Aristotle University of Thessaloniki

Faculty of Sciences

School of Informatics

Multi-Document Summarization of Comment Pools with Transformers

Περίληψη Πολλαπλών Σχολίων με την Χρήση Μετασχηματιστών

Arelakis Aggelos

ID: 2291

Supervisor: Grigorios Tsoumakas

August 30, 2022

Thanks to

I would like to express my gratitude towards my supervisor, Professor Grigorios Tsoumakas, during the development of this thesis for providing me with critical feedback and guidance which led to the completion of the thesis.

I would also like to thank my family for providing both emotional and financial support during this time.

Abstract

The amount of information such as articles and blogs available in the Internet today is mind boggling. Daily a huge number of comments is made in social media platforms. This large collection of data and information requires years or even hundreds of years of human labor in order to be categorized and analyzed. Effectively summarizing a large piece of a document like a blog, an article or even a collection of comment pools offers the benefit of being able to acquire the knowledge containing within it without having to spend massive amounts of time to properly read the whole document. Of course the next thing that comes to mind is being able to automatically summarize massive amounts of text in order for this process to be profitable. This is the main focus of this thesis, where various Transformer models are presented, explained and analyzed. Different nuances of the said models are mentioned and how these models can aid the summarization process of multiple documents is discussed. Finally experiments are carried out with real life twitter comment polls which aim to summarize these lengthy posts. Various models are trained and tested until the best model is selected. These experiments which are presented and analyzed show promising results and are able to generalize to the unseen data remarkably well.

Περίληψη

Ο όγκος των πληροφοριών όπως τα άρθρα και τα ιστολόγια που διατίθενται σήμερα στο Διαδίκτυο είναι συγκλονιστικός. Καθημερινά ένας τεράστιος αριθμός σχολίων γίνεται στις πλατφόρμες κοινωνικής δικτύωσης. Αυτή η μεγάλη συλλογή δεδομένων και πληροφοριών απαιτεί χρόνια ή και εκατοντάδες χρόνια ανθρώπινης εργασίας για να κατηγοριοποιηθεί και να αναλυθεί. Η ουσιαστική σύνοψη ενός μεγάλου τμήματος ενός εγγράφου όπως ένα ιστολόγιο, ένα άρθρο ή ακόμα και μια συλλογή από ομάδες σχολίων προσφέρει το πλεονέκτημα της δυνατότητας απόκτησης της γνώσης που περιέχει χωρίς να χρειάζεται κάποιος να ξοδέψει τεράστιο χρόνο για να διαβάσει ολόκληρο το έγγραφο. Φυσικά, το επόμενο πράγμα που έρχεται στο μυαλό είναι η δυνατότητα αυτόματης σύνοψης τεράστιων ποσοτήτων κειμένου προκειμένου αυτή η διαδικασία να είναι κερδοφόρα. Αυτός είναι ο κύριος στόχος της παρούσας διπλωματικής εργασίας, όπου παρουσιάζονται, επεξηγούνται και αναλύονται διάφορα μοντέλα μετασχηματιστών. Αναφέρονται διάφορες αποχρώσεις των εν λόγω μοντέλων και συζητείται πώς αυτά τα μοντέλα μπορούν να βοηθήσουν τη διαδικασία σύνοψης πολλαπλών εγγράφων. Τέλος, πραγματοποιούνται πειράματα με σχόλια από το twitter που στοχεύουν να συνοψίσουν αυτές τις μακροσκελείς δημοσιεύσεις. Διάφορα μοντέλα εκπαιδεύονται και δοκιμάζονται μέχρι να επιλεγεί το καλύτερο μοντέλο. Αυτά τα πειράματα που παρουσιάζονται και αναλύονται δείχνουν πολλά υποσχόμενα αποτελέσματα και είναι σε θέση να γενικεύουν εξαιρετικά καλά στα δεδομένα που δεν έχουν δει ξανά.

Contents

1 Introduction **10**

2 Background **12**

2.1 Text Summarization 12

2.1.1 Opinion and Argument Mining 12

2.2 Datasets 14

2.3 Sequence-to-Sequence (Seq2Seq) Modeling and Tasks 17

2.4 Tokenization methods 17

2.5 Word Embeddings 19

2.6 Denoising Autoencoders 21

2.7 Transformer-Based Architectures 22

2.8 Loss functions 23

2.9 Transformer 24

2.9.1 BERT 28

2.9.2 RoBERTa 32

2.9.3 Knowledge distillation & DistillBERT 33

2.9.4 BART 35

2.9.5 T5 38

2.9.6 PEGASUS 41

2.9.7 Longformer 44

2.10 Decoding Strategies 47

2.11 Evaluation 49

2.11.1 Precision & Recall 50

2.11.2 ROUGE 50

3 Experiments **53**

3.1 Dataset 54

3.2 DistillBART 55

3.3 DistillPEGASUS 58

3.4 $\boldsymbol{T5_{Base}}$ 60
3.5 Results 61
3.6 Selected Models 64
3.7 Testing (zero-shot) dataset 65

4 Conclusion and final thoughts **66**

List of Figures

2.1 The Seq2Seq Model with an Encoder - Decoder architecture. 17
2.2 CBOW & Skip-gram architectures. 20
2.3 Illustration of the denoising auto-encoder model architecture (Robinet, Nov 26, 2020). 22
2.4 The Transformer - model architecture. 24
2.5 (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. 26
2.6 The positional encoding matrix for $n = 10000$, $d = 512$, sequence length=100. 27
2.7 BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. (Devlin et al., 2018) 29
2.8 Pre-training BERT. 30
2.9 Using BERT for different tasks. 31
2.10 BERTSUM encoder. 32
2.11 The generic teacher-student framework for knowledge distillation (Gou et al., 2021). 33
2.12 ReLU and GELU. 36
2.13 36
2.14 A schematic comparison of BART with BERT and GPT. (source (Lewis et al., 2019)) 37
2.15 Transformations for corrupting the input sequence as seen in the BART model. 38
2.16 T5 model size variants. 40
2.17 T5 framework. 40
2.18 Schematic of the unsupervised objective of T5 (Raffel et al., 2019) using masked language modeling. 41

2.19 The base architecture of PEGASUS. Both GSG and MLM are applied simultaneously. Originally there are three sentences. One sentence is masked with **[MASK1]** and used as target generation text (GSG). The other two sentences remain in the input, but some tokens are randomly masked by **[MASK2]** (MLM) (Zhang et al., 2019). 42
2.20 Effect of pre-training corpus. $\boldsymbol{PEGASUS_{BASE}}$ pre-trained on C4 (350M Web-pages) and HugeNews (1.5B news-like documents) (Zhang et al., 2019). 42
2.21 Fine-tuning with limited supervised examples. The solid lines are $\boldsymbol{PEGASUS_{LARGE}}$ fine-tuned on 0 (zero shot), 10, 100, 1k,10k examples. The dashed lines are $\boldsymbol{PEGASUS_{BASE}}$ models without pre-training and trained using the full supervised datasets (Zhang et al., 2019). 44
2.22 Sliding window attention in the Longformer. 45
2.23 Configuration of the various attention patterns as introduced by the Longformer (Beltagy et al., 2020). 46
2.24 Greedy search. The algorithm greedily chooses the next word of highest probability "nice" and so on, so that the final generated word sequence is ("The", "nice", "woman") having an overall probability of $\mathbf{0.5 \times 0.4 = 0.20}$ 48
2.25 Example of performing the beam search algorithm in a sequence with number of beams $\boldsymbol{k = 2}$. 49
2.26 precision-recall-accuracy 50
2.27 example LCS 52

3.1 ROUGE scores during validation of the E-mails subset. 55
3.2 ROUGE scores during validation of the Nyt subset. 56
3.3 ROUGE scores during validation of the Reddit subset. 56
3.4 ROUGE scores during validation of the Stack Exchange subset. 57
3.5 ROUGE scores during validation on the whole ConvoSumm dataset with 512 tokens. 57
3.6 ROUGE scores during validation on the whole ConvoSumm dataset. 58
3.7 ROUGE scores during validation of the E-mails subset. 59
3.8 ROUGE scores during validation of the Nyt subset. 59
3.9 ROUGE scores during validation of the Reddit subset. 59
3.10 ROUGE scores during validation of the Stack Exchange subset. 60
3.11 ROUGE scores during validation of the E-mails subset. 60
3.12 ROUGE scores during validation of the Nyt subset. 61

3.13 ROUGE scores during validation of the Reddit subset. 61
3.14 Loss and validation loss while fine-tuning DistillBART. 61
3.15 Loss and validation loss while fine-tuning DistillPEGASUS. 62
3.16 Loss and validation loss while fine-tuning $\boldsymbol{T5_{Base}}$. 62

List of Tables

3.1 Results of all models. 63
3.2 Twitter post summarized 65

4.1 E-mails subset with $DistillBARTfull$ - best. 78
4.2 E-mails subset with $DistillBARTfull$ - best. 79
4.3 Reddit subset with $DistillBARTfull$ - worst. 80
4.4 Nyt subset with $DistillPegasus\,solo$ - worst. 81
4.5 Twitter posts summaries generated by the best model. 82

Chapter 1

Introduction

As the amount of internet users increases, on-line information produced increases as well. Imagine a blog or a site with many users like a social platform, creating multiple posts and articles each day. There is an imminent need to categorize and summarize this information with an automated system as human resources are not enough to do so. In this regard various document summarization techniques are introduced.

A summary is a text produced from one or more documents that conveys important information from the original texts and is significantly smaller in size. Extraction is the process of identifying important material in the text, abstraction the process of reformulating it in novel terms, fusion the process of combining extracted portions, and compression the process of squeezing out unimportant material (Radev et al., 2002). Our main focus on this thesis will be abstractive summarization in multi–document format, like posts in twitter.

Abstractive summarization is the technique of creating a summary using not the most salient sentences of a document but understanding the context and the meaning of them, paraphrasing and compressing it to a smaller size. Multi–Document Abstractive summarization (MDS) is a similar process where the summary is to be created from multiple documents.

Summarization is an important challenge in NLP (natural language processing). Despite all the progress in the last few years with the advent of the Transformer architecture (Vaswani et al., 2017), there are still many problems that need to be resolved for an one for all model capable of solving all similar tasks of summarization. The models based on the aforementioned architecture are pre-trained on huge datasets and are later fine-tuned for downstream tasks on specific datasets in order to capture the linguistic attributes, traits and specificities contained in the latter. Furthermore, transfer learning (Zhuang et al., 2019) allows for model re-usage in different subsequent tasks, fine- tuning is one form of transfer learning. This idea allows to train a model from a certain starting point, without the need to train all models from scratch. Not only does this process save resources, it also allows for training models on much smaller datasets, since general linguistic interactions are modeled at the initial training step. Fine-tuning is the next optional step in this process, which allows for, given the same objective, fitting new data into the existing pre-trained model. This ensures better domain modeling, e.g., tuning a model initially developed for working with Wiki data for working with Twitter data instead, which is different in its nature (Blekanov et al., 2022).

Research on summarization datasets created by gathering text from social media is minimal. Still the results of using transfer learning and pre-trained Transformer based models looks very promising since these models produced state-of-the-art results on various NLP tasks including summarization on other datasets. The assumption we make is that if we fine-tune these models on a dataset such as a Twitter comment-pool we should expect similar results. A potential problem is the lack of annotated datasets, but some Transformer based models like (PEGASUS Large) are known to be able to reach state-of-art-results when fine-tuned with as little as 100 examples.

Chapter 2

Background

In this chapter definitions about text summarization will be presented as well as basic concepts relevant such as opinion and argument mining. Some perquisites necessary for fully understanding the inner-workings of the various transformers models will be discussed such as tokenization methods and word embeddings. Various transformers models will be introduced and explained along with their specificities. Lastly the loss functions and the decoding strategies utilized in these models will be analyzed and evaluation metrics will be presented and explained.

2.1 Text Summarization

Text summarizing is the process of extracting the most crucial information from one or more sources to create a condensed version for a certain audience or a specific purpose. There are essentially two distinct methods used in text summarization:

- Extractive Summarization: The goal of extractive summarization is to locate the key passages, which are then extracted and compiled to create a concise summary.
- Abstractive Summarization: A summary of a text can be produced by summarizing the main concepts in the text than by simply repeating its most crucial passages.

2.1.1 Opinion and Argument Mining

The goal of opinion mining (Liu and Zhang, 2012) is to analyze texts that are written in natural language about a given proposition in order to categorize them according to the opinion they are intended to communicate or, more precisely, according to the polarity (negative, positive, or neutral) or emotions of the writers of those texts. These documents are susceptible to three degrees of analysis: (Dragoni et al., 2018):

- **At a sentence level** — in this instance we are considering extracting the context of one single sentence.
- **At a document level** — in this instance we are considered by the overall subject of the entire document.
- **At an aspect level** — in this case, we're taking into account a more nuanced attitude or opinion connected to the document's most important elements or qualities.

One of the core components of human communication is argumentation, or the process of conveying preferences, attitudes, or ideas in an effort to persuade the other person to accept or even adopt them. Since they are employed to support, contest, or even defend claims, arguments are meant to make decision-making easier (Rahwan and Simari, 2009). Cognitive agents are in a similar predicament when they cooperate to tackle a complex problem as when the "division of labor" needs to be considered. Software agents argue in a more suitable formal language whilst humans do debate in everyday language. However, argumentation is becoming more and more crucial for automated processes and, by extension, cognitive computing as a result of recent developments in text mining applications and methodology (Peldszus and Stede, 2013).

Clearly argumentation mining (Peldszus and Stede, 2013; Lippi and Torroni, 2016) shares many similarities as well as differences with opinion mining and sentiment analysis, as highlighted also by (Habernal et al., 2014), "*even if the goals of the two differ: the goal of opinion mining is to understand what people think about something, while the aim of argumentation mining is to understand why, which implies looking for causes and reasons rather than just for opinions and sentiments*."

Opinion mining, sentiment categorization, and opinion summarization are all parts of the larger discipline known as sentiment analysis. The practice of automatically summarizing several perspectives on a single subject is known as opinion summarization (Moussa et al., 2018). Opinion summarization, which differs from typical summarizing (Felciah and Anbuselvi, 2015), needs several pre-processing stages such as tokenization and stemming. It is one of the most interesting and valuable NLP techniques (Lloret et al., 2015). Finding the entries with the most relevant thoughts on a particular topic on social media is important (Liao et al., 2014). Sentiment Summarization differs from factual data summarization in that statements that may have been considered instructive from a factual point of view may not at all include sentiment, rendering them meaningless from a sentimental standpoint (Balahur et al., 2012).

2.2 Datasets

In this section various Datasets used for training models for text summarization will be presented as well as key information regarding them. These datasets are valid candidate datasets to be selected later on for fine-tuning the various transformer models on the downstream task of multi-document summarization. Many of these datasets contain similar structure to what is needed for MDS and that is a dialogue-like structure, since we are treating each part of the dialogue as an individual document.

- **The ICSI Meeting Corpus** (Janin et al., 2003): This corpus is a collection of data from meetings that occured at the *International Computer Science Institute* from approximately 1999 to 2002. In ICSI, the meeting transcript has 464 turns, averaging 10,189 words, while the summary has 534 words. Word error rate ranks at 37% , which was caused by an ASR (Automatic Speech Recognition) system (Shang et al., 2018).
- **The AMI Meeting Corpus** (Mccowan et al., 2005): ***A**ugmented **M**ulti-party **I**nteraction* project. The corpus includes audio and texts of real meetings that were concurrently captured using tabletop and headworn microphones. It includes 53 different speakers and 75 meetings of four main categories. During these meetings, the meeting participants received detailed information about their roles and they were also asked to provide textual summaries after each meeting with a maximum limit of 200 words. The meeting scripts has an average of 4,757 words with 289 turns and 322 words in the summary. The ASR technology used to create the transcript has a word error rate of 36% (Shang et al., 2018).
- **SENSEI** (Barker et al., 2016): The SENSEI Project was a 3-year EU-FP7 funded project (2013-2016), which aimed to develop summarization/analytics technology to help users make sense of human conversation streams from diverse media channels. The researchers chose a limited sample for use in the summary corpus out of an initial collection of 3,362 Guardian news stories published in June–July 2014 and related comment sets. As a result, 1,845 comments totaling 87,559 words were sampled. The length of condensed summary comment sets ranges from 2,384 to 8,663 words. 3,879 comment labels are included in the corpus totaling 99.46 per annotation set (av. 99.46/AS). There are, in total, 329 group annotations (av. 8.44/AS) and 218 subgroups (av. 5.59/AS). Each of the 547 groups and subgroups has a brief group label that describes what it is

composed of. Such tags vary from general terms ("midges", "UK climate", "fining directors", "Air conditioning/fans") to whole statements such as ("Not fair that SE gets the investment", "Why use the fine on WiFi", "Not fair that SE gets the investment"). There are two summaries for each of the 39 sets of annotations, the restricted summaries are shorter (237.74 words on average) than the unconstrained ones, which have an average length of 321.41 words. Each summary is linked back to one or more groups that provided the information, with comment labels.

- **Webis-TLDR-17** (Völske et al., 2017): The Webis-TLDR-17 dataset is the result of a large Reddit crawl. The summaries included in this corpora are accurate summaries supplied by the post's writers, they are abstractions over a subject, and cover a considerably larger spectrum of subjects than those typically seen in news stories. The dataset consists of 3,848,330 posts (1.6 million submissions and 2.4 million comments) with an average length of 270 words for content, and 28 words for the summary, from the Reddit dataset (286 million submissions and 1.6 billion comments posted to Reddit between 2006 and 2016). The dataset consists of both Submissions and Comments that have been combined together. As a result, the majority of comments that are not associated with any post have the title null.

- **Reddit TIFU** (Kim et al., 2018): Contains 122,933 postings of casual narratives from the *TIFU* sub-reddit of the online discussion board Reddit between 2013-Jan and 2018-Mar. According to a manual review, the sub-reddit postings had a higher quality than Webis-TLDR-17 (Völske et al., 2017) (which utilised more subreddits) since they carefully adhere to the guideline of creating a "TL;DR" description (Zhang et al., 2019).

Dataset	**#posts**	**#words/post**	**#words/summ**
TIFU-short	79, 949	342.4 (269)	9.33 (8)
TIFU-long	42, 984	432.6 (351)	23.0 (21)

- **CQASUMM** (Chowdhury and Chakraborty, 2018) : The *Community Question Answering **Summ**arization*. In order to locate question threads in which the best response can function as a summary for the other replies, the Yahoo! Answers L6 corpora is filtered to create CQASUMM. The authors obtain a collection of 100,000 question thread-summary pairings using a number of methods. The best replies are modified, and the most relevant sentences are taken to create the summary. The remaining responses function as potential sources for summarization and together make up a sizable, varied, and extremely abstract dataset. The corpus has an average of 12.02 replies per question thread and 65.03 words in each answer. The best answer is used as a reference summary, and each response within a thread is treated as a distinct document. The corpus contains both opinion and fact based answers in approximately 65%–35% ratio.

- **SAMSum** (Gliwa et al., 2019): ***S***_amsung_ ***A***_bstractive_ ***M***_essenger_ ***Sum***_marization_ is a Human-annotated Dialogue Dataset for Abstractive Summarization. It contains 16,369 natural messenger-like dialogues created by linguists fluent in English. Linguists were instructed to generate dialogues that reflected the proportion of themes of their real-life messenger discussions and were parallel to those they write on a daily basis. Conversations can take on a variety of styles and forms, including casual, semi-formal and they may also use slang terms, emoticons, and typos. Then, summaries were added as annotations to the dialogues. The average length of dialogues is 119 tokens and that of the summaries is 23 tokens. 75% of the conversations are between two interviewers, the rest are between three or more people who took on average 11 turns (Zhao et al., 2021). The dialogues cover a wide range of diverse real-life subjects, and the summaries include speaker profiles.

- **TLDR9+** (Sotudeh et al., 2021): A large summarization dataset collected from the Reddit discussion forum with 9 million training examples[1]. The goal of assembling this dataset was to conduct extreme summarization (i.e., creating concise summaries with high abstraction and compression). It contains 9,227,437 instances, each post has 310 tokens and the TL;DR summary has 35.6 tokens.

- **TLDRHQ**: Human annotations enable the distillation of a more precise dataset, TLDRHQ (Sotudeh et al., 2021), by sampling **H**igh-**Q**uality instances from TLDR9+. It contains 1,671,099 instances, each post has 332 tokens and the TL;DR summary is 27 tokens long.

- **ConvoSumm** (Fabbri et al., 2021): A dataset composed of news article comments (NYT Comments dataset (Kolhatkar and Taboada, 2017)), discussion forums and debate (Reddit data from CoarseDiscourse (Zhang et al., 2017)), community question answering (StackExchange (Hoogeveen et al., 2015)) and email threads (W3C corpus (Craswell et al., 2005)). After pre-processing and cleaning of the data the authors crowdsourced workers to write abstractive summaries for each of the four datasets motivated by work in summarizing viewpoints contained in online discussions (Barker and Gaizauskas, 2016) namely "***issues*** *that individuals discuss,* ***viewpoints*** *that they hold about these issues*

[1] https://github.com/sajastu/reddit_collector

and ***assertions*** *that they make to support their viewpoints*". The dataset as a result consists of 2,021 dialog - summary pairs, the average dialog length is 1,096 tokens, the turns the speakers took are 9.8 and the summaries are 72.8 tokens long.

2.3 Sequence-to-Sequence (Seq2Seq) Modeling and Tasks

A Seq2Seq model is one that generates another series of items from an input sequence, this concept was initially proposed in 2014 (Sutskever et al., 2014; Cho et al., 2014). The objective of a sequence-to-sequence model is to map a fixed-length input to a fixed-length output, even when the lengths of the input and output may vary. Many NLP tasks are such problems, in particular text summarization can be defined as a problem where provided with an arrangement of words the model must build a summary which is also another sequence of words. Common such problems are translation, natural language correction, text generation, speech recognition, video captioning etc.

Typically a Seq2Seq model (Figure 2.1) consists of two main components, the encoder and the decoder. The encoder is responsible for processing each item in the input sequence and compiling the information it captures into a vector (context) which is then fed into the decoder to produce the output.

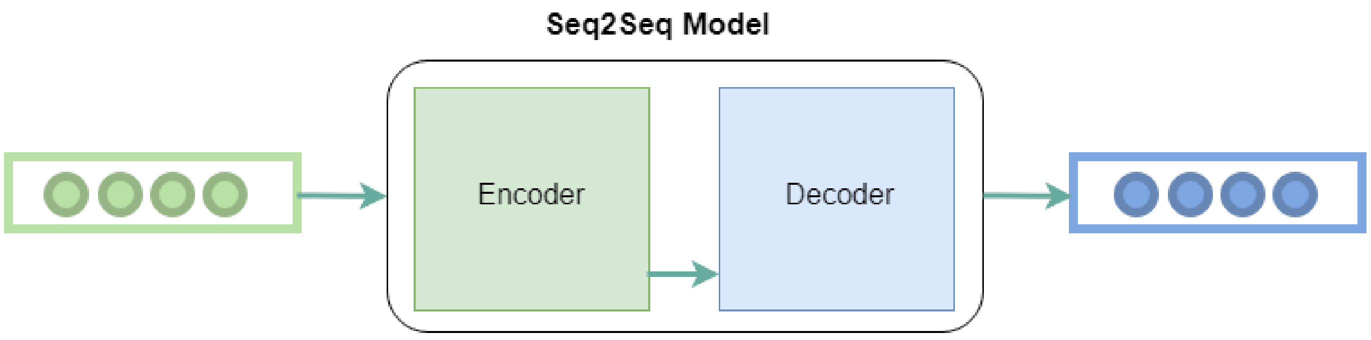


Figure 2.1: The Seq2Seq Model with an Encoder - Decoder architecture.

2.4 Tokenization methods

Before computer processing can start, the concept of a token must be specified. Tokenizing the input sequence (Webster and Kit, 1992) is the initial step in every NLP work. Tokenization is the division of a text or sequence into smaller pieces. A token is any of these discrete parts, which can be a word, a subword, or simply a letter.

- ***Word Tokenization (Word Embeddings)*** splits a piece of text into words based on a delimiter, like space and punctuation marks. Some problems that may arise from this approach are that the same word in plural will have a different ID, the creation of a huge vocabulary or even **OOV** (Out Of Vocabulary) words (Bengio et al., 2003; Mikolov et al., 2013).

- ***Character Tokenization*** splits the text into individual characters and will result in a very small vocabulary and very few unknown or OOV words. The time complexity and memory may be considerably minimized by this kind of tokenization. However, when the input's and output's length booms, it gets more difficult to grasp the relationships between the letters and construct coherent words and thus sentences. Character-level RNNs and their correlation with word boundaries were manually analyzed as early as the 90s by (Elman, 1990).

- ***Subword Tokenization*** or in some cases syllables tokenization (Mikolov et al., 2011), splits the rare words into smaller meaningful subwords while retaining the frequently used ones (e.g. "tokenization", "token" + "##ization"). The model can learn coherent context-independent representations and have a reasonable vocabulary size. Since the decomposition can produce recognized subwords, a model may even be able to process a word that it has never encountered before.

Expanding on the last point, different subword tokenization algorithms are:

- ***Byte-Pair-Encoding (BPE) (Wu et al., 2016; Sennrich et al., 2016).*** The pre-tokenizer employed by BPE divides the training data into words. After pre-tokenization, a collection of distinct words is produced, and the frequency at which each word appeared in the training data is calculated. After that, BPE builds a base vocabulary out of all the symbols that appear in the collection of unique words and learns how to join tokens from the base vocabulary using merge rules. It continues doing this until the vocabulary has reached the desired size. Keep in mind that one must set the required vocabulary size as a hyperparameter prior to training the tokenizer. After calculating the frequency of each potential symbol pair, BPE then selects the pair that appears most frequently[2].

- ***WordPiece*** (Schuster and Nakajima, 2012) used for BERT, DistillBERT, and Electra, is very similar to BPE. The vocabulary of WordPiece is first initialized to contain each character found in the training data, and it then gradually learns a certain amount of merging rules. In contrast to BPE, WordPiece selects the symbol pair that optimizes the probability of the training data once added to the lexicon, not the most common

[2]https://huggingface.co/docs/transformers/tokenizer_summary#bytepair-encoding-bpe

symbol pair. WordPiece differs slightly from BPE in that it assesses what is lost when two symbols are combined to determine whether it is worthwhile[3].

- ***SentencePiece Unigram algorithm (Unigram) (Kudo, 2018)*** is different form the previous approaches as it starts with an enormous base vocabulary and step by step removes symbols to build a smaller vocabulary. None of the transformers models explicitly use Unigram instead SentencePiece is combined with it[4].

2.5 Word Embeddings

Since machine learning models take vectors of numbers as input the text must be vectorized in some way before being fed to the model. One naive approach would be to use one-hot encodings and encode each word in the vocabulary. However, this approach is inefficient. A one-hot encoded vector is sparse (meaning, most indices are zero). In a case where there are 10,000 words in the vocabulary, to one-hot encode each word, the result would be a vector where 99.99% of the elements are zero. Furthermore relationship between words cannot be retained. More sophisticated methods can be used such Word Embeddings (Almeida and Xexéo, 2019). There exists multiple such methods, of which most popular and achieving SOTA results in NLP tasks are:

- Word2Vec
- GLoVe
- BERT
- ELMo

Word2Vec

Word2Vec (Mikolov et al., 2013) is constructed using two main routines: Skip Gram and Common Bag Of Words (CBOW) (Figure 2.2). Using a range of words as input, the neural network model in CBOW predicts the target word that is closely connected to the input words' context. The Skip-gram architecture, on the other hand, uses one word as input and predicts all of the nearly associated surrounding words.

[3]https://huggingface.co/docs/transformers/tokenizer_summary#wordpiece
[4]https://huggingface.co/docs/transformers/tokenizer_summary#unigram

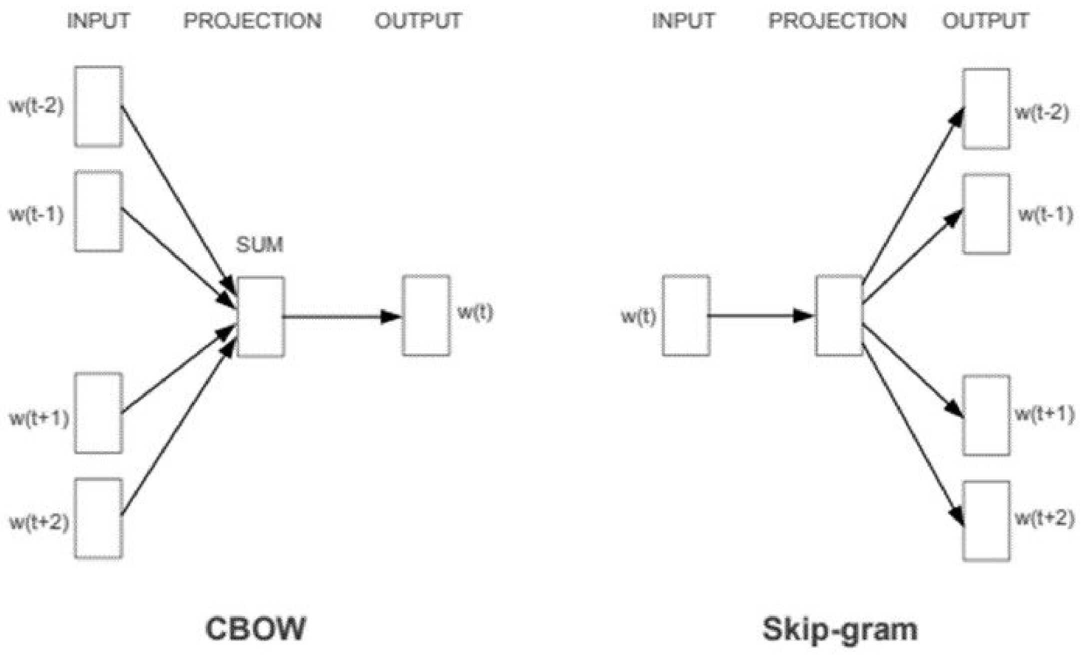


Figure 2.2: CBOW & Skip-gram architectures.

While Skip Gram may effectively represent uncommon words, CBOW is faster and finds superior numerical descriptions for frequent words. Word2Vec models do a decent job of capturing word semantic associations. Word2Vec works well for doing semantic analysis, which is used in knowledge discovery and recommendation systems.

GLoVe

GLoVe (Pennington et al., 2014) extends the work of Word2Vec to capture global contextual information in a text corpora by calculating a global word-word co-occurrence matrix. Word2Vec just records a word's local context. Only nearby words are taken into account when training to capture context. In order to capture the co-occurrence of terms within the corpus, GloVe takes into account the whole corpus and builds a large matrix.

GloVe combines the benefits of two word vector learning methods: local context window approach (Skip-gram) and matrix factorization similar to latent semantic analysis (LSA) (Dumais, 2004). The computational cost of training the model is decreased by the GloVe technique's use of a less complex least square cost error function. The word embeddings that arise are enhanced and different.

Word analogy and named entity identification issues are more easier for GloVe to solve than they are for Word2Vec, but Word2Vec is faster in the computation than GloVe. In certain tasks, GloVe outperforms Word2Vec, while competing in others. However, both methods are effective in identifying semantic data in a corpus (Agarwal, 2021).

BERT

BERT (Devlin et al., 2018) is a pre-trained encoder-based transformer model that comes in two versions, $\boldsymbol{BERT_{Base}}$ which has 110 million parameters and $\boldsymbol{BERT_{Large}}$ which has 340 million parameters and will be discussed further in subsection 2.9.1. In contrast with the previous approaches the embeddings generated by BERT are context aware, therefore they will vary based on the whole sentence or sequence being processed at the time.

ELMo

On top of a two-layer bidirectional language model (biLM), *ELMo* (Peters et al., 2018) word vectors are calculated (bidirectional LSTM (Schuster and Paliwal, 1997)). For the language model to understand both the previous and the next word, each layer has two passes: a forward pass and a backward pass. ELMo creates the contextualized embedding by assembling the hidden states in a certain manner (concatenation followed by weighted summation) (Alammar, 2021).

Words with numerous meanings are combined into a single representation, and polysemy and homonym are improperly handled by word2vec and GLoVe, which is one of their key drawbacks. However, contextually meaningful embeddings like ELMo and BERT have been created, which use the context of a phrase to distinguish between polysemes, in other words, the vector representation of each word is dynamic and adapts depending on the context of a phrase.

2.6 Denoising Autoencoders

An artificial neural network called an autoencoder is used to develop effective codings for unlabeled input (unsupervised learning) (Kramer, 1991). By making an attempt to recreate the input from the encoding, the encoding is validated and improved. By teaching the network to discard irrelevant input (“noise”), the autoencoder learns an encoding for a data point. There exist various types of Autoencoders like:

- **Denoising autoencoder**
- **Sparse Autoencoder**
- **Deep Autoencoder**

- **Contractive Autoencoder**
- **Undercomplete Autoencoder**
- **Convolutional Autoencoder**
- **Variational Autoencoder**

In the case of a Denoising Autoencoder (Vincent et al., 2008) (Figure 2.3), by adding noise in a random manner to the input vector, the data gets partially distorted. The model is then taught to accurately predict the original, unaltered data point as its output. Some of the tranformer models, like BART (subsection 2.9.4), are in part a form of denoising autoencoders.

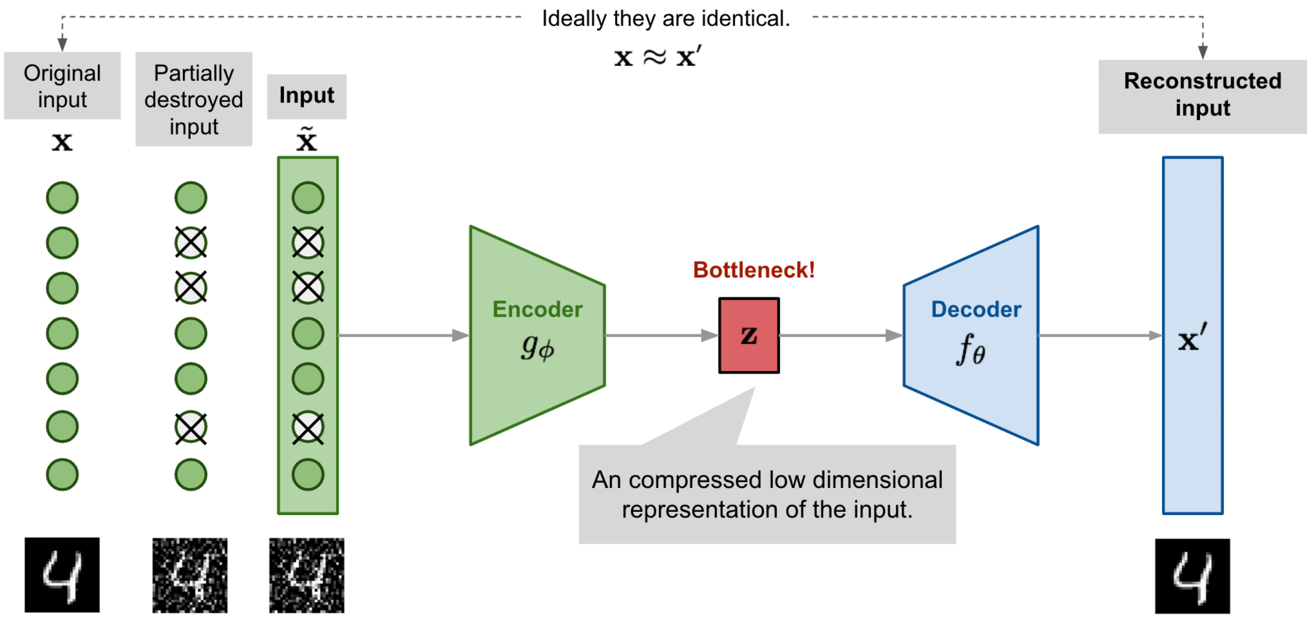

Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Original input
$\mathbf{x}$
Partially destroyed input
Input
$\tilde{\mathbf{x}}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
Reconstructed input
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 2.3: Illustration of the denoising auto-encoder model architecture (Robinet, Nov 26, 2020).

2.7 Transformer-Based Architectures

The Transformer (Vaswani et al., 2017) is a model created by the Google Brain and Google Research team in 2017 and was mainly introduced to tackle the issue of parallel computation plaguing the recurrent neural networks, long short-term memory (Hochreiter and Schmidhuber, 1997) and gated recurrent neural networks (Chung et al., 2014) as well as the vanishing gradients problem. This model is based entirely on the various attention mechanisms and abandoned the idea of recurrence, which was at the time the state-of-the-art for NLP and seq2seq problems. At a high level, it's the same as the previous sequence-to-sequence models with an encoder-decoder pair. The placement and sequencing of words are crucial components of every language. They specify a sentence's syntax, hence defining the sentence's real meaning. One major difference between LSTMs and Transformers is that using positional encoding the latter can capture longer dependencies in a sentence while being more computation efficient.

Language models before transformers could only read text input sequentially (either left-to-right or right-to-left) but couldn't do both at the same, with the exception of bidirectional RNNs, Transformer's encoders on the other hand are bidirectional because they are designed to read in both directions at once. The Transformer can directly access all positions in the sequence, equivalent to having full random access memory of the sequence during encoding/decoding. Transformers may process input in any sequence, making it feasible to train on far bigger data sets than was previously conceivable. This results in improved results since more connections and information are preserved.

2.8 Loss functions

For summarization tasks using the Transformer architecture in the final layers the vector containing the probability distribution of the various words is created. Each probability distribution is represented by a vector of width equal to the vocabulary size. Subtracting the two probability distributions (the one of the gold summary and the predicted one) will get us the loss function. Cross-Entropy builds upon entropy and calculates the difference between two probability distributions. Categorical crossentropy is a loss function that is used in multi-class classification tasks.

$$L_{Cross} = -\sum_{i=1}^{n} y_i * log(\hat{y_i}) \tag{2.1}$$

where $\hat{y_i}$ is the i-th scalar value in the model output, y_i is the corresponding target value, and n is the number of scalar values in the model output.

The model chooses the word with the greatest probability from that probability distribution and discards the remaining words because the model only generates one output at a time (greedy decoding). There are various other ways that this selection can be done and will discussed in the next sections.

2.9 Transformer

The Transformer model consists of two main parts, the encoder and the decoder components. In both parts, there are multiple layers (6 layers as per the original paper) stacked on top of each other which are identical but do not share the same weights.

The encoder block has two sub-layers, the self-attention, hence the novelty, followed by a simple feed-forward layer. Each sub-layer is followed by a residual connection (He et al., 2016) and by layer normalization (Ba et al., 2016). An overview of the architecture can be seen in Figure 2.4.

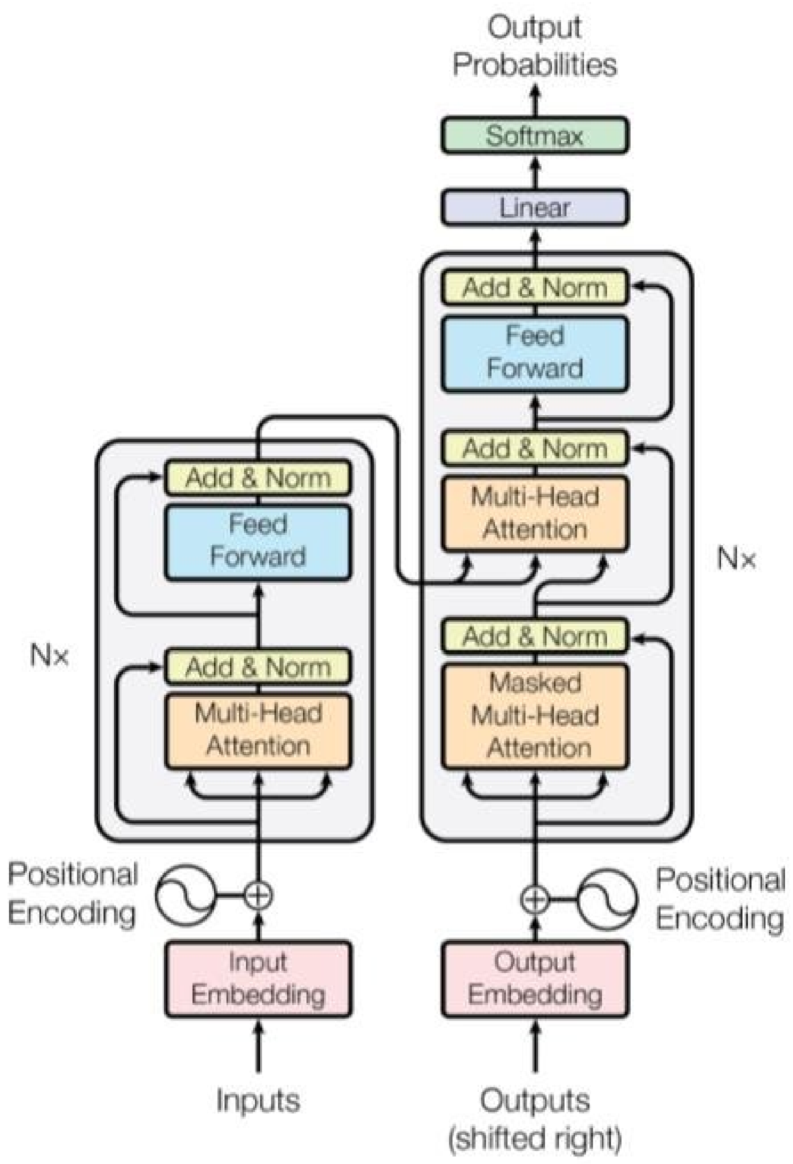


Figure 2.4: The Transformer - model architecture.

After the text gets vectorized using an embedding algorithm, a positional vector is added to the vectorized text in order to capture the order of the words in the sentence. Each word is now a vector and for each word 3 distinct Query, Key and Value vectors are created. This concept comes from information retrieval systems and is the first step towards self-attention. In this attention technique, the context vector is calculated as a weighted sum of the values, with each value's weight determined by the query's compatibility with its associated key.

$$ScaledDot - ProductAttention(Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

Where d_k is the dimension of the Key vector.

There are 8 different Q, K and Value vectors in an attempt to better capture the context of the input, hence the multi head attention (Figure 2.5). This improves the performance of the attention layer in two ways, it expands the model's ability to focus on different positions and it gives the attention layer multiple "representation subspaces", in addition these different heads are calculated in parallel. Finally, the results of the 8 pairs are concatenated and multiplied by an additional weight matrix before the feed-forward layer. In this layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder since the encoder component consists of 6 stacked encoders. Each position in the encoder can attend to all positions in the previous layer of the encoder. The encoder does not have self-attention masking. Therefore is designed not to have any dependency limitation: the token representation obtained at one position depends on all the tokens in the input.

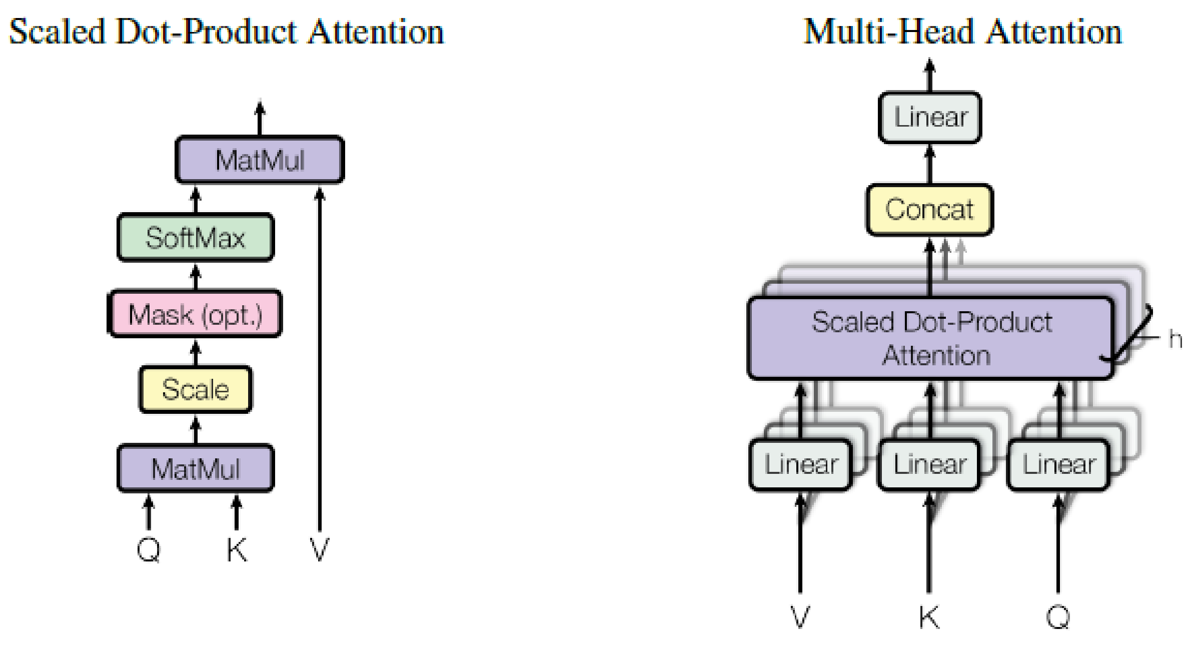


Figure 2.5: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

At this point, which is the end of the encoder stack, the model has taken an arbitrary input of embeddings and has managed to put some context to these embeddings, the resulting vector and in particular the keys and values of it will now be fed to the decoder component. Each decoding layer gets as input the queries from the previous decoding layer and the keys and values of the last encoding layer. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models (Wu et al., 2016; Bahdanau et al., 2014; Gehring et al., 2017). Each position in the decoder is able to attend to all positions up to and including it thanks to the self-attention layers in the decoder. Masked multi head attention is used to stop leftward information flow in the decoder and to maintain the auto-regressive characteristic[5]. The self-attention layer in the decoder is only permitted to focus on earlier points in the output sequence. This is accomplished by hiding future places in the self-attention computation by setting them to -inf before the SoftMax phase (Figure 2.5).

Finally the decoder's output is passed through a fully connected layer that projects this vector into a larger "logits" vector with dimensions equal to the size of the vocabulary used by the model, each neuron corresponding to the score of a unique word. The SoftMax layer turns those scores into probabilities and the word associated with the highest probability is chosen.

[5]The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term.

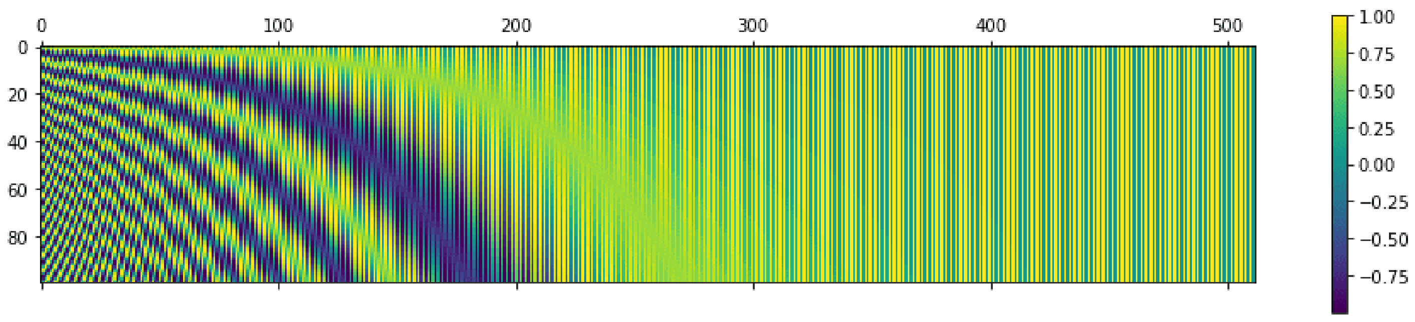


Figure 2.6: The positional encoding matrix for $n = 10000$, $d = 512$, sequence length=100.

The idea of the positional vector (Figure 2.6) was introduced previously but was not explained, in the rest of this section the positional vector used in the Transformer will be analyzed and explained in greater detail. The positional vector is a d-dimensional vector that contains information about a specific position in a sentence. Let t be the desired position in an input sentence, $\overrightarrow{p_t} \in \mathbb{R}^d$ be its corresponding encoding and d be the encoding dimension. Then $f : \mathbb{N} \rightarrow \mathbb{R}^d$ will be the function that produces the output vector and it is defined as follows (Kazemnejad, 2019):

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases} \tag{2.3}$$

where

$$\omega_k = \frac{1}{10000^{2k/d}} \tag{2.4}$$

The frequencies are decreasing along the vector dimension, forming a geometric progression from 2π to $10000 \cdot 2\pi$ on the wavelengths. The positional embedding $\overrightarrow{p_t}$ is a vector containing pairs of sines and cosines for each frequency (d is divisible by 2):

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1} \tag{2.5}$$

Because of the intermediate residual connections in the Transformer, the information from the input of the model containing the positional embeddings can efficiently propagate to further layers and thus the position information is maintained (Kazemnejad, 2019).

2.9.1 BERT

BERT (Devlin et al., 2018) which stands for ***B**idirectional* ***E**ncoder* ***R**epresentations from* ***T**ransformers*, presented a new self-supervised[6] task for pre-training transformers in order to fine-tune them on downstream tasks such as Neural Machine Translation, question answering, sentiment analysis, text summarization and more. BERT is essentially a trained group of stacked encoders following the first half of the Transformer architecture. BERT pre-trains by performing two unsupervised tasks simultaneously, namely **Masked Language Modeling (MLM)** and **Next Sentence Prediction (NSP)**.

- **Masked Language Modeling (MLM):** BERT randomly masks words in the sentence and predicts them. For each example, 15% of the tokens are selected uniformly at random to be masked. When a token is masked it is replaced with a special **[MASK]** token. The problem with this approach is that the model will inherently produce not so rich hidden states if there is no presence of a **[MASK]** token. So the authors in an attempt to force the model to try to predict the correct tokens regardless of what token is present in the input, they sometimes replace words in the sentence with random words instead of the **[MASK]** token. This is why BERT swaps 10% of the 15% tokens selected for masking and leaves 10% of the tokens intact. The remaining 80% are replaced with

[6]Self-supervised learning is a machine learning process where the model trains itself to learn one part of the input from another part of the input.

the **[MASK]** token. The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words.

  - 80% of the time with **[MASK]** tokens.
  - 10% of the time with a random tokens.
  - 10% of the time with the unchanged input tokens that were being masked.

- **Next Sentence Prediction (NSP)**. When taking two sentences as input in order to capture the relationship between them, a unique **[SEP]** token is used to separate the sentences. During training, BERT is given two sentences, with the second one appearing after the first 50% of the time and a randomly selected sentence from the corpus the other 50% of the time. In order to determine if the second sentence is random or not, BERT must then make a prediction.

Same as with the regular Transformer, the input is passed to BERT and then it is tokenized adding the position embeddings with the difference being that now the segment embeddings are added as well, as can be seen in Figure 2.7.

Whether a sentence is a component of sentence A or B is indicated by the segment embeddings. A learned embedding $\boldsymbol{E_A}$ is concatenated to every token of the first sentence and another learned vector $\boldsymbol{E_B}$ to every token of the second one.

A special **[CLS]** token is provided as the first input token, CLS here stands for Classification and is a start of the sentence or sequence token. This token can be used as input to a classifier.

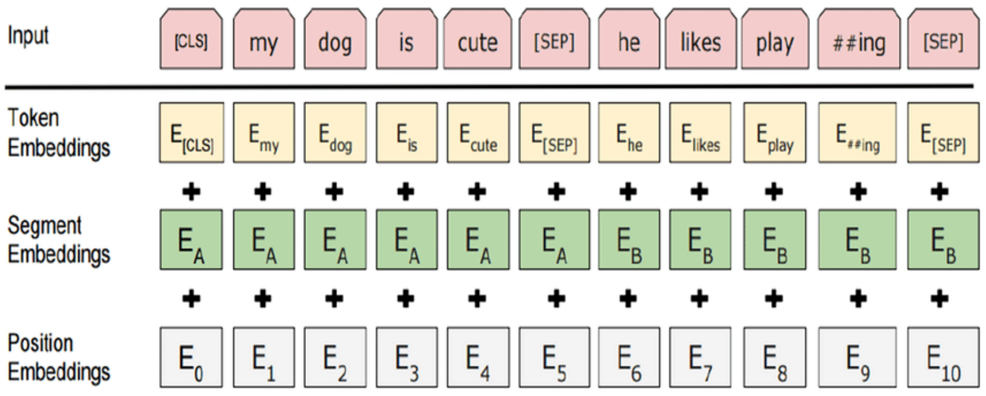

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.7: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. (Devlin et al., 2018)

A key point is that BERT uses word pieces instead of words (e.g. sunny, becomes sun + ##ny or hotter becomes hot + ##ter), which reduces the vocabulary size and further information is made available for each word.

BERT (Figure 2.8) comes in two flavors, the $\boldsymbol{BERT_{Base}}$ and the $\boldsymbol{BERT_{Large}}$ models which have 12 and 24 encoders stacked on top of each other respectively and the total number of parameters sum up to 110 million and 340 million. These also have larger feed forward-networks (768 and 1024 hidden units respectively) and more attention heads (12 and 16 respectively) than the default configuration in the reference implementation of the Transformer in the initial paper (6 encoder layers, 512 hidden units and 8 attention heads).

Finally, altough the BERT encoder produces a sequence of hidden states, the authors decided to use only the hidden state corresponding to the first token for classification tasks. The pre-training schema as well the usage of BERT for different tasks such as question answering and sentence classification (Figure 2.9).

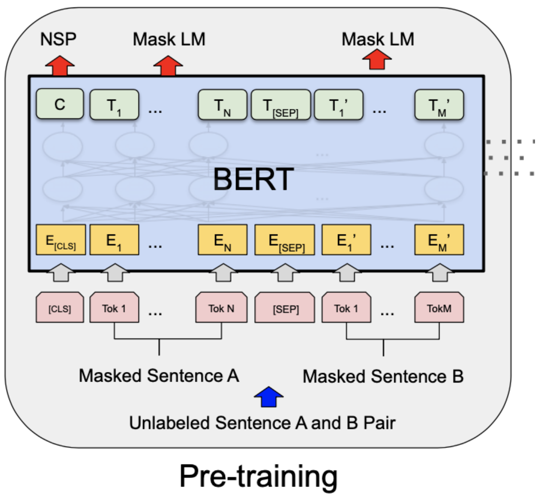
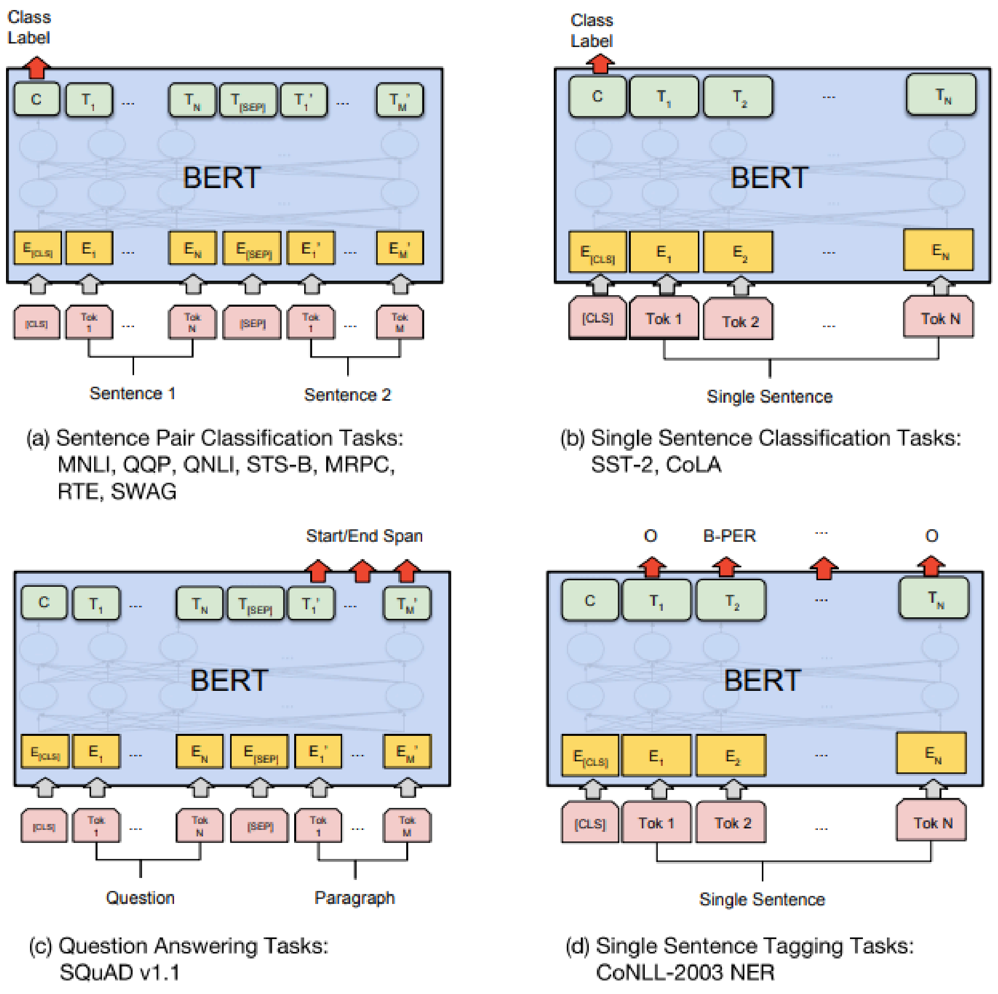


Figure 2.8: Pre-training BERT.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 2.9: Using BERT for different tasks.

Since BERT utilizes the encoder segment from the vanilla Transformer only, it excels at comprehending natural language, but struggles at generating text. BERT can also be fine-tuned for summarization tasks. BERTSUM (Liu, 2019) is a variant of BERT for solving that task and focuses on extractive summarization, while BERTSUMABS is an encoder-decoder Transformer variant for solving abstractive text summarization using BERTSUM as the encoder component (Liu and Lapata, 2019).

As can be seen in Figure 2.10, an illustration of the architecture used in BERTSUM, the sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. Contextual vectors are created for each token using the summarized vectors as input embeddings to several bidirectional Transformer layers. By adding several **[CLS]** tokens to learn sentence representations and utilizing interval segmentation embeddings (red/green color) to distinguish between multiple sentences. The vector Ti which is the vector of the $i-th$ **[CLS]** symbol from the top BERT layer will be used as the representation for $sentence_i$.

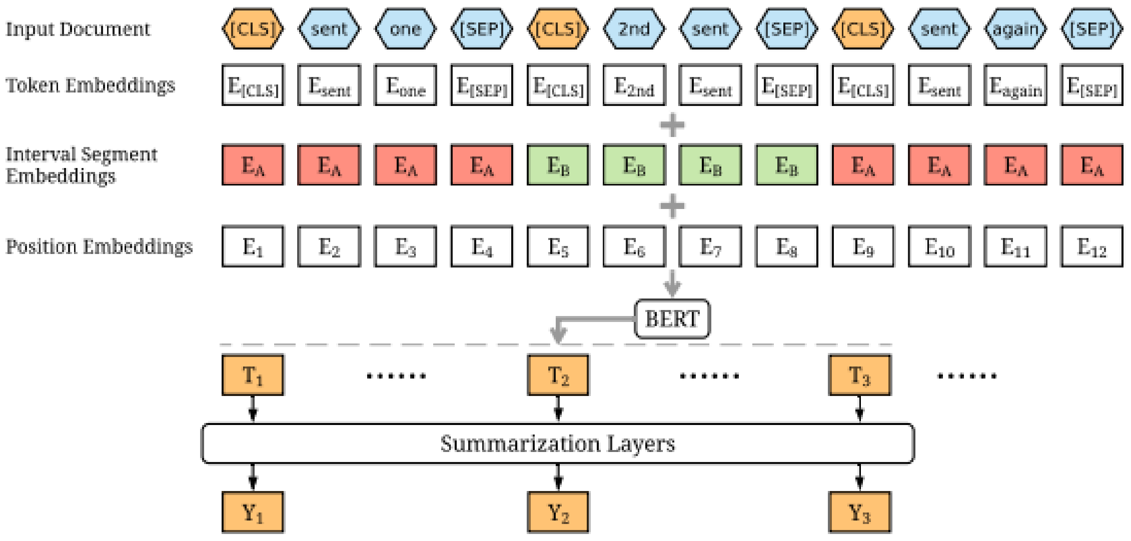


Figure 2.10: BERTSUM encoder.

2.9.2 RoBERTa

RoBERTa (Liu et al., 2019) which stands for ***R**obustly* ***o**ptimized* ***BERT*** ***a**pproach* was introduced by researchers at Facebook and Washington University and as the name implies it is an optimized version of BERT. Key changes made to achive this are:

- **Removing the Next Sentence Prediction (NSP) objective.**
- **Training with more data, bigger batch sizes & longer sequences**: With a batch size of 256 sequences, BERT is first trained for 1M steps. The model was trained by the authors of this study for 125K steps of 2K sequences and 31K steps of batch size of 8K sequences. This has two benefits: the huge batches increase perplexity on masked language modelling objective as well as end-task accuracy. Through distributed parallel training, large batches are also simpler to parallelize.
- **Dynamic masking using different masking patterns of the tokens.** In BERT, each text is masked 10 times in different ways during the pre-processing stage. This ensures that the model will only see these 10 variants throughout training. Additionally, the aforementioned mask is a single static mask. The number of possibly distinct masked versions of each sentence, however, is not constrained as in BERT because the suggested technique applied the masks during the training.

2.9.3 Knowledge distillation & DistillBERT

Knowledge distillation

The task of converting information from a huge model or group of models to a single, more manageable model that can be used realistically under real-world limitations is known as knowledge distillation. It is a form of model compression (Bucilua et al., 2006).

With neural network models that have complex designs with several layers and model parameters, knowledge distillation is employed more frequently. As a result of deep learning's growth over the past ten years and its success in a variety of fields, including speech recognition, image recognition, and natural language processing, knowledge distillation techniques have become more and more popular (Gou et al., 2021).

The difficulties of large deep network implementation are particularly pertinent for systems with little memory and processing capacity. An original solution to this issue was developed by using a model compression approach (Bucilua et al., 2006) to train a smaller model with data from a larger one without suffering a significant performance loss. This process of learning a smaller model from a larger one is officially described by a framework called "Knowledge Distillation." (Hinton et al., 2015)

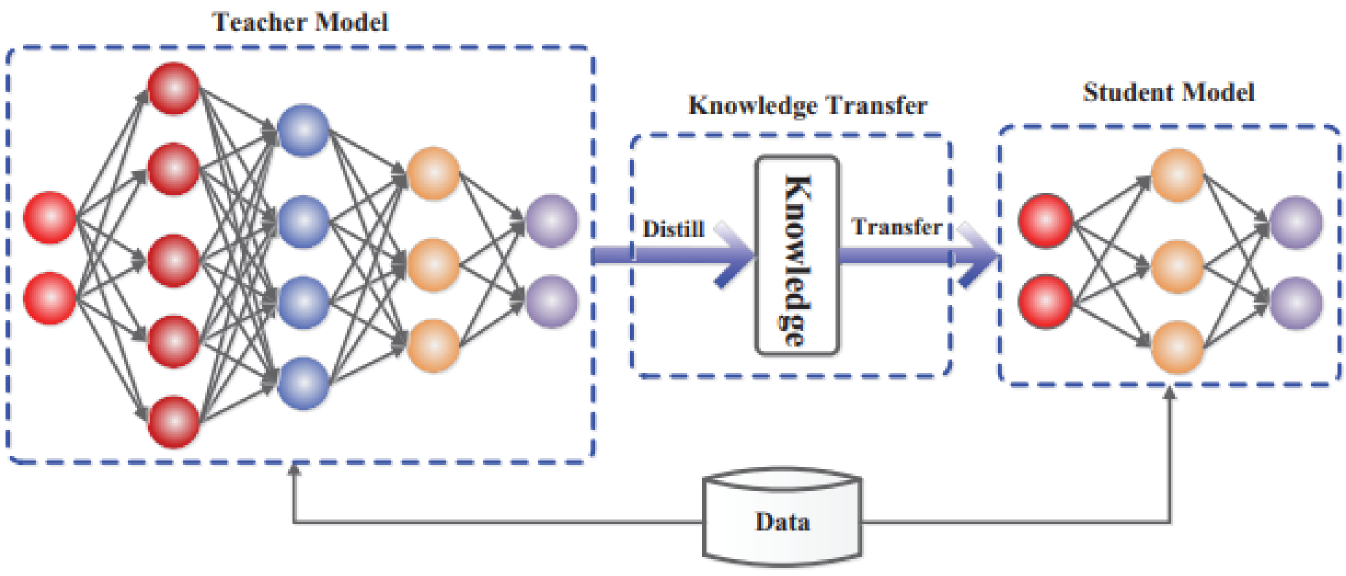


Figure 2.11: The generic teacher-student framework for knowledge distillation (Gou et al., 2021).

DistillBERT

DistillBERT (Sanh et al., 2019) is a smaller, pre-trained general-purpose language representation model that performs well and can be fine-tuned on a variety of downstream NLP tasks, similar to its larger rivals. DistillBERT, which was developed by the Hugging Face group, uses the knowledge-distillation framework to obtain results that are almost identical to those of the original BERT model while being smaller, more effective, and requiring less training time. In comparison to BERT, DistillBERT is 40% smaller, 60% quicker, and preserves 97% of the language understanding skills.

The BERT model, with a few minor alterations, was specifically utilized as the student. The classifier *(pooler)* for the **[CLS]** token and the *token-type embeddings* were also removed, along with the segment embeddings (Figure 2.7) and the number of layers is reduced by a factor of 2. Since there is common dimensionality between teacher and student networks the initialization of the student network is done directly by copying one layer out of two from the teacher. The training employed during the distillation process followed best practices for teaching BERT model as provided in RoBERTa (Liu et al., 2019). The instructor model used was BERT. It was trained using the same corpora as the initial BERT model: a synthesis of English Wikipedia and Toronto Book Corpus, which totals 3.3 billion words (Zhu et al., 2015).

An important thing to note here is the loss function used during back propagation in order for the distillation process to work correctly. Assuming that the Teacher and Student for an input x, produce *T(x)*

$$T(x) = (t_1, ..., t_n) = SoftMax(\hat{t_1}, ..., \hat{t_n}) \tag{2.6}$$

and *S(x)* results respectively

$$S(x) = (s_1, ..., s_n) = SoftMax(\hat{s_1}, ..., \hat{s_n}) \tag{2.7}$$

A triple loss function, which is the combination of the following three loss functions was used:

- *Teacher-Student cross-entropy loss*, which involves minimizing the loss of the student's predictions against the teacher's "soft predictions", by using the SoftMax temperature function as suggested by (Hinton et al., 2015).

$$L_{Cross} = -\sum_{i=1}^{n} t_i * log(s_i) \tag{2.8}$$

- *Cosine Embedding Loss*

$$L_{Cosine} = 1 - cos(T(x), S(x)) \tag{2.9}$$

- *Classic train loss of the student's model*

The full distillation loss which is a combination of the three losses mentioned above can be then be defined as:

$$L_{Distillation} = \frac{L_{Cross} + L_{Cosine} + L_{Classic}}{3} \tag{2.10}$$

2.9.4 BART

BART (Lewis et al., 2019) which stands for ***B****idirectional* ***A****uto-***R****egressive* ***T****ransformers* is a self-supervised denoising auto-encoder for pretraining sequence-to-sequence models based on the transformer architecture (Vaswani et al., 2017) and it was first introduced by the Facebook AI research team in 2019. As the BART authors write, BART can be seen as generalizing BERT (Devlin et al., 2018) (bidirectional encoder) and the auto-regressive GPT-2 (left to right decoder) (2.13a, 2.13b). BART uses a noise corrupted source text as input and an LM (Language Model) for reconstructing the original text by predicting the true replacement of corrupted tokens. The authors provided two variants of the BART model the $\boldsymbol{BART_{Base}}$ with 6 layers in the encoder and decoder and the $\boldsymbol{BART_{Large}}$ consisting of 12 layers and a hidden size of 1024. BART was pre-trained following the RoBERTa (Hendrycks and Gimpel, 2016) pre-training guidelines discussed in subsection 2.9.2. A notable feature is that instead of using ***Re****ctified* ***L****inear* ***U****nits* (ReLUs) (Agarap, 2018) as activation functions, ***G****aussian* ***E****rror* ***L****inear* ***U****nits* (GELUs) (Hendrycks and Gimpel, 2016) are used instead in their place, GELU can be thought of as a smoother ReLU (Figure 2.12).

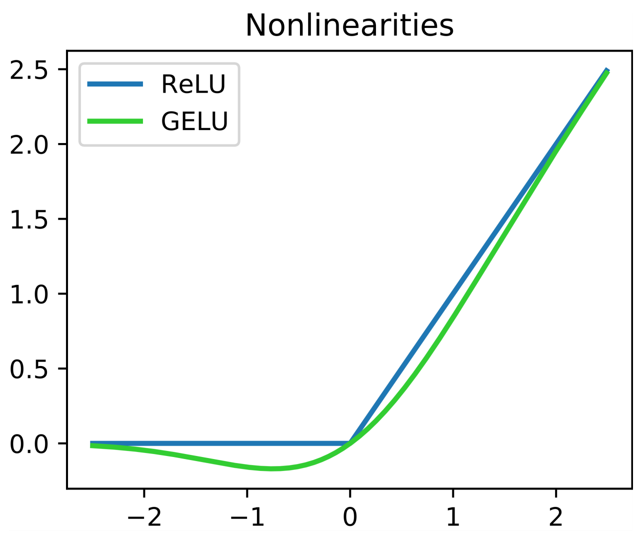
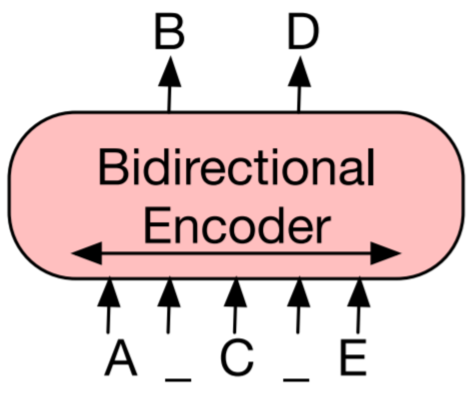


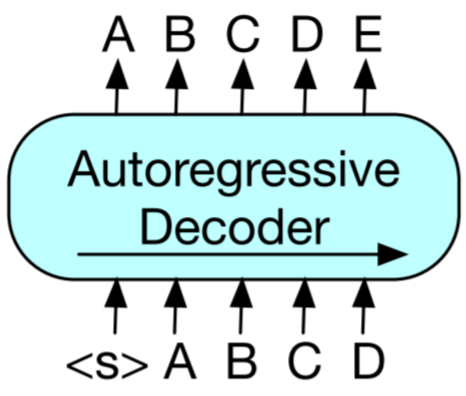
Figure 2.12: ReLU and GELU.

BERT's basic pre-training objectives are **Masked Language Modeling (MLM)** and **Next Sentence Prediction (NSP)** (subsection 2.9.1). As a result it utilizes the whole input sequence to try to predict masked tokens. Since every token is predicted independently and simultaneously, it is excellent for tasks where the prediction at position *i* is permitted to use information from positions before and after *i*.

GPT-2, on the other hand, was trained to guess the next word in sentences and the predictions are made in an auto-regressive fashion, meaning that GPT can be used for text generation, however words can only attend to leftwards (2.13b).



(a) BERT



(b) GPT

Figure 2.13

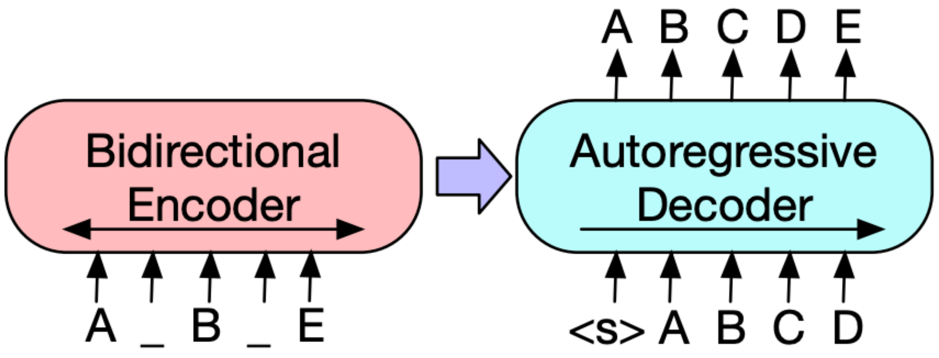


Figure 2.14: A schematic comparison of BART with BERT and GPT. (source (Lewis et al., 2019))

As in the vanilla Transformer, in BART each layer of the decoder performs cross-attention over the final hidden layer of the encoder in addition to the previous decoder layer. BART is trained by corrupting documents and then optimizing a reconstruction loss—the cross-entropy between the decoder's output and the original document. The methods that are employed for corrupting with noise the original input text are (Figure 2.15):

- **Token Masking**: Tokens are sampled in a stochastic manner and substituted with the **[MASK]** tokens. In particular 30% of tokens in each document are masked. This enables the model to learn efficient context aware representations.
- **Token Deletion**: Tokens at random are removed from the input.
- **Text Infilling**: In order for the model to learn how many tokens are missing from a certain spot, text spans of intervals sampled from the Poisson distribution are replaced with a **[MASK]** token (Joshi et al., 2019).
- **Sentence Permutation**: Full stops are used to segment a document into sentences, which are then randomly rearranged. This tests the model's comprehension of causal and temporal linkages given the challenge of creating a meaningful text from a group of phrases that have been randomly rearranged (Chowdhury et al., 2021).
- **Document Rotation**: The document is rotated to start with a token that is uniformly selected at random, in order to learn to recognize the document's beginning.

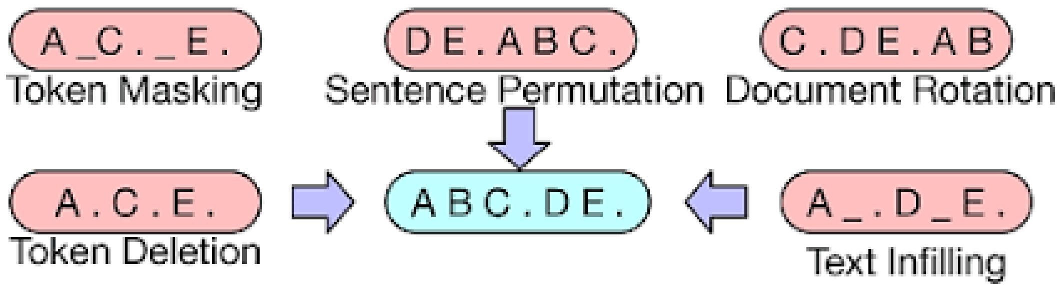


Figure 2.15: Transformations for corrupting the input sequence as seen in the BART model.

BART can be fine-tuned for downstream tasks such as :

- **Sequence Classification Tasks**: The same input is sent into the encoder and decoder for sequence classification tasks, and the final hidden state of the last decoder token is fed into a new multi-class linear classifier. This method is similar to the **[CLS]** token in BERT, but an extra token is added at the end so that the token's representation in the decoder can take into account decoder states from the entire input.
- **Token Classification Tasks**: For token classification tasks, the entire text is put into the encoder and decoder, and each word is represented by the top hidden state of the decoder. The token is categorized using this representation.
- **Sequence Generation Tasks**: BART can be fine-tuned for tasks like answering questions and abstractive summarizing since it features an auto-regressive decoder. Both of these tasks involve copying information from the input and manipulating it, which is strongly connected to the denoising pre-training objective. The input sequence serves as the encoder's input in this case, while the decoder produces auto-regressive outputs.
- **Machine Translation**: A new encoder is introduced that accepts the source sequence as input and is supposed to feed its output to the decoder which is now the whole pretrained encoder-decoder framework. As the encoder attempts to learn the alignment between the source and target sequences, the parameters of the pretrained architecture are frozen throughout training and only the new encoder's parameters are learnt. The model is then trained for a certain number of iterations.

2.9.5 T5

The *Text-to-Text Transfer Transformer* (T5), is a unified framework that converts all text-based language problems into a text-to-text format, as proposed by the GOOGLE AI team in 2019. *"Every task such as translation, question answering, and classification—is cast as feeding the* *model text as input and training it to generate some target text. As a result, it is possible to use the same model, loss function, hyperparameters, etc. to a variety of tasks."* (Raffel et al., 2019). Instead of training the model independently on many different tasks, the names of those same tasks are used as a prefix to the original input when inserting them to the model (McCann et al., 2018; Keskar et al., 2019) (Figure 2.17).

The model is trained with a maximum likelihood objective ("teacher forcing" (Williams and Zipser, 1989)) and a cross-entropy loss.

⋙ Recurrent neural networks may be trained using the teacher forcing method, which substitutes the model's output from a previous time step for ground truth as input (Lamb et al., 2016).

⋙ The strategy was originally designed as an alternate method to back propagation across time for recurrent neural network training. (Williams and Zipser, 1989).

⋙ Instead of using the output produced by the network, teacher forcing uses the actual output from the training dataset at the current time step y(t) as input in the following time step X(t+1). (Brownlee, April 8, 2021)

The model as a result is able to generalize better, albeit the growing memory requirements. The maximum allowed length of the tokens in the input is 512 tokens and at test time greedy decoding is used as well as beam search (Sutskever et al., 2014) (section 2.10) with a beam size of 4 and a length penalty of $\alpha = 0.6$ (Wu et al., 2016).

The authors did an extensive survey of many of the previous frameworks and architectures used for NLP tasks and after experimenting and establishing the SOTA models and techniques used by them, they tested various permutations of them. In particular they investigated:

- **Model architectures.** Where it was determined that "encoder-decoder" language models performed better in general than "decoder-only" language models.
- **Pre-training objectives.** Which verified that computation cost is the most crucial component and that fill-in-the-blank style denoising targets are the best approach.
- **Unlabeled datasets.** They noticed that training on in-domain data can be good and that smaller datasets may cause overfitting.
- **Training strategies.** They discovered that fine-tuning following pre-training on a variety of tasks led to performance that was on par with unsupervised pre-training.

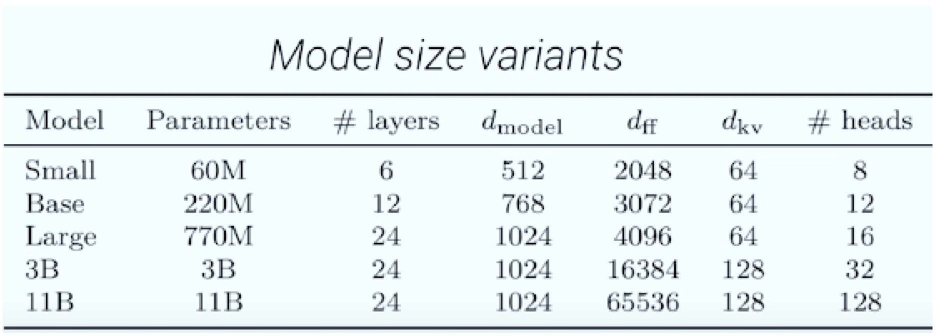
- **Scaling the model size.**

The authors trained 5 versions after integrating all of these principles and scaling things up: small model, base model, large model and models with 3 billion and 11 billion parameters.

The T5 model was trained on the *Colossal Clean Crawled Corpus* (C4) dataset which was developed by the team since previous datasets didnt't match the criteria. C4 is a cleaned-up version of Common Crawl which is orders of magnitude larger than Wikipedia.

The 5 models trained by the GOOGLE AI team (Figure 2.16):

- $T5_{Small}$, with 60 million parameters.
- $T5_{Base}$, with 220 million parameters.
- $T5_{Large}$, with 770 million parameters.
- $T5_{3B}$, with 3 billion parameters.
- $T5_{11B}$, with 11 billion parameters.



Model size variants

Model	Parameters	# layers	d_{model}	d_{ff}	d_{kv}	# heads
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

Figure 2.16: T5 model size variants.

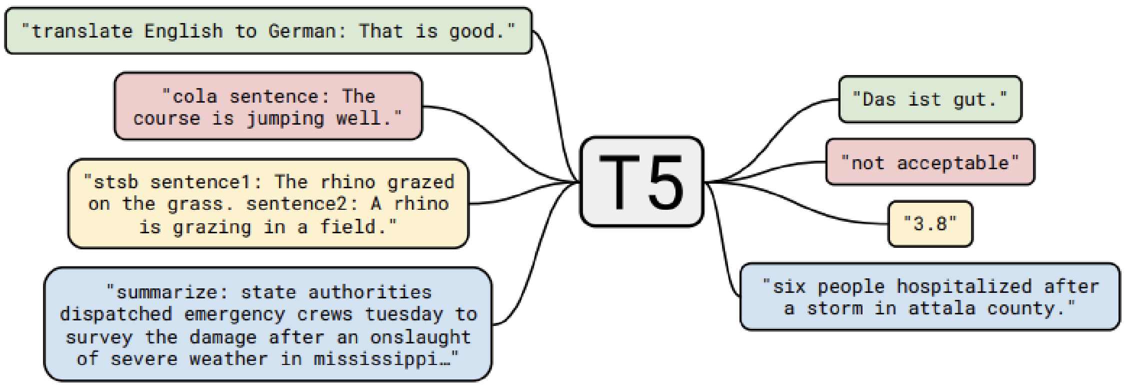

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 2.17: T5 framework.

The architecture of T5 closely resembles that of the vanilla Transformer consisting of the encoder and the decoder components. The only differences are that a simpler layer normalization is used where the activations are only rescaled and no additive bias is applied as well as "*a different position embedding scheme*".

The authors choose to employ relative position embeddings instead of the original Transformer's sinusoidal position signal or learnt position embeddings (Shaw et al., 2018; Huang et al., 2018). They utilized a streamlined version of position embeddings, where each "embedding" is only a scalar added to the relevant logit that is used to calculate the attention weights. Each attention head utilizes a distinct learnt position embedding inside a particular layer, however they share the position embedding parameters between all levels of the model to maximize efficiency (Raffel et al., 2019).

The T5 model is trained based on the unsupervised objective of predicting tokens that are masked (Figure 2.18), inspired by BERT's "masked language modeling" and SpanBERT (Joshi et al., 2019). Tokens are randomly sampled from the input sequence and dropped at a rate of 15% with a span length of 3.

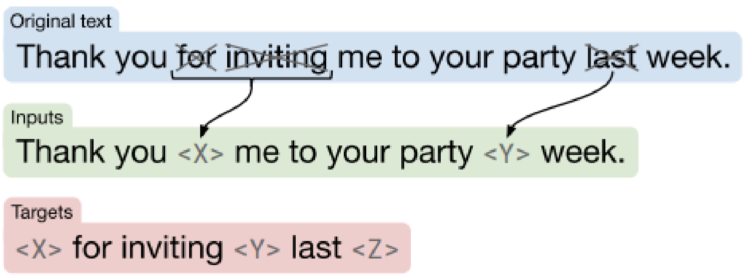

Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.
Inputs
Thank you <X> me to your party <Y> week.
Targets
<X> for inviting <Y> last <Z>

Figure 2.18: Schematic of the unsupervised objective of T5 (Raffel et al., 2019) using masked language modeling.

2.9.6 PEGASUS

So far the notion was that the various Transformer based models will be pre-trained on a task with self-supervised auto regressive or language modeling objectives and then be fine-tuned for downstream tasks. However, text summarization objectives did not get explored that much, but using the pre-trained models these objectives could be solved to some degree. In order to address this issue, the GOOGLE AI research team presented a novel pre-training technique, the PEGASUS model/technique for abstractive summarization (Zhang et al., 2019). PEGASUS stands for ***P***re-training with ***E***xtracted ***G***ap-sentences for ***A***bstractive ***Su***mmarization as per the title of the original paper. The authors hypothesized that "*using a pre-training objective that closely resembles the downstream task leads to better and faster fine-tuning*". As a result instead of using previously employed techniques such as Masked Language Modeling (MLM) (Devlin et al., 2018) they introduced a new pre-training task, the Gap Sentences Generation (GSG).

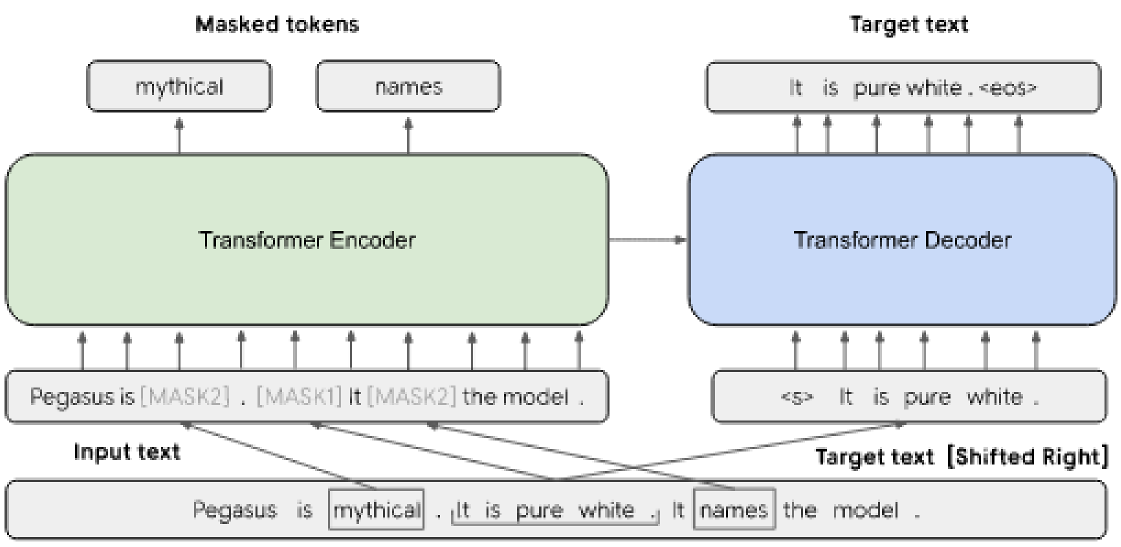

Masked tokens
Target text
mythical
names
It is pure white . <eos>
Transformer Encoder
Transformer Decoder
Pegasus is [MASK2] . [MASK1] It [MASK2] the model .
<s> It is pure white .
Input text
Target text [Shifted Right]
Pegasus is mythical . It is pure white . It names the model .

Figure 2.19: The base architecture of PEGASUS. Both GSG and MLM are applied simultaneously. Originally there are three sentences. One sentence is masked with **[MASK1]** and used as target generation text (GSG). The other two sentences remain in the input, but some tokens are randomly masked by **[MASK2]** (MLM) (Zhang et al., 2019).

The authors noticed that pre-training models on certain datasets can transfer more effectively to downstream tasks when their domains are aligned better as can be in seen in Figure 2.20.

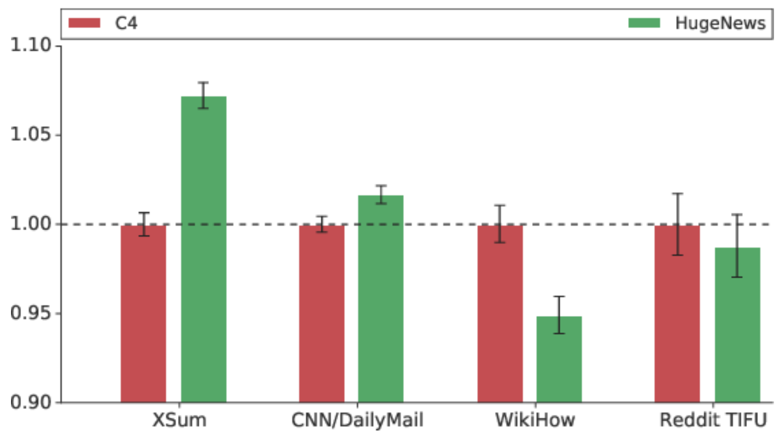


Figure 2.20: Effect of pre-training corpus. $\boldsymbol{PEGASUS_{BASE}}$ pre-trained on C4 (350M Web-pages) and HugeNews (1.5B news-like documents) (Zhang et al., 2019).

Two variants of the PEGASUS model were proposed:

- $\boldsymbol{PEGASUS_{BASE}}$ with L = 12, H = 768, F = 3072 and A = 12
- $\boldsymbol{PEGASUS_{LARGE}}$ with L = 16, H = 1024, F = 4096 and A = 16

where L denotes the number of layers for encoder and decoder (i.e. Transformer blocks), H for the hidden size, F for the feed-forward layer size and A for the number of self-attention heads. They were both trained for 500, 000 steps with 256 and 8192 batch sizes respectively.

For gap-sentence selection methods various methods were tested such as:

- **Lead**: Selecting the first n sentences as a pseudo-summary of the document.
- **Random**: Uniformly selecting sentences at random.
- **Ind-Orig**: The top-n ranked sentences are selected in accordance with importance.
- **Ind-Uniq**, **Seq-Orig** and lastly **Seq-Uniq.**

"A significant hyper-parameter in GSG is the gap-sentences ratio (GSR). A low GSR makes the pre-training less challenging and computationally efficient. On the other hand, choosing gap sentences at a high GSR loses contextual information necessary to guide the generation." It was found that the best ratios are in the range of 15-45% and it depends on various datasets what ratio will yield the best results. For $\boldsymbol{PEGASUS_{LARGE}}$ a GSR of 30% was chosen and it was pre-trained using Ind-Orig as a gap-sentence selection method.

The tokenizations methods tested was Byte-pair encoding algorithm (BPE) (Wu et al., 2016; Sennrich et al., 2016) and SentencePiece Unigram algorithm (Unigram) proposed in (Kudo, 2018). Unigram was chosen as it yielded best results and a Unigram vocabulary size of 96k was used.

The $\boldsymbol{PEGASUS_{LARGE}}$ with 568M parameters model includes the changes:

- The model was pre-trained on the mixture of C4 and HugeNews weighted by their number of examples.
- The model dynamically chose gap sentences ratio uniformly between 15%-45%.
- Importance sentences were stochastically sampled with 20% uniform noise on their scores.
- The model was pre-trained for 1.5M steps.
- The SentencePiece tokenizer was updated to encode the newline character.

In conclusion the authors demonstrated that the model was able to adapt to unseen summarization datasets very quickly, achieving strong results in as little as 1000 examples as can be seen in Figure 2.21.

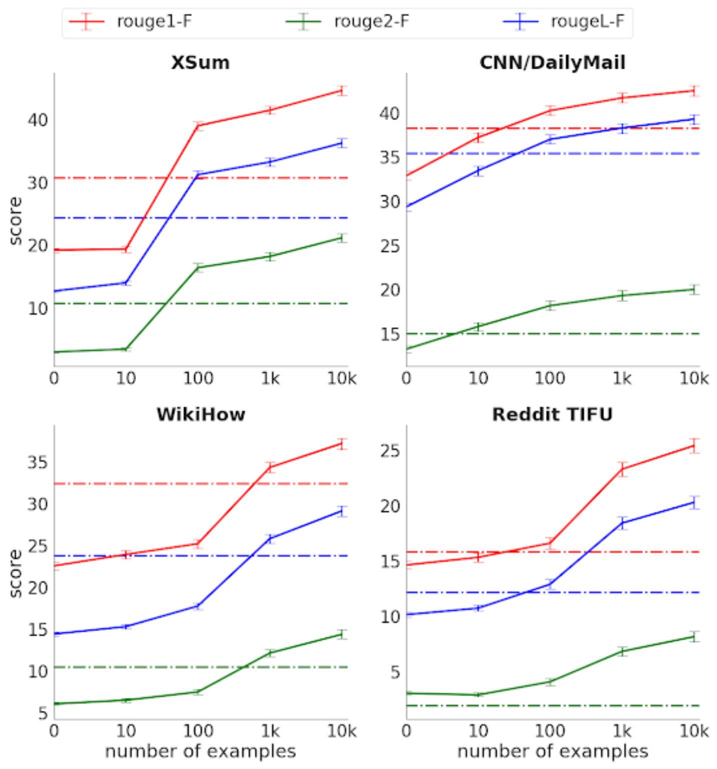


Figure 2.21: Fine-tuning with limited supervised examples. The solid lines are $PEGASUS_{LARGE}$ fine-tuned on 0 (zero shot), 10, 100, 1k,10k examples. The dashed lines are $PEGASUS_{BASE}$ models without pre-training and trained using the full supervised datasets (Zhang et al., 2019).

2.9.7 Longformer

The Transformer architecture's self-attention mechanism, which allows the network to gather contextual information from the whole input sequence, is one of the largest weaknesses of the design but also one of its greatest strengths. Each encoded word in the input sequence attends to every other token while computing the self-attention weights, resulting in a computational complexity of $\mathcal{O}(n^2)$ for sequence lengths of n.

For example, popular models like BERT and RoBERTa can attend to a maximum of 512 tokens at a time, while others like T5 and GPT-2 can utilize a maximum of up to 1024 tokens. If the input sequence is larger than that, different methods, such as Transformer-XL (Dai et al., 2019), Adaptive Span (Sukhbaatar et al., 2019) and Compressive (Rae et al., 2019) have been proposed. In these methods, the input sequence is divided into smaller chunks, and each chunk, moving from left to right (ltr approach), is then individually passed through the Transformer. Despite their effectiveness in auto-regressive language modeling, these models are inappropriate for transfer learning strategies with tasks that profit from bidirectional context. Information is lost during this process because temporal information is not kept and each separate chunk may only attend to previous tokens from the same chunk and not to other chunks.

This problem was beaten with the use of the Longformer (Beltagy et al., 2020), which has an attention mechanism that grows linearly with sequence length and makes it simple to handle documents with thousands of tokens or more.

The Longformer utilizes a fixed-size window attention around each token instead of computing the whole quadratic attention matrix multiplication, much like the Sparse Transformer (Child et al., 2019), which use a type of dilated sliding window of blocks.

A form of self-attention that is sparsified, as proposed by Longformer, involves sparsifying the entire self-attention matrix in accordance with an "attention pattern" that designates pairs of input locations paying attention to one another. The suggested attention pattern grows linearly with the input sequence, unlike the complete self-attention, making it effective for longer sequences. The "attention pattern" proposed in this paper as shown in Figure 2.23 are:

- **Full n^2 attention**: Standard self-attention mechanism as seen in the vanilla Transformer (Vaswani et al., 2017).

- **Sliding window attention**: This enables tokens within a constrained window span w to attend to other tokens. A token at position i can attend to all tokens in the range of $i - w$ and $i + w$ (Figure 2.22). This approach although demands the model to go to a deeper level or layer in order to capture the context in full since at the top layer a cone like structure can be seen forming, which theoretically enables the token to attend to a bigger portion of the input sequence the deeper we go. The computation complexity is $\mathcal{O}(n * w)$.

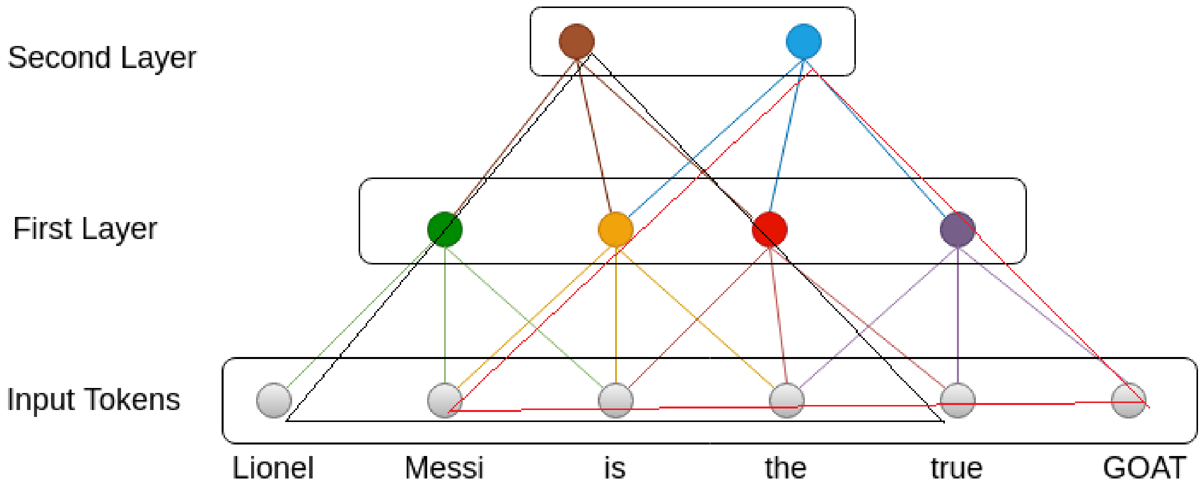


Figure 2.22: Sliding window attention in the Longformer.

- **Dilated sliding window**: The attention is applied only to every j-th token and enables the model to learn features for longer gaps. To better capture a more diversified long-term context, the authors advise utilizing various dilation heads in the multi-head self-attention.

- **Global + sliding window**: Windowed attention overcomes the complexity issue and retains local context, but it is still unable to adapt to learning task-specific representations. Some tokens may therefore attend globally in an even fashion with the longformer (a token with global attention attends to all other tokens as well as in itself). In Question Answering, global attention is supplied on all question tokens, while for classification, it is utilized for the **[CLS]** token. Even though defining global attention is task-specific, it is a simple technique to provide the model's attention inductive bias. The selection of the Global attention tokens is a hyper parameter and in most cases those tokens are special tokens that demand the global attention to properly capture the context needed for accurate results.

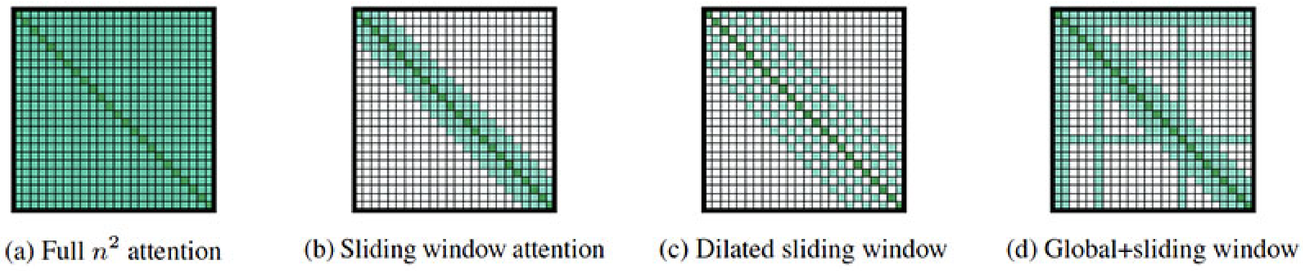

(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

Figure 2.23: Configuration of the various attention patterns as introduced by the Longformer (Beltagy et al., 2020).

Small window sizes are used for the bottom layers and are then increased for the top layers (Sukhbaatar et al., 2019), thus enabling the top layers to learn higher-level representation of the document while the lower layers capture local context. It must be noted here that dilated sliding windows are not used in the first layers in order to maximize their potential to make use of local information (Beltagy et al., 2020).

Two models were created:

- **Small model**: 12 layers, 512 hidden size, 41M parameters.
- **Large model**: 30 layers, 512 hidden sizes, 102M parameters.

The sliding window attention size used was 512, therefore using the same amount of computation as RoBERTa (Liu et al., 2019) (subsection 2.9.2). During the model evaluation, the model can run on a sequence length of 32, 256 tokens (63x more than the vanilla

BERT). It is trained with the **MLM** objective. The authors also tried freezing the weights of the RoBERTa model and continuing the training from the same checkpoint using the novel attention patterns.

***L**ongformer-**E**ncoder-**D**ecoder* **(LED)** was also proposed, a Longformer variant with both the encoder and decoder component utilizing both local(1,024 tokens) and global(first **<s>** token) attention on the encoder and full attention on the decoder. It was initialized from BART, trained using teacher forcing and evaluated on the summarization task using the arXiv summarization dataset, achieving SOTA results.

It is worth mentioning here that **LED** outperforms the BigBird (Zaheer et al., 2020) model on summarization tasks which supports sequences length of up to 4,000 tokens and is a model specifically designed and pre-trained for summarization and has Pegasus as the backbone.

2.10 Decoding Strategies

Auto-regressive language generation is based on the assumption that the probability distribution of a word sequence can be decomposed into the product of conditional next word distributions[7]:

$$P(w_{1:T}|W_0) = \prod_{t=1}^{T} P(w_t|w_{1:t-1}, W_0), \ \ with \quad w_{1:0} = \varnothing, \tag{2.11}$$

and $\boldsymbol{W_0}$ being the initial context word sequence. The length $\boldsymbol{T}$ of the word sequence is usually determined on-the-fly and corresponds to the timestep $\boldsymbol{t = T}$ the end-of-sequence token **[EOS]** is generated from $\boldsymbol{P(w_t|w_{1:t-1})}$.

Some of the most prominent decoding methods used are :

- **Greedy Search**: At each time-step $\boldsymbol{t}$, greedy search simply picks the word with the highest probability as its next entry. The following sketch shows greedy search.

[7]https://huggingface.co/blog/how-to-generate

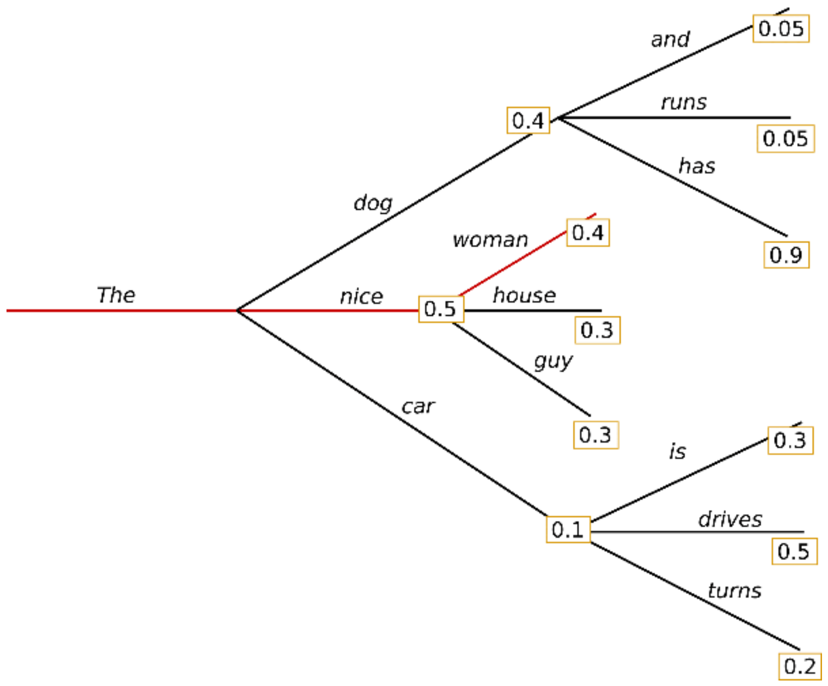


Figure 2.24: Greedy search. The algorithm greedily chooses the next word of highest probability "nice" and so on, so that the final generated word sequence is ("The", "nice", "woman") having an overall probability of $\mathbf{0.5 \times 0.4 = 0.20}$

The primary issue with greedy search is that it overlooks high probability terms that are concealed by low probability. In the previous example the sequence ("The", "dog", "has") results in a conditional probability of 0.90.

- **Beam Search**: By preserving the most plausible hypotheses at each time step and ultimately selecting the one with the highest overall probability, beam search lowers the chance of missing "hidden" high probability word sequences. While beam search will almost always discover an output sequence with a greater probability than greedy search, the most ideal output is not always found by beam search. The dynamic selection of the beam size effectively creates a trade-off between accuracy and computational expense. The term "beam search" was coined by Raj Reddy of Carnegie Mellon University in 1977.

  The beam size $\boldsymbol{k}$ which is a hyperparameter of Beam Search determines how many tokens are to be considered while performing the algorithm. At each time step $\boldsymbol{k}$ tokens with the highest conditional probabilities are to be selected and each of them along with their respective output sequences will be the candidate tokens for the next iterations. At each subsequent time step, based on the $\boldsymbol{k}$ candidate output sequences at the previous time step, we continue to select $\boldsymbol{k}$ candidate output sequences with the highest conditional probabilities.

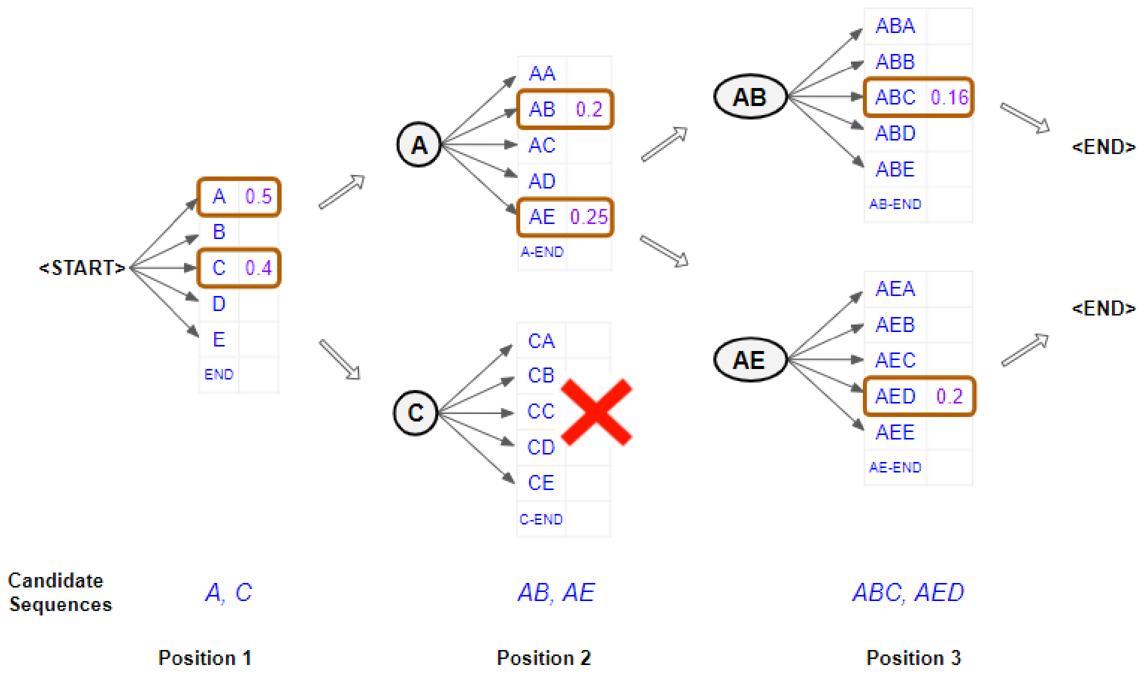


Figure 2.25: Example of performing the beam search algorithm in a sequence with number of beams $k = 2$.

- **Top-K Sampling** (Fan et al., 2018)
- **Top-p Sampling**

2.11 Evaluation

In general the two core types of evaluating summaries as proposed by (Over et al., 2007) are:

- *Intrinsic evaluation:* The immediate evaluation of the summary's quality is what is prioritized. To do this, judgments must be made on what metrics to utilize, what exactly is being measured and how to operationalize the evaluation.
- *Extrinsic evaluation:* When assessing how effectively the summary improves performance on a certain job is the main focus. A task that requires summarization is provided, and the impact of employing automated summaries in place of the original text must be assessed. This happens in order to prove that the produced summaries can help in the real-world task and even in cases where it is essential to detect differences across automatic systems producing the summaries.

It should be noted that both intrinsic and extrinsic evaluations are components of a broad spectrum of assessments, starting with basic intrinsic evaluations like how readable a summary is, moving through more task-dependent intrinsic evaluations like how much of a document's important content was covered.

2.11.1 Precision & Recall

In classification problems, we can measure the accuracy of a given model simply by dividing the total number of correct predictions by the total number of predictions made. Accuracy although is inappropriate for imbalanced classification problems as a performance measure. The main reason is that the majority class will overwhelm the number of examples in the minority class. An alternative to using accuracy is to use precision and recall metrics. Such metrics are used to calculate the ROUGE scores which can give a better indicator as to how close the reference summary is to the machine-generated one.

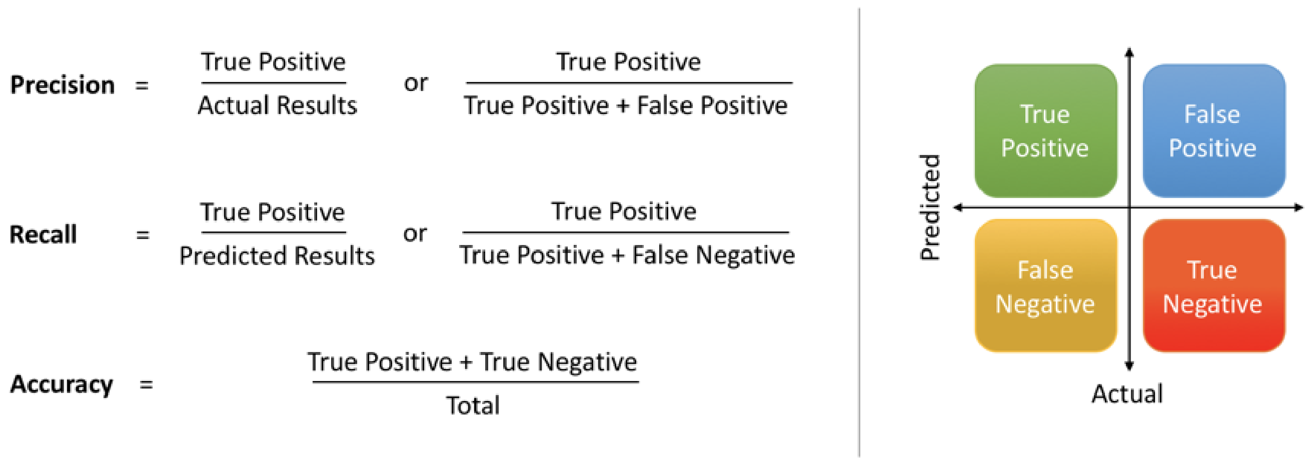

Precision = True Positive / Actual Results or True Positive / True Positive + False Positive
Recall = True Positive / Predicted Results or True Positive / True Positive + False Negative
Accuracy = True Positive + True Negative / Total
Predicted
True Positive
False Positive
False Negative
True Negative
Actual

Figure 2.26: precision-recall-accuracy

2.11.2 ROUGE

ROUGE (Lin, 2004; Liu and Liu, 2010) stands for *Recall-Oriented Understudy for Gisting Evaluation*, it is mainly used for evaluating automatic summarization of texts as well as machine translations. ROUGE is an intrinsic metric for automatically evaluating summaries and is based on BLUE (Papineni et al., 2002), even though it is not as good as human evaluation it is much more convenient. It is calculated by comparing a machine generated transcript with a set of reference summaries. The collection of reference summaries is also known as a ground-truth or gold-standard set of summaries. It includes the following 5 evaluation metrics:

1. **ROUGE-N**
2. **ROUGE-L**
3. **ROUGE-W**
4. **ROUGE-SU**
5. **ROUGE-S**

The main metrics that will be used for the purpose of this thesis are **ROUGE-1**, **ROUGE-2** and **ROUGE-L**.

- **ROUGE-N** is a recall-related measure and it measures the overlap of n-grams between the automatically generated summary and the reference summary. An n-gram is a neighbouring series of n tokens from a document. In n-grams value of N can vary from 1 to n but as the value of n increases computation cost also increase rapidly. The most common used n-grams are uni and bi-gram.

  For example given the following sentence : "The cat was sitting on the desk"

  The 1-grams or uni-grams are : [The | cat | was | sitting | on | the | desk]

  The 2-grams or bi-grams are : [The cat | cat was | was sitting | sitting on | on the | the desk] and so on.

  Given a ground-truth summary R and a generated one S containing R_N and S_N N-grams respectively, we can calculate the ROUGE-N score simply by dividing the number of overlapping n-grams by the number of n-grams in the reference summary. In similar we can calculate the recall, precision and F1-scores of ROUGE.

$$ROUGE - N_{Recall} = \frac{R_N \cap S_N}{R_N} \tag{2.12}$$

$$ROUGE - N_{Precision} = \frac{R_N \cap S_N}{S_N} \tag{2.13}$$

$$ROUGE - N_{F1-Score} = 2 * (\frac{Recall * Precision}{Recall + Precision}) \tag{2.14}$$

  If the summaries are longer the ROUGE-2 scores will be generally be smaller than the ROUGE-1 scores because there are fewer bi-grams to match which is also true for abstractive summarization, so in general both ROUGE-1 and ROUGE-2 scores are reported.

- **ROUGE-L** on the other hand doesn't consider n-grams but instead treats each summary as a sequence of words and looks for the longest common sub-sequence of words in the reference and the generated summary, using the Longest Common Sub-sequence (LCS). A sub-sequence is a sequence that appears in the same relative order but not necessarily contiguous. In comparison to ROUGE-N, ROUGE-L and employing LCS have the key benefit of not requiring consecutive matches but rather in-sequence matches that represent sentence level word order, which captures sentence structure considerably more accurately.

$$ROUGE - L_{Recall} = \frac{LCS(R_N, S_N)}{R_N} \tag{2.15}$$

$$ROUGE - L_{Precision} = \frac{LCS(R_N, S_N)}{S_N} \tag{2.16}$$

For example in the following summaries the ROUGE-L score will be calculated as follows. The recall will be 6/6 and the precision score will be 6/7.

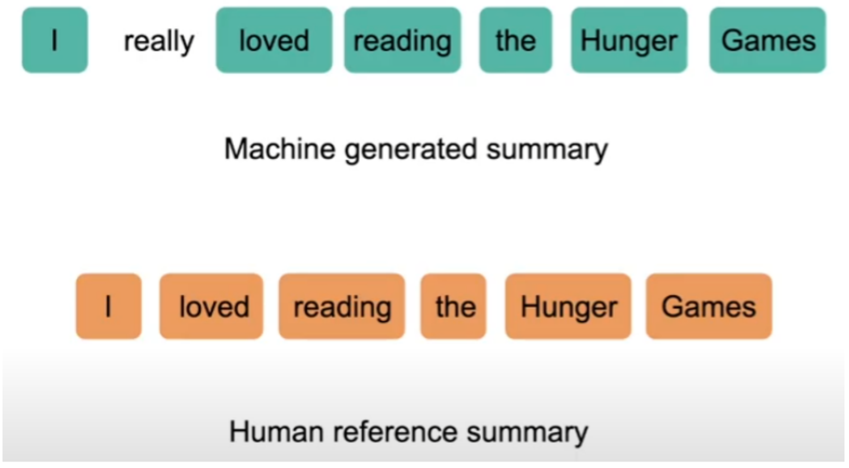


Figure 2.27: example LCS

Chapter 3

Experiments

In this chapter the experiments conducted using various transformer based models will be discussed in greater detail, and the results will be presented and analyzed. The experiments made used a great deal of the HuggingFace library and they were conducted using software and hardware from the Google Colab pro which allows anybody to write and execute arbitrary python code through the browser using hardware devices like expensive GPUs much needed for demanding transformer models.

For these experiments the main language used is Python. The core libraries used are HuggingFace, NumPy and Tensorflow. The majority of text manipulation was made using core (built-in) Python libraries, albeit the nltk library was employed sparsely. For visualization of some of the results the Matplotlib library was used. Lastly for the metrics used to evaluate the results, namely BERT-score and ROUGE-score, the datasets library was used.

The experiments made are focused on Multi-Document Summarization and in particular summarization of dialogues. These dialogues may consist of email threads where people talk with each other, or Facebook or even Twitter posts where people express their opinions in the comments. In short any platform where people may interact with each other and produce some form of communication. The word dialogue is a little bit vague, since in a Tweet post it's true that people will most likely talk about the main subject in that post and they will most likely express their opinions but it's not necessary that there will be a dialogue. In the scope of this thesis the "Multi" in the "Multi-Document" aspect can be seen therefore as multiple opinions or multiple comments, if we define a single document as a single comment in a post.

The main purpose is to build a system capable of extracting the consensus or the core ideas in a conversation between many peoples, much like opinion mining.

For this purpose 3 transformer based models were fine-tuned and tested with various datasets. And finally the best model will be selected with zero-shot settings to parse 4 distinct posts from Twitter.

3.1 Dataset

The dataset used is the ConvoSumm (Fabbri et al., 2021) dataset (section 2.2). There are multiple versions of this dataset, but for the purposes of this thesis the arg-filtered version is chosen which is a less noisy version of the original, since as per the creators of the dataset the results are more promising with this version. In particular the authors created the argument graph of each document identifying the claims and premises and the relationship between them (Lenz et al., 2020; Chesñevar et al., 2006). Later they did some prepossessing in order to extract the arguments from the main documents and to remove the non-argumentative units and they used a three-way classifier for the task of argument extraction, following (Chakrabarty et al., 2019).

ConvoSumm is a dataset which consists of 4 different parts, one can say 4 different datasets. That is because in the said dataset there exist email threads, news article comments, question answering threads and Reddit threads. The authors of ConvoSumm tested each part of the dataset in isolation but they never tested the dataset as a whole, meaning that they used BART in each part of it but never on the whole. One novel approach in this thesis is that besides testing the models on each part of the dataset, the dataset as a whole is also tested with the various models.

The majority of the samples (dialogue-summary pairs) in this dataset produce an average of 1000 tokens when tokenized, but almost 24% of the dataset produce more tokens. Considering the limitation of the transformers used there are two option:

- Discard the samples that when tokenized would produce more tokens than a threshhold.
- Use truncation to keep all the samples and consequently suffer some information loss.

Since the dataset is by no means big, it consists of a total of 2021 dialog-summary pairs, using the first approach would result in almost 500 samples being discarded. One can argue about whatever approach is the best, nevertheless the latter approach is chosen here. Also putting into perspective that some of the models used require a max of 512 tokens, since Longformer based models are not used, that would result in more than half the dataset being discarded. Another reason for this choice is hardware limitations.

3.2 DistillBART

The first model used is DistillBART trained on the Extreme Summarization (XSum) Dataset (Narayan et al., 2018) and consisting of 12 layers in the encoder component and 6 layers in the decoder. Thanks to the HuggingFace library finding and downloading the model was effortlessly.

The model was fine-tuned using the Adam optimizer with weight decay with a learning rate of 0.00005 (lr = 5e-5) and weight decay of 0.01, the same optimizer was used for the rest of the models. The maximum number of tokens allowed in this model is 1024 tokens, so the samples exceeding that threshold were truncated to the maximum of 1024.

The model was fine-tuned and tested on each of the four subsets of the ConvoSumm dataset. In addition the model was trained on the whole ConvoSumm and then tested on each subset separately. The splits chosen for this process are 70/20/10 for fine-tuning/validating and testing.

For generating the summaries beam search was used with beam size of 4, the minimum length of summaries was set to 100 words and the maximum to 125.

- E-mails subset

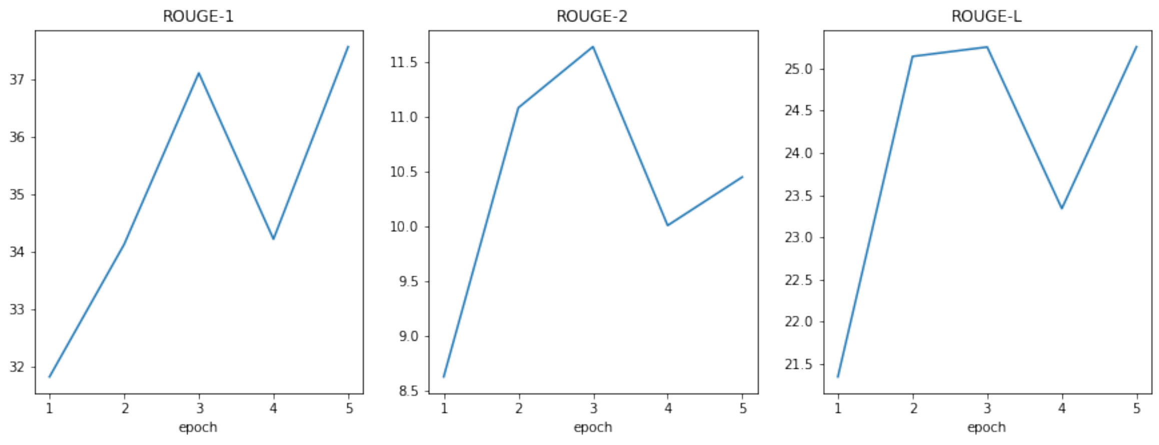


Figure 3.1: ROUGE scores during validation of the E-mails subset.

- Nyt subset

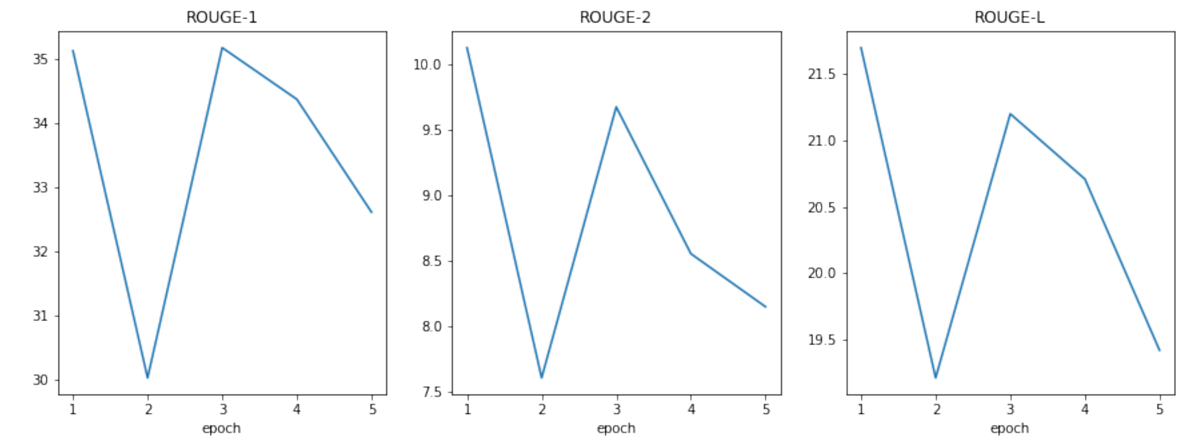


Figure 3.2: ROUGE scores during validation of the Nyt subset.

- Reddit subset

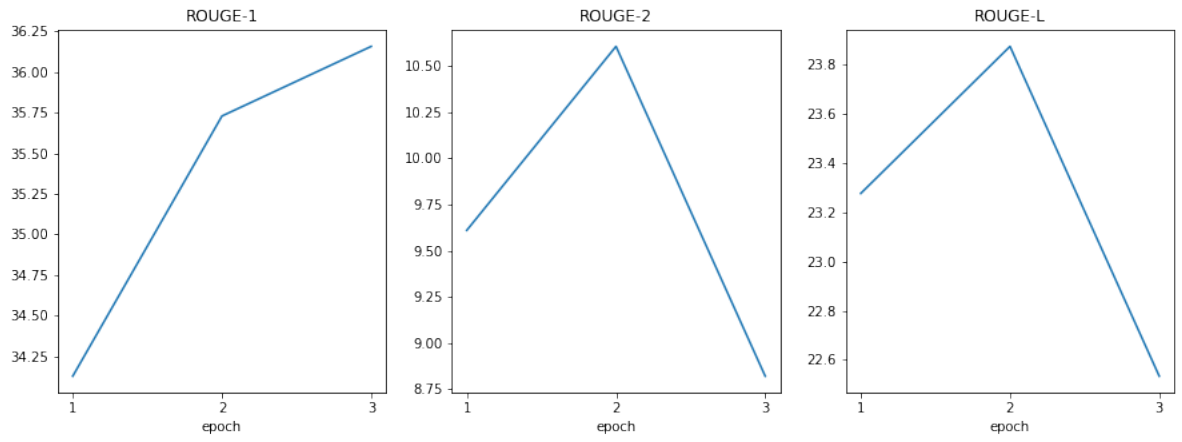


Figure 3.3: ROUGE scores during validation of the Reddit subset.

- Stack subset

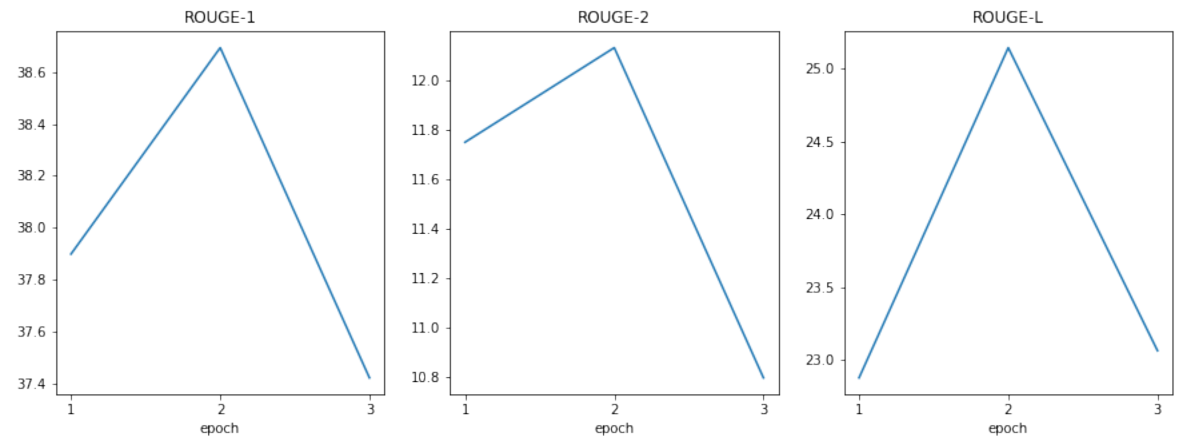


Figure 3.4: ROUGE scores during validation of the Stack Exchange subset.

- On the whole ConvoSumm dataset using 512 tokens as input

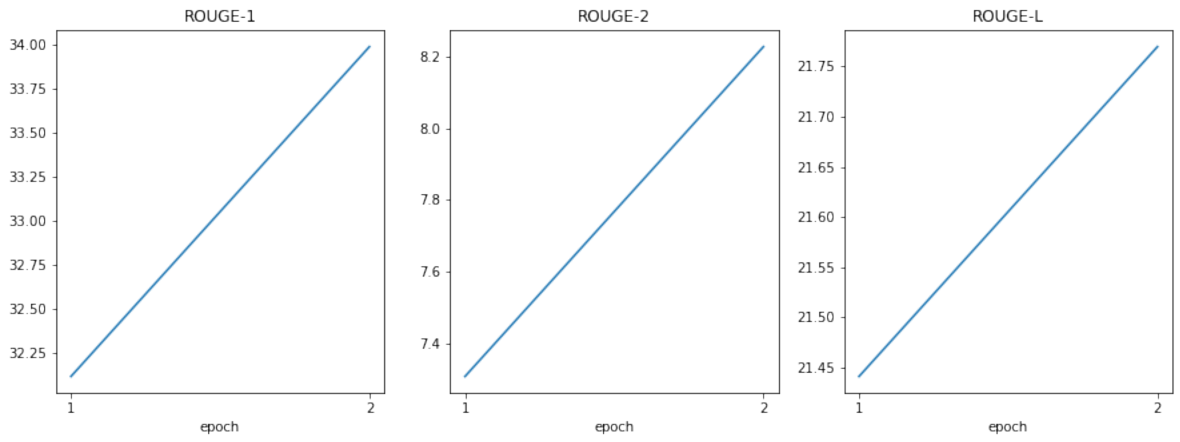


Figure 3.5: ROUGE scores during validation on the whole ConvoSumm dataset with 512 tokens.

- On the whole ConvoSumm using full 1024 tokens as input

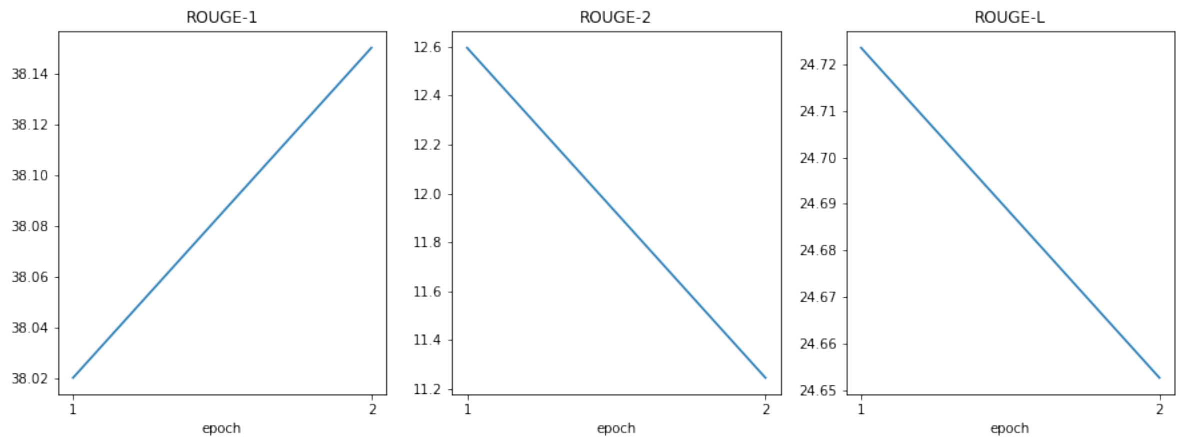


Figure 3.6: ROUGE scores during validation on the whole ConvoSumm dataset.

As can be seen in (Figure 3.14) the DistillBART model was fine-tuned for various epochs using a batch size of 4 and after each epoch a portion of the validation dataset was used to calculate the ROUGE scores as can be seen in the previous graphs, providing us with an idea of how effectively the model is learning. All the weights were saved, but for the final selection the weights from the epoch which had the lowest validation loss were chosen. Finally one more model was trained on the whole ConvoSumm using 512 tokens instead of 1024 and thus using truncation even more.

3.3 DistillPEGASUS

The second model used is Distill PEGASUS trained on the Extreme Summarization (XSum) dataset consisting of 16 layers in the encoder component and 8 layers in the decoder. For this model the maximum allowed tokens were used which are 512 tokens.

In the following images the ROUGE scores during validation are presented.

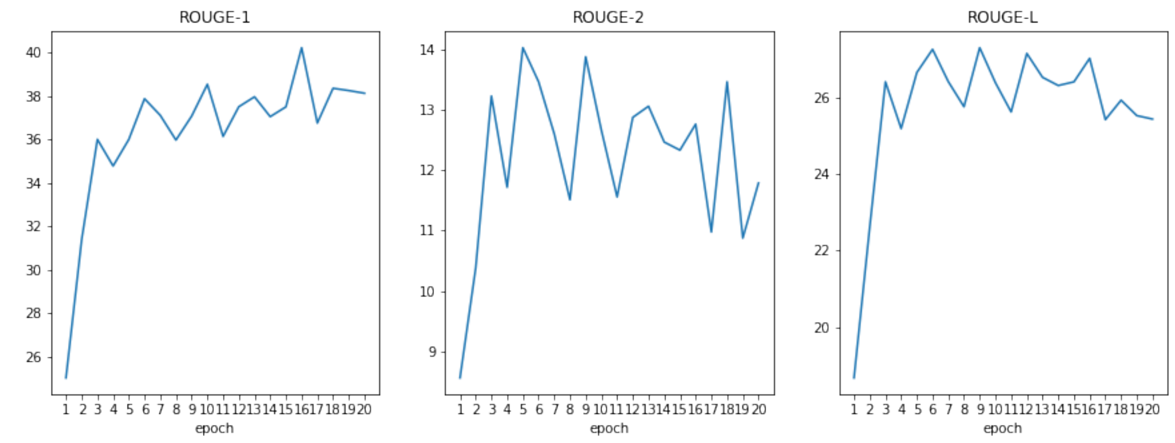


Figure 3.7: ROUGE scores during validation of the E-mails subset.

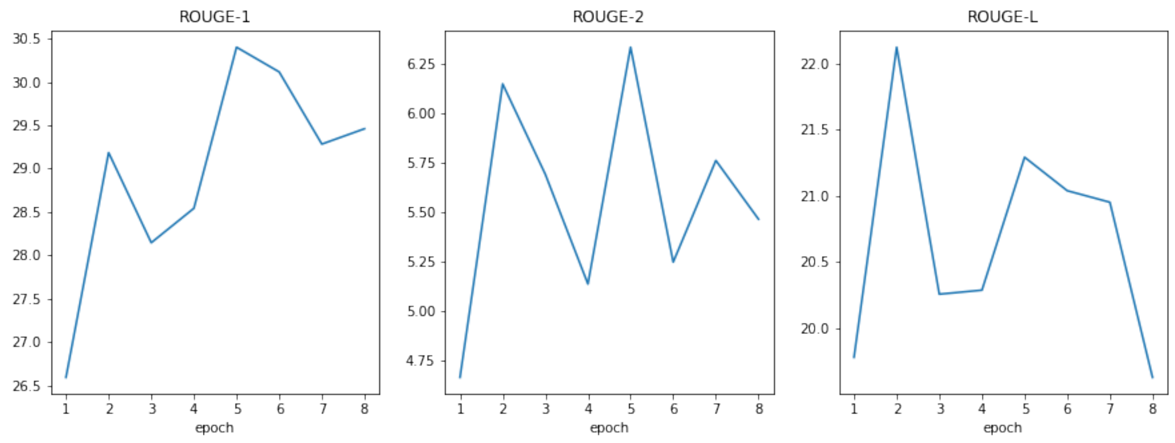


Figure 3.8: ROUGE scores during validation of the Nyt subset.

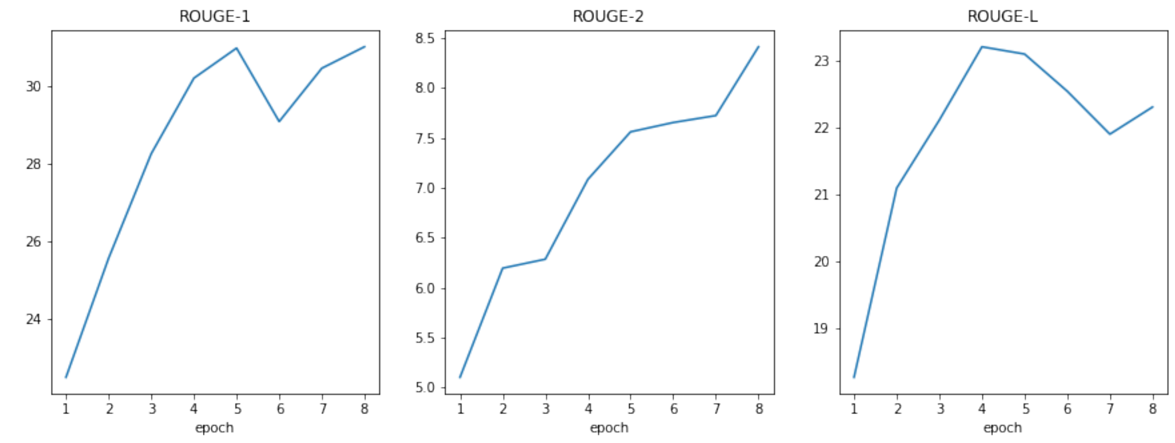


Figure 3.9: ROUGE scores during validation of the Reddit subset.

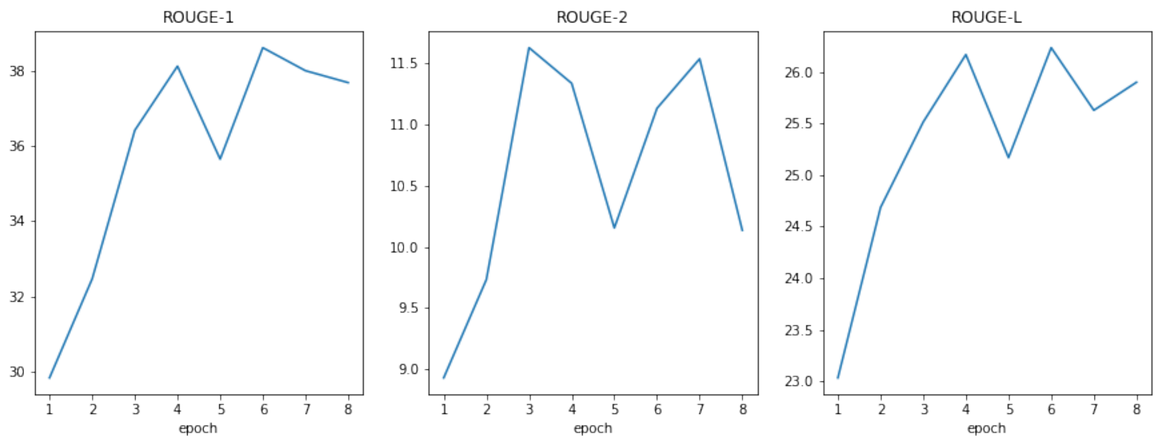


Figure 3.10: ROUGE scores during validation of the Stack Exchange subset.

3.4 $T5_{Base}$

The last model fine-tuned is $\boldsymbol{T5_{Base}}$ which used 512 tokens as inputs.

In the following images the ROUGE scores during validation are presented.

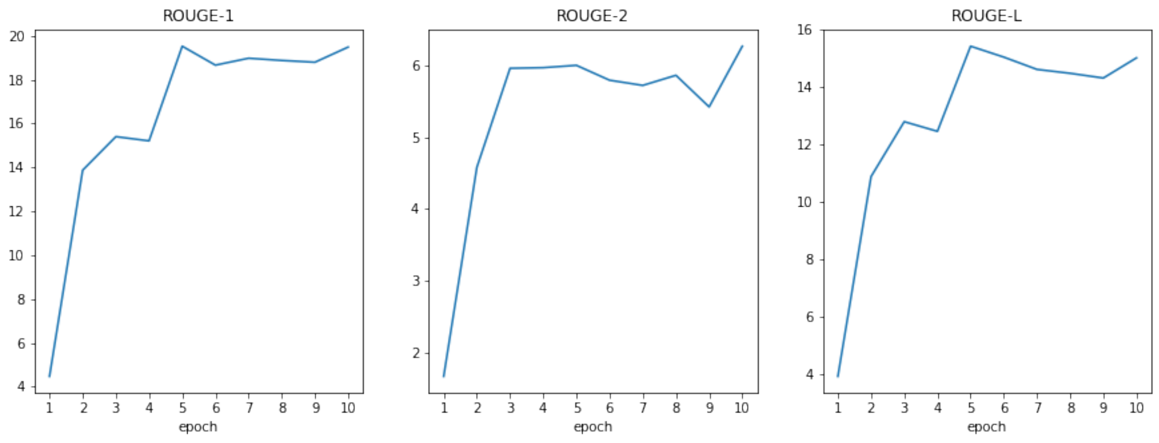


Figure 3.11: ROUGE scores during validation of the E-mails subset.

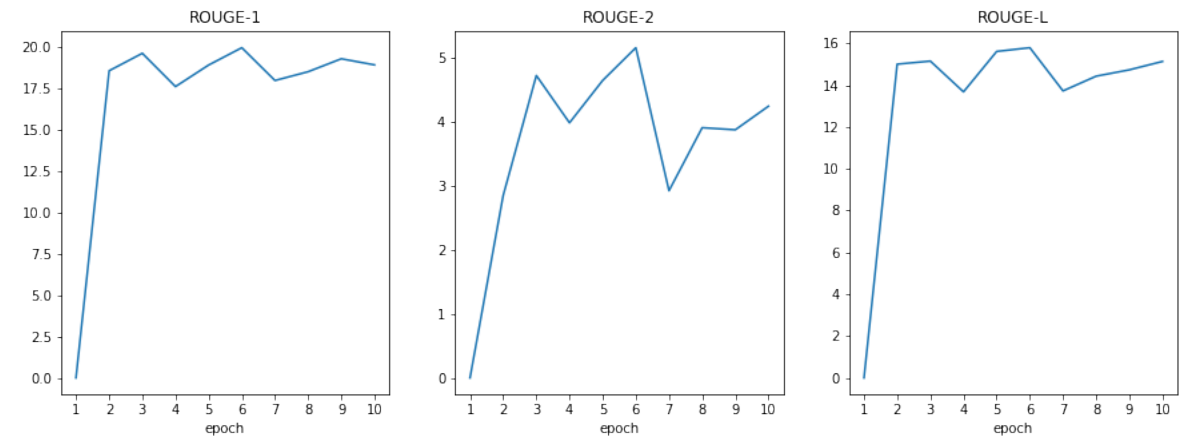


Figure 3.12: ROUGE scores during validation of the Nyt subset.

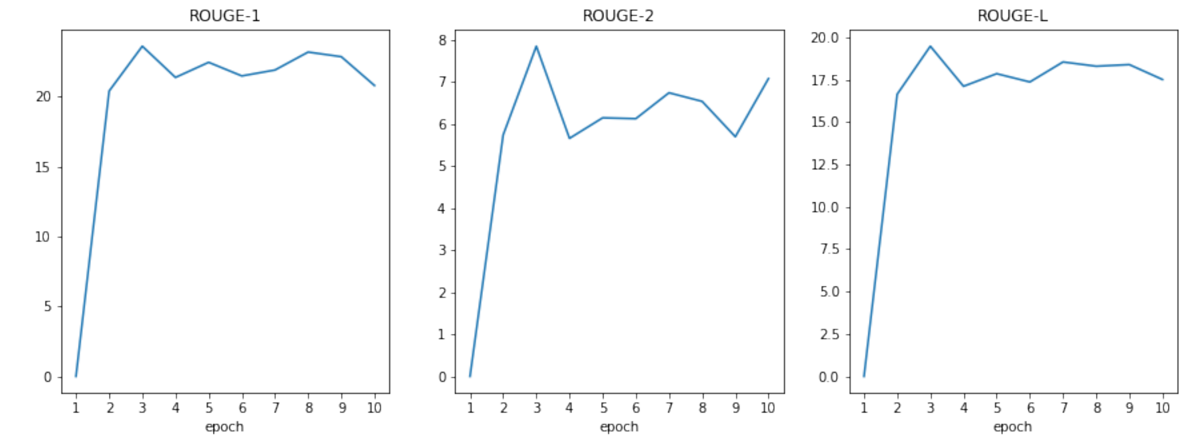


Figure 3.13: ROUGE scores during validation of the Reddit subset.

3.5 Results

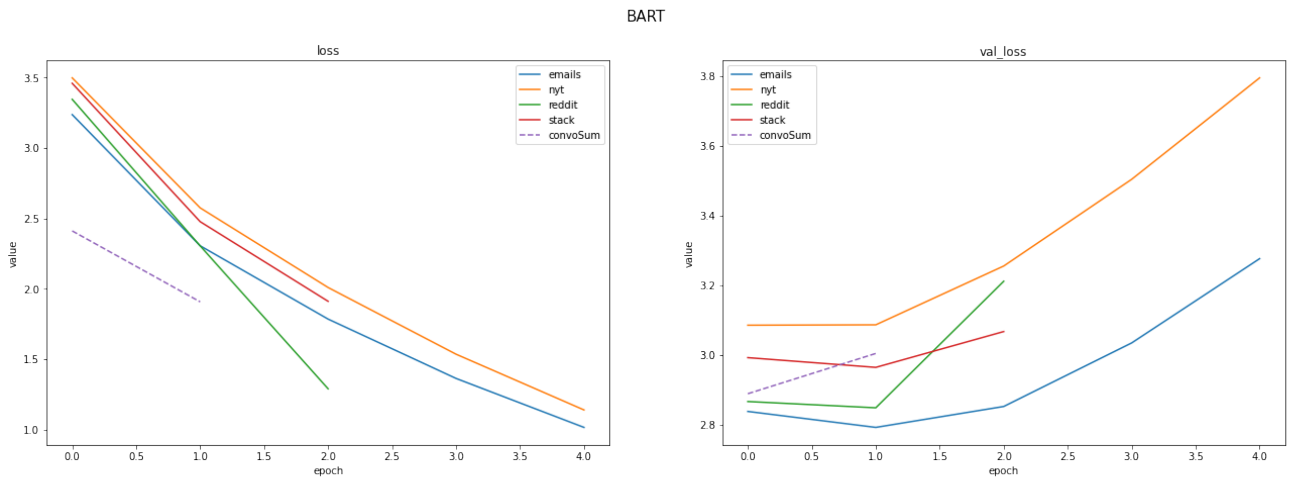


Figure 3.14: Loss and validation loss while fine-tuning DistillBART.

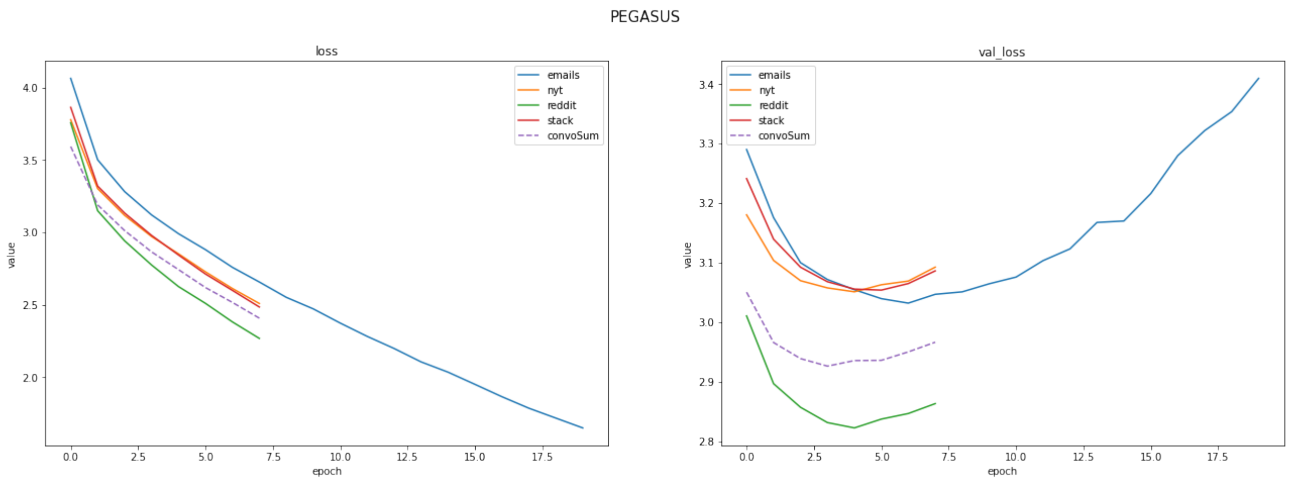


Figure 3.15: Loss and validation loss while fine-tuning DistillPEGASUS.

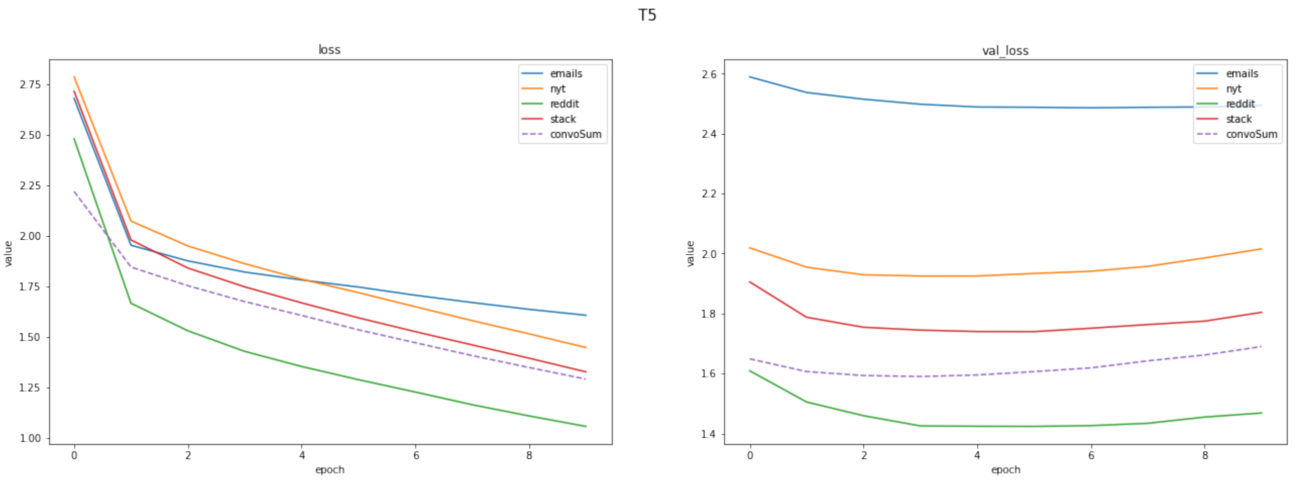


Figure 3.16: Loss and validation loss while fine-tuning $\boldsymbol{T5_{Base}}$.

		ROUGE Scores				BERT Scores		
		R-1	R-2	R-L	R-Lsum*	F1	Prec	Rec
E-mails	$BART - solo$	38.96	12.2	**25.99**	35.35	87.47	**87.7**	87.25
	$BART - full$	40.01	12.11	24.96	**36.96**	**87.68**	87.47	**87.89**
	$BART - 512 - full$	**40.49**	**13.15**	25.26	36.64	87.56	87.25	87.89
	$PEGASUS - solo$	34.28	9.52	24.09	30.9	86.56	86.16	86.97
	$PEGASUS - full$	32.01	9.52	23.25	28.02	86.18	85.84	86.55
	$T5_{Base} - solo$	38.92	12.52	25.55	34.81	87.22	86.88	87.58
	$T5_{Base} - full$	36.53	11.37	24.68	33.08	86.67	86.15	87.21
NYT	$BART - solo$	35.98	9.33	22.11	32.16	87.83	88.03	87.63
	$BART - full$	**37.68**	**10.43**	**22.14**	**33.93**	**88.11**	**88.05**	**88.18**
	$BART - 512 - full$	35.3	7.96	20.61	31.28	87.62	87.49	87.77
	$PEGASUS - solo$	24.62	5.27	18.69	20.63	85.35	84.81	85.91
	$PEGASUS - full$	26.32	5.71	19.69	22.09	85.77	85.4	86.15
	$T5_{Base} - solo$	31.14	6.94	20.76	27.84	86.81	86.47	87.15
	$T5_{Base} - full$	28.19	6.36	19.77	24.99	86.16	85.95	86.37
Reddit	$BART - solo$	35.49	**10.77**	22.02	32.39	87.77	87.44	88.1
	$BART - full$	**35.85**	10.17	**22.16**	**32.65**	**87.84**	**87.55**	**88.14**
	$BART - 512 - full$	34.84	9.64	20.9	31.31	87.5	87.03	87.97
	$PEGASUS - solo$	25.57	6.77	19.88	21.82	85.81	85.21	86.44
	$PEGASUS - full$	24.51	5.96	19.8	20.7	85.33	84.62	86.08
	$T5_{Base} - solo$	31.16	7.91	21.78	27.7	87.02	86.8	87.25
	$T5_{Base} - full$	27.38	6.21	20.11	23.93	86.25	85.85	86.66
Stack	$BART - solo$	**39.06**	11.46	**23.62**	**34.75**	87.92	87.78	**88.06**
	$BART - full$	38.01	**11.55**	23.61	33.91	**87.98**	**88.02**	87.94
	$BART - 512 - full$	37.76	10.76	22.49	33.6	87.62	87.49	87.76
	$PEGASUS - solo$	29.51	7.01	22.12	24.91	85.96	85.45	86.5
	$PEGASUS - full$	29.68	8.21	22.29	25.43	86.13	85.88	86.4
	$T5_{Base} - solo$	34.96	9.93	22.53	31.12	87.38	87.07	87.7
	$T5_{Base} - full$	33.23	9.78	23.49	28.18	86.7	86.58	86.83

*(The "sum" in rougeLsum refers to the fact that this metric is computed over a whole summary, while rougeL is computed as the average over individual sentences.)

**(The *solo* and the $full$ keywords after the models refer to the fact that in the case of *solo* the model was only fine-tuned and tested on a specific subset of the ConvoSumm dataset whereas the $full$ keyword indicates that the same model was fine-tuned on the whole ConvoSumm dataset but was only tested on the particular subset of ConvoSumm.)

Table 3.1: Results of all models.

The best model overall seems to be $DistillBART full$ (Table 3.1). It is quite interesting that in many cases the models that are trained on the whole dataset are over performing compared to when they are specialized in the specific part of the dataset.

3.6 Selected Models

The epochs during which the fine-tuning of the models had the lowest validation loss and thus were selected to perform the experiments are the following:

- E-mails subset:
  - $DistillBART$ epoch 2
  - $DistillPEGASUS$ epoch 7
  - $T5_{Base}$ epoch 8
- Nyt subset:
  - $DistillBART$ epoch 1
  - $DistillPEGASUS$ epoch 5
  - $T5_{Base}$ epoch 6
- Reddit subset:
  - $DistillBART$ epoch 2
  - $DistillPEGASUS$ epoch 8
  - $T5_{Base}$ epoch 6
- Stack subset:
  - $DistillBART$ epoch 2
  - $DistillPEGASUS$ epoch 6
  - $T5_{Base}$ epoch 6
- ConvoSumm:
  - $DistillBART$ epoch 1
  - $DistillPEGASUS$ epoch 8
  - $T5_{Base}$ epoch 4

It must be noted here that in rare occurrences where the validation loss was marginally worst the latter epochs were chosen and they produced slightly better ROUGE scores.

3.7 Testing (zero-shot) dataset

The best model ($DistillBARTfull$) trained on the whole ConvoSumm dataset is thus selected and is tested with zero shot settings on 4 distinct proprietary twitter posts and corresponding gold summaries. The resulting summaries along with the gold summaries are presented in Table 4.5. Some of the best generated summaries are presented in Table 4.1 and Table 4.2 and some of the worst summaries can be found in Table 4.3 and Table 4.4.

Since the Twitter posts provided greatly outscale the 1024 tokens, in particular the larger one produces almost 137,000 tokens in total, the tokens are split into equal intervals and then are passed into the model.

A divide and conquer method is used where the posts are split into chunks that do not produce more than 1024 tokens and the summaries are generated from those chunks which in turn go into a recursive function until the desired length of 100-125 words is reached. Since each comment is treated as a document, when the splits occur they occur in such a way that no one solid document will be split in half, instead in that case that document will go into the next chunk thus preserving some coherence, albeit far from perfect.

One more method is used where the Twitter posts are passed through the best model using only truncation but is not explored further since the better approach is used which is the divide and conquer approach. (Table 3.2).

Table 3.2: Twitter post summarized

GOLD	TRUNCATION
The easy access of your office can lead to just keep working and also to not taking breaks at all. Many remote workers indicate the unsatisfied need for communication, so if you are very sociable, you might find yourself in trouble too. If you have small children at home with you then you will not be able to focus on your work because of the many distractions and issues associated with them. One big challenge is internet connectivity, it is very confusing when this problem occurs only to you. Working from home can help you to save time for other things in your personal life.	Commenters agree that working from home is a difficult task and that it can be difficult to keep up with the team. One commenter says that the team is not working well enough and that they need to be more connected to each other. Another commenter says it is difficult to stay motivated when working remotely. Another says that it is very difficult to focus on the job and that there are many issues that make it difficult for remote workers to complete tasks and that the company is not able to.

Chapter 4

Conclusion and final thoughts

In this thesis the transformer model was presented and explained along with its variations. Metrics for evaluation of text similarity were established and later employed to assess the results of the models. The topic of multi document summarization in the form of comment pools was tackled. Multiple datasets of interest were introduced. Even thought most of the datasets are relatively close to what was needed for the specific purpose of summarizing comment pools, they are not quite there yet. Better datasets without a doubt are much needed to produce better training or fine tuning grounds for the models already available.

Various experiments were carried out and satisfactory results were achieved, which only pointed out the need for further tests. Out of the many models investigated, BART seemed to be the best suited for the current dataset. Playing around further with the plethora of hyper parameters may yield even more adequate results. In particular during the generation of the summaries by the model, the beam size and the min/max length of the summaries was not tested extensively, a process which required a lot of compute power and which is quite lengthy indeed. In that note using a Longformer model may have procured better results.

In the final experiments the model was tested on zero shot samples and produced acceptable results. The grammar and the syntax produced by the model are not suffering, yet the meaning may not be optimal.

It may be the case that ”going larger” will help. Building bigger models with trillions of parameters will help, but no one can say with certainty that that's the sole reason of improvement. In the latest experiment a simple divide and conquer algorithm was used to generate the summaries and produced acceptable results.

In conclusion, the field of NLP and ML surrounding the topic of multi document summarization is quite vast and interesting and much more research and experiments are required, as there is always room for improvement.

Bibliography

[1] A. F. Agarap. Deep learning using rectified linear units (relu), 2018. URL https://arxiv.org/abs/1803.08375.

[2] N. Agarwal. The ultimate guide to different word embedding techniques in nlp, 2021.

[3] J. Alammar. The illustrated bert, elmo, and co. (how nlp cracked transfer learning), 2021. URL https://jalammar.github.io/illustrated-bert/.

[4] F. Almeida and G. Xexéo. Word embeddings: A survey, 2019. URL https://arxiv.org/abs/1901.09069.

[5] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL https://arxiv.org/abs/1409.0473.

[7] A. Balahur, M. Kabadjov, J. Steinberger, R. Steinberger, and A. Montoyo. Challenges and solutions in the opinion summarization of user-generated content. *Journal of Intelligent Information Systems*, 39, 10 2012. doi: 10.1007/s10844-011-0194-z.

[8] E. Barker and R. Gaizauskas. Summarizing multi-party argumentative conversations in reader comment on news. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 12–20, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2802. URL https://aclanthology.org/W16-2802.

[9] E. Barker, M. L. Paramita, A. Aker, E. Kurtic, M. Hepple, and R. Gaizauskas. The SENSEI annotated corpus: Human summaries of reader comment conversations in on-line news. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 42–52, Los Angeles, Sept. 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-3605. URL https://aclanthology.org/W16-3605.

[10] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003. ISSN 1532-4435.

[12] I. S. Blekanov, N. Tarasov, and S. S. Bodrunova. Transformer-based abstractive summarization for reddit and twitter: Single posts vs. comment pools in three languages. *Future Internet*, 14(3), 2022. ISSN 1999-5903. doi: 10.3390/fi14030069. URL https://www.mdpi.com/1999-5903/14/3/69.

[13] J. Brownlee. What is teacher forcing for recurrent neural networks?, April 8, 2021. URL https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/.

[14] C. Bucilua, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL https://doi.org/10.1145/1150402.1150464.

[15] T. Chakrabarty, C. Hidey, S. Muresan, K. McKeown, and A. Hwang. AMPERSAND: Argument mining for PERSuAsive oNline discussions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2933–2943, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1291. URL https://aclanthology.org/D19-1291.

[16] C. Chesñevar, J. Mcginnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott. Towards an argument interchange format. *Knowledge Eng. Review*, 21:293–316, 12 2006. doi: 10.1017/S0269888906001044.

[17] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

[18] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

[19] S. B. R. Chowdhury, F. Brahman, and S. Chaturvedi. Is everything in order? a simple way to order sentences, 2021. URL https://arxiv.org/abs/2104.07064.

[20] T. Chowdhury and T. Chakraborty. Cqasumm: Building references for community question answering summarization corpora, 2018. URL https://arxiv.org/abs/1811.04884.

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

[22] N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the trec 2005 enterprise track. In *TREC*, 2005.

[23] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019. URL https://arxiv.org/abs/1901.02860.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

[25] M. Dragoni, C. da Costa Pereira, A. Tettamanzi, and S. Villata. Combining argumentation and aspect-based opinion mining: The smack system1. *AI Communications*, 31:1–21, 01 2018. doi: 10.3233/AIC-180752.

[26] S. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38:188–230, 01 2004. doi: 10.1002/aris.1440380105.

[27] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 0364-0213. doi: https://doi.org/10.1016/0364-0213(90)90002-E. URL https://www.sciencedirect.com/science/article/pii/036402139090002E.

[28] A. R. Fabbri, F. Rahman, I. Rizvi, B. Wang, H. Li, Y. Mehdad, and D. Radev. Convosumm: Conversation summarization benchmark and improved abstractive summarization with argument mining, 2021. URL https://arxiv.org/abs/2106.00829.

[29] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation, 2018. URL https://arxiv.org/abs/1805.04833.

[30] L. Felciah and R. Anbuselvi. A study on sentiment analysis of social media reviews. In *A study on sentiment analysis of social media reviews*, pages 1–3, 03 2015. doi: 10.1109/ICIIECS.2015.7192949.

[31] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning, 2017. URL https://arxiv.org/abs/1705.03122.

[32] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL https://aclanthology.org/D19-5409.

[33] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, mar 2021. doi: 10.1007/s11263-021-01453-z. URL https://doi.org/10.1007%2Fs11263-021-01453-z.

[34] I. Habernal, J. Eckle-Kohler, and I. Gurevych. Argumentation mining on the web from information seeking perspective. In *ArgNLP*, 2014.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[36] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus), 2016. URL https://arxiv.org/abs/1606.08415.

[37] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

[38] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[39] D. Hoogeveen, K. M. Verspoor, and T. Baldwin. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340403. doi: 10.1145/2838931.2838934. URL https://doi.org/10.1145/2838931.2838934.

[40] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music transformer, 2018. URL https://arxiv.org/abs/1809.04281.

[41] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003 ieee international conference on acoustics, speech, and signal processing: Proceedings: April 6-10, 2003, hong kong exhibition and convention centre, hong kong, 2003. URL https://arxiv.org/abs/1811.04884.

[42] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. Spanbert: Improving pre-training by representing and predicting spans, 2019. URL https://arxiv.org/abs/1907.10529.

[43] A. Kazemnejad. Transformer architecture: The positional encoding. *kazemnejad.com*, 2019. URL https://kazemnejad.com/blog/transformer_architecture_positional_encoding/.

[44] N. S. Keskar, B. McCann, C. Xiong, and R. Socher. Unifying question answering, text classification, and regression via span extraction, 2019. URL https://arxiv.org/abs/1904.09286.

[45] B. Kim, H. Kim, and G. Kim. Abstractive summarization of reddit posts with multi-level memory networks, 2018. URL https://arxiv.org/abs/1811.00783.

[46] V. Kolhatkar and M. Taboada. Using New York Times picks to identify constructive comments. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 100–105, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4218. URL https://aclanthology.org/W17-4218.

[47] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. doi: https://doi.org/10.1002/aic.690370209. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

[48] T. Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates, 2018. URL https://arxiv.org/abs/1804.10959.

[49] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks, 2016. URL https://arxiv.org/abs/1610.09038.

[50] M. Lenz, P. Sahitaj, S. Kallenberg, C. Coors, L. Dumani, R. Schenkel, and R. Bergmann. Towards an argument mining pipeline transforming texts to argument graphs. *ArXiv*, abs/2006.04562, 2020.

[51] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019. URL https://arxiv.org/abs/1910.13461.

[52] X. Liao, Y. Huang, J. Wei, Z. Yu, and G. Chen. A heterogeneous graph model for social opinion detection. In *A Heterogeneous Graph Model for Social Opinion Detection*, volume 481, pages 175–185, 07 2014. ISBN 978-3-662-45651-4. doi: 10.1007/978-3-662-45652-1_19.

[53] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[54] M. Lippi and P. Torroni. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2), mar 2016. ISSN 1533-5399. doi: 10.1145/2850417. URL https://doi.org/10.1145/2850417.

[55] B. Liu and L. Zhang. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, 2012.

[56] F. Liu and Y. Liu. Exploring correlation between rouge and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1): 187–196, 2010. doi: 10.1109/TASL.2009.2025096.

[57] Y. Liu. Fine-tune bert for extractive summarization, 2019. URL https://arxiv.org/abs/1903.10318.

[58] Y. Liu and M. Lapata. Text summarization with pretrained encoders, 2019. URL https://arxiv.org/abs/1908.08345.

[59] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

[60] E. Lloret, E. Boldrini, T. Vodolazova, P. Martínez-Barco, R. Muñoz, and M. Palomar. A novel concept-level approach for ultra-concise opinion summarization. *Expert Syst. Appl.*, 42:7148–7156, 2015.

[61] B. McCann, N. S. Keskar, C. Xiong, and R. Socher. The natural language decathlon: Multitask learning as question answering, 2018. URL https://arxiv.org/abs/1806.08730.

[62] I. Mccowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska Masson, W. Post, D. Reidsma, and P. Wellner. The ami meeting corpus. *Int'l. Conf. on Methods and Techniques in Behavioral Research*, 01 2005.

[63] T. Mikolov, I. Sutskever, A. Deoras, H. S. Le, S. Kombrink, and J. H. Cernocký. Subword language modeling with neural networks, 2011.

[64] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

[65] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[66] M. E. Moussa, E. H. Mohamed, and M. H. Haggag. A survey on opinion summarization techniques for social media. *Future Computing and Informatics Journal*, 3(1): 82–109, 2018. ISSN 2314-7288. doi: https://doi.org/10.1016/j.fcij.2017.12.002. URL https://www.sciencedirect.com/science/article/pii/S2314728817300582.

[67] S. Narayan, S. B. Cohen, and M. Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206.

[68] P. Over, H. Dang, and D. Harman. Duc in context, 2007-11-21 2007. URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=50955.

[69] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for*

Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

[70] A. Peldszus and M. Stede. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, January 2013. URL https://ideas.repec.org/a/igg/jcini0/v7y2013i1p1-31.html.

[71] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

[72] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations, 2018. URL https://arxiv.org/abs/1802.05365.

[73] D. R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408, dec 2002. ISSN 0891-2017. doi: 10.1162/089120102762671927. URL https://doi.org/10.1162/089120102762671927.

[74] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

[75] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019. URL https://arxiv.org/abs/1910.10683.

[76] I. Rahwan and G. R. Simari. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 0387981969.

[77] L. Robinet. Autoencoders and the denoising feature: From theory to practice, Nov 26, 2020. URL https://lilianweng.github.io/posts/2018-08-12-vae/.

[78] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. URL https://arxiv.org/abs/1910.01108.

[79] M. Schuster and K. Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

[80] M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

[81] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

[82] G. Shang, W. Ding, Z. Zhang, A. Tixier, P. Meladianos, M. Vazirgiannis, and J.-P. Lorré. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1062. URL https://aclanthology.org/P18-1062.

[83] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations, 2018. URL https://arxiv.org/abs/1803.02155.

[84] S. Sotudeh, H. Deilamsalehy, F. Dernoncourt, and N. Goharian. TLDR9+: A large scale resource for extreme summarization of social media posts. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 142–151, Online and in Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.newsum-1.15. URL https://aclanthology.org/2021.newsum-1.15.

[85] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin. Adaptive attention span in transformers, 2019. URL https://arxiv.org/abs/1905.07799.

[86] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014. URL https://arxiv.org/abs/1409.3215.

[87] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

[88] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *DAE*, pages 1096–1103, 01 2008. doi: 10.1145/1390156.1390294.

[89] M. Völske, M. Potthast, S. Syed, and B. Stein. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4508. URL https://aclanthology.org/W17-4508.

[90] J. J. Webster and C. Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*, COLING '92, page 1106–1110, USA, 1992. Association for Computational Linguistics. doi: 10.3115/992424.992434. URL https://doi.org/10.3115/992424.992434.

[91] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

[92] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, □. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016. URL https://arxiv.org/abs/1609.08144.

[93] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[94] A. Zhang, B. Culbertson, and P. Paritosh. Characterizing online discussion using coarse discourse sequences. In *Characterizing Online Discussion Using Coarse Discourse Sequences*, 2017.

[95] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2019. URL https://arxiv.org/abs/1912.08777.

[96] L. Zhao, Z. Yang, W. Xu, S. Gao, and J. Guo. Improving abstractive dialogue summarization with conversational structure and factual knowledge, 2021. URL https://openreview.net/forum?id=uFk038O5wZ.

[97] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015. doi: 10.1109/ICCV.2015.11.

[98] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning, 2019. URL https://arxiv.org/abs/1911.02685.

Table 4.1: E-mails subset with *DistillBARTfull* - best.

Gold summaries	**Generated summaries**
Scott Luebking asks if he should give a demo of some of the keyboard navigation during one of the conference calls. Scott Luebking then tells Denis that it would be easy to set up the demo and says that it might be interesting to have Raman's Emacs-speak do the demo for fifteen minutes then have five minutes for questions. Then he tells Jon that an advantage of using telnet is that you do not have to have it installed on each computer and says that he isn't sending the password.	Scott Luebking asks if he can give a demo of some of the keyboard navigation in his prototype browser. He replies that it's easy to set up, and that he can't send out the password. He then asks if people can try it outside of the telecon. He asks if someone can use EmacSpeak for 15 minutes to demonstrate the functionality of the system. He also asks if the user can access the browser via telnet, and he replies that he'll post a demo.
Andre asks if DOM IDL definitions can be mapped to C++ abstract classes and wants a way provided if there isn't. Joe replies that some have existed, but people can't agree on behavior and the UTF-16 version to use, then tells him to check existing C++ implementations. Joe explains to Michael why DOMString type needs to be defined so that the language can have a binding.	Andre wants to know if C++ DOM applications are conformal to the DOM standard. Joe replies that there have been several DOM bindings for C++, but that they are not clear on whether to make them behave like Java and Javascript bindings or like a more traditional C/C++ design. Michael asks why DOMString type needs to be defined. Joe says that DOMString should be defined to say what type DOMString is mapped to and that it will run against all instances of DOMString.
John proposes to add two required provisions to checkpoint 6.4 and details his proposal. Hansen agrees with the changes and notes that one part is a bit vague. Richard wants info about an object's bounding rectangle and active elements' bounding rectangles for magnification purposes. Ian notes that Richard's proposed changes are not needed. Richard replies that even an event notification would benefit from the bounding rectangle of the element.	Jon wants to know if he agrees or disagrees with the new checkpoint requirements for checkpoint 6.4. Richard replies that the user agent should also provide access to information about graphical objects. He then replies that he thinks that the new requirement is a good idea. He also likes the change and says that it's possible to get the best possible version of the document. He adds that the API should be able to access both structured information and pixel-level information. Eric likes the changes.
Charles can't get Amaya to work on Win98. Irene tells him to wait for the upcoming Amaya 2.2 release next week, which will include a Win98 version. Paul laughs that every version has memory problems and that they are at least consistent. Irene notes that it was a typo and that the version fixes said memory problems.	Coder tries to get Amaya to work on Win98 and Irene replies that they are preparing a new release for the next week. Paul replies that Amaya will work on Windows 98 and that it's a memory problem. Irene tells Coder to be patient and that they're preparing for a release next week, and Paul thanks Irene for the reply and says that they can't fix the problem. Some of the users get their private email.
Julian wants to hide a "skip links" option for sighted users, but make sure that screen readers can find the option. Simon suggests making the link the same color as the background. He also notes that an arrow has been used in the past, as some users with good vision also use that feature. Jon replies that display:none won't let screen readers have access to the element. Phill suggests putting styling on the image with a tool tip that appears on tab or mouseover.	Julian wants to hide a hidden link on a page and wants to make it visible to blind and visually-impaired users. Phillip replies that the link skipper is not just for visually impaired users, but for people with screenreaders. Simon says that it's possible to add the link to the background to hide it from blind people. thenthenthen. Julian replies that he wants to add an arrow to skip to the main content to make sure screen readers can see it.

Table 4.2: E-mails subset with *DistillBARTfull* - best.

Gold summaries	**Generated summaries**
Loretta laments that she won't be able to make it to the weekly meeting. Maurizio likewise says that he won't be able to attend. Gian also can't make it to the meeting. Eugenia replies that he can't make it to the call today.	Loretta Guarino Reid says that she won't be able to attend a meeting at Beacon Technologies. Maurizio Vittoria and Eugenia Slaydon say that they will not be attending this week's meeting. Gian says that he was unable to make the call because of a lack of time. He then states that he'll not be available to attend the meeting. He also says that they're not able to call the call today.
Judy gives a draft for a reference card that's business-card sized. Daniel thinks that the intro and close are too long, then gives some specific tips on how to do it, such as rewording a phrase and removing redundant words and sentences.	Judy wants to know how to make a website accessible for people with disabilities. Daniel replies that the intro and closing paragraphs are too long and that they need to be shortened. He replies that millions of people have access to the site. He says that millions more people will benefit from having an accessible Web Site. He gives some tips to make it work better for those with disabilities, such as mobile devices and a lack of audio output. He then says that people with mobile devices can't use it.
Patrick Stickler requests that those who support a new datatyped literal node to provide the technical justification for why URIs cannot be used. He believes that they must demonstrate why URIrefs are not appropriate. In addition, he thinks that URIs would have a similar level of restriction that RDF level processing has, and both can be treated equally.	Patrick wants to know why URIs can't be used for a new type of node type. He replies that RDF already has a type of labeled node. He then adds that a URI would not be restrictive to RDF level processing. He also notes that URIrefs do not do the job as a compact, single node denotation of datatypes and that a new class of node label will be treated just as opaquely as URIs and not restrictive.
Barb Fox shares some rewording that should be done in section 6.1 to remove any ambiguity. Barb discusses the changes she wants to make with the user tgindin@us.ibm.com, who replies that most of it is good but there should be a change in the third sentence. Barb, however, does not agree with these changes.	Barb wants to add a change to paragraph two in section 6.1 of the spec. Tom replies that the wording should be changed to remove any ambiguity and leave the door open for non-cryptographic signatures. Barb replies that she was trying to close a hole in the wording. Tom says that it's not clear that pure digest algorithms do not rule out using a cryptographic key and that it clarifies the issue. Barb then replies that it doesn't clarify.
Mark quotes Steve about GroupWise not using range locking, then asks that it is an expedient feature instead of a good idea in general. Steve replies that range locking allows an operating system to share semaphore quickly. Mark answers that range locking gets special treatment among legacy system behaviors and thinks that it's useless. Larry replies to Steve, saying that his operating system example is a special case. Steve agrees that the protocol doesn't need range locking and considers the issue settled. Yaron summarizes some problems and solutions.	Mark wants to know if range locking is a good idea or not. Steve replies that GroupWise doesn't use it. Mark replies that it's not a bad idea. Jim replies that they've been having a discussion about range locking and that the arguments for and against range locking can be collected in one message. Mark says that the argument does not hold water when a non-web-based document repository is used. Steve says that range locking should be supported.

Table 4.3: Reddit subset with *DistillBARTfull* - worst.

Gold summaries	**Generated summaries**
Many commenters think that the work of the original poster is really good. There is a balance between those who may be interested and those who are not. The original poster states they will post more later. They also state that they appreciate tips shared with them.	One commenter says that they like the sub, and another says they like it. Another commenter says they would like to add some people to the sub. Another user says they are just starting to build their PC..and that they are looking for deals on Black Friday. One user says that the sub is great and that they have been lurking about r/buildc for a long time, but they are unsure if they will be able to post on the subreddit or on the Xbox.
Most commentators ask the OP questions about specific players and whether to drop them or not. A few other commentators ask about possible trades, such as for Dubnyk. One commentator wants to know who to drop for Gagner. Another commentator wants to know if he should keep Tomas Fleischmann, which he should. A third asks about Crawford and Varly, who should be kept.	Most of the commenters think that Dubnyk is a useless goalie and that he needs to be replaced. One commenter says that he would like to trade him for a forward. Another commenter suggests that he should stay with the player. Other commenters say that the player is a good player and that they would like the player to be on the team. One user says that they are trying to get a player to replace him. Another suggests that they should offer him with someone else.
The commenters don't see people praying only when they want something as an issue. They see it as not a big deal and mention the teaching of asking and receiving. A couple of commenters state good believers pray all the time and not just when they want something.	One commenter says that they pray to God only when they need. Another says that theists pray all the time, even when they do not need it. Another commenter says they pray only when needed. Another user says that prayer is a way to communicate with God and that they get what they want. Another comment says that it is not possible to pray without feeling like they need it, and that prayer can be used as a placebo effect, and another says that God is a good thing.
One commentator posts a detailed review of a coffee place that roasts their coffee, complete with maps, images, and details on what they sell. Another commentator says that he's opening up a roaster, which interests some of the other commentators. Another commentator gives several options, complete with images and addresses, for the OP to try.	Commenters offer various places to buy coffee beans from in Guangzhou. One commenter says that they found a good roaster in the middle of the street. Another commenter says they have bought beans from two different places, including Bank and Rose Cafe, and another says they've bought coffee from a few other places. A couple of commenters say that they have been buying beans from both of the places and that they've been trying to find them. One user says they found the best coffee roasters.
Several commenters respond to the original poster's surprise that he has been unable to find any PC Leagues of dirtgame. The same commenters claim to have found active games though they also state their difficulties. The original poster shares his ups and downs as he finds a game to join, which follows immediate notice that racenet is not available.	Most commenters say that they have not been able to join leagues on the game Dirt game. One commenter says that they are not sure if there is an active league. Another commenter says they have been kicked out of the game, and another says they are unable to find active leagues. A few commenters say they have never seen an active event in the game and that it is just luck that they can join a league. One user says that the league is alive and well, but not always.

Table 4.4: Nyt subset with *DistillPegasus solo* - worst.

Gold summaries	**Generated summaries**
Several commenters chide readers for being so obsessed with Westbrook and suggest they are ignoring more significant events. One commenter responds in a lengthy post suggesting that the character gets his unwarranted publicity by being obnoxious. Others counter the position by admitting Westbrook's talent on the field while saying he is not so amazing as to deserve all the attention he has received.	Most commenters agree that Russell Westbrook is talented, entertaining, enigmatic, quirky, and caring human being who is involved in his community and puts on a great show. One commenter says that basketball is a form of entertainment and that Westbrook puts on a great show. One commenter says that basketball is a form of entertainment and that Westbrook puts on a great show. One commenter says that basketball is a form of entertainment and that Westbrook puts on a great show. One commenter says that basketball is a form of entertainment and that Westbrook puts on a great show.
Several commentators note that Balotelli used to be a poor, lazy player. Several commentators remark that Balotelli has improved over time and is doing well. A few commentators think that the article was enjoyable. A few other commentators disdain Balotelli, calling him overrated.	Commenters praise the article, saying that Mario Balotelli is a great player and that the team is doing well. One commenter says that he has never seen him play for Nice. Another commenter says that he has never seen him play for Nice. Another commenter says that he did not know that Mario Balotelli was playing for Nice. Another commenter says that he did not know that Mario Balotelli was playing for Nice. Another commenter says that he did not know that Mario Balotelli was playing for Nice. Another commenter says that he did not know that Mario Balotelli was playing for Nice.
One commentator thinks that the president has no knowledge and that the population must resist government policies, which he considers unscrupulous. One commentator reports that he watched John Legend and liked his attitude of being able to expose his points of view, respecting those who think differently. Another comment says that people who fought for civil law in the 1960s should be an inspiration. Others talk about the importance of art to keep society together and consider the artist to be a humble person.	Many commenters praise John Legend's artistry and activism. Some commenters praise his work, including his role in the film "La La Land" and his involvement in the civil rights movement. Others praise his work, including his role in the film "La La Land" and his involvement in the civil rights movement. A few commenters praise his work, including his role in the film "La La Land" and his role in the civil rights movement. A few commenters praise his work, including his role in the film "La La Land" and his role in the civil rights movement.
Several commentators suggest using eyeshades or blackout curtains to help the person sleep at night with the bright light nearby. A few commentators see a parallel between the person's situation and a Seinfeld episode with a similar premise. One commentator says that building codes are the way to address the situation, another noting that hallway lighting is different from outdoor lights. One commentator suggests that the person find a better apartment. One commentator sees a similar issue with reflected sunlight.	Several commenters discuss the issue of excessive light coming from a building and how to address it. Some commenters discuss the issue of light pollution and how to address it. A few commenters discuss the issue of light pollution and how to address it. A few commenters discuss the issue of light pollution and how to address it. A few commenters discuss the issue of light pollution and how to address it. A few commenters discuss the issue of light pollution and how to address it. A few commenters discuss the issue of light pollution and how to address it. A few commenters discuss the issue of light pollution and how to address it.

Table 4.5: Twitter posts summaries generated by the best model.

Gold summaries	**Generated summaries**
The monetary system (fiat currency. Bottled water in countries with potable tap water and gasoline taxes. Property Taxes. The Federal Reserve, ECB and other central banks are examples of the Federal Reserve system. Getting taxed on your paycheck and then paying taxes on everything you buy with the money that they already taxed.	Most commenters talk about the US governmentś role in the economy and taxes, such as the US dollar,the US healthcare system, and banksf́ees. One commenter says that the government is responsible for high rates of interest rates and fees. Another commenter suggests that banks should be able to charge more money, and another says that banks are a ”money machine” and that the American Dream is still alive and well despite the Ukraine war. One user says that taxes are a scam.
You need to create value for yourself, and you can hand over the company to a professional management team later, so that you can have more time to develop your own company. When the team and startup starts to secure customers, finding, and steady progress, they pivot to Fulltime startup! Gary Vee's children would be adults at this point, I would take the jump and branch out on my own,weighing out all of my options of course. For me...it's about loving what you do, having a good work life balance, and growing enough financially that my children won't have to suffer. If work/life balance is a priority, staying put may be a good plan.	Commenters offer advice on how to start a company, such as working for yourself and finding a side-hust. One commenter advises quitting his job so he can start his own company. Another commenter advises working for himself to get a job and work as a freelancer to get started a business. A few commenters give advice on starting his own business. One user says that quitting is the best decision, and another says that it takes courage to take a risk and make a decision to start your own firm.
We need absolute freedom of speech more than anything else. Limit the edit time window to 5 minutes directly after tweet. Show "edited" and if you really want, you could even show the edit history. Only a small time limit to edit your tweet. Bots are way more annoying than no edit button imo Elon...	Commenters agree that the edit button on Twitter is a dumb idea and that it should be limited to just five minutes to fix typos. One commenter says that Trump should be banned from the platform. Another commenter says it is too late to edit the original poster's post and that editing the original post should be edited for 15 minutes to make it easier. Another user says it's too late for Trump to be allowed back on the platform, then says that he should ban Trump from Twitter.
The easy access of your office can lead to just keep working and also to not taking breaks at all. Many remote workers indicate the unsatisfied need for communication, so if you are very sociable,you might find yourself in trouble too. If you have small children at home with you then you will not be able to focus on your work because of the many distractions and issues associated with them. One big challenge is internet connectivity, it is very confusing when this problem occurs only to you. Working from home can help you to save time for other things in your personal life.	Commenters agree that working from home can be a difficult task, especially if you have children or are a remote worker. One commenter says that working alone and working alone can be difficult. Another commenter says it is difficult to keep up with the team and that the only disadvantage is losing power at home. Another user says that they are having a hard time working life because they are a loner and not able to be productive when there are other people at work. A few commenters discuss the pros and cons of having to work from home.