Fine-tuning a General Transformer Model on Story-lines of IMDB Movies Database

by

Hojat Ghasemi

A thesis submitted in partial fulfillment of the

requirements for the degree of

Master of Science (M.Sc.) in Computational Sciences

The Office of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Office of Graduate Studies/Bureau des études supérieures

Title of Thesis Titre de la thèse	Fine-tuning a General Transformer Model on Story-lines of IMDB Movies Database	
Name of Candidate Nom du candidat	Ghasemi, Hojat	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance January 13, 2022

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur(trice) de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Luckny Zephyr
(Committee member/Membre du comité)

Dr. Vishvjit Thakar
(External Examiner/Examinateur externe)

Approved for the Office of Graduate Studies
Approuvé pour le Bureau des études supérieures
Tammy Eger, PhD
Vice-President Research (Office of Graduate Studies)
Vice-rectrice à la recherche (Bureau des études supérieures)
Laurentian University / Université Laurentienne

ABSTRACT

Recent transformer-based language models pre-trained on huge text corpora have shown great success in performing downstream Natural Language Processing (NLP) tasks such as text summarization when fine-tuned on smaller labeled datasets. However, the impact of fine-tuning on improving the performance of pre-trained language models in summarizing movie storylines have not been explored. Moreover, there is a lack of extensive labelled datasets containing movies storylines to allow pre-trained language models delving deeper in this realm. In this research work we propose a novel labelled dataset containing IMDB movie storylines alongside their summaries for teaching pre-trained language models how to perform text summarization on movie storylines. Furthermore, we showcase the potential of this dataset by fine-tuning a T5-base model with the use of this dataset. Our results show that fine-tuning a T5-base model on this dataset can significantly improve the performance in summarizing movie storylines.

KEYWORDS:

Fine-tuning, Language Model, Text Summarization, Transfer Learning, Transformer, Pre-training, Abstractive Summarization, IMDB, Movie Storyline, Natural Language Processing, Attention Based Models, Deep Learning.

ACKNOWLEDGEMENTS

I need to firstly thank Professor Kalpdrum Passi, whose patience and respectful manner was the best side of my study experience at Laurentian University. I would also like to thank my committee members for reviewing this thesis and serving on the defense committee.

I am sincerely thankful to my parents for financially supporting my studies and making this possible. Last but not least, I need to thank both of my brothers for always encouraging me to try harder and keep moving forward.

DEDICATION

In loving memory of
Naser Ghasemi
Beloved Uncle
1959-2015

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

NLP	Natural Language Processing
TPLM	Transformer-based Pre-trained Language Model
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
BiRNN	Bidirectional Recurrent Neural Network
LSTM	Long Short-Term Memory
CLM	Causal Language Modeling
MLM	Masked Language Modeling
RTD	Replaced Token Detection
STD	Shuffled Token Detection
RTS	Random Token Substitution
SLM	Swapped Language Modeling
TLM	Translation Language Modeling
ALM	Alternate Language Modeling
SBO	Sentence Boundary Objective
NSP	Next Sentence Prediction
SOP	Sentence Order Prediction
Seq2SeqLM	Sequence-to-Sequence Language Model
DAE	Denoising Auto Encoder
IMDB	Internet Movie Database

CHAPTER 1
INTRODUCTION

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) which aims at enabling computing machines to automatically analyze and represent human language. Since its inception in 1950s, the most important tasks that NLP has been focusing on are machine translation, information retrieval, text summarization, question answering, information extraction, topic modeling, etc., [1]. In order for a machine to deeply understand a document, important features of the text need to be extracted from raw text and fed to the NLP model. Hence, feature extraction and feature selection are crucial parts of all state-of-the-art NLP systems. However, both of those operations require substantial computing power which hinders the development of reliable NLP models via general algorithms of machine learning; that is where deep learning algorithms and architectures come into play [2]. Simply explained, a language model is any kind of system that can predict the next word in a sentence. Figure 1.1 shows an example of a "language modelling" task. Imagine the sentence "the students opened their ….." was given to a person and they were asked to put a suitable word in the blank space to complete the sentence in a meaningful way. Many possible answers could be given in this situation. Figure 1.1 provides a few examples.

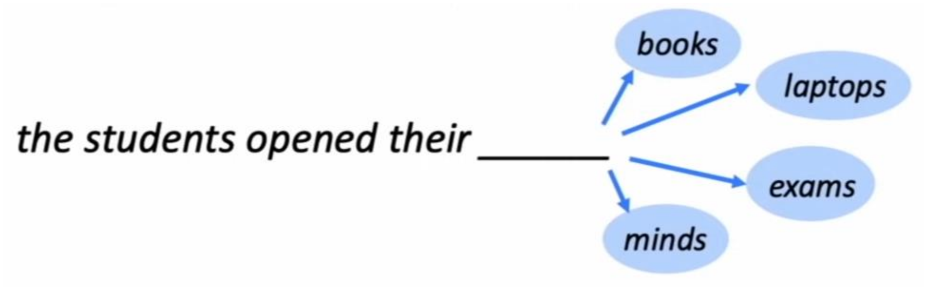


Figure 1.1 – Illustration of language modeling [3]

When a human (or a machine) chooses one of those possible answers for completing the sentence, this task is called **language modeling** and that person (or machine) has performed the role of a language model. In the realm of computer science, many various architectures have been suggested for performing language modeling but at the time of this writing, the most successful language models have been created with the use of **transformers** technology.

Transformer is an extremely complicated software system with so many internal building blocks working with each other in order to make a language model system as a whole; it is out of the scope of this chapter to meticulously explain everything in transformers. Simply explained, transformer is a new family of deep neural network architectures, which was introduced in 2017 to address some of the important problems of language model technologies before it. Unlike their forerunners, transformers could be efficiently parallelized (which means they can be trained with parallel computing cores of a machine), hence their popularity. For more details about transformers, refer to Vaswani et al. [4].

One important matter which needs to be taken into account about transformers architecture is that they contain numerous neural networks inside them and each neural network has some **hyperparameters** which are the weights of the model. The number of these parameters refer to the size of that transformer-based language model. For example, Figure 1.2 shows a few transformer-based language models with their corresponding number of parameters. As a general rule, when the number of parameters in a transformer-based language model increases, the size and accuracy of that model increases as well.

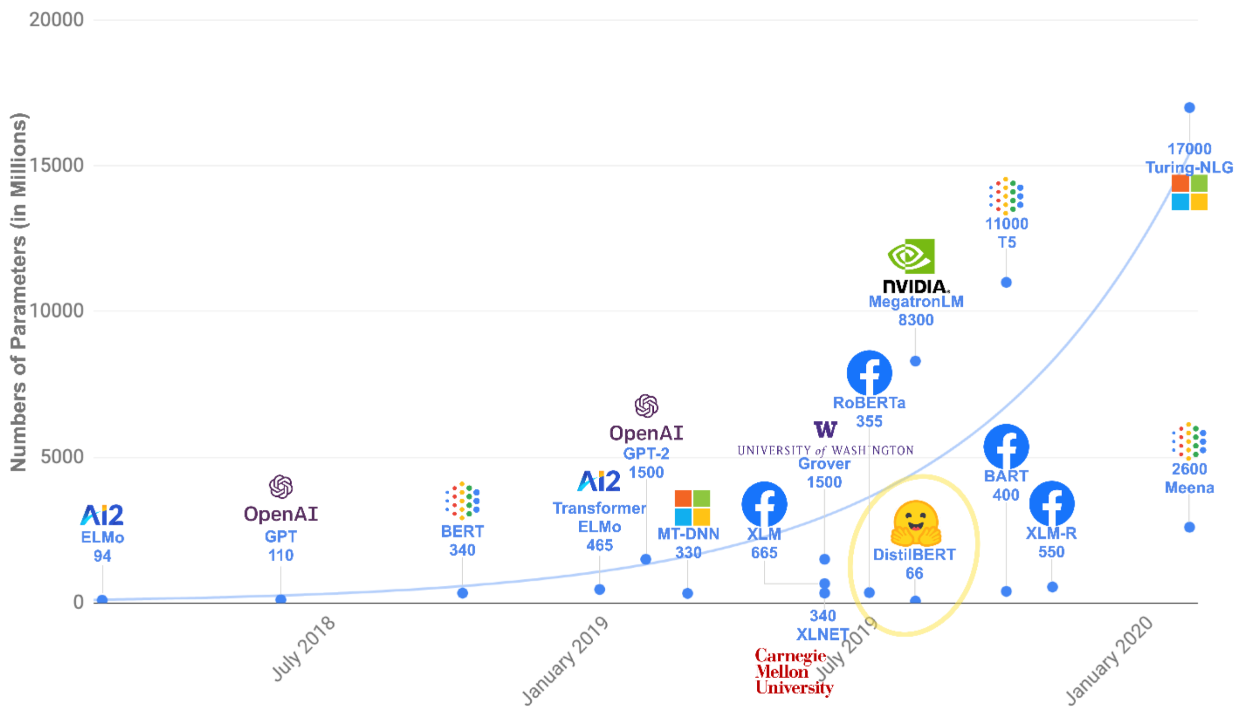


Figure 1.2 - Model parameters [9]

The current most popular approach for performing language modeling with the use of transformers is with Transformer-based Pre-trained Language Models (**TPLM)**. In this technique, the research groups that are developing transformer-based language models train their model on huge corpora in a self-supervised fashion for teaching the model the general knowledge of language syntax and semantics. All the language models mentioned in Chapter 2 are TPLMs.

While using a TPLM for performing any kind of NLP tasks, one needs to remember that all of these language models are already pre-trained with some pre-training tasks and on some huge corpora (both of these topics are comprehensively explained in Chapter 2). Hence, those TPLMs already have some basic understanding about the language. This initial knowledge of model about the language is beneficial for any kinds of NLP tasks one wants to put the model into, but it is not enough.

For getting a high quality performance out of a TPLM in a specific downstream task, that TPLM needs to be further trained with a task-oriented, relatively small **labelled dataset**; that is why task-oriented and labelled datasets (such as the one created in this research) are so important for improving the performance of TPLMs in specific fields. This process of teaching a labelled dataset to a TPLM is called **fine-tuning**.

Creating a huge labelled dataset in the context of Natural Language Understanding (NLU) is usually very expensive/time-consuming but unlabeled corpora are abundant. That is why current de facto approach of using transformer based language models in NLP is to use a TPLM which is already trained (by its developers) on a huge unlabeled corpus and then fine-tune that model on a downstream labelled dataset. This procedure has been used for performing all the common NLP tasks (and even some computer vision tasks) and proved to be highly effective.

Transformers are a new family of neural network architectures which were introduced to address some of the drawbacks of Recurrent Neural Networks (RNNs) as the main approach of computational linguists to various Natural Language Understanding (NLU) tasks. Prior to the introduction of transformers by researchers at Google Brain team in 2017 [4], RNNs were the primary tools for using deep learning to understand text.

The main advantage of transformer model architecture over plain RNNs is that unlike the inherent sequential structure of RNNs, transformers are conveniently parallelizable which makes them the perfect fit for training a model on huge amounts of data. Transformer achieves this parallelizability by exploiting a few previously proposed inventions such as positional encoding, attention [5], [6] and self-attention. These innovations help the transformer model to move the burden of understanding word order from the structure of neural network to the data itself; thus, getting around the intrinsic sequence of words in a sentence.

Apart from all of their advantages, transformers should not be considered as a panacea for all kinds of problems in the field of Natural Language Processing (NLP). Among others, these are some of the most prominent challenges of putting transformer-based language models in use:

1- Transformers are extremely memory and compute intensive:

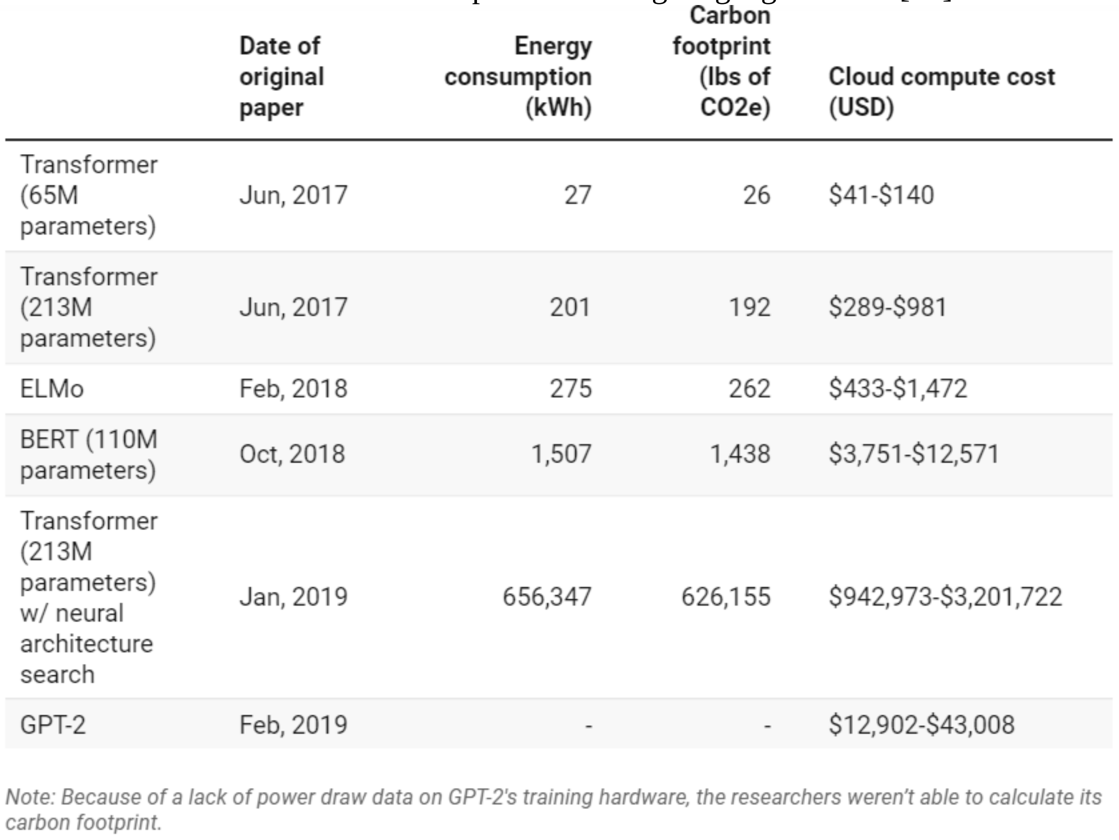
   It requires massive computing power and a huge corpus for a pre-trained transformer-based language model to achieve an acceptable performance. Figure 1.2 depicts a few language models with their corresponding number of hyper parameters. Vertical axis represents the number of parameters in a given model and horizontal axis shows the date that model was published. As it is shown in Figure 1.2, apart from a few outliers, the general strategy to achieve better performance (while training a transformer-based language model) is by increasing the model sizes as well as the amount of data they are pre-trained on. Even very small transformer models still have tens of millions of parameters.

2- Training a big transformer model from scratch might have environmental side effects on ecosystem:

In order for transformer models to have acceptable accuracy while performing NLP tasks, they need to be trained on abundant data via exceptionally large computational resources. Such a procedure requires substantial energy consumption; as a result, these models are costly to train and develop, both financially (due to the cost of hardware and electricity or cloud compute time) and environmentally (due to the carbon footprint required to fuel modern processing hardware) [7].

As it is shown in Table 1.1, training a model, especially a large one may have environmental impacts. The projects shown in this table were led by a team, consciously trying to reduce the environmental impact of pre-training. The footprint of running lots of trials to get the best hyperparameters would be even higher. In practice, it is much more likely that researchers would develop a new model from scratch or adapt an existing model to a new dataset, either of which can require many more rounds of training than shown here. According to [7], the normal procedure of building a model worthy of being published in a scientific paper requires training 4,789 models over a six-month period. If converted to CO2 equivalent, it emits more than 78,000 pounds and could be considered as a representative of typical works in this field.

Table 1.1 – Environmental impact of training language models [10]



	Date of original paper	Energy consumption (kWh)	Carbon footprint (lbs of CO2e)	Cloud compute cost (USD)
Transformer (65M parameters)	Jun, 2017	27	26	$41-$140
Transformer (213M parameters)	Jun, 2017	201	192	$289-$981
ELMo	Feb, 2018	275	262	$433-$1,472
BERT (110M parameters)	Oct, 2018	1,507	1,438	$3,751-$12,571
Transformer (213M parameters) w/ neural architecture search	Jan, 2019	656,347	626,155	$942,973-$3,201,722
GPT-2	Feb, 2019	-	-	$12,902-$43,008

Note: Because of a lack of power draw data on GPT-2's training hardware, the researchers weren't able to calculate its carbon footprint.

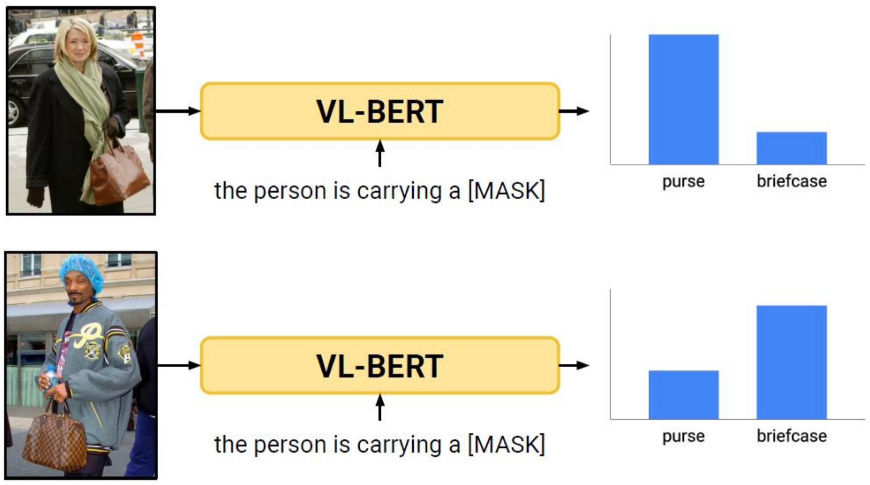
3- Transformers are a relatively new and not-yet-known class of models:

There is not a good and principled understanding of how changing every hyper-parameter will change the overall output of a given transformer model. But with older models such as RNN, there is a better understanding than with transformers as there have been extensive testing and modifications to the architecture. Another aspect of the problem is that transformers tend to have a steep learning curve, especially for novice NLP practitioners.

4- Transformer models are susceptible to bias and limitation:

Although pre-trained language models are powerful tools, they have intrinsic limitations/flaws. The most noticeable of them is that, in order to enable pre-training on large amounts of data, researchers usually scrape all the online contents they can find, taking the best as well as the worst of what is available on the internet.

While using these tools, it must be kept in mind that the original model could very easily generate racist, sexist, or homophobic contents. For instance, it can be seen in Figure 1.3 that in the model VL-BERT, the word "purse" is highly associated with female which overtakes the explicit visual evidence. These biases happen even though the basic BERT model is among rare transformer models built by using apparently neutral data (It was trained on the English Wikipedia and BookCorpus datasets [8]).


VL-BERT
the person is carrying a [MASK]
purse
briefcase
VL-BERT
the person is carrying a [MASK]
purse
briefcase
Figure 1.3 - Gender bias in VL-BERT [11]

One effective remedy which can be considered as a partial answer to some of the problems above was first introduced in [12] and is widely known as "Transfer Learning". Transfer learning is considered to be a method of machine learning where a model developed for a first learning task is reused as the starting point for another model in a somewhat other related learning task. The main justification for doing transfer learning is that we intend to store knowledge gained while solving one problem and apply it to a different but related problem [13]. The most common context in which transfer learning is being used in the field of deep learning is to use a pre-trained neural network for achieving better results in another learning task. In such a scenario, often the previous learning is referred to as "source" and the next learning as "target" [14].

Simply explained, during the process of transfer learning we choose a previous model which has already been trained on a huge corpus, and then re-train this model on a specific smaller dataset that we have for a task at hand. This method which is usually called "fine-tuning" has been adopted as de facto approach to deep learning problems by lots of researchers and has various benefits [14]. Firstly, the process of fine-tuning a model consumes a lot less time and computing power compared to training a model from scratch, but is capable of producing state-of-the-art results which in some cases can even be of higher precision than a pre-trained model. Second, since fine-tuning consumes less power, it can be a solution for environmental side-effects of training a huge model from scratch. Lastly, transfer learning can be a partial solution to the problem of intrinsic bias in pre-trained models. By fine-tuning our model on an egalitarian dataset, we can alleviate the biases of our models. But keep in mind that fine-tuning the model will not make the intrinsic bias disappear.

The main purpose of this research is to transfer the knowledge of pre-trained language models to the field of summarizing movie storylines by training the T5 language model on a specific dataset of Internet Movie Database (IMDB) movie storylines (for more information about this online database refer to section 3.1). T5 is a general purpose model which considers all common NLP problems to have a text-to-text format and has been pre-trained on a huge dataset named "C4" [15].

IMDB website is considered to be the world's most popular and authentic source for movie, TV and celebrity content. It is a comprehensive online database which contains information about movies, TV series, video games, and streaming content online including cast, production crew and personal biographies, plot summaries, trivia, ratings, and fan and critical reviews.

This research introduces a middle-sized dataset containing links to almost 5,000 movie pages on the IMDB website, alongside their corresponding storyline. In order to make it a useful database for fine-tuning pre-trained models on the task of text summarization, the PEGASUS model [16] has been applied to all the storylines for making an abstractive summary of the plot for each movie. Furthermore, for some entries of the database, the machine generated summary has been optimized by a human editor.

Higher ROUGE [17] scores achieved by the fine-tuned T5 model in comparison with un-tuned models prove the previous hypothesis that fine-tuning will be beneficial for improving the accuracy of models on the task of text summarization. In addition, the most important contributions of this research are as follows:

1- This work introduces a new comprehensive database of IMDB storylines for movies. Furthermore, our database includes summarized versions of those storylines, which are sporadically improved by human editors. According to the results of this research (documented and discussed in Chapter 4) this novel dataset can be an optimal candidate for fine-tuning any pre-trained transformer models for the task of summarizing movie storylines.

2- Throughout this project, PEGASUS model (with more than 568 M parameters [16]) is used for producing the summaries of the dataset. Later on, a different smaller model is fine-tuned (basic checkpoint of T5 with roughly 220 M parameters [15]) on this dataset. By following this method, at least in theory, it was possible to use the knowledge gained by a bigger model, but at a lower cost in terms of memory requirements and the computational burden of a smaller model.

3- The most popular summarization models (such as PEGASUS or BertSum) are usually pre-trained on news articles and their summaries [16, 18]. Those datasets usually have a more formal set of words than found in a movie storyline. As can be seen in the dataset, storylines of movies tend to contain informal or sometimes even out-of-vocabulary words which are not usually found in news articles. By fine-tuning a T5 base model on the storyline summaries, we introduce a new summarization model which is specifically trained on movie storylines.

CHAPTER 2
LITERATURE REVIEW

2.1. Transformers

Since its introduction in 2017 by Vaswani, et al. [4], the “transformer” model architecture has shown great promise in tackling modern NLP tasks. This early success urged researchers for further improvements on the original architecture. The main difference between transformers and other model architectures is the exclusive usage of attention mechanism (a technique of machine learning that allows the model to focus on the relevant parts of input sequence as needed [5], [6]) for implementing various parts of the model.

Even before the emergence of transformers, attention mechanism had become an integral part of sequence modelling in various NLP tasks, allowing modelling of dependencies in the input/output sequences without regard to their distance from each other [5, 19]. However, such attention mechanisms were used in conjunction with Recurrent Neural Networks (RNNs). The main contribution of transformers to this pattern was to avoid RNNs completely and relying exclusively on the attention mechanism for realizing dependencies in both input and output sequences [4].

Transformers attracted immense interest from NLP researchers who put a lot of efforts to develop more efficient variants of the model [20], [21], [22], [23], [24], [25], [26]. Transformer-based pre-trained language models (TPLMs) which are built on top of transformers and transfer learning have achieved great success in all NLP tasks [27].

The idea behind TPLMs is to first teach universal language representations to a model (via self-supervised learning over a large corpus of unlabeled text data) and then transfer this background knowledge to downstream tasks (which is usually performed by further training the pre-trained model over a small labelled task-specific dataset). Some of the benefits of pre-training are:

- The model will learn universal language representations using a large corpus of unlabeled text data.
- By providing a good initialization, the pre-trained model avoids training on the downstream tasks from scratch.
- Help the model perform acceptable even on small datasets, hence reducing the requirement for large labeled datasets.
- Since large deep learning models have so many parameters, they tend to overfit on small datasets and not generalize well. Pre-training the model provides a good initialization point and can be viewed as a form of regularization [28].

Pre-training a transformer-based language model over a huge corpora has become the de facto approach of NLP research in recent years, as a result, many researchers have tried to study and further develop this method.

2.2. Pre-training

According to Kalyan, Rajasekharan and Sangeetha [27], pre-training a transformer-based language model involves five necessary steps described below in sections 2.2.1 – 2.2.5.

2.2.1 Prepare the pre-training corpus

The dataset used for pre-training tasks is always an unlabeled, cleaned text corpus. [15, 29, 30] showed that the larger and more diverse the corpus, the higher the performance of the resulting model. Furthermore, according to [31], there are numerous duplicate sentences and repetitive substrings in all pre-training corpora; by deduplicating the corpus, the required number of training steps for achieving a specific level of performance will be reduced. Kalyan, Rajasekharan and Sangeetha [27] categorizes all the pre-training corpora into four types:

1- General corpus:

In this category, the text is less noisy and usually written by professionals in a formal idiolect. Examples of general pre-training corpora include OpenWebText (it is approximately 32GB and was used for pre-training RoBERTa [30]) and C4 (a huge collection of common crawl text with the size of 750GB which was filtered and used for pre-training T5 [15]).

2- Social media corpus:

This category is known to be noisier than others, as it is comprised of informal conversations of the general public on social media. Two of the most famous corpora in this category are Rale-E (a collection of hateful comments in English posted on Reddit which is a discussion website; this corpus was used for pre-training HateBERT [32]) and Amazon reviews [33] (a huge collection of 233M Amazon product reviews covering around 29 domains; it was used for pre-training SentiX [34]).

3- Language-based corpus:

These types of corpora are categorized according to the language used in them and as a result, they have various sub-categories as described below.

Monolingual: As the name suggests, this corpus focuses on just one specific language. Some examples of monolingual corpus are Indo4B [35] (23GB of Indonesian text), BrWaC [36] (17.5GB of Brazilian Portuguese web text), CLUECorpus2020 [37], and WuDaoCorpora [38] (both are huge collections of Chinese web text).

Multilingual: This corpus contains texts of more than one language. Some examples of multilingual corpus are OSCAR [39] (common crawl text for more than 166 languages; it was used in pre-training MuRIL [40]), mC4 (considered to be multilingual equivalent of C4; it contains texts of 101 languages and was used for pre-training mT5 [41]), CC-100 [42] (2.5 TB of common crawl text for 100 languages; it was used for pre-training XLM-R [42], infoXLM [43], and XLM-E [44]), and IndCorpus (1.17GB of Wikipedia text data for 10 indigenous languages; it was used for pre-training IndT5 [45]).

Parallel: The most important difference between parallel and multilingual corpora is that in a parallel corpus, some text data is aligned with its translation in one or more other languages than its original language [46]; multilingual corpora do not necessarily have this characteristic. Some of the well-known parallel corpora are IIT Bombay [47] (1.49M pairs of English-Hindi parallel sentences; it has been used as the pre-training corpus for many TPLMs such as mT6 [48], XLM [49], ALM [50], and Unicoder [51]), Multi-UN [52] (parallel corpus of UN official documents in six languages; used as the pre-training corpus in some TPLMs such as mT6 [48], ALM [50], XNLG [53], and XLM-E [44]), and CC-Aligned [54] (Parallel corpus of 292 million non-English document pairs and 100 million English document pairs; used as a pre-training corpus of XLM-E [44]).

4- Domain-specific corpus:

There are numerous corpora of specific domains such as biomedical, finance, and academic, which contain various domain-specific words, not found in the general domain. As an outcome of this fact, the performance of TPLMs pre-trained only on general corpora is so limited in domain-specific tasks [55]. In order to address this issue, some domain-specific corpuses have been introduced, for example:

MIMIC-III [56]: A biomedical corpus of anonymized ICU patient records gathered over a decade. It has been used as the pre-training corpus of BioBERT [55], BLUE BERT [57], and ClinicalBERT [58].

RealNews [59]: 120GB of de-duplicated common crawl corpus on news text data. It was used for pre-training Roberta-base-news [60].

S2ORC [61]: A collection of more than 80M academic research papers in English; it was used for pre-training the models Roberta-base-biomed [60] and Roberta-base-cs [60].

In closing, as the characteristics of text differ from one type of corpus to another, it is important to appropriately choose the pre-training corpus depending on the target domain.

2.2.2 Generating the vocabulary

In order to use a corpus for pre-training a model, it firstly needs to be converted into a unique set of characters and commonly used words/subwords which is usually referred to as “vocabulary”. This vocabulary is created by applying any of the tokenizers on the pre-training corpus. Some tokenizers that are usually used in TPLMs are WordPiece [62], Byte Pair Encoding (BPE) [63], Byte Level BPE [64], and SentencePiece [65].

After creating the vocabulary of the model, one level of abstraction over input vocabulary is the embedding. An embedding is a relatively low-dimensional space into which we usually map high-dimensional vocabulary vectors. The reason for using embedding is to more easily operate over inputs like sparse vectors representing words. An embedding can be learned and reused across models [66].

There are various types of embedding that can be used in a specific language model; the type of embedding used is important because it impacts the size of the vocabulary and also determines the overall size of the pre-trained language model [67]. As a general rule, in TPLMs character or sub-word embedding are preferred over word embedding because of three reasons [27]:

1- Character and sub-word embedding will result in a smaller vocabulary size compared to word embedding.
2- Character and sub-word embedding can represent any words, thus overcoming the problem of out-of-vocabulary words.
3- They can encode fine-grained information at character and subword level.

There's a category of embedding referred to as main embedding. They are meant to represent the input sequence in dense and low dimensional vectors; as the name suggests, they provide insight about the main concepts of input text. The main embedding themselves are divided into three groups.

1- Character embedding:

   It maps each one of the characters of the input sequence to a dense low dimensional vector. The vocabulary of this embedding includes all of the characters of that specific language (all the letters, punctuations, and numbers of that language). Some models such as CharacterBERT [68] and AlphaBERT [69] use character embedding.

2- Sub-word embedding:

   Unlike character embedding, sub-word embedding have a vocabulary which consists of frequently occurring sub-words/words in addition to characters. Most of the popular TPLMs such as BERT [8], RoBERTa [30], XLNet [29], T5 [15], and BART [70] use sub-word embedding (but each one of them has its own specific tokenizer).

3- Hybrid embedding:

   The main goal of these embedding was to leverage the benefits of both character and sub-word embedding. Models such as CharBERT [71] use a combination of character and subword embedding.

Some specific-domain TPLMs expect the input sequence to contain a series of codes recognizable by the model. For example, in a medical domain, many concepts such as diseases, drugs, symptoms, and many other concepts are represented in the electronic health records using special medical codes. These models have a vocabulary comprised of standard clinical ontologies and make use of code embedding. For example, some of the TPLMs in the biomedical domain are BERTEHR [72], Med-BERT [73], and BEHRT [74]; this category is usually called as code embedding [27].

In contrast to main embedding, auxiliary embedding are meant for providing additional information to the model and each has its own specific purpose. The first category of auxiliary embedding are referred to as position embedding. Since transformers do not have any RNN or

CNN layers to learn the order of input sentences, they need to make use of position embedding as a means of learning the order of input sequences; these position embedding can be divided into absolute or relative as follows:

1) **Absolute:** Models such as BERT [8], RoBERTa [30], and ELECTRA [75] use these embedding and learn them during the training process. Other models such as BEHRT [74] and BERTEHR [72] use absolute position embedding in a pre-determined manner.
2) **Relative:** In this type of position embedding, the order of input sentences is defined relative to each other; XLNet [29] makes use of this embedding.

Segment embedding is always used in conjunction with position embedding and is meant to distinguish between tokens of two input sentences. Position embedding is different for different tokens in the input sentence, but segment embedding will be the same for all tokens in each one of the input sentences [27]. BERT [8], RoBERTa [30], and ELECTRA [75] make use of segment embedding.

Language embedding is usually used in models with multilingual pre-training corpus and their goal is to explicitly inform the model about the language of input sentences [27]. XLM [49], Unicoder [51], and XNLG [53] use these embedding.

Entity type embedding provide information about various entities to the model during the pre-training process. For instance, OAG-BERT [76] is a transformer-based language model pre-trained on academic text as well as augmented with information on various academic entities such as paper, published venue, author affiliation, research domain, and authors.

Age and gender embedding are usually studied/used in the pre-trained language models in conjunction with each other. For example, in medical pre-trained models such as BEHRT [74] and BERTEHR [72] input sequences are a series of patient medical records. Apart from code embedding, it is useful to provide temporal information like age and gender to the model; age and gender embedding provide that additional knowledge in an explicit manner.

UmlsBERT [77] is a medical TPLM which was obtained by pre-training ClinicalBERT [58] model on the UMLS data. UMLS is a collection of more than 100 medical concepts; during the process of continual pre-training, the semantic type information is provided to the model explicitly via a specific category of embedding referred to as semantic group embedding.

2.2.3 Designing the pre-training tasks

During the self-supervised learning (SSL) process, the language model will learn universal language representations by performing one or more pre-training tasks. It is of great importance to decide which pre-training task to use for a given TPLM because those tasks should be challenging enough so that they provide as many training signals to the model as possible; moreover, a pre-training task should be similar to the downstream task. Kalyan, Rajasekharan and Sangeetha [27] suggested a comprehensive list of some pre-training tasks usually used in modern TPLMs.

Causal Language Modeling (CLM): In this task, the model tried to predict the next word in a sequence, based on the context. GPT-1 [78] was the first TPLM to use this task and after that, UniLM [79] used both left-to-right and right-to-left CLM as a pre-training task.

Masked Language Modeling (MLM): The main disadvantage of CLM is its inability to leverage both right-to-left and left-to-right contexts during the pre-training process of the model.

Bidirectional contextual information has proved to be far more effective compared to unidirectional information; but it is impossible to use this information in the standard CLM. MLM is an improved version of CLM which allows the usage of tokens from both contexts; and BERT [8] was the first model to use this pre-training task.

Replaced Token Detection (RTD): MLM is a great improvement upon CLM but it has two main drawbacks:

1- MLM provides training signals only for 15% of the tokens in input sequences (it wastes system resources).

2- MLM uses the special mask tokens only during the pre-training process, which causes discrepancy between pre-training and fine-tuning stages.

RTD tackles these issues by identifying the replaced tokens. MLM involves predicting the original tokens by processing masked tokens; while RTD is in fact a binary classification task, which tries to classify each one of the input tokens as replaced or not replaced. ELECTRA [75] was the first model to use this pre-training task.

Shuffled Token Detection (STD): Similar to RTD, STD is a token level task that tries to discriminate between shuffled and non-shuffled tokens; hence, avoids discrepancy between pre-training and fine-tuning stages of a model. Panda et al. [80] performed continual pre-training on RoBERTa using STD and showed that this improves the performance of the model in many NLP tasks.

Random Token Substitution (RTS): RTD is more sample efficient compared to MLM, but the problem with this task is that it requires a separate generator for corrupting the input sequences; and training such a generator model will be computationally expensive. In order to address this issue, Di Liello, Gabburo and Moschitti [81] proposed RTS which tries to identify the randomly substituted tokens. In this method, 15% of tokens are randomly substituted with other tokens of the vocabulary. RTS is sample efficient like RTD but does not require any separate generator model to corrupt the input sequence.

Swapped Language Modeling (SLM): SLM is another category of pre-training tasks that tries to overcome the drawbacks of MLM by corrupting the sequence with random tokens from vocabulary at a probability of 0.15 [81]. SLM replaces those tokens with random tokens which is similar to RTS, but unlike RTS which is sample efficient by involving every token in the input sequence, SLM is not sample efficient as it involves only 15% of input tokens.

Translation Language Modeling (TLM): This is an extension of MLM for using parallel data in cross-lingual pre-training. XLM [49] was the first language model to use TLM as a task of pre-training, followed by XNLG [53]. In this task, input is a pair of sentences which are parallels of each other (one sentence is translation of the other one in a given language). TLM masks tokens of both of those sentences in a random way. Since prediction of masked tokens involves context from both of the sentences, TLM helps the model to learn cross-lingual mapping.

Alternate Language Modeling (ALM): ALM involves predicting the masked tokens in the code-switched sentences generated from parallel sentences [50]. For a given parallel sentence pair, a code-switched sentence is generated by randomly substituting some phrases of the first sentence with their translations from the second sentence. After that, the input tokens will be masked following the same settings as standard MLM. By pre-training the model on code-switched sentences, it will learn the relationships between those languages in a much better way. Yang et al. [50] proved that a model pre-trained by ALM outperforms one that has used TLM as its pre-training task.

Sentence Boundary Objective (SBO): SBO involves predicting the masked tokens based on the span boundary tokens and position embedding [82]. Similar to MLM, SBO tries to predict the masked tokens as well; but it is different from MLM in three aspects:

1. SBO masks only contiguous spans of tokens while MLM masks tokens randomly.
2. Unlike MLM, in SBO, the prediction of masked tokens involves span boundary tokens and position embedding as well (in MLM it only involves masked tokens).
3. SBO is more challenging compared to standard MLM because it tries to predict entire spans of text (MLM tries to predict just the next word in a sentence).

Next Sentence Prediction (NSP): NSP is a binary sentence pair classification task that involves identifying consecutive sentences; thus helping the model to learn relationships between sentences [8]. Pre-training the model at the sentence level is useful in downstream tasks like question answering, natural language inference (NLI), and semantic textual similarity (STS) which involve sentence pair inputs [27].

Sentence Order Prediction (SOP): NSP takes into account both topic and coherence prediction but since topic prediction is easier, effectiveness of NSP is questionable [30]. SOP is a sentence-level pre-training task that tackles this issue by only paying attention to sentence coherence. ALBERT [83] was the first ever model to use SOP as a pre-training task. This task involves identifying whether the given sentences are swapped or not.

Sequence-to-Sequence LM (Seq2SeqLM): In a standard MLM task, the original words are predicted by feeding the vector of masked tokens to a softmax layer over the vocabulary. Seq2SeqLM is an extension to this architecture for pre-training encoder-decoder-based models such as T5 [15], mT5 [41], or MASS [84]. In Seq2SeqLM, the context includes all words in the input masked sequence and the left side words in the predicted target sequence. With a masked sequence as input to the encoder, the decoder predicts the masked words from left to right sequentially.

Denoising Auto Encoder (DAE): In DAE, the pre-training task for the model is to reconstruct the original text from a corrupted input sequence [70]. This input text could be corrupted at token, phrase, sentence, or document level; and just like Seq2SeqLM, this task can be used for training encoder-decoder-based models.

However, the main advantage of DAE over Seq2SeqLM is that it is more sample efficient because DAE involves reconstructing the entire original text while Seq2SeqLM is concerned only with reconstructing masked tokens. BART [70] was one of the TPLMs using DAE as a pre-training task.

2.2.4 Choosing the pre-training methods

Training a new model from scratch by only using SSL is highly expensive and consumes a lot of pre-training time. Instead, there are various methods that can be used alongside SSL for a more efficient pre-training process. Kalyan, Rajasekharan and Sangeetha [27] have categorized all the pre-training methods into 5 major groups:

1. Pre-training from Scratch (PTS)

PTS is the most straightforward method of pre-training; in this method, all the layer parameters of the model are randomly initialized and then learned during the pre-training process by minimizing their loss over the pre-training tasks. Some of the most famous TPLMs pre-trained with this method are BERT [8], RoBERTa [30], ELECTRA [75], T5 [15], and SciBERT [85]. As mentioned previously, the main drawback of PTS is that it is computationally expensive and requires a large number of GPUs/TPUs.

2. Continual Pre-training (CPT)

CPT tries to tackle issues of PTS by initializing weights of the model from an existing TPLM and then further pre-training it. Unlike PTS, in continual pre-training model parameters are not learned from scratch; hence, it consumes less pre-training time and computing power. CPT is commonly used to develop TPLMs in specific domains and according to whether the continual pre-training is performed with or without new vocabulary, it can be divided into three sub-categories:

a) CPT with existing vocabulary:

   The main drawback of this variant is lack of target domain-specific vocabulary, especially in cases where the primary corpus does not include many of the domain-specific words. In such a situation, domain-specific words are usually split into various sub-words which severely decreases model performance in downstream tasks. BioBERT [55], LEGAL-BERT [86], CodeBERT [87], HateBERT [32], and SentiX [34] are among the models pre-trained with this method.

b) CPT with new vocabulary:

   This method tries to overcome the aforementioned issues by continual pre-training over target domain or language-specific vocabulary. In this method, during continual pre-training, the embedding layer is randomly initialized but all the other layers will be initialized with existing language model parameters. Pay attention that the performance of the model obtained by continual pre-training with new vocabulary is slightly less but on par with the performance of the model acquired by PTS. Some of the models pre-trained using this method are RuBERT [88], BERTimbau [89], Slavic-BERT [90] (all initialized from multilingual BERT [8] but further pre-trained with language-specific vocabulary), and PTT5 [91] (initialized from T5 [15] and further pre-trained with language specific vocabulary).

c) CPT with existing and new vocabulary:

Yao et al. [92] were the first ones to perform this method by proposing the "adapt and distill" approach for adapting general models to a specific domain with the use of vocabulary expansion and knowledge distillation. The main benefit of this approach over the previous methods is that it not only adapts general models to a specific domain, but also reduces the model size.

3. Simultaneous Pre-training (SPT)

Both PTS and CPT methods require large volumes of domain-specific unlabeled text to pre-train the model and this limits usability of these methods in cases that there is a lack of domain-specific text. For instance, domain-specific text in some languages other than English is available only in small quantities. Using PTS or CPT over a small amount of domain-specific text data will cause the model to overfit and not generalize well on new input data. Simultaneous pre-training (SPT) tackles this issue by pre-training the model from scratch using both the general and domain-specific text. This method ensures having a balanced pre-training by up-sampling the domain-specific corpus. As an example, Wada et al. [93] showed that Japanese clinical BERT pre-trained using SPT outperforms Japanese clinical BERT trained from scratch.

4. Task Adaptive Pre-training (TAPT)

The common point between PTS, CPT, and SPT is that all of them require the model to be pre-trained over large volumes of general or domain-specific or combined text data. As all these methods involve training over a large amount of text, they are considered to be expensive. Unlike these approaches, TAPT allows the model to learn fine-grained task-specific knowledge along with domain-specific knowledge by pre-training on a rather small dataset of task-specific unlabeled text [60]. Since it requires a comparatively smaller amount of text compared to its forerunners, TAPT is considered a cheaper approach and a more efficient solution for cases where the cost of finding large domain-specific text is unjustifiably high. TAPT is usually used as a complementary approach to other pre-training methods like PTS and CPT.

5. Knowledge Inherited Pre-training (KIPT)

All the pre-training methods discussed so far, solely depended on self-supervised learning (SSL) in order to pre-train the model; which will be highly expensive and time consuming if the model is larger than a specific size. Inspired by the real-world observation that humans learn not only through self-learning but also by gaining knowledge from other people, Qin et al. [94] proposed a novel pre-training method which pre-trains the model using both self-supervised learning and knowledge distillation, referred to as KIPT. Using KIPT, it is possible to reuse the knowledge available in existing pre-trained models to pre-train a new model. By learning from a teacher model in addition to self-supervised learning, the student model gains more knowledge and also converges faster; this makes KIPT more proficient compared to previous methods which only relied on self-supervised learning. Furthermore, Qin et al. [94] have shown that models pre-trained with KIPT outperform models trained using self-supervised learning only. CPM-2 [95] was the first large scale TPLM pre-trained with the use of KIPT.

2.2.5 Choosing the pre-training dynamics

Liu et al. [30] showed that carefully choosing the pre-training dynamics such as dynamic masking, larger batch sizes, more pre-training steps, and longer input sequences can further enhance the performance of the pre-trained model. One important subject to pay attention to is that using large batch sizes may cause difficulty in optimization. In order to solve this issue, it is recommended to (1) Linearly increase the learning rate in the early steps of pre-training and (2) Use different learning rates in different layers which also helps speeding up convergence [96].

Materials discussed in this chapter so far were centered on how to pre-train a model and create a TPLM. But it is a valuable effort to provide a taxonomy for distinguishing between various families of TPLMs and better understand their different characteristics. Kalyan, Rajasekharan and Sangeetha [27] have classified TPLMs according to four different perspectives and here three of them are mentioned Section 2.3.

2.3. Taxonomy of TPLMs

2.3.1 Architecture

The original transformer architecture proposed by Vaswani et al. [4] consists of both encoder and decoder layers. According to whether a TPLM was trained with encoders, decoders, or both of them, they can be categorized into three groups:

(1) Encoder based: These models consist of an embedding layer followed by a stack of encoder layers; and output of the last encoder layer is considered to be the final context which represents the processed input sequence. As a general rule, encoder based TPLMs are used in natural language understanding (NLU) tasks. BERT [8], XLNet [29], RoBERTa [30],

ELECTRA [75], ALBERT [83], and XLM-E [44] are among the models that fall into this group.

(2) Decoder based: These models consist of an embedding layer followed by a stack of decoder layers. Decoder based models are usually used in natural language generation tasks. GPT-1 [78], GPT-2 [97], and GPT-3 [98] are among the models that fall into this group.

(3) Encoder-Decoder based: This category of models are more suitable for modeling tasks which require both understanding and generation of natural language; such as machine translation, text summarization, and so on. T5 [15], mT5 [41], mT6 [48], BART [70], mBART [99], PLBART [100], PEGASUS [16], and PALM [101] are just a few of the pre-trained language models that fall into this group.

2.3.2 SSL

The self-supervised learning technique used in a TPLM is one of their key ingredients and according to the SSL technique used for training the models, they can be categorized into four major groups.

(1) Generative SSL

In the generative SSL technique, the model will be trained by trying to predict tokens in a given sentence. Generative SSL can be performed in three different scenarios:

1- The model tries to predict the next token based on current tokens (Causal Language Modeling - CLM).

2- Model predicts the masked tokens in a given sequence of text data (MLM and its variants such as TLM, Seq2SeqLM).

3- Model reconstructs original text from the corrupted text (DAE).

GPT-1 [78], GPT-2 [97], GPT-3 [98] (based on CLM), RoBERTa [30], XLM [49], XLM-R [42] (based on MLM and TLM), BART [70], mBART [99] (based on DAE), MASS [84], T5 [15], and mT5 [41] (all based on Seq2SeqLM) are among the models that fall into generative SSL category.

(2) contrastive SSL

In contrastive SSL technique, the model learns by comparison. Currently, there are no TPLMs solely trained using contrastive SSL, but rather use it in continual pre-training to further improve the model (usually for learning the sentence-level semantics). For instance, CERT [102] uses contrastive SSL to improve BERT model by injecting more sentence-level semantics, and outperforms BERT in many NLP tasks. CERT, Mirror-BERT [103], and SimCSE [104] are among the models that fall into this category.

(3) Adversarial SSL

Adversarial SSL technique trains the model by distinguishing corrupted tokens (which can be replaced or shuffled). ELECTRA [75] is pre-trained using replaced token detection (RTD) and XLM-E [44] is pre-trained using multi-lingual replaced token detection (MRTD) and translation replaced token detection (TRTD). Panda et al. [80] on the other hand, used adversarial SSL via shuffled token detection (STD) to further improve RoBERTa model.

(4) Hybrid SSL

Hybrid SSL technique is just an amalgamated version of other SSL techniques applied to a single TPLM. For instance, BERT [8] uses generative (MLM) and contrastive SSL (NSP), and ALBERT [83] uses generative (MLM) and contrastive SSL (SOP). For example, infoXLM [43] makes use of generative (MLM, TLM) and contrastive SSL (cross lingual contrastive pre-training). Moreover, CLINE [105] was developed by further pre-training RoBERTa using generative (MLM), contrastive, and adversarial SSL (RTD).

2.3.3 Extensions

TPLMs discussed in this section couldn't be classified into any other groups but they all extend the capabilities of a normal TPLM in specific ways.

Kaplan et al. [106] proved that the performance of a TPLM can be highly improved by increasing the batch size, volume of pre-training corpus, and number of training steps. But the main problem here is that large models will cause higher levels of latency in real-world applications where resources are limited and require fast responses. Compact TPLMs are a category that try to address this issue by compressing the size of TPLM. In order to reduce the size of TPLMs and make them faster, many compression techniques have been proposed in recent years; Gupta and Agrawal [107] give a comprehensive list of all of these techniques and divide them into four main groups:

1- Pruning:

   Michel, Levy and Neubig [108] and Voita et al. [109] have shown that most of the attention heads in a TPLM are redundant and can be removed during the inference stage of the model. Furthermore, Fan, Grave and Joulin [110] proved that lots of encoder layers can be dropped during pre-training process without influencing model performance. All deep learning models (including TPLMs) are over parameterized i.e., some of their components (such as weights, attention heads, or layers) can be removed during or after the pre-training process without much impact on the performance [27]. Doing so, will reduce the size and inference time of the model.

2- Quantization:

   Comparably similar to pruning which compresses a model by removing less important weights, quantization compresses the model by using fewer bits to represent each weight. By default, TPLMs use either 32 or 16 bits for representing each parameter, but quantized models might use 8 bits (e.g. Q8BERT [111]) or even lesser (e.g. Q-BERT [112], TernaryBERT [113], and Gobo[114]). Various methods have been proposed to reduce the performance drop in cases where the number of bits is less than 8, such as mixed-bit quantization [112, 114], combining knowledge distillation with quantization [113], and product quantization [115].

3- Knowledge Distillation:

Firstly introduced by Buciluǎ, Caruana and Niculescu-Mizil [116] and later generalized by Ba and Caruana [117] and Hinton, Vinyals and Dean [118], this model compression method allows for training compact student models from larger teacher models. During this procedure, the student model tries to learn the generalization ability of the teacher by reproducing its behavior. Hence, the performance of the student model will be on par with the teacher model. The generalization approach of [117] was to train the student model using L2 loss between output logits of teacher and student model. But [118] on the other hand, uses cross-entropy between softmax logits of teacher and student models (a.k.a. soft loss) as well as cross-entropy of student model's predictions and actual labels (hard loss). DistilBERT [119], TinyBERT [120], PKD-BERT [121], MobileBERT [122] and MiniLM [123] are among the models which have used knowledge distillation as a means of compressing the model.

4- Parameter Sharing:

ALBERT [83] has managed to achieve a lite version of BERT model by performing cross layer parameter sharing and factorized embedding parameterization. Using those two techniques, ALBERT has managed to achieve 18x fewer parameters and 1.7x faster training compared to the BERT-large model.

The current standard approach for adapting a general model to a specific domain is continual pre-training (CPT); although the resulting models of CPT achieve good performance, this process is computationally expensive and not environmentally friendly with CO2 emissions [7]. This problem fostered research focus on developing a new family of methods for adapting general models to a specific domain in a less computationally expensive fashion; this category of models are collectively referred to as green TPLMs. Some of the example efforts for developing these green models include:

1- GreenBioBERT [124]: Was developed by extending the vocabulary of the BERT-base model using domain-specific word embeddings. It achieves comparable performance with BioBERT [55] which was developed by continual pre-training during 10 days using eight v100 NVIDIA GPUs.
2- exBERT [125]: It was developed by extending general BERT with domain-specific WordPiece embeddings and an extension module. The technique used in this model is to freeze all the other parameters while performing continual pre-training over the extra WordPiece embeddings and extension module parameters; hence, reducing the price of calculations.
3- E-BERT [126]: This model was developed by extending BERT model vocabulary with the Wikipedia2Vec [127] entity vectors. This model does not need any further pre-training but has managed to outperform some pre-trained models such as ERNIE [128] and KnowBERT [129] over the LAMA [130] benchmark.

Most TPLMs use sub-word embeddings which are based on the idea that only rare and misspelled words should be represented using sub-words while frequently used words should be represented as they are. This architecture is advantageous enough to make sub-words the default embeddings used for TPLMs; but they have two intrinsic drawbacks:

1- They cannot encode fine-grained character-level information in the word representation.
2- They are susceptible to noise; even simple typos in text can change the representation of words and impact model learning.

In order to address the issues above, a character-based category of TPLMs have been proposed which includes models such as CharacterBERT [68], CharBERT [71], and AlphaBERT [69].

Almost all the TPLMs discussed in this paper so far make use of some kind of embeddings (sub-words, words, characters, etc.) for converting input text into tokens. Apart from all of their advantages, these tokenization methods have some drawbacks:

1- Sub-word embeddings require a fixed number of vocabulary which tends to be so large in multilingual models. In this case, the model needs to have a vocabulary matrix in which each token is mapped with a vector and softmax matrix in the output layer. These huge matrices will occupy a significant amount of model parameters. For instance, in a multilingual model such as mT5, these two matrix parameters are about 66% of all the model parameters [41].
2- Having a fixed vocabulary makes the adaptation of a general model to new domains inefficient because many domain-specific words are not represented properly in the vocabulary. This problem will impact the model adaptation and downstream performance on new tasks [131].
3- Both sub-word and character embeddings require an explicit tokenizer for splitting the input sequence based on the white spaces or punctuations in the text. This becomes a major problem in case of some languages that do not use white space or punctuation for separating words. For

example, Chinese and Thai do not use white space as separators; moreover, Hawaiian and Twi languages use punctuations as consonants [132].

There has been an increasing interest towards overcoming these issues with tokenization-free TPLMs. This category of models have no need for language-specific tokenizers, are more robust to noise, and do not suffer from a large vocabulary or significant amount of model parameters. Some examples of tokenization-free models are:

CANINE [132]: The first ever tokenization-free TPLM, it directly operates on character sequence. The main method of this model is to apply convolution layers on character sequence (in order to reduce input sequence length) and then apply transformer encoder layer stack.

ByT5 [133]: This is an improved version of the T5 model for handling input text at a byte-level without using any fixed vocabulary.

Charformer [134]: This model follows a different procedure compared to the aforementioned examples. It makes use of a novel tokenizer that leverages gradients for automatically learning sub-words from characters. Doing so, eliminates the requirement for a fixed vocabulary. Charformer performs on par with models such as T5 while outperforming byte-level models like ByT5. Moreover, gradient-based tokenizers have an interpretable output (unlike CANINE).

General purpose TPLMs like BERT have failed in creating effective sentence-level embeddings. In order to address that issue, many supervised and self-supervised methods have been proposed recently for leveraging sentence-based TPLMs. Some examples of sentence-level TPLMs are as follows:

1- SBERT [135]:

One of the first supervised approaches for extending BERT in order to generate high-quality sentence embeddings, SBERT fine-tunes BERT using Siamese network over NLI [136, 137] and STSb [138] datasets (which are sentence pair classification tasks); hence, the model learns sentence-level semantics and is capable of generating quality sentence vectors.

2- DvBERT [139]:

It uses multi-view learning for further extending SBERT by adding word-level interaction features across two sentences.

3- IS-BERT [140]:

Zhang et al. [140] showed that performance of SBERT and Dv-BERT methods is limited when the training data is scarce or distribution of test data differs significantly from that of training data. Moreover, the necessity of labeled datasets limits the application of these models in domain-specific scenarios where labeled data is not abundant. In order to overcome these problems, many SSL methods were proposed; for instance, IS-BERT uses mutual information maximization strategy for learning quality sentence embeddings.

4- Mirror-BERT [103]:

Another SSL method that trains BERT by using a contrastive learning-based objective to push positive sentence pairs (which are obtained by random masking in input space or applying dropout in feature space) to have similar representations.

5- TSDAE [141]:

It firstly gives a corrupted sentence to the original model and then tries to train the model by reconstructing the original sentence from the embedding vector created by the encoder layer of the initial model.

6- SimCSE [104]:

A contrastive-learning based framework for improving TPLMs' functionality on generating sentence embeddings; it can work on both labelled and unlabeled data. Historically, TPLMs are known to require pre-training on large amounts of data for long durations; recently, some new architectures have been proposed for achieving similar (or better) performance using less pre-training data and less costs. Some examples of these models which are commonly referred to as efficient TPLMs are these:

7- DeBERTa [142]:

Improves upon basic BERT model by using disentangled attention mechanism (for representing a word with separate vectors to encode its content and position information) and enhanced masked decoder (it predicts masked tokens instead of softmax layer during pre-training).

These two novel methods have improved the model's pre-training efficiency and the DeBERTa model which was pre-trained on 78GB of data, has managed to outperform RoBERTa which is a model pre-trained on 160GB of data.

8- ConvBERT [143]:

It tries improving BERT by using a mixed attention block, consisting of self-attention module (for modeling global dependencies) and span based dynamic convolution modules (for modeling local dependencies). Furthermore, authors of ConvBERT observed that some attention heads are needed to model only local dependencies; hence, they replaced these attention heads with span-based dynamic convolution modules. Using these refined model architectures, ConvBERT managed to outperform ELECTRA-base model while having less than ¼ of its pre-training cost.

The self-attention mechanism in transformers updates the representation of each input token by attending to all the other tokens in the input sequence [4]. But the main problem is with the scalability of transformer models (in long input sequences) because of the quadratic time complexity of the self-attention module. In order to overcome this drawback and extend usage of TPLMs to even long input sequences, two main methods were proposed:

1- sparse self-attention:

It tries to reduce the complexity of self-attention mechanism by including sparsity bias, which reduces the number of query-key pairs that each query attends to. Longformer [22], ETC [144], Big-Bird [145], and Reformer [20] are some of the more-known TPLMS that implement this technique.

2- linearized self-attention:

It tries to achieve reduced complexity by disentangling the attention with kernel feature maps and then computing the attention in reverse order. Performer [146] is one of the models that makes use of this technique.

During the pre-training process, model learns the knowledge available in a large volume of text by solving one or more challenging pre-training tasks over that corpus. Recent works have tried to further improve TPLMs by integrating the knowledge already available in external knowledge bases. These TPLMs which are enhanced with knowledge from external sources are collectively referred to as Knowledge Enriched TPLMs. Some of these external knowledge bases usually leveraged in these models are WordNet [147], Wikidata (both are used for the general domain), and UMLS [148] (in the biomedical domain). Some examples of knowledge enriched TPLMs are CausalBERT [149] (injects causal knowledge into BERT model using two novel pre-training tasks on cause-effect pairs), KnowBERT [129], SenseBERT [150], LIMIT-BERT [151], LIBERT [152] (is pre-trained from scratch using lingual relation classification-LRC- task along with MLM and NSP), SentiLARE [153] (introduces label-aware MLM to inject POS tag and word polarity information into BERT), ERNIE [128], E-BERT [126] (all of them in the general domain); and Clinical KB-BERT [154], Clinical Kb-ALBERT [154], SAPBERT[155], UmlsBERT [77], and CODER[156] (all of them in medical domain).

As it is shown in Figure 1.2, initial TPLMs had a size range around 110M to 340m parameters. Kaplan et al. [106] showed that the performance of TPLMs is strongly related to three factors:

1- Number of parameters.
2- Size of the pre-training corpus.
3- Amount of pre-training compute.

According to that paper, all of these three must be scaled up at the same time in order to achieve an optimal performance. These observations triggered the sparks for developing many large or very large models. Some of the latest very large-scale TPLMs are these: GPT-3 (175B parameters) [98], PANGU (200B parameters) [157], GShard (600B parameters) [158], and Switch-Transformers (1.6T parameters) [159]. Apart from all of their advantages over smaller models, large-scale TPLMs are not easily usable (mostly because of their size and needed computing power) and could cause serious environmental impacts.

According to the author's personal experience of working with TPLMs during this research work, if researchers are mainly relying on the free version of Google Colab [160] for their computations, it is almost impossible to train a model with more than 500M parameters in a reasonable amount of time. The first section of this chapter mainly focused on transformer based language models, their characteristics, and how they can be used for performing NLP tasks via transfer learning, and a comprehensive taxonomy of all kinds of TPLMs and how they differ from each other. But since during this research work various TPLMs have been used for performing text summarization, we need to study the current state of research work in this field as well. Hence, the rest of this chapter focuses on neural text summarization and revisits most recent research papers published in this area.

2.4. Text summarization

Automatic text summarization task aims at automatically producing a shorter version of a text while preserving most of its meaning [161]. Such a task could be performed in two main ways:

1) Extractive Summarization:

    It is usually considered as a binary classification problem which decides whether to include some span of input text (usually a sentence) in the final summary or not. It is considered to be the easier version of summarization since it merely copies informative parts of input text [16].

2) Abstractive Summarization:

Unlike the extractive type, this technique may generate novel words which did not exist in the input. Hence, it is harder to perform compared to extractive summarization. A good abstractive summary covers principal information in the input and is linguistically fluent [16]. Some previous works suggest that combining extractive summarization objectives with abstractive techniques can further boost the performance of abstractive summarization [162], [163].

Most of the earlier research work in this field focused on extractive summarization but then the research trend gradually shifted towards more abstractive techniques of text summarization [164]. According to the taxonomy proposed by [165], the modern abstractive approaches for summarizing single-document texts could be classified to the following main groups:

a) Attention-based summarization:

Rush, Chopra and Weston [166] were one of the first to make use of attention mechanism in a summarization system. The architecture proposed was a feed-forward neural language model combined with a contextual input decoder (and an attention-based encoder) for generating headlines from each input sentence. Moreover, the authors experimented with tuning a very small set of additional features that trade off the abstractive/extractive tendency of the system.

Recurrent Attentive Summarizer (RAS): Originally introduced in [167], RAS is an extension of the attention-based summarization model which tries to leverage improved attention models. The decoder of RAS is created with vanilla RNN and the convolutional attention-based encoder convolves over the full embeddings.

Another extension of the ABS model uses a technique called “Abstract Meaning Representation” or AMR; which was first introduced in [168]. It is a rooted, directed, and acyclic graph that tries to encode the meaning of a sentence.

AMR usually can represent some important information of the sentence such as predicate-argument structures or named entities. This information contained by AMR provides effective clues for the model to produce output summaries.

b) Encoder-decoder models with various attention mechanisms:

Lopyrev’s model [169] was one of the first architectures that included encoder-decoder in the summarization model by utilizing two deep stacks of four LSTM (Long Short-Term Memory) units as the decoder and encoder. Furthermore, this architecture tested with two different attention mechanisms. The first one was the dot attention. The other one consisted of a small set of neurons for computing the attention weights, which made the output of the model more interpretable and easier to be studied.

Chen et al. [170] incorporated a coverage mechanism (mentioned as “distraction”) for implementing encoder-decoder architecture into their model. This architecture proposed a novel attention mechanism that not only considers specific regions and contents of input documents, but also distracts them to traverse between different contents of a document. This model performs three different kinds of distraction for implementing its encoder-decoder architecture:

1) Kullback-Leibler (KL) divergence [171] of attention weights.
2) Distraction on the attention-primed input content vector.
3) Distraction on the hidden output vector.

One of the major drawbacks in the models mentioned above is that they often generate repetitive and incoherent phrases when trained on longer documents. In order to mitigate these issues, Paulus, Xiong and Socher [172] proposed an intra-attention-based encoder-decoder architecture coupled with reinforcement learning objectives. Moreover, an intra-decoder attention mechanism is introduced to prevent the model from attending over the same parts of the input on different decoding steps. This was the first end-to-end abstractive summarization model trained on NYT dataset.

Graph based attention mechanism was initially introduced by Tan, Wan and Xiao [173], this technique generates a hidden state graph containing representations of all the sentences in an input sequence. Afterwards, this graph is used to determine the attention score of each sentence. This model uses two different LSTMs as the word encoder and the sentence encoder respectively. The LSTM-based decoder receives the preprocessed document as its initial state and predicts the sentence representations sequentially.

Gehrmann, Deng and Rush [162] invented “Bottom-up attention” technique as an effort for improving the selection process of summary content; in order to do so, a data-efficient content selector is employed for determining salient phrases in a source document. The content selector computes the selection probability for each word of input sequence; these probabilities were later used for modifying the attention distribution of the model.

Hsu et al. [174] proposed a unified model which tried to combine the strength of both extractive and abstractive realms of summarization. It firstly applies the SummaRuNNer [175] (as the abstractive part of architecture) to all the input sentences and the output of this layer of the model will be treated as sentence-level attention. Similarly, the word-level attention vectors of this model are obtained by the pointer-generator model [176] (the extractive part of architecture) and finally, these sentence-level and word-level attention vectors are modulated with each other in order to decrease the chance of words in less attended sentences to appear in the output text.

Fan, Grangier and Auli [177] designed an encoder-decoder model where both of those elements were Convolutional Neural Networks (CNNs). The connecting network between encoder and decoder was a multi-hop attention module. This model delved deeper into implementing all blocks via attention, by adopting self-attention in the decoder to enable the model to refer back to previously generated words.

RC-Transformer (RCT) was proposed by Cai et al. [178] as an extension of transformer architecture with an additional RNN-based encoder; which was meant to capture the sequential context representations of input sequence. As a result, this model architecture features two encoders (for capturing contextual semantic representations and modeling sequential context, respectively) and single decoder. RCT can not only learn long-term dependencies, but also address the inherent shortcomings of Transformer. This model has proved to be speedier than classical Seq2Seq models [165].

c) Methods tackling repetition problem:

The model introduced in [179] tried to address the problem of unknown words in the final summary by using the idea that some of the words in this generated summary also appear in the input text. Their suggested architecture can be trained to pick up a word from the input sequence and copy it to the final summary. At each time step, this model first determines whether to take a word from the predefined shortlist or to copy a word from the source sentence.

Gu et al. [180] further developed the copying mechanism mentioned above by using a Bidirectional Recurrent Neural Network (BiRNN) as its encoder and also a vanilla RNN as the decoder. In the proposed architecture, the decoder reads the context vector and creates summary sentences with either copying or generation.

The model suggested by See, Liu and Manning [176] took a different approach than the previous papers for overcoming the unknown words and repetition problem in summarization. Their suggested model augmented the attentional encoder-decoder model with a hybrid pointer-generator (PG) network. They also used a bidirectional LSTM as the encoder and another LSTM as the decoder. In their proposed model, the PG network will calculate the generation probability from the context vector, decoder state, and decoder input. Then, the coverage mechanism is used to keep track of what has been summarized; hence, discouraging repetition in the final summary.

d) Inspirations by human summarizers:

Zeng et al. [181] were inspired by human summarizers who read the text numerous times before generating the summaries, proposed the "read-again" model. This architecture first reads the input text and then does a second read for paying special attention to the words that are relevant to generate the final summary. The procedure for doing so is to use the information acquired from the first read to bias the second read representations.

Li et al. [163] drew inspiration from human summarizers who first skim through a document and delete unnecessary materials. They extended the encoder-decoder framework by adding information selection layer which is designed as a filtering network that consists of two parts:

1) Gated global selection for filtering out the unnecessary information.
2) Local sentence selection for picking up salient parts of document sequentially to produce the final summary.

Inspired by how humans summarize long documents, Chen and Bansal [182] devised a model that combined extractive and abstractive summarization with reinforcement learning. It features a temporal convolutional model which is applied to compute the representation of each sentence. Furthermore, it uses a LSTM to train a pointer network for extracting sentences recurrently. Lastly, in order to address the problem of out of vocabulary words, the copying mechanism is used to directly copy some of those words from the input document to the generated summary.

e) Incorporating additional features:

Nallapati et al. [183] tried to address the common problems of classical attentional encoder-decoder architectures by adopting a bidirectional gated recurrent unit (BiGRU) as the encoder and also a gated recurrent unit (GRU) as its decoder. The architecture they proposed, firstly captures all the additional linguistic information (part of speech tags, named entity tags, and TF-IDF scores of the words) for each word along with the original word embedding to concatenate into a single vector as the word representation. Moreover, to capture the hierarchy of sentence-to-word structure, the attention mechanism is operated at both levels (i.e. sentence and word) simultaneously.

In order to improve informativeness of resulting summaries, Jiang et al. [184] incorporated topic-oriented keyword information from the input document via a new attention mechanism. In this architecture, the sum of word embeddings was used as a part of input for the attention distribution. This way, a topic-oriented summary can be generated in a context-aware manner with guidance.

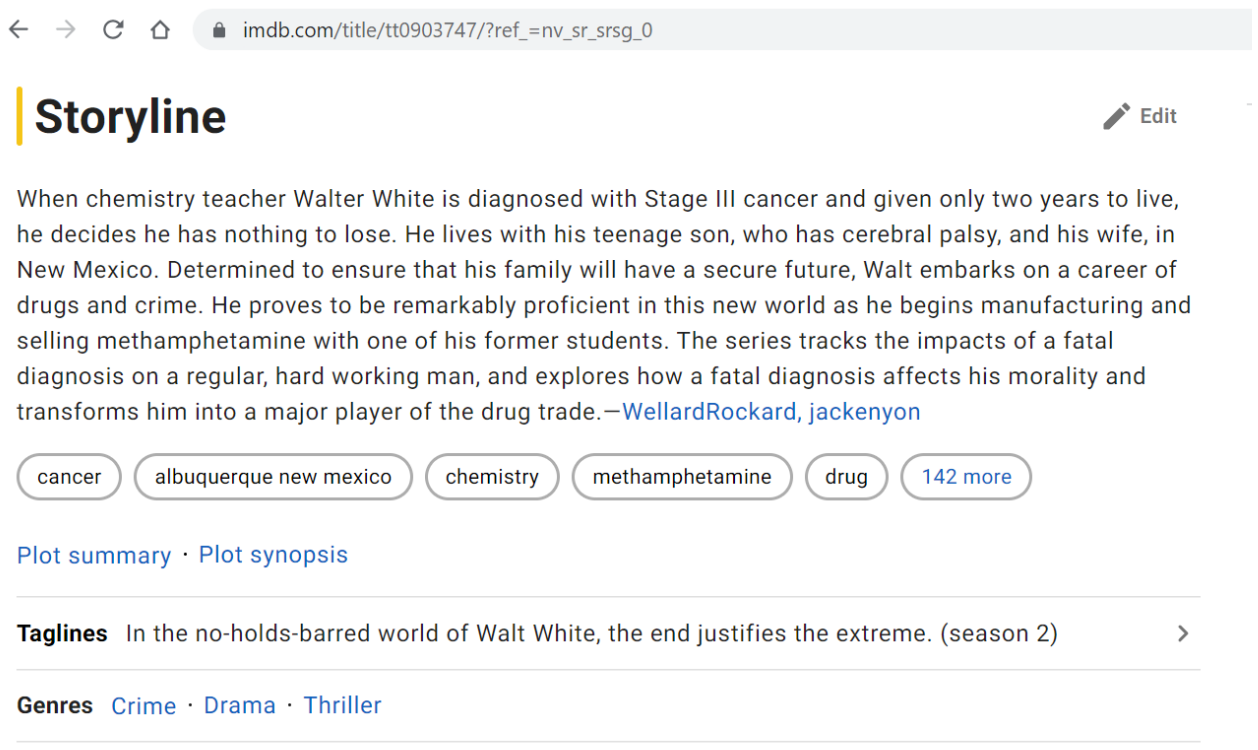
CHAPTER 3
DATA AND METHODS

3.1 Data

The first novel point of this research work is to create a new labelled dataset of movie storylines provided by IMDB online database. From now on, this dataset is referred to as **IMDB storylines**. Since IMDB provides an invaluable body of online information about various media, the obvious result is that so many data scientists have paid academic interest in digging this database and leveraging the raw data for extracting valuable knowledge. Hence, there is a huge volume of academic research centering on data gathered from IMDB.

But the majority of research works currently performed on this colossal web database has mostly focused on sentiment analysis [185, 186, 187], knowledge discovery [188, 189, 190, 191], and recommender systems [192, 193]. To the best of my knowledge, at the time of this writing, there has not been major research efforts for performing abstractive text summarization on the storylines of movies from IMDB.

Figure 3.1 shows the corresponding web page of a specific TV series on IMDB website; including the storyline for that title which is published in IMDB. The first step of this research was to scrape the IMDB website and extract as much storylines as possible and save them into an offline database.


imdb.com/title/tt0903747/?ref_=nv_sr_srsg_0

Storyline

When chemistry teacher Walter White is diagnosed with Stage III cancer and given only two years to live, he decides he has nothing to lose. He lives with his teenage son, who has cerebral palsy, and his wife, in New Mexico. Determined to ensure that his family will have a secure future, Walt embarks on a career of drugs and crime. He proves to be remarkably proficient in this new world as he begins manufacturing and selling methamphetamine with one of his former students. The series tracks the impacts of a fatal diagnosis on a regular, hard working man, and explores how a fatal diagnosis affects his morality and transforms him into a major player of the drug trade.—WellardRockard, jackenyon

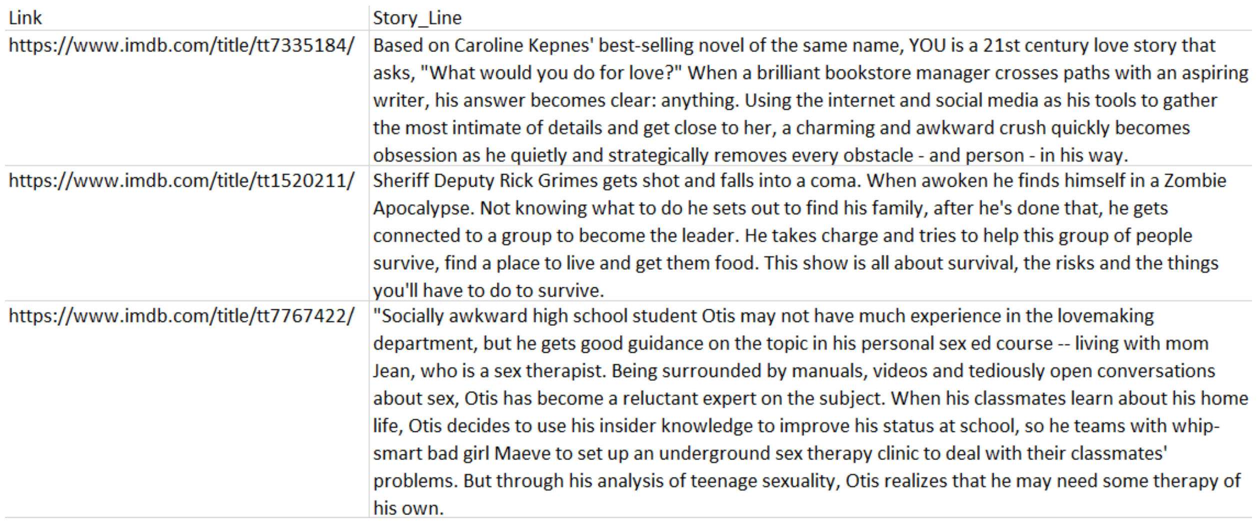
Taglines In the no-holds-barred world of Walt White, the end justifies the extreme. (season 2)

Genres Crime · Drama · Thriller

Figure 3.1 – IMDB example webpage

At the first step of this research, in order to extract storylines of movies from “imdb.com”, a **web scraper script** was written (in Python language) which browsed through the IMDB website and extracted the storylines of given movies then saved them into an offline database. For a complete explanation of how this script is working refer to the source code of the thesis. But as a general walkthrough, the script first goes through a list of web addresses which contain IMDB pages. Then, the script downloads all the html content for any of those pages. Finally, the downloaded content is parsed and the html specific tag that contains the story line is extracted as a text. Each one of these storylines is saved with the corresponding link of its web page as a single entry of the IMDB storylines dataset. The format for saving IMDB storylines dataset is comma separated values (.CSV).

At this step, the IMDB storylines dataset looks like Figure 3.2; it only has two columns, the first column is the link of each title (movie, TV series, etc.) on IMDB and the second column is the storyline of that title.



Link	Story_Line
https://www.imdb.com/title/tt7335184/	Based on Caroline Kepnes' best-selling novel of the same name, YOU is a 21st century love story that asks, "What would you do for love?" When a brilliant bookstore manager crosses paths with an aspiring writer, his answer becomes clear: anything. Using the internet and social media as his tools to gather the most intimate of details and get close to her, a charming and awkward crush quickly becomes obsession as he quietly and strategically removes every obstacle - and person - in his way.
https://www.imdb.com/title/tt1520211/	Sheriff Deputy Rick Grimes gets shot and falls into a coma. When awoken he finds himself in a Zombie Apocalypse. Not knowing what to do he sets out to find his family, after he's done that, he gets connected to a group to become the leader. He takes charge and tries to help this group of people survive, find a place to live and get them food. This show is all about survival, the risks and the things you'll have to do to survive.
https://www.imdb.com/title/tt7767422/	"Socially awkward high school student Otis may not have much experience in the lovemaking department, but he gets good guidance on the topic in his personal sex ed course -- living with mom Jean, who is a sex therapist. Being surrounded by manuals, videos and tediously open conversations about sex, Otis has become a reluctant expert on the subject. When his classmates learn about his home life, Otis decides to use his insider knowledge to improve his status at school, so he teams with whip-smart bad girl Maeve to set up an underground sex therapy clinic to deal with their classmates' problems. But through his analysis of teenage sexuality, Otis realizes that he may need some therapy of his own.

Figure 3.2 – IMDB storyline dataset, the initial step

One major challenge at this step of collecting the IMDB storylines was that the servers of imdb.com (like all the other busy websites) have some professional security layers which try to avoid automatic programs (such as web scraping scripts) from browsing their website. As a result, it was necessary to make sure that the automatic script was not kicked out of connection with the server.

Finally, the script extracted storylines and web links for 5,075 movies and saved all of them inside the aforementioned dataset (all the possible duplicates were removed at this step). One of the main intentions in devising this dataset is to create a high-quality knowledge base, which allows comparably small-sized TPLMs to learn from the labels created by larger models about how to efficiently summarize movie storylines.

In order to accomplish the above goal, in the next step of the project, all the storylines of the database are summarized with the use of a huge language model which is specifically trained for abstractive summarization, i.e. PEGASUS model [16].

PEGASUS is a large TPLM (with 568 M hyperparameters) which is specifically trained by Google Brain for performing abstractive summarization. As previously mentioned in Chapter 2, all of the TPLMs are pre-trained on some huge corpora with one or more pre-training tasks. The pre-training task that is used in PEGASUS model is intentionally similar to summarization; so this model has already learnt how to perform text summarization. The specific pre-training task used in PEGASUS is called Gap Sentence Generation (GSG). During this pre-training task, important sentences are removed/masked from an input document and the model tries to generate those sentences together as one output sequence from the remaining sentences, similar to an extractive summary. Doing so, the PEGASUS model has already been trained about how to perform text summarization (for more information about other common pre-training tasks refer to Chapter 2).

Apart from being specifically pre-trained for text summarization, there are some other reasons that make PEGASUS an optimal language model for performing text summarization and led us to pick this model for creating the summaries inside the IMDB storylines dataset.

Firstly, according to [16], PEGASUS works surprisingly well in low-resource summarization examples, surpassing previous state-of-the-art results on 6 datasets with only 1000 examples. Hence, this model will work better in cases that the input movie storyline is not long enough.

Second, since this research did not have enough funds for using human summarizers on all the entries in the dataset, the best choice of model would be the ones that have a performance on par with human summarization quality. According to [16], PEGASUS achieves human performance on multiple datasets.

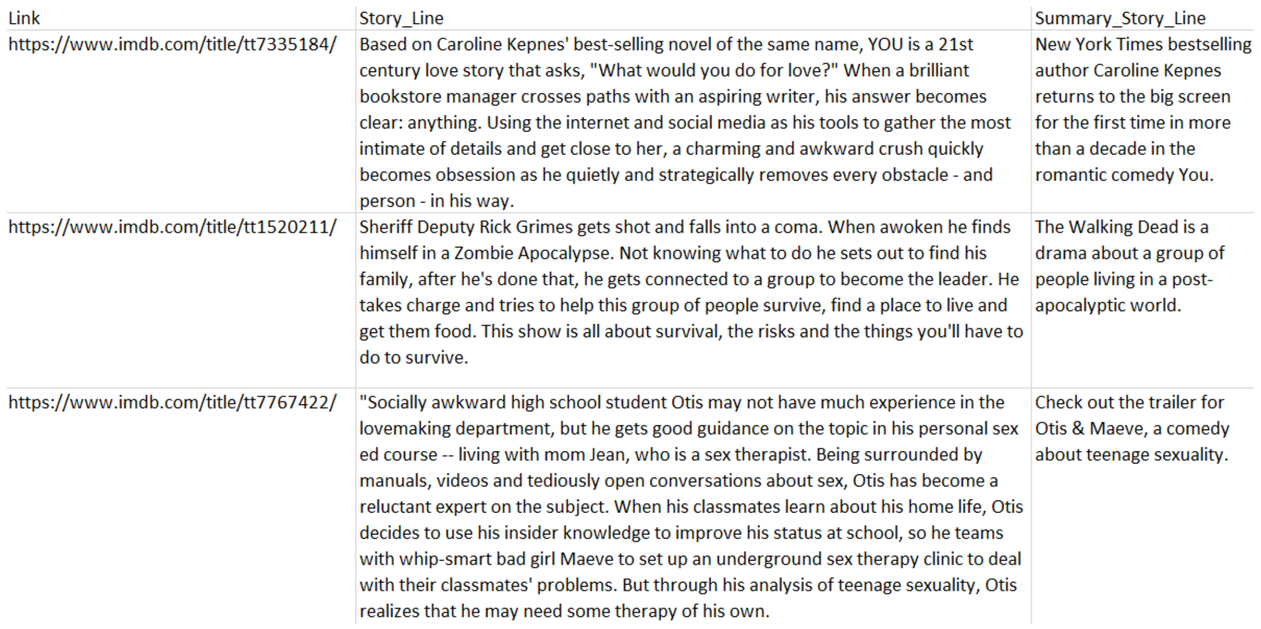
Lastly, PEGASUS is a rather large model (with almost 568 M parameters). The decision made in this research was to use the summaries created by this huge model as a starting point for the labels inside IMDB storylines dataset. These labels inside the dataset could later be used for training smaller models. Thus, the summarization knowledge of a huge model like PEGASUS can be transferred to smaller models with less number of hyperparameters (this point is further explained later in this chapter).

All reasons above led PEGASUS model to be chosen for creating the reference summaries from the storylines already in the IMDB storylines dataset. As mentioned before, in order to fine-tune a TPLM, a labelled dataset is needed and with the use of PEGASUS, it is possible to create the labels of IMDB storylines dataset with high quality.

3.2 Model

Now that the reasons for choosing PEGASUS model are explained, let us see how this model was used in the IMDB storylines dataset. For complete explanation refer to the source code of this project but simply explained, at first, PEGASUS model and its tokenizer will be loaded into 2 Python objects. Next, input text is fed to tokenizer for getting the processed input. The processed input goes through PEGASUS model and the result is logits (i.e., mathematical probabilities). Finally, the tokenizer converts the logits to meaningful text which is the final abstractive summary created by the PEGASUS model. The resulting abstractive summaries are added as a new column of the database so the smaller models could be trained on them.

Figure 3.3 shows a selected view of the IMDB storylines dataset at this stage. As can be seen, the summaries that PEGASUS model has created for each storyline of the dataset have been saved as a new column of dataset. The labelled dataset that has been created so far is a good starting point for the future steps of this project.

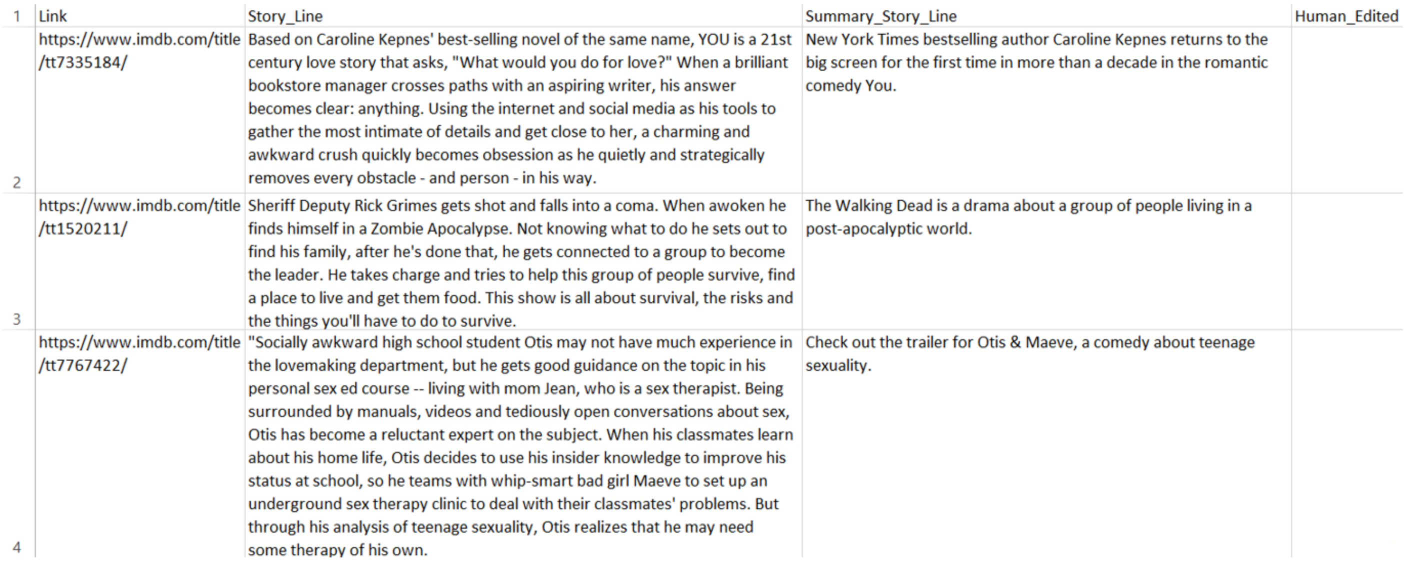


Link	Story_Line	Summary_Story_Line
https://www.imdb.com/title/tt7335184/	Based on Caroline Kepnes' best-selling novel of the same name, YOU is a 21st century love story that asks, "What would you do for love?" When a brilliant bookstore manager crosses paths with an aspiring writer, his answer becomes clear: anything. Using the internet and social media as his tools to gather the most intimate of details and get close to her, a charming and awkward crush quickly becomes obsession as he quietly and strategically removes every obstacle - and person - in his way.	New York Times bestselling author Caroline Kepnes returns to the big screen for the first time in more than a decade in the romantic comedy You.
https://www.imdb.com/title/tt1520211/	Sheriff Deputy Rick Grimes gets shot and falls into a coma. When awoken he finds himself in a Zombie Apocalypse. Not knowing what to do he sets out to find his family, after he's done that, he gets connected to a group to become the leader. He takes charge and tries to help this group of people survive, find a place to live and get them food. This show is all about survival, the risks and the things you'll have to do to survive.	The Walking Dead is a drama about a group of people living in a post-apocalyptic world.
https://www.imdb.com/title/tt7767422/	"Socially awkward high school student Otis may not have much experience in the lovemaking department, but he gets good guidance on the topic in his personal sex ed course -- living with mom Jean, who is a sex therapist. Being surrounded by manuals, videos and tediously open conversations about sex, Otis has become a reluctant expert on the subject. When his classmates learn about his home life, Otis decides to use his insider knowledge to improve his status at school, so he teams with whip-smart bad girl Maeve to set up an underground sex therapy clinic to deal with their classmates' problems. But through his analysis of teenage sexuality, Otis realizes that he may need some therapy of his own.	Check out the trailer for Otis & Maeve, a comedy about teenage sexuality.

Figure 3.3 – IMDB storyline dataset, the second step

At the next step of the project, in order to further enhance the quality of the IMDB storylines dataset, for some entries of the dataset the automatically created summary was further edited/rewritten by a **human summarizer**. By doing so, we can make sure that the quality of the summaries inside the dataset is even higher than what a language model like PEGASUS can create and at least for some entries, those summaries are further enhanced (or maybe even rewritten) by a human summarizer. Thus, the value of the knowledge present in the IMDB storylines dataset is increased at this step.

At this step, the dataset is edited by a human summarizer; a new column, "Human_Edited" (see Figure 3.4) is added to it which indicates whether the summary in that entry of the dataset has been edited/rewritten by human summarizer or not. If for a specific entry the value in the column "Human_Edited" is 1, it means the corresponding summary of that entry in the column "Summary_Story_Line" has been enhanced by a human summarizer. In the entries that the summary is not edited by a human, the value in the "Human_Edited" column is just null.



1	Link	Story_Line	Summary_Story_Line	Human_Edited
2	https://www.imdb.com/title/tt7335184/	Based on Caroline Kepnes' best-selling novel of the same name, YOU is a 21st century love story that asks, "What would you do for love?" When a brilliant bookstore manager crosses paths with an aspiring writer, his answer becomes clear: anything. Using the internet and social media as his tools to gather the most intimate of details and get close to her, a charming and awkward crush quickly becomes obsession as he quietly and strategically removes every obstacle - and person - in his way.	New York Times bestselling author Caroline Kepnes returns to the big screen for the first time in more than a decade in the romantic comedy You.	
3	https://www.imdb.com/title/tt1520211/	Sheriff Deputy Rick Grimes gets shot and falls into a coma. When awoken he finds himself in a Zombie Apocalypse. Not knowing what to do he sets out to find his family, after he's done that, he gets connected to a group to become the leader. He takes charge and tries to help this group of people survive, find a place to live and get them food. This show is all about survival, the risks and the things you'll have to do to survive.	The Walking Dead is a drama about a group of people living in a post-apocalyptic world.	
4	https://www.imdb.com/title/tt7767422/	"Socially awkward high school student Otis may not have much experience in the lovemaking department, but he gets good guidance on the topic in his personal sex ed course -- living with mom Jean, who is a sex therapist. Being surrounded by manuals, videos and tediously open conversations about sex, Otis has become a reluctant expert on the subject. When his classmates learn about his home life, Otis decides to use his insider knowledge to improve his status at school, so he teams with whip-smart bad girl Maeve to set up an underground sex therapy clinic to deal with their classmates' problems. But through his analysis of teenage sexuality, Otis realizes that he may need some therapy of his own.	Check out the trailer for Otis & Maeve, a comedy about teenage sexuality.	

Figure 3.4 – Final state of IMDB storylines dataset

The biggest hardship in this step was that due to lack of time and funding, it was not possible to perform human enhancement on all the entries of the IMDB storylines dataset. At the end, only the summaries that looked somewhat fallacious were enhanced slightly. At the time of this writing, less than 20% of summaries in IMDB storylines dataset have been enhanced by a human summarizer but this issue could be addressed in the future development iterations of this research project.

At the end of this step, the IMDB storylines dataset looks like Figure 3.4. This figure shows a selected view from the final version of this dataset. As it is observable in this figure, the dataset has four columns:

1) Link: Link of the movie's corresponding web page on IMDB.
2) Story_Line: The official story line of the movie from IMDB.
3) Summary_Story_Line: The abstractive summary of the story line which was created by the PEGASUS model and in some cases further enhanced by a human summarizer.
4) Human_Edited: An integer value indicating whether the corresponding summary was further enhanced by human editor or not (if the summary is edited by a human, this column is 1, otherwise it is null).

At this step, the project managed to achieve one of its goals, which was to create a valuable, task-oriented, and labelled dataset (i.e., the IMDB storylines dataset) for summarizing movie storylines. Because this dataset has the knowledge of PEGASUS model and human summarization at the same time, it can be highly beneficial for teaching other TPLMs about how to summarize movie storylines in a more proficient way and enhance the performance of any models in this field.

Now that the IMDB storylines dataset has been created, the second part of this research project is to put it in use and showcase its potential in creating customized TPLMs which can outperform equivalent models that are not trained on this dataset. The process of using a TPLM in a new downstream task is "transfer learning" and the procedure that teaches a TPLM the knowledge of a labelled dataset is referred to as "fine-tuning".

In Chapters 1 and 2 concepts of “transfer learning” and “fine-tuning” were comprehensively explained; and one of the goals of this research was to put in use those guidelines and leverage our novel dataset for fine-tuning some general-purpose language models on the task of summarizing movie storylines.

At this step of the project the IMDB storylines dataset is used to fine-tune a rather small TPLM (T5-base) and create an enhanced language model which performs exceptionally better than its un-tuned equivalent in summarizing movie storylines. The initial TPLM chosen for this part of project was T5-base which is a rather small model with 220 M parameters. Before explaining how the fine-tuning procedure was performed, let us dive deeper in **T5 model** and see its characteristics.

Initially introduced by Google Brain in 2020, T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks. T5 is somewhat different from other common TPLMs because it introduces a unified framework which converts all NLP problems into a text-to-text format (T5 stands for “**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer”). This technique allows the researcher to use the same model, loss function, hyperparameters, etc. across a diverse set of tasks. Hence, it is not hard to leverage this model in new fields such as text summarization because T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task.

Moreover, T5 was the first language model that has been pre-trained on “Colossal Clean Crawled Corpus” (a.k.a. C4), which is one of the biggest corpora (unlabeled text datasets) in the history of

TPLMs consisting of hundreds of gigabytes of clean English text scraped from the web. That means all the different checkpoints of T5 model (no matter how many parameters they have) already have a fairly good general understanding of English language such as grammar, syntax, and even semantics (for more information about various kinds of pre-training corpora, refer to Section 2.2.1 about "prepare the pre-training corpus"). This previous knowledge of the model about the language can be beneficial while using it in downstream tasks (such as text summarization).

Google Brain has published various checkpoints of T5 model with different sizes. Although all of those checkpoints are the same model pre-trained on C4 and are implemented the same way, but the number of hyperparameters are different which leads to different sizes and accuracies. Currently published checkpoints of T5 are as follows [194]:

- T5-Small: 60 million parameters
- T5-Base: 220 million parameters
- T5-Large: 770 million parameters
- T5-3B: 3 billion parameters
- T5-11B: 11 billion parameters

All the characteristics above make T5 a desired example for a model to be taught how to summarize movie storylines with the use of IMDB storylines dataset. Since the main development machine used in this research was the free version of Google Colab with no more than 13 GB of RAM, the final decision made was to choose the basic version of T5 (T5-base) to perform the fine-tuning. Since the fine-tuning implementation for all those different checkpoints of T5 is roughly the same, if the machine used in future development iterations of this project had more powerful hardware, it will be fairly easy to change the code in order to perform the fine-tuning with bigger checkpoints of T5.

The IMDB storylines dataset has been created with the usage of both PEGASUS model (which is a fairly large TPLM with 568 M parameters) and human summarization. As a result, this dataset could be used for enhancing the performance of any models (no matter how big or small they are) in summarizing movie storylines. But the main power of this dataset is to enable smaller TPLMs to perform as good as larger models. This way, the final user of that fine-tuned model only pays the memory and computing price of using a smaller model but receives the result that is typical of a bigger model and is closer to human performance.

So, the “T5-Base” checkpoint of T5 model was chosen for being fine-tuned on IMDB storylines dataset and showcase the power of this dataset for improving the performance of smaller TPLMs in the task of summarizing movie storylines. Remember that by further enhancing IMDB storylines dataset via human editing, the quality of this dataset can be further increased and it will have a more noticeable performance gain for any models that are fine-tuned over this dataset (even for very huge TPLMs such as T5-11B).

As mentioned before, T5-base was chosen as a pioneer model to be fine-tuned on IMDB storylines dataset. At the end, the performance of fine-tuned T5-base model is compared with the un-tuned model in order to better understand how much effective it can be to fine-tune a TPLM over IMDB storylines dataset (results are meticulously discussed in Chapter 4). At the end of this fine-tuning step, a customized version of T5-base model was presented which has comparably better performance on summarizing movie storylines compared to the un-tuned T5-base model.

Now that specific characteristics of T5 model have been discussed, we fine-tune T5-base on the IMDB storylines dataset. The fine-tuning approach used in this research follows the tutorial mentioned in documentation of PyTorch library [195]. The fine-tuning process was performed on a Google Colab [160] with 12.69 GB of RAM, 78.19 GB of disk space, and also a Tesla K80 as a GPU. For complete explanation of fine-tuning process refer to the project's source code but all the major steps that the Python notebook goes through for fine-tuning T5-base on the dataset are as follows:

1) Download the IMDB storylines dataset into a dataframe and perform some preprocessing operations.
2) Split the whole dataset into subsets of training, validation, and test sets (the training and validation sets are used for fine-tuning and test set is used for calculating the ROUGE scores of both fine-tuned and un-tuned models; these scores are used for comparing the performance of those two models).
3) Create a Python class which extends map-style dataset of PyTorch library and allows for working with the dataset more easily (this object abstracts the dataset and helps working with it more easily).
4) Create another Python class which extends Data Module of PyTorch library and is a reusable class for encapsulating all the steps needed for processing data during fine-tuning, such as:

- Download/tokenize/process the data entries.
- Clean and save to disk.
- Load inside a Dataset.
- Apply transforms (rotate, tokenize, etc.).
- Wrap data inside a DataLoader.

5) Load the “T5-base” model and its tokenizer into two separate objects (all of the different checkpoints of T5 model and their corresponding tokenizers are published via “transformers” API from HuggingFace [196]).
6) Create a histogram of the number of tokens in the training dataset, which showed that the majority of storylines in the training dataset have less than 500 tokens and also the majority of summaries in training set have less than 100 tokens. Hence, in the Data Module class (which describes the process of fine-tuning) the maximum number of tokens for storylines is set on 512 and maximum number of tokens for summaries is set on 128.

   Furthermore, a batch size of 2 and epoch number of 3 were selected for fine-tuning process. The selection of the number of epochs and batch size was based on the experience that it is highly time consuming to fine-tune any model which has more than 200M parameters on the free version of Google Colab. Hence, the numbers were chosen in a way that the training process would not be insanely lengthy (nevertheless, it still took hours to fully fine-tune the model with these characteristics on a free version of Google Colab notebook).
7) Create a Python class for implementing the model architecture that performs fine-tuning. The forward pass of the model, loss function, logits, training step, validation step, and test step of the model were all defined as methods of this class.

8) Train the T5-base model over the training dataset via the classes defined earlier (it was the most time-consuming part of the notebook).
9) After training the model and achieving the least global error (by minimizing the loss function), the best checkpoint of the model found is frozen and saved to the disk in order to be used as the fine-tuned version of T5-base model which now has learnt how to summarize movie storylines.
10) The final step of this notebook is to use both the fine-tuned and un-tuned versions of the T5-base model on all the data points in the test dataset and calculate ROUGE scores (further explained in Chapter 4) for both of them; the result of this comparison is shown and discussed in Chapter 4.

CHAPTER 4
RESULTS AND DISCUSSION

Before starting to interpret the results of this research, let us discuss the ROUGE score, its calculation and interpretation. Originally introduced by Lin 2004 [17], ROUGE is an acronym for Recall-Oriented Understudy of Gisting Evaluation. It is a suite of mathematical measures, which intend to determine the quality of an automatically generated summary compared to another (reference) summary created (usually) by humans.

Lin 2004 [17] introduces four different ROUGE measures: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. The first two (ROUGE-N and ROUGE-L) are the most popular and are used for evaluating the results of this research project.

ROUGE-N, as calculated in Equation (1), is considered the n-gram recall between a candidate summary and a set of reference summaries.

$$ROUGE - N = \frac{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match(gram_n)}}{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \qquad (1)$$

In the Equation (1), “gram_n” stands for the length of the n-gram, and “$\text{Count}_{\text{match}}(\text{gram}_n)$” refers to the maximum number of n-grams occurring both in a generated summary and a set of reference summaries. ROUGE-N is a recall-related measure because the denominator of the Equation (1) is the total sum of the number of n-grams occurring in the reference summary.

According to the size of n-grams used, there can be different kinds of ROUGE-N measures but the most commonly used measures in this category are ROUGE-1 and ROUGE-2 (using unigrams and bigrams respectively) which are usually considered a means of assessing **informativeness** of the tested summary.

ROUGE-L on the other hand tries to find the longest common subsequence between the generated summary and reference summary. For example, imagine a sequence Z = $[z_1, z_2, \ldots, z_n]$ is a subsequence of another sequence X = $[x_1, x_2, \ldots, x_m]$, if there is a unique increasing sequence $[i_1, i_2, \ldots, i_k]$ of indices of X such that for all j = 1, 2, …, k, we have $x_{ij} = z_j$ [197]. Given the sequences X and Y, the longest common subsequence (LCS) is a common subsequence of them that has the maximum length. ROUGE-L is usually considered a means of assessing **fluency** in generated summaries.

In order to further discover the effect of fine-tuning T5-base (or any other general purpose TPLMs) on the dataset of IMDB storylines, both the un-tuned and fine-tuned versions of the T5-base model were applied to the entries in test dataset. The average value of ROUGE-1 and ROUGE-L scores for these models have been calculated and compared with each other. While discussing the results, whenever the “un-tuned model” is mentioned, the basic checkpoint of T5 (a.k.a. T5-base) is intended. Furthermore, whenever “fine-tuned model” is mentioned, T5-base which was fine-tuned on the training subset of the dataset is meant.

The ROUGE-1 and ROUGE-L scores used in this research contain 3 different parts:

1- Precision:

Precision is a common indicator of a model’s performance, which refers to the quality of positive predictions made by the model. Simply explained, precision refers to the number of true positives divided by the total number of positive predictions (true positives plus false positives). In the context of ROUGE-1, unigram precision [198] is the proportion of words in generated summary that are also in the reference summary as shown in equation (2).

$$Precision = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)} \qquad (2)$$

2- Recall:

Recall is another mathematical measure for quantifying how well the model has identified true positives. It simply indicates the total number of important sentences retrieved (in the generated summary), divided by the total number of important sentences present (in the reference summary). In the context of ROUGE-1, unigram recall [198] reflects the proportion of words in reference summary that are also present in the generated summary as given in equation (3).

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)} \qquad (3)$$

3- F-measure:

F-Measure or F1-Score [198] provides a way to combine both “Precision” and “Recall” into a single measure which represents a tradeoff between those properties. This measure is usually used at situations that precision and recall are equally important and is given in equation (4).

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \qquad (4)$$

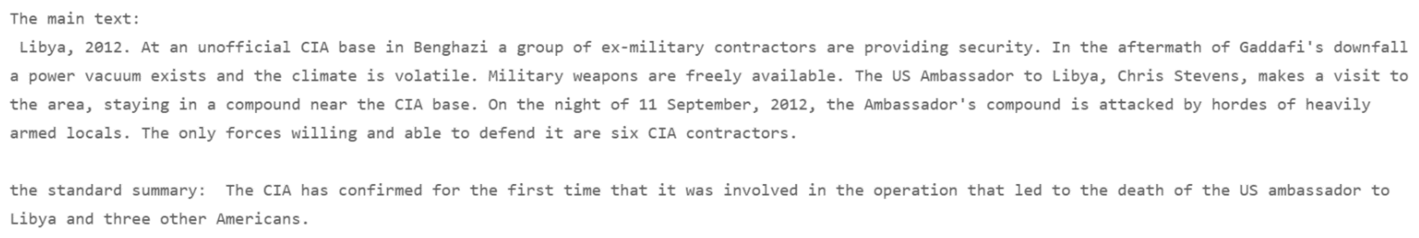
F-measure is easier to work with because instead of trying to make a balance between precision and recall, it is possible to just aim for higher F1 scores and that will imply having good precision and recall values as well. In interpreting the results of this research, only F1 scores have been used for comparing the performance of various models with each other.

Important research efforts on neural text summarization (i.e. performing text summarization via neural networks) such as [16] usually use an additional measure of performance on their models which is known as “human evaluation”. While calculating that measure, reference summaries used are all created by some highly skilled human summarizers.

Research works such as [16] usually use human evaluation for showing that their model summaries are capable of achieving human performance (at least on some specific datasets). Using this measure for studying the performance of some models over a specific dataset (such as the dataset we have) is expensive because it involves employing skilled human summarizers to make high quality summaries for all the entries of that dataset. Unfortunately, this research project did not have enough financial resources to do that; hence, the only measure used for evaluating the performance of various models over IMDB storylines dataset is ROUGE.

After fine-tuning T5-base on the training subset of dataset, the test subset was fed into both fine-tuned and un-tuned versions of T5-base model. Eventually, the “rouge scorer” library [199] from Google was used for calculating ROUGE-1 and ROUGE-L scores for each one of the summaries that both of these models have suggested for entries of test subset. The results were saved into separate files (for the sake of saving memory and computing power, only the ROUGE scores were saved not the generated summaries).

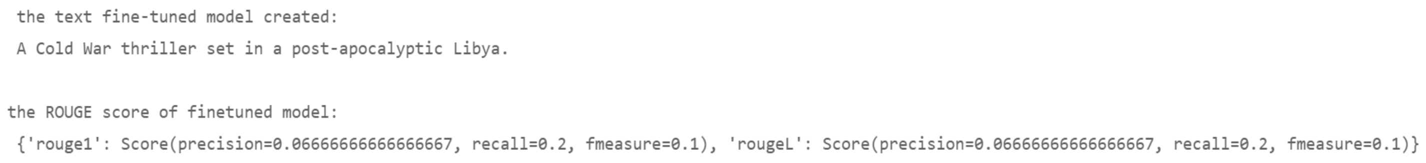
Before discussing the average ROUGE results of un-tuned and fine-tuned models, let us see the outputs these models have created for three example storylines in the test set:



```
The main text:
 Libya, 2012. At an unofficial CIA base in Benghazi a group of ex-military contractors are providing security. In the aftermath of Gaddafi's downfall
a power vacuum exists and the climate is volatile. Military weapons are freely available. The US Ambassador to Libya, Chris Stevens, makes a visit to
the area, staying in a compound near the CIA base. On the night of 11 September, 2012, the Ambassador's compound is attacked by hordes of heavily
armed locals. The only forces willing and able to defend it are six CIA contractors.

the standard summary:  The CIA has confirmed for the first time that it was involved in the operation that led to the death of the US ambassador to
Libya and three other Americans.
```

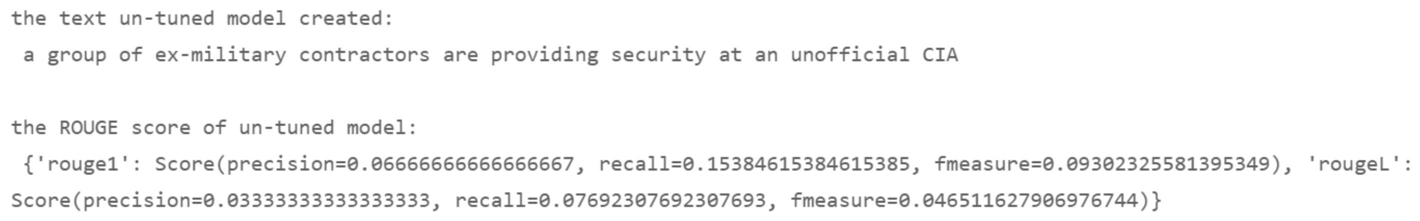
Figure 4.1 – First example, the input storyline and its summary in the dataset



```
 the text fine-tuned model created:
 A Cold War thriller set in a post-apocalyptic Libya.

the ROUGE score of finetuned model:
 {'rouge1': Score(precision=0.06666666666666667, recall=0.2, fmeasure=0.1), 'rougeL': Score(precision=0.06666666666666667, recall=0.2, fmeasure=0.1)}
```

Figure 4.2 – First example, summary created by fine-tuned model and its corresponding ROUGE scores

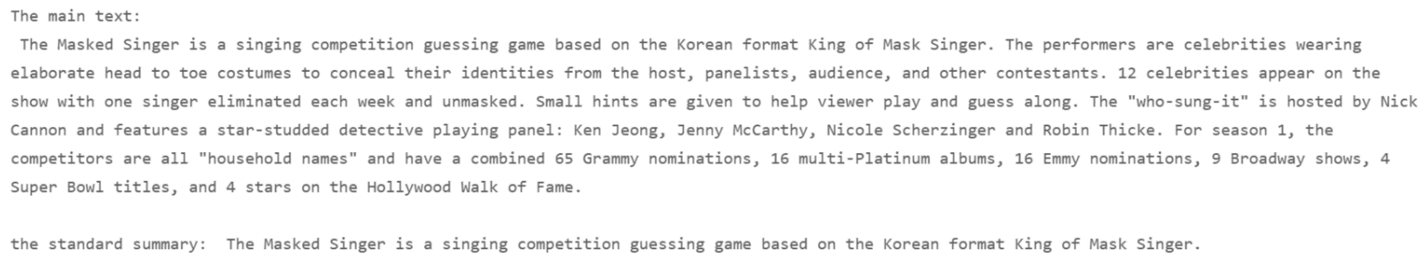


```
 the text un-tuned model created:
  a group of ex-military contractors are providing security at an unofficial CIA

 the ROUGE score of un-tuned model:
  {'rouge1': Score(precision=0.06666666666666667, recall=0.15384615384615385, fmeasure=0.09302325581395349), 'rougeL':
 Score(precision=0.03333333333333333, recall=0.07692307692307693, fmeasure=0.046511627906976744)}
```

Figure 4.3 – First example, summary created by un-tuned model and its corresponding ROUGE scores

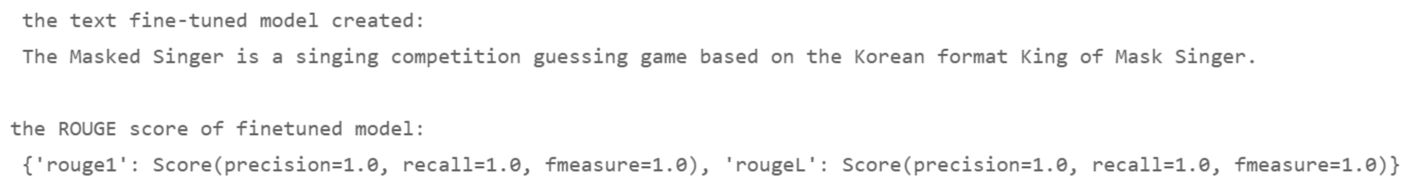
The second example is as follows:



```
The main text:
 The Masked Singer is a singing competition guessing game based on the Korean format King of Mask Singer. The performers are celebrities wearing
elaborate head to toe costumes to conceal their identities from the host, panelists, audience, and other contestants. 12 celebrities appear on the
show with one singer eliminated each week and unmasked. Small hints are given to help viewer play and guess along. The "who-sung-it" is hosted by Nick
Cannon and features a star-studded detective playing panel: Ken Jeong, Jenny McCarthy, Nicole Scherzinger and Robin Thicke. For season 1, the
competitors are all "household names" and have a combined 65 Grammy nominations, 16 multi-Platinum albums, 16 Emmy nominations, 9 Broadway shows, 4
Super Bowl titles, and 4 stars on the Hollywood Walk of Fame.

the standard summary:  The Masked Singer is a singing competition guessing game based on the Korean format King of Mask Singer.
```

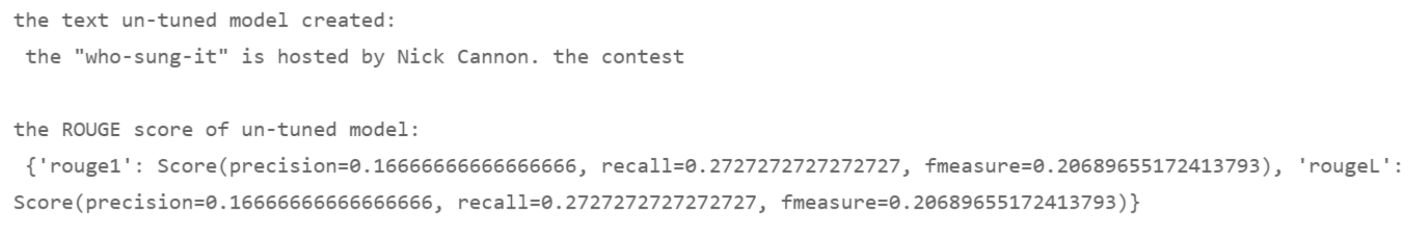
Figure 4.4 - Second example, the input storyline and its summary in the dataset



```
 the text fine-tuned model created:
 The Masked Singer is a singing competition guessing game based on the Korean format King of Mask Singer.

the ROUGE score of finetuned model:
 {'rouge1': Score(precision=1.0, recall=1.0, fmeasure=1.0), 'rougeL': Score(precision=1.0, recall=1.0, fmeasure=1.0)}
```

Figure 4.5 - Second example, summary created by fine-tuned model and its corresponding ROUGE scores

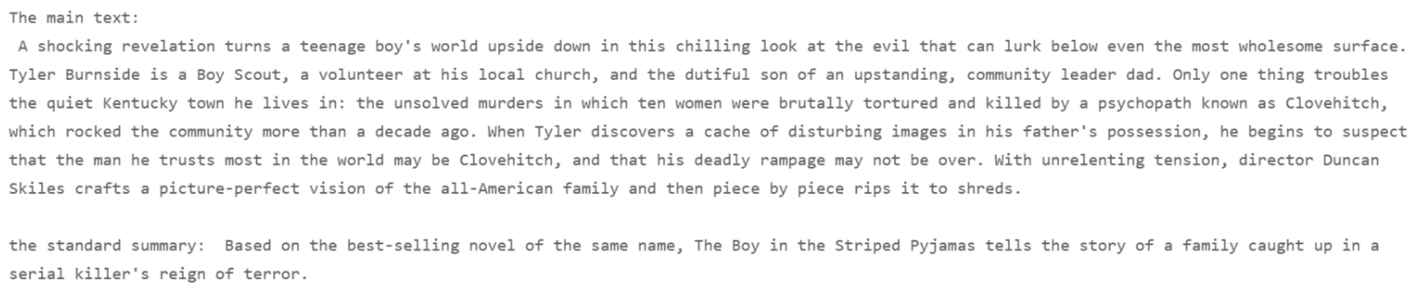


```
 the text un-tuned model created:
  the "who-sung-it" is hosted by Nick Cannon. the contest

 the ROUGE score of un-tuned model:
  {'rouge1': Score(precision=0.16666666666666666, recall=0.2727272727272727, fmeasure=0.20689655172413793), 'rougeL':
 Score(precision=0.16666666666666666, recall=0.2727272727272727, fmeasure=0.20689655172413793)}
```

Figure 4.6 - Second example, summary created by un-tuned model and its corresponding ROUGE scores

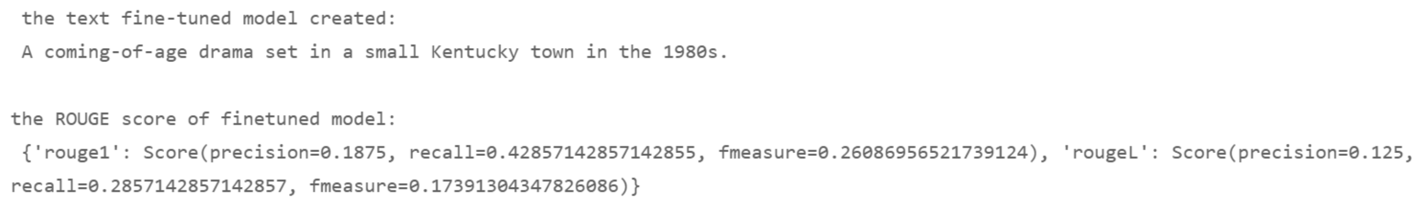
And the third example is like this:



```
The main text:
 A shocking revelation turns a teenage boy's world upside down in this chilling look at the evil that can lurk below even the most wholesome surface.
Tyler Burnside is a Boy Scout, a volunteer at his local church, and the dutiful son of an upstanding, community leader dad. Only one thing troubles
the quiet Kentucky town he lives in: the unsolved murders in which ten women were brutally tortured and killed by a psychopath known as Clovehitch,
which rocked the community more than a decade ago. When Tyler discovers a cache of disturbing images in his father's possession, he begins to suspect
that the man he trusts most in the world may be Clovehitch, and that his deadly rampage may not be over. With unrelenting tension, director Duncan
Skiles crafts a picture-perfect vision of the all-American family and then piece by piece rips it to shreds.

the standard summary:  Based on the best-selling novel of the same name, The Boy in the Striped Pyjamas tells the story of a family caught up in a
serial killer's reign of terror.
```

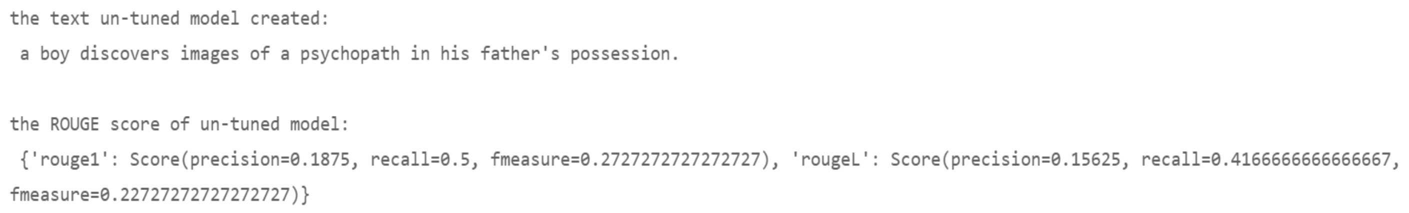
Figure 4.7 - Third example, the input storyline and its summary in the dataset



```
 the text fine-tuned model created:
 A coming-of-age drama set in a small Kentucky town in the 1980s.

the ROUGE score of finetuned model:
 {'rouge1': Score(precision=0.1875, recall=0.42857142857142855, fmeasure=0.26086956521739124), 'rougeL': Score(precision=0.125,
recall=0.2857142857142857, fmeasure=0.17391304347826086)}
```

Figure 4.8 – Third example, summary created by fine-tuned model and its corresponding ROUGE scores



```
the text un-tuned model created:
 a boy discovers images of a psychopath in his father's possession.

the ROUGE score of un-tuned model:
 {'rouge1': Score(precision=0.1875, recall=0.5, fmeasure=0.2727272727272727), 'rougeL': Score(precision=0.15625, recall=0.4166666666666667,
fmeasure=0.22727272727272727)}
```

Figure 4.9 – Third example, summary created by un-tuned model and its corresponding ROUGE scores

Studying the examples above can give a general idea that the fine-tuned model achieves higher F1 scores in the majority of examples but in order to have a precise measurement about how these models have performed over the test set, the average of F1 scores is calculated for both un-tuned and fine-tuned models. All ROUGE scores were saved as comma separated values (CSV files) and evaluated by Microsoft Excel (for more information refer to project's result files). Tables 4.1, 4.2, 4.3, and 4.4 show the average ROUGE scores achieved by un-tuned and fine-tuned versions of T5-base model on the test subset of IMDB storylines dataset.

Table 4.1 – Un-tuned model - ROUGE-1

	precision	recall	F1-score
Average	0.189695	0.255092	**0.205835**

Table 4.2 – Fine-tuned model - ROUGE-1

	precision	recall	F1-score
Average	0.305691	0.327997	**0.29691**

Table 4.3 – Un-tuned model - ROUGE-L

	precision	recall	F1-score
Average	0.165378	0.22072	**0.178769**

Table 4.4 – Fine-tuned model - ROUGE-L

	precision	recall	F1-score
Average	0.277557	0.298345	**0.270169**

As mentioned before, while studying the effect of fine-tuning on T5-base model, only F1 scores are considered (bold numbers in tables above). Studying Tables 4.1 and 4.2 shows that fine-tuning has increased the average F1-score of the model from 0.205835 to 0.29691 (by 44.247%) according to ROUGE-1. Moreover, Tables 4.3 and 4.4 show that fine-tuning has improved the average F1-score of T5-base model from 0.178769 to 0.270169 (by 51.127%) according to ROUGE-L.

The axiomatic conclusion of calculations above is that fine-tuning has highly improved both informativeness (according to ROUGE-1) and fluency (according to ROUGE-L) of summaries for movie storylines generated by the T5-base model.

It was not feasible for this research project to use human evaluation for further investigating the effect of fine-tuning on t5-base model, but since some of the reference summaries in IMDB storylines dataset were edited by a human summarizer, it can be hypothesized that human evaluation will prove the positive effect of fine-tuning on helping the model to achieve human performance in producing concise and meaningful story lines but further verification of this assumption needs more research.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This research managed to create a dataset of movie storylines and further optimize it. This task-oriented, labelled, and relatively small dataset is valuable for teaching all pre-trained transformer-based language models about how to efficiently summarize movie story lines. The majority of those language models are pre-trained on news datasets and Wikipedia text which have formal tone but as we witnessed in this project, movie storylines tend to contain slang or even out of vocabulary words. Datasets like this can teach that specific tone of speech to any pre-trained models and help further customizing the model.

By fine-tuning T5-base on the dataset created in this research, an optimized version of this model was created with better performance at summarizing movie storylines. Since the summaries used in the dataset were created by PEGASUS and further enhanced by human summarizing, this research managed to train a smaller model (like T5-base) with the labels created (i.e. the knowledge provided) by a larger model and human summarizer.

5.2 Future work

With more time and funding, this project could be further developed in multiple dimensions, such as:

1- The dataset currently contains 5075 entries, its size can be increased to augment the knowledge gained by fine-tuning a model over this dataset.

2- One possible improvement is to use human editing on more entries of the dataset. This way, fine-tuned models will create summaries of better quality which are closer to human performance.

3- Another prospective improvement is to use human evaluation as another way of more accurately measuring the effects of fine-tuning on the performance of pre-trained models.

4- T5-base which was used in this project is a rather small model (compared to others) and is not considered to be the best/most accurate pre-trained model for academic or business usage. In the future development iterations of this project, it is possible to use bigger models with more hyperparameters (but remember that doing so requires a more powerful development machine compared to Google Colab notebook used for this project).

REFERENCES

1- Cambria, Erik, and Bebo White. "Jumping NLP curves: A review of natural language processing research." *IEEE Computational intelligence magazine* 9.2 (2014): 48-57.

2- Martinez, Hector P., Yoshua Bengio, and Georgios N. Yannakakis. "Learning deep physiological models of affect." *IEEE Computational intelligence magazine* 8.2 (2013): 20-33.

3- https://youtu.be/iWea12EAu6U accessed December 13th 2021

4- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

5- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).

6- Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).

7- Strubell, Emma, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for deep learning in NLP." *arXiv preprint arXiv:1906.02243* (2019).

8- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

9- https://huggingface.co/course/chapter1 Accessed November 29, 2021.

10- https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/ Accessed November 29, 2021.

11- Srinivasan, Tejas, and Yonatan Bisk. "Worst of Both Worlds: Biases Compound in Pre-trained Vision-and-Language Models." *arXiv preprint arXiv:2104.08666* (2021).

12- S. Bozinovski, A. Fulgosi (1976). The influence of pattern similarity and transfer of learning upon training of a base perceptron B2. (original in Croatian: Utjecaj slicnosti likova i transfera ucenja na obucavanje baznog perceptrona B2), Proc. Symp. Informatica 3-121-5, Bled.

13- Tan, Chuanqi, et al. "A survey on deep transfer learning." *International conference on artificial neural networks*. Springer, Cham, 2018.

14- Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering* 22.10 (2009): 1345-1359.

15- Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." *arXiv preprint arXiv:1910.10683* (2019).

16- Zhang, Jingqing, et al. "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization." *International Conference on Machine Learning*. PMLR, 2020.

17- Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." *Text summarization branches out*. 2004.

18- Liu, Yang. "Fine-tune BERT for extractive summarization." *arXiv preprint arXiv:1903.10318* (2019).

19- Kim, Yoon, et al. "Structured attention networks." *arXiv preprint arXiv:1702.00887* (2017).

20- Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." *arXiv preprint arXiv:2001.04451* (2020).

21- Roy, Aurko, et al. "Efficient content-based sparse attention with routing transformers." *Transactions of the Association for Computational Linguistics* 9 (2021): 53-68.

22- Beltagy, Iz, Matthew E. Peters, and Arman Cohan. "Longformer: The long-document transformer." *arXiv preprint arXiv:2004.05150* (2020).

23- Katharopoulos, Angelos, et al. "Transformers are rnns: Fast autoregressive transformers with linear attention." *International Conference on Machine Learning*. PMLR, 2020.

24- Tay, Yi, et al. "Sparse sinkhorn attention." *International Conference on Machine Learning*. PMLR, 2020.

25- Wang, Sinong, et al. "Linformer: Self-attention with linear complexity." *arXiv preprint arXiv:2006.04768* (2020).

26- Rae, Jack W., et al. "Compressive transformers for long-range sequence modelling." *arXiv preprint arXiv:1911.05507* (2019).

27- Kalyan, Katikapalli Subramanyam, Ajit Rajasekharan, and Sivanesan Sangeetha. "Ammus: A survey of transformer-based pretrained models in natural language processing." *arXiv preprint arXiv:2108.05542* (2021).

28- Erhan, Dumitru, et al. "Why does unsupervised pre-training help deep learning?." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010.

29- Yang, Zhilin, et al. "Xlnet: Generalized autoregressive pretraining for language understanding." *Advances in neural information processing systems* 32 (2019).

30- Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692* (2019).

31- Lee, Katherine, et al. "Deduplicating training data makes language models better." *arXiv preprint arXiv:2107.06499* (2021).

32- Caselli, Tommaso, et al. "Hatebert: Retraining bert for abusive language detection in english." *arXiv preprint arXiv:2010.12472* (2020).

33- Ni, Jianmo, Jiacheng Li, and Julian McAuley. "Justifying recommendations using distantly-labeled reviews and fine-grained aspects." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019.

34- Zhou, Jie, et al. "Sentix: A sentiment-aware pre-trained model for cross-domain sentiment analysis." *Proceedings of the 28th International Conference on Computational Linguistics*. 2020.

35- Wilie, Bryan, et al. "IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding." *arXiv preprint arXiv:2009.05387* (2020).

36- Wagner Filho, Jorge A., et al. "The brwac corpus: A new open resource for brazilian portuguese." *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*. 2018.

37- Xu, Liang, Xuanwei Zhang, and Qianqian Dong. "CLUECorpus2020: A large-scale Chinese corpus for pre-training language model." *arXiv preprint arXiv:2003.01355* (2020).

38- Yuan, Sha, et al. "WuDaoCorpora: A Super Large-scale Chinese Corpora for Pre-training Language Models." *AI Open* (2021).

39- Suárez, Pedro Javier Ortiz, Benoît Sagot, and Laurent Romary. "Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures." *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache, 2019.

40- Khanuja, Simran, et al. "Muril: Multilingual representations for indian languages." *arXiv preprint arXiv:2103.10730* (2021).

41- Xue, Linting, et al. "mt5: A massively multilingual pre-trained text-to-text transformer." *arXiv preprint arXiv:2010.11934* (2020).

42- Conneau, Alexis, et al. "Unsupervised cross-lingual representation learning at scale." *arXiv preprint arXiv:1911.02116* (2019).

43- Chi, Zewen, et al. "Infoxlm: An information-theoretic framework for cross-lingual language model pre-training." *arXiv preprint arXiv:2007.07834* (2020).

44- Chi, Zewen, et al. "Xlm-e: Cross-lingual language model pre-training via electra." *arXiv preprint arXiv:2106.16138* (2021).

45- Chen, Wei-Rui, Muhammad Abdul-Mageed, and Hasan Cavusoglu. "IndT5: A Text-to-Text Transformer for 10 Indigenous Languages." Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas. 2021.

46- Williams, Philip; Sennrich, Rico; Post, Matt; Koehn, Philipp (2016). Syntax-based Statistical Machine Translation. Morgan & Claypool. ISBN 978-1-62705-502-4.

47- Kunchukuttan, Anoop, Pratik Mehta, and Pushpak Bhattacharyya. "The iit bombay english-hindi parallel corpus." arXiv preprint arXiv:1710.02855 (2017).

48- Chi, Zewen, et al. "mT6: Multilingual Pretrained Text-to-Text Transformer with Translation Pairs." *arXiv preprint arXiv:2104.08692* (2021).

49- Lample, Guillaume, and Alexis Conneau. "Cross-lingual language model pretraining." *arXiv preprint arXiv:1901.07291* (2019).

50- Yang, Jian, et al. "Alternating language modeling for cross-lingual pre-training." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 05. 2020.

51- Huang, Haoyang, et al. "Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks." *arXiv preprint arXiv:1909.00964* (2019).

52- Ziemski, Michał, Marcin Junczys-Dowmunt, and Bruno Pouliquen. "The united nations parallel corpus v1. 0." *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016.

53- Chi, Zewen, et al. "Cross-lingual natural language generation via pre-training." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 05. 2020.

54- El-Kishky, Ahmed, et al. "CCAligned: A massive collection of cross-lingual web-document pairs." *arXiv preprint arXiv:1911.06154* (2019).

55- Lee, Jinhyuk, et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining." *Bioinformatics* 36.4 (2020): 1234-1240.

56- Johnson, Alistair EW, et al. "MIMIC-III, a freely accessible critical care database." *Scientific data* 3.1 (2016): 1-9.

57- Peng, Yifan, Shankai Yan, and Zhiyong Lu. "Transfer learning in biomedical natural language processing: an evaluation of BERT and ELMo on ten benchmarking datasets." *arXiv preprint arXiv:1906.05474* (2019).

58- Alsentzer, Emily, et al. "Publicly available clinical BERT embeddings." *arXiv preprint arXiv:1904.03323* (2019).

59- Zellers, Rowan, et al. "Defending against neural fake news." *arXiv preprint arXiv:1905.12616* (2019).

60- Gururangan, Suchin, et al. "Don't stop pretraining: adapt language models to domains and tasks." *arXiv preprint arXiv:2004.10964* (2020).

61- Lo, Kyle, et al. "S2ORC: The semantic scholar open research corpus." *arXiv preprint arXiv:1911.02782* (2019).

62- Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." *arXiv preprint arXiv:1609.08144* (2016).

63- Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." *arXiv preprint arXiv:1508.07909* (2015).

64- Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

65- Kudo, Taku, and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." *arXiv preprint arXiv:1808.06226* (2018).

66- https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture Accessed November 29, 2021.

67- Ganesh, Prakhar, et al. "Compressing large-scale transformer-based models: A case study on bert." *Transactions of the Association for Computational Linguistics* 9 (2021): 1061-1080.

68- Boukkouri, Hicham El, et al. "CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters." *arXiv preprint arXiv:2010.10392* (2020).

69- Chen, Yen-Pin, et al. "Modified bidirectional encoder representations from transformers extractive summarization model for hospital information systems based on character-level tokens (AlphaBERT): development and performance evaluation." *JMIR medical informatics* 8.4 (2020): e17787.

70- Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." *arXiv preprint arXiv:1910.13461* (2019).

71- Ma, Wentao, et al. "CharBERT: Character-aware Pre-trained Language Model." *arXiv preprint arXiv:2011.01513* (2020).

72- Meng, Yiwen, et al. "Bidirectional Representation Learning from Transformers using Multimodal Electronic Health Record Data to Predict Depression." *IEEE Journal of Biomedical and Health Informatics* (2021).

73- Rasmy, Laila, et al. "Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction." *NPJ digital medicine* 4.1 (2021): 1-13.

74- Li, Yikuan, et al. "BEHRT: transformer for electronic health records." *Scientific reports* 10.1 (2020): 1-12.

75- Clark, Kevin, et al. "Electra: Pre-training text encoders as discriminators rather than generators." *arXiv preprint arXiv:2003.10555* (2020).

76- Liu, Xiao, et al. "OAG-BERT: Pre-train Heterogeneous Entity-augmented Academic Language Models." *arXiv preprint arXiv:2103.02410* (2021).

77- Michalopoulos, George, et al. "Umlsbert: Clinical domain knowledge augmentation of contextual embeddings using the unified medical language system metathesaurus." *arXiv preprint arXiv:2010.10391* (2020).

78- Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

79- Dong, Li, et al. "Unified language model pre-training for natural language understanding and generation." *arXiv preprint arXiv:1905.03197* (2019).

80- Panda, Subhadarshi, et al. "Shuffled-token Detection for Refining Pre-trained RoBERTa." *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. 2021.

81- Di Liello, Luca, Matteo Gabburo, and Alessandro Moschitti. "Efficient pre-training objectives for Transformers." *arXiv preprint arXiv:2104.09694* (2021).

82- Joshi, Mandar, et al. "Spanbert: Improving pre-training by representing and predicting spans." *Transactions of the Association for Computational Linguistics* 8 (2020): 64-77.

83- Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." *arXiv preprint arXiv:1909.11942* (2019).

84- Song, Kaitao, et al. "Mass: Masked sequence to sequence pre-training for language generation." *arXiv preprint arXiv:1905.02450* (2019).

85- Beltagy, Iz, Kyle Lo, and Arman Cohan. "Scibert: A pretrained language model for scientific text." *arXiv preprint arXiv:1903.10676* (2019).

86- Chalkidis, Ilias, et al. "LEGAL-BERT: The muppets straight out of law school." *arXiv preprint arXiv:2010.02559* (2020).

87- Feng, Zhangyin, et al. "Codebert: A pre-trained model for programming and natural languages." *arXiv preprint arXiv:2002.08155* (2020).

88- Kuratov, Yuri, and Mikhail Arkhipov. "Adaptation of deep bidirectional multilingual transformers for russian language." *arXiv preprint arXiv:1905.07213* (2019).

89- Souza, Fábio, Rodrigo Nogueira, and Roberto Lotufo. "BERTimbau: pretrained BERT models for Brazilian Portuguese." *Brazilian Conference on Intelligent Systems*. Springer, Cham, 2020.

90- Arkhipov, Mikhail, et al. "Tuning multilingual transformers for language-specific named entity recognition." *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*. 2019.

91- Carmo, Diedre, et al. "PTT5: Pretraining and validating the T5 model on Brazilian Portuguese data." *arXiv preprint arXiv:2008.09144* (2020).

92- Yao, Yunzhi, et al. "Adapt-and-Distill: Developing Small, Fast and Effective Pretrained Language Models for Domains." *arXiv preprint arXiv:2106.13474* (2021).

93- Wada, Shoya, et al. "Pre-training technique to localize medical BERT and enhance biomedical BERT." *arXiv preprint arXiv:2005.07202* (2020).

94- Qin, Yujia, et al. "Knowledge Inheritance for Pre-trained Language Models." *arXiv preprint arXiv:2105.13880* (2021).

95- Zhang, Zhengyan, et al. "CPM-2: Large-scale Cost-effective Pre-trained Language Models." *arXiv preprint arXiv:2106.10715* (2021).

96- You, Yang, et al. "Large batch optimization for deep learning: Training bert in 76 minutes." *arXiv preprint arXiv:1904.00962* (2019).

97- Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

98- Brown, Tom B., et al. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020).

99- Liu, Yinhan, et al. "Multilingual denoising pre-training for neural machine translation." *Transactions of the Association for Computational Linguistics* 8 (2020): 726-742.

100- Ahmad, Wasi Uddin, et al. "Unified Pre-training for Program Understanding and Generation." *arXiv preprint arXiv:2103.06333* (2021).

101- Bi, Bin, et al. "Palm: Pre-training an autoencoding&autoregressive language model for context-conditioned generation." *arXiv preprint arXiv:2004.07159* (2020).

102- Fang, Hongchao, et al. "Cert: Contrastive self-supervised learning for language understanding." *arXiv preprint arXiv:2005.12766* (2020).

103- Liu, Fangyu, et al. "Fast, Effective and Self-Supervised: Transforming Masked LanguageModels into Universal Lexical and Sentence Encoders." *arXiv preprint arXiv:2104.08027* (2021).

104- Gao, Tianyu, Xingcheng Yao, and Danqi Chen. "SimCSE: Simple Contrastive Learning of Sentence Embeddings." *arXiv preprint arXiv:2104.08821* (2021).

105- Wang, Dong, et al. "Cline: Contrastive learning with semantic negative examples for natural language understanding." *arXiv preprint arXiv:2107.00440* (2021).

106- Kaplan, Jared, et al. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

107- Gupta, Manish, and Puneet Agrawal. "Compression of Deep Learning Models for Text: A Survey." *arXiv preprint arXiv:2008.05221* (2020).

108- Michel, Paul, Omer Levy, and Graham Neubig. "Are sixteen heads really better than one?." *arXiv preprint arXiv:1905.10650* (2019).

109- Voita, Elena, et al. "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned." *arXiv preprint arXiv:1905.09418* (2019).

110- Fan, Angela, Edouard Grave, and Armand Joulin. "Reducing transformer depth on demand with structured dropout." *arXiv preprint arXiv:1909.11556* (2019).

111- Zafrir, Ofir, et al. "Q8bert: Quantized 8bit bert." *arXiv preprint arXiv:1910.06188* (2019).

112- Shen, Sheng, et al. "Q-bert: Hessian based ultra low precision quantization of bert." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 05. 2020.

113- Zhang, Wei, et al. "Ternarybert: Distillation-aware ultra-low bit bert." *arXiv preprint arXiv:2009.12812* (2020).

114- Zadeh, Ali Hadi, et al. "Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference." *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020.

115- Fan, Angela, et al. "Training with quantization noise for extreme model compression." *arXiv preprint arXiv:2004.07320* (2020).

116- Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006.

117- Ba, Lei Jimmy, and Rich Caruana. "Do deep nets really need to be deep?." arXiv preprint arXiv:1312.6184 (2013).

118- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

119- Sanh, Victor, et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *arXiv preprint arXiv:1910.01108* (2019).

120- Jiao, Xiaoqi, et al. "Tinybert: Distilling bert for natural language understanding." *arXiv preprint arXiv:1909.10351* (2019).

121- Sun, Siqi, et al. "Patient knowledge distillation for bert model compression." *arXiv preprint arXiv:1908.09355* (2019).

122- Sun, Zhiqing, et al. "Mobilebert: a compact task-agnostic bert for resource-limited devices." *arXiv preprint arXiv:2004.02984* (2020).

123- Wang, Wenhui, et al. "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers." *arXiv preprint arXiv:2002.10957* (2020).

124- Poerner, Nina, Ulli Waltinger, and Hinrich Schütze. "Inexpensive domain adaptation of pretrained language models: case studies on biomedical NER and COVID-19 QA." *arXiv preprint arXiv:2004.03354* (2020).

125- Tai, Wen, et al. "exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 2020.

126- Poerner, Nina, Ulli Waltinger, and Hinrich Schütze. "E-BERT: Efficient-yet-effective entity embeddings for BERT." *arXiv preprint arXiv:1911.03681* (2019).

127- Yamada, Ikuya, et al. "Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia." *arXiv preprint arXiv:1812.06280* (2018).

128- Zhang, Zhengyan, et al. "ERNIE: Enhanced language representation with informative entities." *arXiv preprint arXiv:1905.07129* (2019).

129- Peters, Matthew E., et al. "Knowledge enhanced contextual word representations." *arXiv preprint arXiv:1909.04164* (2019).

130- Petroni, Fabio, et al. "Language models as knowledge bases?." *arXiv preprint arXiv:1909.01066* (2019).

131- Gu, Yu, et al. "Domain-specific language model pretraining for biomedical natural language processing." *ACM Transactions on Computing for Healthcare (HEALTH)* 3.1 (2021): 1-23.

132- Clark, Jonathan H., et al. "Canine: Pre-training an efficient tokenization-free encoder for language representation." *arXiv preprint arXiv:2103.06874* (2021).

133- Xue, Linting, et al. "ByT5: Towards a token-free future with pre-trained byte-to-byte models." *arXiv preprint arXiv:2105.13626* (2021).

134- Tay, Yi, et al. "Charformer: Fast character transformers via gradient-based subword tokenization." *arXiv preprint arXiv:2106.12672* (2021).

135- Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint arXiv:1908.10084* (2019).

136- Bowman, Samuel R., et al. "A large annotated corpus for learning natural language inference." *arXiv preprint arXiv:1508.05326* (2015).

137- Williams, Adina, Nikita Nangia, and Samuel R. Bowman. "A broad-coverage challenge corpus for sentence understanding through inference." *arXiv preprint arXiv:1704.05426* (2017).

138- Cer, Daniel, et al. "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation." *arXiv preprint arXiv:1708.00055* (2017).

139- Cheng, Xingyi. "Dual-View Distilled BERT for Sentence Embedding." *arXiv preprint arXiv:2104.08675* (2021).

140- Zhang, Yan, et al. "An unsupervised sentence embedding method by mutual information maximization." *arXiv preprint arXiv:2009.12061* (2020).

141- Wang, Kexin, Nils Reimers, and Iryna Gurevych. "TSDAE: Using Transformer-based Sequential Denoising Auto-Encoder for Unsupervised Sentence Embedding Learning." *arXiv preprint arXiv:2104.06979* (2021).

142- He, Pengcheng, et al. "Deberta: Decoding-enhanced bert with disentangled attention." *arXiv preprint arXiv:2006.03654* (2020).

143- Jiang, Zihang, et al. "Convbert: Improving bert with span-based dynamic convolution." *arXiv preprint arXiv:2008.02496* (2020).

144- Ainslie, Joshua, et al. "ETC: Encoding long and structured inputs in transformers." *arXiv preprint arXiv:2004.08483* (2020).

145- Zaheer, Manzil, et al. "Big Bird: Transformers for Longer Sequences." *NeurIPS*. 2020.

146- Choromanski, Krzysztof, et al. "Rethinking attention with performers." *arXiv preprint arXiv:2009.14794* (2020).

147- Fellbaum, Christiane. "WordNet." *Theory and applications of ontology: computer applications*. Springer, Dordrecht, 2010. 231-243.

148- Bodenreider, Olivier. "The unified medical language system (UMLS): integrating biomedical terminology." *Nucleic acids research* 32.suppl_1 (2004): D267-D270.

149- Li, Zhongyang, et al. "Causalbert: Injecting causal knowledge into pre-trained models with minimal supervision." *arXiv preprint arXiv:2107.09852* (2021).

150- Levine, Yoav, et al. "Sensebert: Driving some sense into bert." *arXiv preprint arXiv:1908.05646* (2019).

151- Zhou, Junru, et al. "LIMIT-BERT: Linguistic informed multi-task bert." *arXiv preprint arXiv:1910.14296* (2019).

152- Lauscher, Anne, et al. "Specializing unsupervised pretraining models for word-level semantic similarity." *arXiv preprint arXiv:1909.02339* (2019).

153- Ke, Pei, et al. "SentiLARE: Sentiment-aware language representation learning with linguistic knowledge." *arXiv preprint arXiv:1911.02493* (2019).

154- Hao, Boran, Henghui Zhu, and Ioannis Paschalidis. "Enhancing clinical bert embedding using a biomedical knowledge base." *Proceedings of the 28th international conference on computational linguistics*. 2020.

155- Liu, Fangyu, et al. "Self-alignment pretraining for biomedical entity representations." *arXiv preprint arXiv:2010.11784* (2020).

156- Yuan, Zheng, et al. "CODER: Knowledge infused cross-lingual medical term embedding for term normalization." *arXiv preprint arXiv:2011.02947* (2020).

157- Zeng, Wei, et al. "PanGu-$\alpha $: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation." *arXiv preprint arXiv:2104.12369* (2021).

158- Lepikhin, Dmitry, et al. "Gshard: Scaling giant models with conditional computation and automatic sharding." *arXiv preprint arXiv:2006.16668* (2020).

159- Fedus, William, Barret Zoph, and Noam Shazeer. "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity." *arXiv preprint arXiv:2101.03961* (2021).

160- https://colab.research.google.com/ Accessed November 29, 2021.

161- Qiu, Xipeng, et al. "Pre-trained models for natural language processing: A survey." *Science China Technological Sciences* (2020): 1-26.

162- Gehrmann, Sebastian, Yuntian Deng, and Alexander M. Rush. "Bottom-up abstractive summarization." *arXiv preprint arXiv:1808.10792* (2018).

163- Li, Wei, et al. "Improving neural abstractive document summarization with explicit information selection modeling." *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2018.

164- Syed, Ayesha Ayub, Ford Lumban Gaol, and Tokuro Matsuo. "A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization." *IEEE Access* 9 (2021): 13248-13265.

165- Hou, Sheng-Luan, et al. "A Survey of Text Summarization Approaches Based on Deep Learning." *Journal of Computer Science and Technology* 36.3 (2021): 633-663.

166- Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." *arXiv preprint arXiv:1509.00685* (2015).

167- Chopra, Sumit, Michael Auli, and Alexander M. Rush. "Abstractive sentence summarization with attentive recurrent neural networks." *Proceedings of the 2016*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.

168- Takase, Sho, et al. "Neural headline generation on abstract meaning representation." *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016.

169- Lopyrev, Konstantin. "Generating news headlines with recurrent neural networks." *arXiv preprint arXiv:1512.01712* (2015).

170- Chen, Qian, et al. "Distraction-Based Neural Networks for Modeling Document." *IJCAI*. Vol. 16. 2016.

171- Inan, Hakan, Khashayar Khosravi, and Richard Socher. "Tying word vectors and word classifiers: A loss framework for language modeling." *arXiv preprint arXiv:1611.01462* (2016).

172- Paulus, Romain, Caiming Xiong, and Richard Socher. "A deep reinforced model for abstractive summarization." *arXiv preprint arXiv:1705.04304* (2017).

173- Tan, Jiwei, Xiaojun Wan, and Jianguo Xiao. "Abstractive document summarization with a graph-based attentional neural model." *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.

174- Hsu, Wan-Ting, et al. "A unified model for extractive and abstractive summarization using inconsistency loss." *arXiv preprint arXiv:1805.06266* (2018).

175- Celikyilmaz, Asli, et al. "Deep communicating agents for abstractive summarization." *arXiv preprint arXiv:1803.10357* (2018).

176- See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." *arXiv preprint arXiv:1704.04368* (2017).

177- Fan, Angela, David Grangier, and Michael Auli. "Controllable abstractive summarization." *arXiv preprint arXiv:1711.05217* (2017).

178- Cai, Tian, et al. "Improving transformer with sequential context representations for abstractive text summarization." *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, Cham, 2019.

179- Gulcehre, Caglar, et al. "Pointing the unknown words." *arXiv preprint arXiv:1603.08148* (2016).

180- Gu, Jiatao, et al. "Incorporating copying mechanism in sequence-to-sequence learning." *arXiv preprint arXiv:1603.06393* (2016).

181- Zeng, Wenyuan, et al. "Efficient summarization with read-again and copy mechanism." *arXiv preprint arXiv:1611.03382* (2016).

182- Chen, Yen-Chun, and Mohit Bansal. "Fast abstractive summarization with reinforce-selected sentence rewriting." *arXiv preprint arXiv:1805.11080* (2018).

183- Nallapati, Ramesh, et al. "Abstractive text summarization using sequence-to-sequence rnns and beyond." *arXiv preprint arXiv:1602.06023* (2016).

184- Jiang, Xiaoping, et al. "Improving pointer-generator network with keywords information for chinese abstractive summarization." *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, Cham, 2018.

185- Yenter, Alec, and Abhishek Verma. "Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis." 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON). IEEE, 2017.

186- Kumar, H. M., B. S. Harish, and H. K. Darshan. "Sentiment Analysis on IMDb Movie Reviews Using Hybrid Feature Extraction Method." International Journal of Interactive Multimedia & Artificial Intelligence 5.5 (2019).

187- Topal, Kamil, and Gultekin Ozsoyoglu. "Movie review analysis: Emotion analysis of IMDb movie reviews." 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2016.

188- Oghina, Andrei, et al. "Predicting imdb movie ratings using social media." European conference on information retrieval. Springer, Berlin, Heidelberg, 2012.

189- Otterbacher, Jahna. "Gender, writing and ranking in review forums: a case study of the IMDb." Knowledge and information systems 35.3 (2013): 645-664.

190- Hsu, Ping-Yu, Yuan-Hong Shen, and Xiang-An Xie. "Predicting movies user ratings with imdb attributes." *International Conference on Rough Sets and Knowledge Technology*. Springer, Cham, 2014.

191- Bristi, Warda Ruheen, Zakia Zaman, and Nishat Sultana. "Predicting imdb rating of movies by machine learning techniques." 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2019.

192- Jung, Jason J. "Attribute selection-based recommendation framework for short-head user group: An empirical study by MovieLens and IMDB." Expert Systems with Applications 39.4 (2012): 4049-4054.

193- Hu, Yajie, et al. "Recommendation for movies and stars using YAGO and IMDB." 2010 12th International Asia-Pacific Web Conference. IEEE, 2010.

194- https://github.com/google-research/text-to-text-transfer-transformer Accessed November 29, 2021.

195- https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html Accessed November 29, 2021.

196- https://huggingface.co/transformers/model_doc/t5.html Accessed November 29, 2021.

197- Cormen, T. H. "CE leiserson, and RL Rivest." *Introduction to Algorithms* (1990): 550.

198- https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/ Accessed November 29, 2021.

199- https://github.com/google-research/google-research/tree/master/rouge Accessed November 29, 2021.