesis`.

[4]Source Lines Of Code, excluding comments, blank lines, and whitespace.

Appendix A

Human vs. AI: a Summarization Comparison

In this appendix chapter we want to give the reader access to a comparison of some of the summaries generated from the UniSum test set with the model chosen as the best, that is LED_{BASE} with domain adaptation finetuned on the extended UniSum flavor.

Table A.1 contains 8 out of 159 blocks[1], separated by thick horizontal rules, each with three rows[2]:

1. the first row, in smaller print, contains all the metadata useful to get the context for the lecture, including its subset, the writing frequency classification of the course it belongs to, its macrocategory, category, the title of the course it belongs to, and its title;
2. the second row contains the summary generated by our trained model;
3. and the third row contains the original summary.

We note that absolutely no modification was applied to the summaries, both generated and original, unless there was a specific clash with the formatting rules of LaTeX (this occurred in around ten cases in total, for special characters). We also avoid any transformations of the course and lecture titles.

[1]The full table is available in our repository.

[2]We choose this format over a two-column table for readability purposes.

Yale | Medium Frequent Writing | Scientific | Physics | Frontiers/Controversies in Astrophysics with Charles Bailyn | 3. Our Solar System and the Pluto Problem

Planets and exoplanets are introduced and classified according to their orbital period. Different categories of objects are defined, such as planets, comets, asteroids, and other celestial bodies. The lecture ends with a discussion of the geography of the Solar System and how it evolved.

Class begins with a review of the first problem set. Newton's Third Law is applied in explaining how exoplanets are found. An overview of the Solar System is given; each planet is presented individually and its special features are highlighted. Astronomy is discussed as an observational science, and the subject of how to categorize objects in the Solar System is addressed. The Pluto controversy is given special attention and both sides of the argument regarding its status are considered.

MIT | Frequent Writing | Scientific | Computer Science | Artificial Intelligence | Lecture Videos LECTURE 14: LEARNING: SPARSE SPACES, PHONOLOGY

In this lecture, Dr. Winston discusses the miracles of learning, including the Sussman-Yip machine. He begins with a discussion of phonological mechanisms and examples. He then moves on to talk about some naive biological mimicry.

Why do "cats" and "dogs" end with different plural sounds, and how do we learn this? We can represent this problem in terms of distinctive features, and then generalize. We end this lecture with a brief discussion of how to approach AI problems.

MIT | Frequent Writing | Scientific | Mathematics | Graph Theory and Additive Combinatorics | Video Lectures LECTURE 18: ROTH'S THEOREM I: FOURIER ANALYTIC PROOF OVER FINITE FIELD

In this lecture, Prof. Zhao discusses the Fourier analytic proof of Roth's theorem, using similarities, graph regularity lemma, and the triangle removal lemma.

The finite field model is a nice sandbox for methods and tools in additive combinatorics. Professor Zhao explains how to use Fourier analysis to prove the analog of Roth's theorem in the finite field setting.

Yale | Infrequent Writing | Humanities | History | Yale University's Lectures: The Early Middle Ages, 284-1000 | 07. Barbarian Kingdoms

In this lecture, Professor Freedman discusses the collapse of the Roman Empire in 530 and the rise of the Visigoths in the West. The barbarian kingdoms are discussed in detail, along with the reasons for their existence and their resistance to Roman rule. The lecture ends with a discussion of the Eastern Roman Empire and the Burgundian Code.

Continues →

In this lecture, Professor Freedman considers the various barbarian kingdoms that replaced the Western Roman Empire. Oringinally the Roman reaction to these invaders had been to accommodate them, often recruiting them for the Roman army and settling them on Roman land. Now, however, they were the rulers of the previously Roman lands of the West. These tribes included the Ostrogoths and Visigoths in Italy, the Franks in Gaul, and the Vandals in North Africa. As most sources about these groups come from the Roman perspective, it's unclear how coherent each group was. In general, the barbarian groups characterized by disorganization, internal fighting and internecine feuds, and lack of economic development. Professor Freedman closes with some remarks on the Burgundian Code as evidence of barbarian society and institutions.

MIT | Medium Frequent Writing | Scientific | Biology | Introductory Biology | Lecture Videos LECTURE 17: GENOMES AND DNA SEQUENCING

In this lecture, Prof. Martin talks about how knowing the DNA sequence can be helpful in identifying a gene through functional complementation. He then talks about more modern techniques to sequencing, including Sanger sequencing.

Professor Martin talks about DNA sequencing and why it is helpful to know the DNA sequence, followed by linkage mapping and then the different methods of sequencing DNA.

Yale | Infrequent Writing | Humanities | Literature | Dante in Translation with Giuseppe Mazzotta | 5. Inferno IX, X, XI

Professor Mazzotta begins with a discussion of the cantos IX-X. The pilgrim's encounter with the so-called heretics and the guide Virgil leads him to the city of Dis. The lecture then moves on to discuss the relationship between the pilgrim and the philosopher Guido Cavalcanti. The relationship between these two types of allegories is discussed in detail in the context of Inferno IX, where Dante's anti-Pagans, neo-Platonists, and theologians are introduced. The discussion of Dante's failure to master his own will is continued in the second canto of Inferno, where the poet enters the City of Dis, a soulless city described by the three Erinyes of Greek mythology: Alecto, Tesiphone, and Megaera. This lecture concludes with the discussion of Purgatory IX.

In this lecture, Professor Mazzotta discusses Inferno IX-XI. An impasse at the entrance to the City of Dis marks Virgil's first failure in his role as guide (Inferno IX). The invocation of Medusa by the harpies that descend while they wait for divine aid elicits Dante's first address to the reader. The question of literary mediation, posed in the previous lecture in the context of Inferno V, is explored further, and the distinction Dante draws between the "allegory of poets" and the "allegory of theologians" is introduced. Inferno X is read with a view to the uniqueness of the sin it deals with - heresy. The philosophical errors of the shades encountered here, Farinata and Cavalcante, are tied to the political turmoil they prophecy for Florence. From the disorder of the earthly city, Dante moves on to the order on its infernal counterpart, mapped by Virgil in Inferno XI. The moral system of Dante's Hell is then discussed with a view to its classical antecedents.

Yale | Frequent Writing | Scientific | Physics | Fundamentals of Physics II with Ramamurti Shankar | 22. Quantum mechanics IV: Measurement theory, states of definite energy

---

Continues →

The allowed energy states of a particle in a ring are discussed. The allowed energy of a free particle is quantized and the wave function ψ(x) is explained. The quantization of energy is explained using the Schrödinger equation. Finally, the allowed wave functions are explained.
It is shown how to extract the odds for getting different values of momentum from a generic wave function by writing it as a sum over functions of definite momentum. A recipe is given for finding states of definite energy, which requires solving a differential equation that depends on what potential the particle is experiencing. The particle in a box is considered and the allowed energies derived.
Yale | Infrequent Writing | Humanities | Economics | Financial Markets (2008) with Robert Shiller | 23. Options Markets
In this lecture, Professor Shiller discusses options. Options are traded on the Chicago Mercantile Exchange (CME) as a hedging mechanism against possible counterparty risks. The lecture begins with a discussion of the Black-Scholes formula, which is used to price single-family home price options. He then discusses the arbitrage relation between the stock price and the exercise date. He ends with an example of an option exercised under the condition of no arbitrage opportunity.
Options introduce an essential nonlineary into portfolio management. They are contracts between buyers and writers, who agree on exercise prices and dates at which the buyer can buy or sell the underlying (such as a stock). Options are priced based on the price and volatility of the underlying asset as well as the duration of the option contract. The Black-Scholes options pricing model is one of the most famous equations in finance and offers a useful first approximation for prices for option contracts. Options exchanges and futures exchanges both are involved in creating a liquid and transparent market for options. Options are not just for stocks; they are also important for other asset classes, such as real estate.

Table A.1: A comparison of generated summaries with their original counterparts for eight lectures of the UniSum test set ($N = 159$), obtained with LED$_{BASE}$ with domain adaptation trained on the extended UniSum variant.

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Ł ukasz Kaiser, and Illia Polosukhin. «Attention is All you Need». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on pp. 1, 5, 21, 23, 24, 26, 27, 33, 37).

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805 (cit. on pp. 1, 5, 27–29, 33, 35, 45, 49, 60).

[3] Wikimedia Foundation. *Wikimedia Downloads*. URL: https://dumps.wikimedia.org (cit. on pp. 2, 29, 49).

[4] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. *Expanding Language-Image Pretrained Models for General Video Recognition*. 2022. DOI: 10.48550/ARXIV.2208.02816. URL: https://arxiv.org/abs/2208.02816 (cit. on pp. 2, 31, 41, 44, 51, 66, 69).

[5] Tengchao Lv, Lei Cui, Momcilo Vasilijevic, and Furu Wei. *VT-SSum: A Benchmark Dataset for Video Transcript Segmentation and Summarization*. 2021. DOI: 10.48550/ARXIV.2106.05606. URL: https://arxiv.org/abs/2106.05606 (cit. on pp. 2, 42, 43, 45, 52).

[6] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. DOI: 10.48550/ARXIV.2004.05150. URL: https://arxiv.org/abs/2004.05150 (cit. on pp. 2, 31, 37–40, 46, 50, 89).

[7] Charles Condevaux and Sébastien Harispe. *LSG Attention: Extrapolation of pretrained Transformers to long sequences*. 2022. DOI: 10.48550/ARXIV.2210.15497. URL: https://arxiv.org/abs/2210.15497 (cit. on pp. 2, 31, 38, 50, 89).

[8] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013 (cit. on pp. 3, 83, 84).

[9] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2019. DOI: 10.48550/ARXIV.1904.09675. URL: https://arxiv.org/abs/1904.09675 (cit. on pp. 3, 85).

[10] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. DOI: 10.48550/ARXIV.1910.13461. URL: https://arxiv.org/abs/1910.13461 (cit. on pp. 3, 5, 31, 32, 34–36, 38, 49, 50, 60, 67, 89).

[11] Alfredo Canziani and Yann LeCun. *NYU Deep Learning, Spring 2021*. https://github.com/Atcold/NYU-DLSP21. [Online; accessed January 2023]. 2021 (cit. on pp. 5, 21, 24).

[12] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. *Deep contextualized word representations*. 2018. DOI: 10.48550/ARXIV.1802.05365. URL: https://arxiv.org/abs/1802.05365 (cit. on pp. 5, 27).

[13] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. *Improving Language Understanding by Generative Pre-Training*. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. 2018. URL: https://openai.com/blog/language-unsupervised (cit. on pp. 5, 27, 28).

[14] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692 (cit. on pp. 5, 34, 38, 85).

[15] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2019. DOI: 10.48550/ARXIV.1906.08237. URL: https://arxiv.org/abs/1906.08237 (cit. on p. 5).

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. 2019. URL: https://openai.com/blog/better-language-models (cit. on p. 5).

[17] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2019. DOI: 10.48550/ARXIV.1910.10683. URL: https://arxiv.org/abs/1910.10683 (cit. on pp. 5, 32, 33, 103).

[18] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. 2019. DOI: 10.48550/ARXIV.1912.08777. URL: https://arxiv.org/abs/1912.08777 (cit. on pp. 5, 32–34, 39, 49, 103).

[19] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. DOI: 10.48550/ARXIV.2005.14165. URL: https://arxiv.org/abs/2005.14165 (cit. on pp. 5, 7).

[20] Manzil Zaheer et al. «Big Bird: Transformers for Longer Sequences». In: (2020). DOI: 10.48550/ARXIV.2007.14062. URL: https://arxiv.org/abs/2007.14062 (cit. on pp. 5, 27, 38).

[21] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. 2021. DOI: 10.48550/ARXIV.2107.03374. URL: https://arxiv.org/abs/2107.03374 (cit. on p. 5).

[22] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. DOI: 10.48550/ARXIV.2203.02155. URL: https://arxiv.org/abs/2203.02155 (cit. on p. 5).

[23] Kurt Shuster et al. *BlenderBot 3: a deployed conversational agent that continually learns to responsibly engage*. 2022. DOI: 10.48550/ARXIV.2208.03188. URL: https://arxiv.org/abs/2208.03188 (cit. on pp. 5, 7).

[24] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. *End-to-End Object Detection with Transformers*. 2020. DOI: 10.48550/ARXIV.2005.12872. URL: https://arxiv.org/abs/2005.12872 (cit. on p. 5).

[25] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: 10.48550/ARXIV.2010.11929. URL: https://arxiv.org/abs/2010.11929 (cit. on p. 5).

[26] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. DOI: 10.48550/ARXIV.2103.00020. URL: https://arxiv.org/abs/2103.00020 (cit. on pp. 5, 41).

[27] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. *Zero-Shot Text-to-Image Generation*. 2021. DOI: 10.48550/ARXIV.2102.12092. URL: https://arxiv.org/abs/2102.12092 (cit. on p. 5).

[28] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. DOI: 10.48550/ARXIV.2204.06125. URL: https://arxiv.org/abs/2204.06125 (cit. on p. 5).

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. «High-Resolution Image Synthesis With Latent Diffusion Models». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10684–10695 (cit. on p. 5).

[30] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. *HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units*. 2021. DOI: 10.48550/ARXIV.2106.07447. URL: https://arxiv.org/abs/2106.07447 (cit. on p. 5).

[31] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. *Robust Speech Recognition via Large-Scale Weak Supervision*. 2022. DOI: 10.48550/ARXIV.2212.04356. URL: https://arxiv.org/abs/2212.04356 (cit. on p. 5).

[32] Raúl Rojas. «The Computer Programs of Charles Babbage». In: *IEEE Annals of the History of Computing* 43.1 (2021), pp. 6–18. DOI: 10.1109/MAHC.2020.3045717 (cit. on p. 6).

[33] Alan Mathison Turing. «Computing Machinery and Intelligence». In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433 (cit. on pp. 6, 13).

[34] Guardian staff and agency. *Google fires software engineer who claims AI chatbot is sentient*. July 2022. URL: https://web.archive.org/web/20220723081645/https://www.theguardian.com/technology/2022/jul/23/google-fires-software-engineer-who-claims-ai-chatbot-is-sentient (cit. on p. 7).

[35] Mengmi Zhang et al. *Human or Machine? Turing Tests for Vision and Language*. 2022. DOI: 10.48550/ARXIV.2211.13087. URL: https://arxiv.org/abs/2211.13087 (cit. on pp. 7, 8).

[36] W. John Hutchins. «The Georgetown-IBM Experiment Demonstrated in January 1954». In: *Machine Translation: From Real Users to Research*. Ed. by Robert E. Frederking and Kathryn B. Taylor. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 102–114. ISBN: 978-3-540-30194-3 (cit. on pp. 9, 10).

[37] «IBM Type 701 Electronic Data Processing Machine». In: *Digital Computer Newsletter* 5.4 (Oct. 1953), pp. 7–8. URL: http://www.bitsavers.org/pdf/onr/Digital_Computer_Newsletter/Digital_Computer_Newsletter_V05N04_Oct53.pdf (cit. on p. 9).

[38] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. DOI: 10.48550/ARXIV.1609.08144. URL: https://arxiv.org/abs/1609.08144 (cit. on p. 10).

[39] Quoc V. Le and Mike Schuster. *A Neural Network for Machine Translation, at Production Scale*. Sept. 2016. URL: https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html (cit. on p. 10).

[40] Joseph Weizenbaum. «ELIZA: A Computer Program For the Study of Natural Language Communication Between Man and Machine». In: *Communications of the ACM* 9 (Jan. 1966), pp. 35–36. URL: http://www.universelle-automation.de/1966_Boston.pdf (cit. on p. 10).

[41] *FLOPS*. URL: https://en.wikipedia.org/wiki/FLOPS (cit. on p. 11).

[42] *GeForce 30 series*. URL: https://en.wikipedia.org/wiki/GeForce_30_series (cit. on p. 11).

[43] Giovanni Guida and Giancarlo Mauri. «Evaluation of natural language processing systems: Issues and approaches». In: *Proceedings of the IEEE* 74.7 (1986), pp. 1026–1035. DOI: 10.1109/PROC.1986.13580 (cit. on p. 11).

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep Learning». In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on pp. 12, 15).

[45] James L. McClelland, David E. Rumelhart, and Geoffrey E. Hinton. «The appeal of parallel distributed processing». In: *MIT Press, Cambridge MA* 3 (1986), p. 44 (cit. on p. 12).

[46] Risto Miikkulainen and Michael G. Dyer. «Natural Language Processing With Modular PDP Networks and Distributed Lexicon». In: *Cognitive Science* 15.3 (1991), pp. 343–399. DOI: https://doi.org/10.1207/s15516709cog1503_2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1503_2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1503_2 (cit. on p. 12).

[47] John Joseph Hopfield. «Neural networks and physical systems with emergent collective computational abilities.» In: *Proceedings of the National Academy of Sciences* 79.8 (Apr. 1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554 (cit. on pp. 12, 19).

[48] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. «Neocognitron: A neural network model for a mechanism of visual pattern recognition». In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983), pp. 826–834. DOI: 10.1109/TSMC.1983.6313076 (cit. on p. 12).

[49] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. «A Learning Algorithm for Boltzmann Machines». In: *Cognitive Science* 9.1 (1985), pp. 147–169. DOI: https://doi.org/10.1207/s15516709cog0901_7. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog0901_7. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog0901_7 (cit. on p. 12).

[50] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. «Handwritten Digit Recognition: Applications of Neural Net Chips and Automatic Learning». In: *IEEE Communication* (Nov. 1989), pp. 41–46 (cit. on p. 12).

[51] Paul Werbos. «Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Appl. Math. Harvard University». PhD thesis. Jan. 1974 (cit. on p. 12).

[52] D. B. Parker. *Learning Logic*. Tech. rep. TR-47. Cambridge, MA: Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, 1985 (cit. on p. 12).

[53] Yann LeCun. «Une procédure d'apprentissage pour réseau a seuil asymmetrique (a Learning Scheme for Asymmetric Threshold Networks)». In: *Proceedings of Cognitiva 85*. [In French]. Paris, France, 1985, pp. 599–604 (cit. on pp. 12, 17).

[54] Yann LeCun. «Learning Processes in an Asymmetric Threshold Network». In: *Disordered systems and biological organization*. Ed. by E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch. Les Houches, France: Springer-Verlag, 1986, pp. 233–240 (cit. on p. 12).

[55] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. «Learning representations by back-propagating errors». In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0 (cit. on pp. 12, 17, 19).

[56] Charles Babbage. *Babbage's Calculating Engines: Being a Collection of Papers Relating to them; their History and Construction*. Ed. by Henry P. Babbage. Cambridge Library Collection - Mathematics. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511694721 (cit. on p. 12).

[57] Douglas R. Hartree. *Calculating Instruments and Machines*. University of Illinois Press, Sept. 1949 (cit. on p. 12).

[58] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957 (cit. on p. 15).

[59] Warren S. McCulloch and Walter Pitts. «A logical calculus of the ideas immanent in nervous activity». In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259 (cit. on p. 15).

[60] *Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award*. 2019. eprint: https://www.acm.org/binaries/content/assets/press-releases/2019/march/turing-award-2018.pdf. URL: https://www.acm.org/media-center/2019/march/turing-award-2018 (cit. on p. 15).

[61] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. «Deep Sparse Rectifier Neural Networks». In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html (cit. on pp. 16, 34).

[62] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2016. DOI: 10.48550/ARXIV.1606.08415. URL: https://arxiv.org/abs/1606.08415 (cit. on pp. 17, 34).

[63] Léon Bottou. «Online Algorithms and Stochastic Approximations». In: *Online Learning and Neural Networks*. Ed. by David Saad. revised, oct 2012. Cambridge, UK: Cambridge University Press, 1998. URL: http://leon.bottou.org/papers/bottou-98x (cit. on p. 17).

[64] Mark Harris. «Many-Core GPU Computing with NVIDIA CUDA». In: *Proceedings of the 22nd Annual International Conference on Supercomputing*. ICS '08. Island of Kos, Greece: Association for Computing Machinery, 2008, p. 1. ISBN: 9781605581583. DOI: 10.1145/1375527.1375528. URL: https://doi.org/10.1145/1375527.1375528 (cit. on p. 18).

[65] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. «Large-Scale Deep Unsupervised Learning Using Graphics Processors». In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, 873–880. ISBN: 9781605585161. DOI: 10.1145/1553374.1553486. URL: https://doi.org/10.1145/1553374.1553486 (cit. on p. 18).

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 18).

[67] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. «Learning long-term dependencies with gradient descent is difficult». In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 5 (Feb. 1994), pp. 157–66. DOI: 10.1109/72.279181 (cit. on pp. 19, 37).

[68] Salah El Hihi and Yoshua Bengio. «Hierarchical Recurrent Neural Networks for Long-Term Dependencies». In: *Proceedings of the 8th International Conference on Neural Information Processing Systems*. NIPS'95. Denver, Colorado: MIT Press, 1995, 493–499 (cit. on p. 19).

[69] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 19, 37).

[70] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. 2012. DOI: 10.48550/ARXIV.1211.5063. URL: https://arxiv.org/abs/1211.5063 (cit. on p. 20).

[71] Ilya Sutskever. «Training Recurrent Neural Networks». PhD thesis. 2013. URL: https://hdl.handle.net/1807/36012 (cit. on p. 20).

[72] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. 2014. DOI: 10.48550/ARXIV.1409.1259. URL: https://arxiv.org/abs/1409.1259 (cit. on pp. 20, 37).

[73] Rico Sennrich, Barry Haddow, and Alexandra Birch. «Neural Machine Translation of Rare Words with Subword Units». In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162 (cit. on p. 23).

[74] Taku Kudo. «Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates». In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 66–75. DOI: 10.18653/v1/P18-1007. URL: https://aclanthology.org/P18-1007 (cit. on p. 23).

[75] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. «Massive Exploration of Neural Machine Translation Architectures». In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1442–1451. DOI: 10.18653/v1/D17-1151. URL: https://aclanthology.org/D17-1151 (cit. on p. 23).

[76] Andrej Karpathy. *Let's build GPT: from scratch, in code, spelled out.* Code: https://github.com/karpathy/nanoGPT. 2023. URL: https://www.youtube.com/watch?v=kCc8FmEb1nY (cit. on p. 24).

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on p. 25).

[78] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450 (cit. on p. 25).

[79] Ondrej Bojar et al. «Findings of the 2014 Workshop on Statistical Machine Translation». In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 12–58. URL: http://www.aclweb.org/anthology/W/W14/W14-3302 (cit. on p. 25).

[80] Austin Huang, Suraj Subramanian, Jonathan Sum, Khalid Almubarak, and Stella Biderman. *The Annotated Transformer*. 2022. URL: http://nlp.seas.harvard.edu/annotated-transformer/ (cit. on p. 27).

[81] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. «How Transferable Are Features in Deep Neural Networks?» In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, 3320–3328 (cit. on p. 27).

[82] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 27).

[83] Wilson L. Taylor. «"Cloze Procedure": A New Tool for Measuring Readability». In: *Journalism & Mass Communication Quarterly* 30 (1953), pp. 415–433 (cit. on p. 28).

[84] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. «Extracting and Composing Robust Features with Denoising Autoencoders». In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, 1096–1103. ISBN: 9781605582054. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294 (cit. on p. 29).

[85] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2019. DOI: 10.48550/ARXIV.1910.03771. URL: https://arxiv.org/abs/1910.03771 (cit. on pp. 29, 60, 66, 67, 73).

[86] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. «Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books». In: *The IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015 (cit. on pp. 29, 49).

[87] Will Kay et al. *The Kinetics Human Action Video Dataset*. 2017. DOI: 10.48550/ARXIV.1705.06950. URL: https://arxiv.org/abs/1705.06950 (cit. on pp. 31, 41, 44).

[88] Medhini Narasimhan, Anna Rohrbach, and Trevor Darrell. «CLIP-It! Language-Guided Video Summarization». In: (2021). DOI: 10.48550/ARXIV.2107.00650. URL: https://arxiv.org/abs/2107.00650 (cit. on p. 32).

[89] Hao Jiang and Yadong Mu. «Joint Video Summarization and Moment Localization by Cross-Task Sample Transfer». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 16388–16398 (cit. on p. 32).

[90] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, and De Rosal Ignatius Moses Setiadi. «Review of automatic text summarization techniques & methods». In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (2022), pp. 1029–1046. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2020.05.006. URL: https://www.sciencedirect.com/science/article/pii/S1319157820303712 (cit. on p. 32).

[91] H. P. Luhn. «A Statistical Approach to Mechanized Encoding and Searching of Literary Information». In: *IBM Journal of Research and Development* 1.4 (1957), pp. 309–317. DOI: 10.1147/rd.14.0309 (cit. on p. 32).

[92] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. «Teaching Machines to Read and Comprehend». In: *NIPS*. 2015, pp. 1693–1701. URL: http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend (cit. on pp. 33–35).

[93] BigScience Workshop et al. *BLOOM: A 176B-Parameter Open-Access Multilingual Language Model*. 2022. DOI: 10.48550/ARXIV.2211.05100. URL: https://arxiv.org/abs/2211.05100 (cit. on p. 33).

[94] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. «Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization». In: *ArXiv* abs/1808.08745 (2018) (cit. on pp. 34, 35, 63).

[95] Max Grusky, Mor Naaman, and Yoav Artzi. «NEWSROOM: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies». In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2018 (cit. on p. 34).

[96] Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. *Multi-News: a Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model*. 2019. arXiv: 1906.01749 [cs.CL] (cit. on p. 34).

[97] David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. «English gigaword». In: *Linguistic Data Consortium, Philadelphia* 4.1 (2003), p. 34 (cit. on p. 34).

[98] Alexander M. Rush, Sumit Chopra, and Jason Weston. «A Neural Attention Model for Abstractive Sentence Summarization». In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015). DOI: 10.18653/v1/d15-1044. URL: http://dx.doi.org/10.18653/v1/D15-1044 (cit. on p. 34).

[99] Mahnaz Koupaee and William Yang Wang. *WikiHow: A Large Scale Text Summarization Dataset*. 2018. DOI: 10.48550/ARXIV.1810.09305. URL: https://arxiv.org/abs/1810.09305 (cit. on p. 34).

[100] Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. *Abstractive Summarization of Reddit Posts with Multi-level Memory Networks*. 2018. arXiv: 1811.00783 [cs.CL] (cit. on p. 34).

[101] Eva Sharma, Chen Li, and Lu Wang. «BIGPATENT: A Large-Scale Dataset for Abstractive and Coherent Summarization». In: *CoRR* abs/1906.03741 (2019). arXiv: 1906.03741. URL: http://arxiv.org/abs/1906.03741 (cit. on pp. 34, 39).

[102] Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. *On the Use of ArXiv as a Dataset*. 2019. arXiv: 1905.00075 [cs.IR] (cit. on pp. 34, 38, 39).

[103] National Library of Medicine. *PubMed*. URL: https://www.nlm.nih.gov/databases/download/pubmed_medline_faq.html (cit. on pp. 34, 39).

[104] Rui Zhang and Joel Tetreault. «This Email Could Save Your Life: Introducing the Task of Email Subject Line Generation». In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 446–456. DOI: 10.18653/v1/P19-1043. URL: https://aclanthology.org/P19-1043 (cit. on p. 34).

[105] Anastassia Kornilova and Vlad Eidelman. *BillSum: A Corpus for Automatic Summarization of US Legislation*. 2019. arXiv: 1910.00523 [cs.CL] (cit. on p. 34).

[106] Mohammad Saleh and Anjuli Kannan. *Auto-generated Summaries in Google Docs*. 2022. URL: https://ai.googleblog.com/2022/03/auto-generated-summaries-in-google-docs.html (cit. on p. 34).

[107] Mia Xu Chen et al. *The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation*. 2018. DOI: 10.48550/ARXIV.1804.09849. URL: https://arxiv.org/abs/1804.09849 (cit. on p. 34).

[108] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. «GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding». In: In the Proceedings of ICLR. 2019 (cit. on pp. 34, 85).

[109] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. «SQuAD: 100,000+ Questions for Machine Comprehension of Text». In: *arXiv e-prints*, arXiv:1606.05250 (2016), arXiv:1606.05250. arXiv: 1606.05250 (cit. on p. 34).

[110] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. *SpanBERT: Improving Pre-training by Representing and Predicting Spans*. 2019. DOI: 10.48550/ARXIV.1907.10529. URL: https://arxiv.org/abs/1907.10529 (cit. on p. 35).

[111] *List of Nvidia graphics processing units*. URL: https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units (cit. on p. 37).

[112] Jason Phang, Yao Zhao, and Peter J. Liu. *Investigating Efficiently Extending Transformers for Long Input Summarization*. 2022. DOI: 10.48550/ARXIV.2208.04347. URL: https://arxiv.org/abs/2208.04347 (cit. on pp. 38, 103).

[113] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. «Revealing the Dark Secrets of BERT». 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)*. Hong Kong, China: Association for Computational Linguistics,

Nov. 2019, pp. 4365–4374. DOI: 10.18653/v1/D19-1445. URL: https://aclanthology.org/D19-1445 (cit. on p. 38).

[114] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. «Constructing Datasets for Multi-hop Reading Comprehension Across Documents». In: *Transactions of the Association for Computational Linguistics* 6 (2018), pp. 287–302. DOI: 10.1162/tacl_a_00021. URL: https://aclanthology.org/Q18-1021 (cit. on p. 38).

[115] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. «triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension». In: *arXiv e-prints*, arXiv:1705.03551 (2017), arXiv:1705.03551. arXiv: 1705.03551 (cit. on p. 38).

[116] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. *Efficient Attentions for Long Document Summarization*. 2021. arXiv: 2104.02112 [cs.CL] (cit. on p. 39).

[117] Yi Tay et al. «Long Range Arena : A Benchmark for Efficient Transformers». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=qVyeW-grC2k (cit. on p. 39).

[118] Uri Shaham et al. «SCROLLS: Standardized CompaRison Over Long Language Sequences». In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 12007–12021. URL: https://aclanthology.org/2022.emnlp-main.823 (cit. on p. 40).

[119] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. «Video Swin Transformer». In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 3192–3201. DOI: 10.1109/CVPR52688.2022.00320 (cit. on p. 41).

[120] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. «ViViT: A Video Vision Transformer». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 6836–6846 (cit. on p. 41).

[121] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. *A Short Note about Kinetics-600*. 2018. DOI: 10.48550/ARXIV.1808.01340. URL: https://arxiv.org/abs/1808.01340 (cit. on p. 44).

[122] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. *A Short Note on the Kinetics-700 Human Action Dataset*. 2019. DOI: 10.48550/ARXIV.1907.06987. URL: https://arxiv.org/abs/1907.06987 (cit. on p. 44).

[123] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. *A Short Note on the Kinetics-700-2020 Human Action Dataset*. 2020. DOI: 10.48550/ARXIV.2010.10864. URL: https://arxiv.org/abs/2010.10864 (cit. on p. 44).

[124] Derek Miller. *Leveraging BERT for Extractive Text Summarization on Lectures*. 2019. DOI: 10.48550/ARXIV.1906.04165. URL: https://arxiv.org/abs/1906.04165 (cit. on pp. 45, 46).

[125] Yang Liu and Mirella Lapata. «Text Summarization with Pretrained Encoders». 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3730–3740. DOI: 10.18653/v1/D19-1387. URL: https://aclanthology.org/D19-1387 (cit. on pp. 45, 46).

[126] Chengpei Xu, Ruomei Wang, Shujin Lin, Xiaonan Luo, Baoquan Zhao, Lijie Shao, and Mengqiu Hu. «Lecture2Note: Automatic Generation of Lecture Notes from Slide-Based Educational Videos». In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. 2019, pp. 898–903. DOI: 10.1109/ICME.2019.00159 (cit. on pp. 45, 46).

[127] Hayden T Housen. «Lecture2Notes: Summarizing Lecture Videos by Classifying Slides and Analyzing Text». In: (2022). eprint: https://haydenhousen.com/media/lecture2notes-paper-v1.pdf. URL: https://lecture2notes.com/ (cit. on pp. 45–47).

[128] S. Lloyd. «Least squares quantization in PCM». In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 46).

[129] Irene Benedetto, Laura Farinetti, Lorenzo Canale, and Luca Cagliero. «Video lectures summarization». MA thesis. July 2021. URL: https://webthesis.biblio.polito.it/19175/ (cit. on p. 47).

[130] Regina Merine and Saptarshi Purkayastha. «Risks and Benefits of AI-generated Text Summarization for Expert Level Content in Graduate Health Informatics». In: *2022 IEEE 10th International Conference on Healthcare Informatics (ICHI)*. 2022, pp. 567–574. DOI: 10.1109/ICHI54592.2022.00113 (cit. on p. 47).

[131] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. *A Brief Review of Domain Adaptation*. 2020. DOI: 10.48550/ARXIV.2010.03978. URL: https://arxiv.org/abs/2010.03978 (cit. on p. 50).

[132] Creative Commons. *Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0)*. URL: https://creativecommons.org/licenses/by-nc-nd/4.0/ (cit. on p. 52).

[133] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.ai. 2020. URL: https://www.wandb.ai/ (cit. on pp. 73–75, 78–81).

[134] Danny Hernandez et al. *Scaling Laws and Interpretability of Learning from Repeated Data*. 2022. DOI: 10.48550/ARXIV.2205.10487. URL: https://arxiv.org/abs/2205.10487 (cit. on p. 74).

[135] Ilya Loshchilov and Frank Hutter. «Decoupled Weight Decay Regularization». In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7 (cit. on p. 76).

[136] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. «Bleu: a Method for Automatic Evaluation of Machine Translation». In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040 (cit. on pp. 83, 85).

[137] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. «DeBERTa: Decoding-enhanced BERT with Disentangled Attention». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=XPZIaotutsD (cit. on p. 85).

[138] Moreno La Quatra and Luca Cagliero. «BART-IT: An Efficient Sequence-to-Sequence Model for Italian Text Summarization». In: *Future Internet* 15.1 (2023). ISSN: 1999-5903. DOI: 10.3390/fi15010015. URL: https://www.mdpi.com/1999-5903/15/1/15 (cit. on p. 103).