Alma Mater Studiorum · Università di Bologna

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Artificial Intelligence

MASTER THESIS
in
Natural Language Processing

Efficient Knowledge Distillation for Green NLP Models: Bridging the Gap with Large Language Models

CANDIDATE:
Stefano Fantazzini

SUPERVISOR:
Prof. Paolo Torroni

CO-SUPERVISOR:
Anna Elisabetta Ziri
Dario Fioravante Alise
Federico Ruggeri

Academic year 2022-2023

Session 3rd

Contents

Introduction

The rise of Large Language Models (LLMs) has catalyzed a massive shift in the AI landscape, reshaping how many fields operate. As students enrolled in the new Master's course in Artificial Intelligence at the University of Bologna (UniBo), we have witnessed firsthand the rapid growth of this discipline.

This thesis stems from a shared aspiration between ourselves and PriceWaterhouseCoopers (PwC), a world leader in business and technology consulting, to innovate but with significant regard to the ecological impact of these new technologies. The substantial investments made by PwC in AI serve as a clear indicator of the field's exponential and widespread growth. This surge underscores the imperative for a more thoughtful approach to the development of these new technologies, as their rapid expansion requires increased awareness of the potential ecological impact.

Our decision to apply to PwC for our internship was driven by a dual purpose - the aspiration to gain invaluable industry experience and the personal commitment to contribute to a more sustainable world.

Renowned for its diverse portfolio of projects, particularly in the realms of legal, audit, and tax branches, PwC provides the perfect environment for automation through AI and Natural Language Processing (NLP). At present, PwC integrates OpenAI's GPT models into various use cases addressing many challenges related to text and document processing.

While AI implementations are a new focal point at PwC, an equally pressing concern is the environmental impact associated with the resource-intensive nature of LLMs. The demand for greener and inter-company alternatives of these models is growing into an essential consideration.

Considering these factors, identifying a suitable thesis topic presented no challenges, thanks to the fascinating realm of distillation and sustainability in LLMs.

This thesis is positioned at the intersection between state-of-the-art AI research and practical implementation, having as its main purpose the development of a small-sized and resource-efficient AI model that retains the capabilities of well-established LLMs.

Throughout the course of our thesis work, we delved into extensive readings and practical experiments. We began with a comprehensive literature review, exploring subjects such as the Transformer architecture and delving into recent research papers, for instance, "Distilling Step-by-Step" and "Element-aware Summarization with Large Language Models". Following that, we conducted fine-tuning experiments using the T5 model and utilized programming libraries such as Huggingface. Ultimately, we compared our fine-tuned models with the latest state-of-the-art results.

Problem Description & Thesis Objectives

The exploration of generative AI applications has become increasingly pertinent in the contemporary business landscape, particularly in the realm of automation. During our internship at PwC, we were exposed to numerous use cases that demonstrated the potential for efficient automation through the application of generative AI.

These applications predominantly revolved around text-centric tasks, currently taking advantage of technologies such as OpenAI's ChatGPT. As far as our knowledge goes, ChatGPT represents the current state of the art (SOTA) in text generation, showcasing its expertise in various NLP tasks.

However, the prevailing question that emerged during our internship at PwC was whether it could be designed an alternative, in-house model, that does not compromise on efficiency, while also being less resource-intensive and contributing to a more sustainable, greener environmental impact. This consideration also derives from the reliance on external providers such as OpenAI for the current internal document-based applications.

One notable advantage of having a private LLM instance is the independence it allows from external providers. This autonomy can lead to increased control over the model's behavior, ensuring alignment with the company's specific requirements and internal policies. However, this autonomy comes at the cost of a significant challenge – achieving a level of performance comparable to, or even surpassing, the current state-of-the-art models. Overcoming these challenges requires the development of a small-sized AI model that can retain the essential capabilities of colossal LLMs.

Hence, the primary objective of this thesis is to delve into the development of a compact AI model, specifically designed to handle the downstream task of text summarization. By focusing on efficiency and environmental sustainability, we aspire to strike a balance between autonomy and performance, presenting a viable alternative for the company's needs.

The thesis will perform a comprehensive exploration of the latest methodologies and approaches, such as the Chain-of-Thoughts technique and LLM knowledge distillation, to achieve the desired objective aspired by PwC.

1. Background and Relevant Literature

This section delves in an in-depth exploration of the theoretical foundations behind our model's architecture, providing a comprehensive examination of the relevant research papers. It provides additional support to assist in the general understandings of these major advancements.

1.1 The Transformer

At the foundation of the majority of current NLP models lies the Transformer mechanism. Thoroughly examining the "Attention Is All You Need" [24] paper, we will unravel the essential components that constitute the Transformer mechanism.

Transformers were developed to solve the problem of sequence transduction [9], or neural machine translation. This comprehend tasks involving the conversion of input sequences into corresponding output sequences.

1.1.1 Tokenizer

One crucial notion of the transformer, and many other NLP approaches, is that they operates on tokens rather than full words. A token is the fundamental unit of input processed by the model. These tokens can represent various linguistic elements, ranging from individual characters to entire words or subwords. For instance, in English, a word like "transformer" may be tokenized into individual subword units such as "trans," "form," and "er," or even further into characters like "t," "r," "a," and so on.

The component responsible for converting raw text into a sequence of tokens is called tokenizer. The tokenizer, and connected to it the chosen tokenization scheme, profoundly influences the model's word understanding and ability to generate sentences.

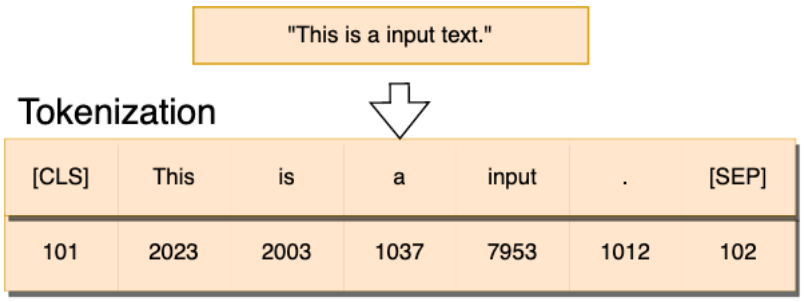


Figure 1.1: Simple example of tokenization

A tokenizer follows a sequential methodology when splitting a string into tokens. The typical approach involves examining the characters of the input string, identifying the first character that hasn't been assigned to a token, and verifying its existence in the vocabulary.

Upon finding a match in the vocabulary, the corresponding token is saved, and the process iteratively repeats, assessing whether the combination of the current character(s) and the next new character forms a valid token present in the vocabulary. If a match is found, a new token is saved, and the process repeats again. In cases where there is no match, the latest saved token is outputted, and a new search process begins with the new current character.

Vocabulary	
Characters	Token
...	...
a	11
ab	12
bc	13
d	14
...	...

Table 1.1: Example of a vocabulary of a tokenizer

Keeping as an example the vocabulary shown in table 1.1 and an input string "abd" to tokenize, we can now analyze the tokenization process.

The tokenizer begins by getting the first character, 'a', and checks its existence in the vocabulary table. As 'a' is listed as a valid token, it is saved, and the process continues. Moving to the next character, 'b,' the tokenizer start to examine the combination 'ab.' Since 'ab' is present in the vocabulary table, the new corresponding token is saved.

The process iterates to the following character, 'd,' which is also present in the table as a standalone token. Therefore, 'd' is saved as another token.

As there are no more characters in the input string, the tokenizer will output our tokenized input string, obtaining 12 and 14 as result.

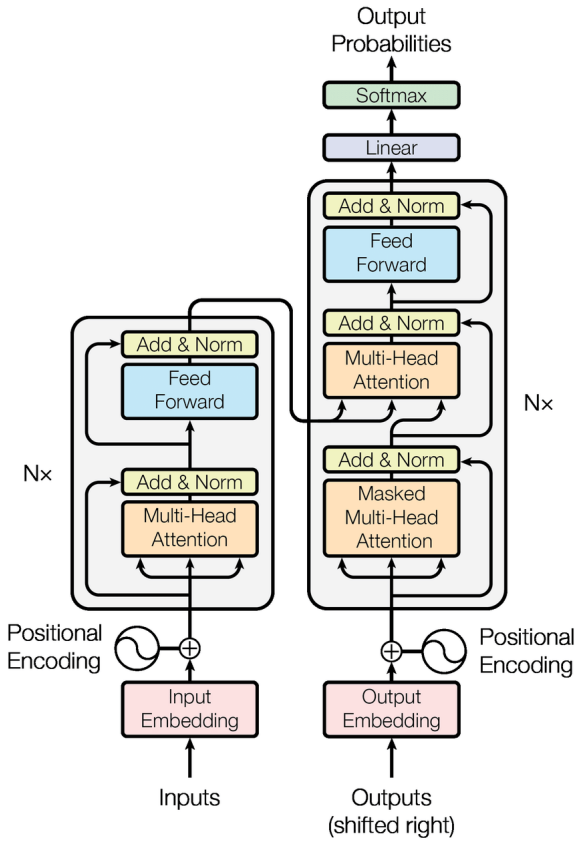


Figure 1.2: The Transformer model architecture

1.1.2 The Transformer Architecture

The Transformer architecture uses an encoder-decoder structure that does not rely on recurrence and convolutions to generate an output.

Since the tokens are not semantically meaningful, the architecture first converts the input data into n-dimensional embedding.

The encoder maps the input embedded sequence into a series of continuous representations, while the decoder receives the encoder's output and the decoder's output at the previous time step, generating an output sequence. We will now expand on these considerations.

- **Encoder**: The encoder consists of a stack of $N = 6$ identical layers, with each layer composed of two sublayers:
  - The first sublayer implements a multi-head self-attention mechanism.
  - The second sublayer is a fully connected feed-forward network consisting of two linear transformations with Rectified Linear Unit (ReLU) activation in between.

  The six layers of the Transformer encoder apply the same linear transformations to all the words in the input sequence, but each layer employs different weight (W_1, W_2) and bias (b_1, b_2) parameters.

  Furthermore, each of these two sublayers has a residual connection around it and a final normalization step.

  Finally, to enforce the knowledge of the words position, a positional encoder is utilized to inject this knowledge into the input embeddings.

- **Decoder**: The decoder shares several similarities with the encoder, having the same logic of $N = 6$ identical layers stacked, but each composed of three sublayers:
  - The first sublayer receives the output of the previous decoder in line, augments it with positional information, and implements multi-head self-attention, modified to attend only to the preceding words in the sequence.

- The second sublayer implements another multi-head self-attention mechanism. This multi-head mechanism receives the queries from the previous decoder sublayer and the keys and values from the output of the encoder. This allows the decoder to attend to all the words in the input sequence.
- The third layer implements a fully connected feed-forward network, similar to the one implemented in the encoder.

As done in the encoder implementation, the three sublayers on the decoder side also have residual connections around them and are succeeded by a normalization layer.

Positional encodings are also added to the input embeddings of the decoder in the same manner as for the encoder.

1.1.3 The Embedding Layer

The embedding layer serves as the initial stage in processing data within a transformer. Its primary purpose is to transform the vectors of discrete tokens representing words or subword units into matrices of dimension $b \times n \times d_{model}$.

Tokens, in isolation, lack meaningful semantic and relational information, which are needed to capture the nuances of a language.

By employing an embedding layer, transformers convert these tokens into high-dimensional vectors, effectively placing them into a continuous vector space where the relationships between words can be more accurately represented.

More precisely, they utilized pre-learned embeddings to convert the input tokens to vectors of dimension d_{model}.

Parameter	Description
d_{model}	Embedding's vector size within the transformer. Each vector represents a single token. A typical value is 2048.
d_k, d_v, d_q	Respective dimension of the key, value and query vectors.
b	Number of sequences processed in parallel by a transformer.
n	Sequence length in number of tokens.
$vocab\_size$	Number of possible tokens of the tokenizer.
n_blocks	Number of repeated blocks in the architecture.
$n\_attention\_head$	Number of attention heads in the architecture.
$attention\_head\_length$	Size of the embedding within an attention head.
$expansion\_ratio$	Ratio of growth of the embedding vector in the MLP.

Table 1.2: Overview of the relevant transformer parameters

1.1.4 Positional Encoding

Since the architecture does not contains recurrence nor convolution, in order for the model to understand and utilize the order of the input sequence, they had to inject the information of the position of the tokens into the sequence. To this end, they introduced "positional encoding" to provide this information to both encoder and decoders.

Positional encoding provides a numerical value which describe the position of a token in a sequence so that each position is assigned a unique representation.

This value was obtained in the original paper [24] utilizing sine and cosine functions of different frequencies.

1.1.5 Multi-Head Attention

Fundamental in the transformer architecture is the concept of attention, enabling the model to selectively concentrate on the most relevant parts of the input sequence. An attention function can be described as a mapping between a query and a set of key-value pairs to an output.

The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

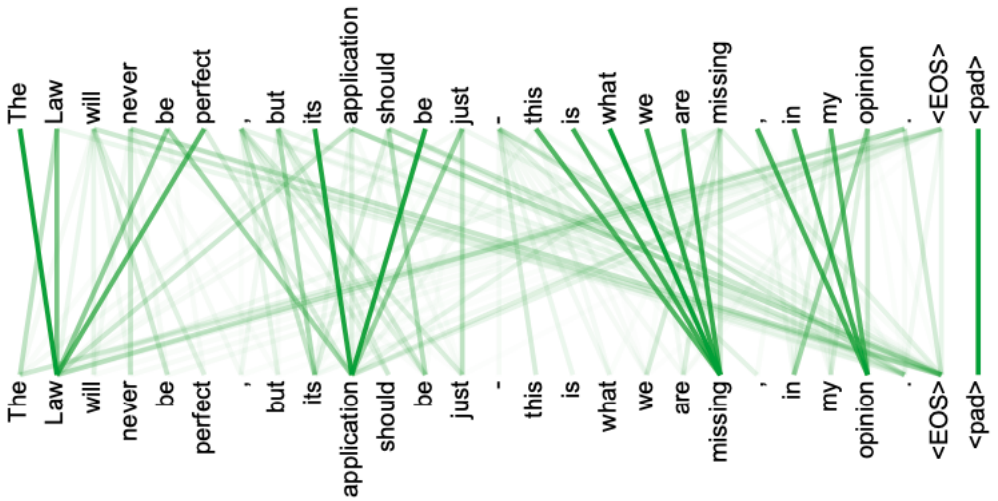


Figure 1.3: Example of self-attention

Within an attention head, the input (dimension $b \times n \times d_{model}$) is replicated in three equal instances. Each instance is then individually processed through a dedicated fully connected layer, reducing their dimension to $attention\_head\_length = \frac{d_{model}}{n\_attention\_head}$ from the original d_{model}.

This change in dimensionality is performed since, instead of carrying out a single attention function with d_{model}-dimensional keys, values and queries, they found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections.

The resulting vectors, denoted as K, Q, and V, are of dimension $b \times n \times attention\_head\_length$.

Subsequently, the computation of the dot product between matrices K and Q produces a matrix with dimensions $b \times n \times n$. This resulting matrix reveals the relevance between each pair of tokens of the input sequence.

At the end, the obtained matrix, is multiplied (dot product) by V, creating a vector of dimension $b \times n \times attention\_head\_length$ once again.

These different representation are then concatenated, allowing the model to jointly attend to information from different representation subspaces at different positions and obtaining again the original dimension $b \times n \times d_{model}$.

With closer examination, the transformer architecture implements an attention mechanism called: "Scaled Dot-Product Attention".

The two commonly most used attention functions are additive attention, and dot-product (multiplicative) attention. Scaled Dot-Product attention is identical to the classical dot-product implementation, but with the addition of a scaling factor $\frac{1}{\sqrt{d_k}}$ in the multiplicative operation.

This action was undertaken to prevent the excessive expansion of the dot product if given large d_k in input, pushing the softmax function into regions where it has extremely small gradients.

1.1.6 Masked Attention-Heads

Unlike the encoder, which contains normal multi-head self-attention layers, the decoder employs a modification of the aforementioned mechanism, called masked self-attention layers.

Similarly, masked self-attention layers allow each position in the decoder to attend to all the positions in the decoder up to a specific position. The idea is to prevent the flow of information to "future" part of the sentence.

Specifically, a token at index i can only access tokens with indexes smaller than i, preserving the auto-regressive property of the decoder.

To implement this property they masked all the "illegal connection", setting

their value to $-\infty$ before their progress through the softmax function.

This design promotes learning by prohibiting the model from simply copying future tokens from neighboring entries, driving the model to actively generate prediction based on its acquired knowledge.

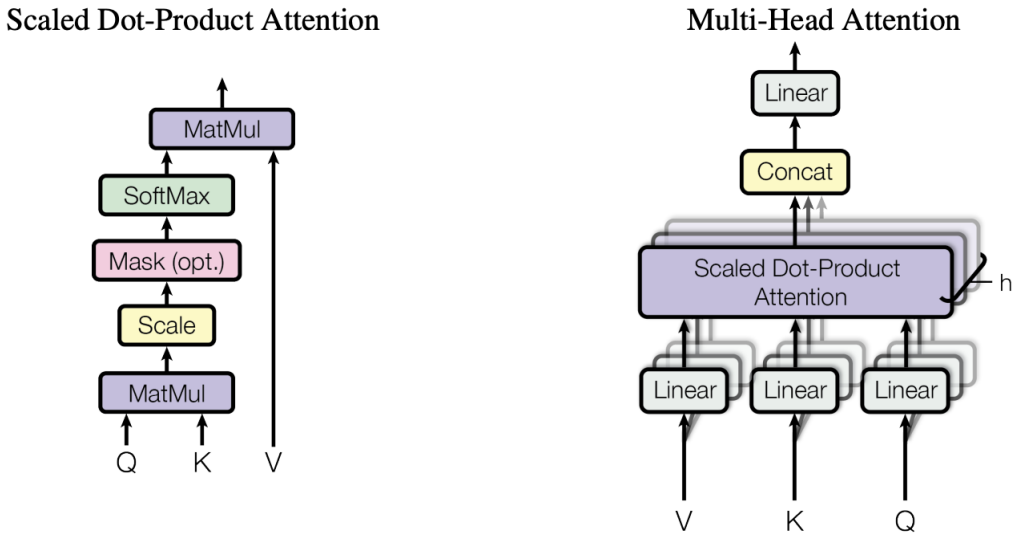


Figure 1.4: Illustration of the "Scaled Dot-Product Attention" and "Multi-Head Attention" mechanisms

1.1.7 Feed Forward Network

In addition to attention sub-layers, each of the layers in the encoder and decoder contains a fully connected feed-forward network, applied to each position separately and identically. This sub-layer consists of two linear transformations with a ReLU activation in between.

This component provides the elaboration needed for the model to process the received information.

Given its input of dimension $b \times n \times d_{model}$, it transforms them to dimension $b \times n \times (expansion\_ratio * d_{model})$ before returning it to the original shape.

1.1.8 Add & Normalization

The "Add & Normalization" layer serves a dual function within the transformer model architecture.

- *Residual Connection*: It combines two streams of the model creating a unified representation of the data [11].
- *Normalize*: Layer normalization, particularly through standardization, is employed to adjust and refine the resulting values[3].

This layer plays a crucial role in addressing issues like vanishing and exploding gradients commonly encountered in deep neural networks.

1.1.9 Output of the Network

With the obtained processed and elaborated data, two final steps remain.

Initially, the data changes its representation, transitioning from dimensions of $b \times n \times d_{model}$ to dimensions of $b \times n \times vocab\_size$. In this process, each of the n predictions at every time step yields a probability distribution across the tokens of the vocabulary.

Subsequently, a softmax function is employed to guarantee that these probabilities collectively sum up to one for each time step.

1.1.10 Decoder-Only Transformer

Modern LLMs and GPTs models employ a variation of the described transformer architecture, called decoder-only transformer.

The structure is identical to the previously analyze encoder-decoder, but omits the encoder part. For this reason these models are often called auto-regressive models.

The distinction between models that use only a decoder and those that employ both an encoder and decoder is somewhat blurred, with no consensus on what the best architecture and pre-training setup should be [21].

Encoder-decoder models handle inputs and outputs separately using distinct parameters and include a cross-attention mechanism that links the input tokens directly to the output tokens.

On the other hand, decoder-only models work by joining inputs and outputs together, resulting in the simultaneous development of representations for both inputs and outputs at each layer as they move through the network. It has been seen that these models are best suited for tasks involving text generation.

This concludes the description of the data flow within a transformer.

1.2 Transfer Learning

Transfer learning, the technique where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a powerful approach in NLP [20] and, more precisely, in the modern creation of LLMs.

In our text-centric context, transfer learning allows a model to leverage its understanding of language and context gained from one task to improve its performance on a different, related task.

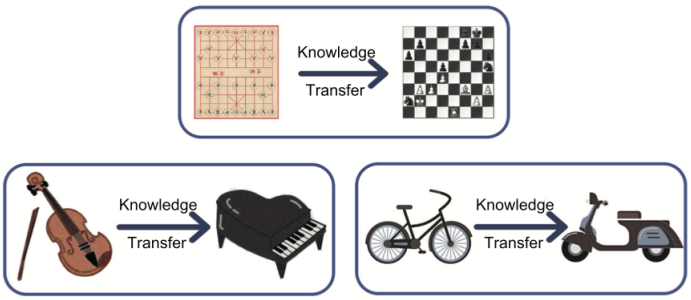


Figure 1.5: Intuitive examples about transfer learning

1.2.1 Theoretical Foundations

The ideal scenario of machine learning or deep learning is, generally, an abundant of labeled training instances, which have the same distribution as the test data.

However, collecting sufficient training data is often expensive, time-consuming, or even unrealistic in many scenarios.

The concept of transfer learning can trace its origin from educational psychology. According to the generalization theory of transfer, as proposed by psychologist C.H. Judd, learning to transfer is the result of the generalization of experience [28].

Inspired by human beings' capabilities to transfer knowledge across domains, transfer learning aims to leverage knowledge from a domain, called source domain, to improve the learning performance or minimize the number of labeled examples required in a related target domain.

More precisely, given a source domain D_s with its task T_s, and a target domain D_t with its task T_t, transfer learning utilizes the knowledge implied in the source domain to improve the performance of the learned decision functions f^T on the target domain.

It is worth mentioning that the transferred knowledge does not always bring a positive impact on new tasks. If there is little in common between domains, knowledge transfer could be unsuccessful.

If the target learner is negatively affected by the transferred knowledge, the phenomenon is defined as negative transfer.

1.2.2 Related Concepts

Here we will analyze some areas related to transfer learning and their relative connection and differences.

- **Semi-Supervised Learning**: The semi-supervised learning method lies between supervised learning and unsupervised learning.

  It utilizes abundant unlabeled instances combined with a limited number of labeled instances to train a learner. Both the labeled and unlabeled instances are drawn from the same distribution, which is in contrast with the concept of transfer learning, where the data distributions of the source and the target domains are usually different.

  The key assumptions in semi-supervised learning, i.e., smoothness, cluster, and manifold assumptions, are also made use of in transfer learning.

- **Multi-View Learning**: Multi-view learning focuses on the machine learn-

ing problems with multi-view data, resulting in abundant input information. A view represents a distinct feature set. An intuitive example about multiple views could be a video object which can be described from two different viewpoints, i.e., the image signal and the audio signal.

Multi-view techniques are also adopted in some transfer learning approaches.

- **Multi-Task Learning**: The idea behind multi-task learning, which is going to be a pivotal concept in this thesis, is to jointly learn a group of related tasks. More specifically, multi-task learning reinforces each task by taking advantage of the interconnections between task. In this way, the generalization of each task is enhanced.

  The main difference between transfer learning and multi-task learning is that the former transfer the knowledge contained in the related domains, while the latter transfer the knowledge via simultaneously learning some related tasks.

  Multi-task learning pays equal attention to each task, while transfer learning pays more attention to the target task than to the source task. Both of them aim to improve the performance of learners via knowledge transfer.

  Most of modern application employs both transfer learning and multi-task learning.

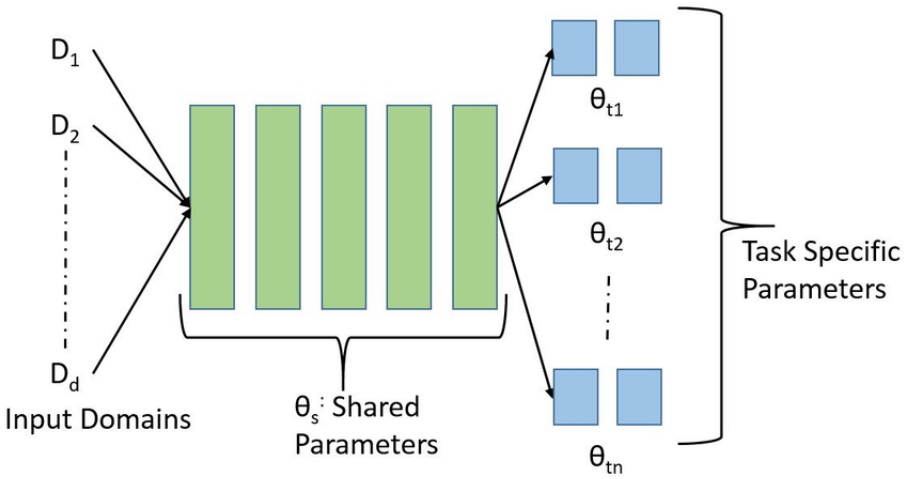


Figure 1.6: General multi-task learning framework

1.3 Fine Tuning

Another closely related concept is fine-tuning.

Fine-tuning involves taking a pre-trained model, trained on a large dataset for a general task such as natural language understanding, and adapting it on a task-specific dataset. This approach aims to optimize the model's performance on a new, related task, without the need of performing the training process anew.

Both transfer learning and fine-tuning allow the transfer of knowledge gained from solving one problem to a related problem. The key distinction lies in the notion that fine-tuning is a specific technique within the transfer learning spectrum. While transfer learning is the broader concept, fine-tuning is a targeted optimization that builds upon the foundational knowledge acquired by the model's previous training, making precise, often minor, adjustments to its parameters.

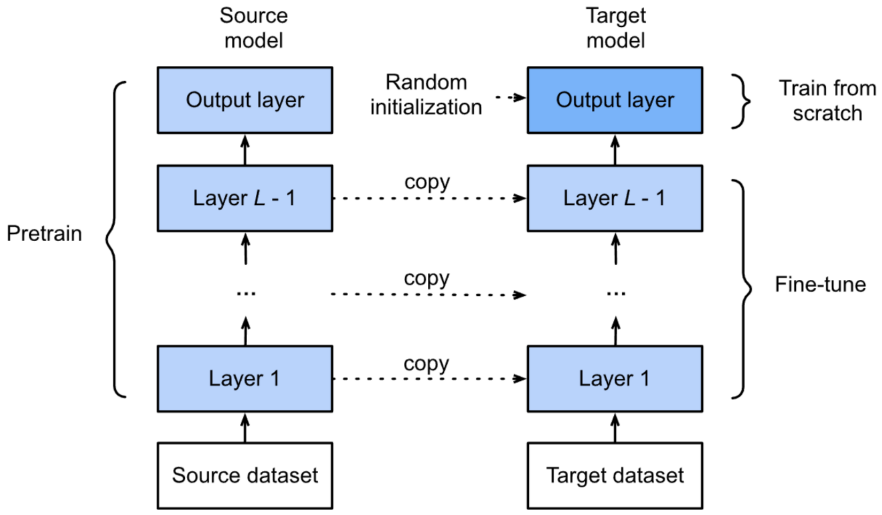


Figure 1.7: Fine-tuning visualization

1.3.1 Fine-Tune Pipeline

The typical strategy for fine-tuning generally comprises these steps:

1. *Choose a pre-trained model*: This pre-trained model has learned a rich representation that can be a valuable starting point for learning new tasks.

2. *Adaptation to the new task*: Typically involves modifying the top layers of the model changing, for example, the final layer to match the needs of the new task.

3. *Freeze or Unfreeze Layers*: Depending on the complexity of the task and the size of the dataset, one may choose to freeze some layers of the pre-trained model. Freezing a layer and preventing it from updating its parameters can be beneficial if they have already learned a stable structure which could be useful for the new task.

4. *Training*: Usually a shorter training with smaller learning rates is performed, preventing drastic changes in the already learned representation in favour of a more gentle adaptation to the new data.

1.4 Knowledge Distillation

In recent years, transformer-based pre-trained language models have achieved astounding results in both industry and academia. However, their large model sizes and high run-time latency are serious impediments for their practical implementation.

To address these challenges, knowledge distillation has been widely adopted as solving strategy.

Knowledge distillation shares many core concepts with transfer learning, label smoothing, ensemble learning and contrastive learning [14].

It is also used to achieved various objectives besides model compression, such as inference acceleration and generalization improvement.

1.4.1 Teacher-Student

The key idea of knowledge distillation is to utilize a larger and elaborate model as teacher, responsible of guiding the learning process of a smaller model, named the student model. Typically, the distillation methods utilize the teacher model's predictions to guide the student model's training.

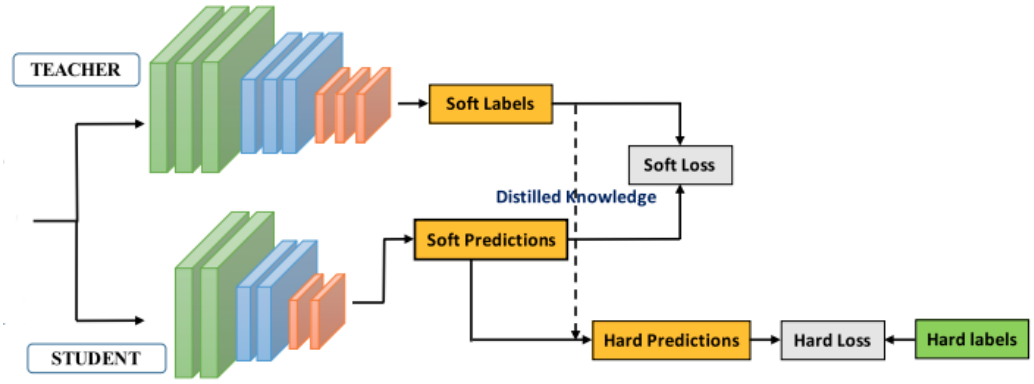


Figure 1.8: Standard knowledge distillation learning flow [12]

The student model is trained to predict both the correct output and the soft targets produced by the teacher model.

The soft targets refer to the probabilities the teacher produces when predicting the output of a given input. This training is performed by minimizing the distillation loss between the targets produced by the teacher model and the predictions produced by the student model.

In general, several types of loss functions exist to measure the difference between student and teacher models, such as Kullback–Leibler (KL) divergence, mean Squared Error (MSE) and Cosine Similarity [6].

Let denote with L_{KL} the knowledge loss and L_T the task loss. The knowledge distillation loss implemented will be:

$$L_{Distillation} = \alpha L_T + (1 - \alpha) L_{KL} \tag{1.1}$$

1.4.2 Distilling LLMs

As previously demonstrated, the representations of teachers and students will be compared and implemented as addition information in the learning process. Consequently, distillation requires the soft labels generated by both models to analyze the differences in their output distributions.

Distilling knowledge from existing LLMs can pose a challenge due to the restricted access one might have to the desired teacher architecture.

Recently, a solution to the aforementioned problem and the potential issue of data scarcity has been explored by utilizing these high-performing models as low-cost data labelers to train other models [25].

Although the data labeled by LLMs can usually be more noisy than human-labeled data, the process is much cheaper, faster and generalizable to multiple tasks.

Empirical evidence suggests that models fine-tuned on data generated by LLMs like GPT-3, thereby distilling the knowledge of the LLM, produce comparable performance across a range of Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks under the same low-budget settings. Moreover, it has also been noticed that these distilled models trained with data labeled by LLMs can outperform the LLM itself under the fewshot setting.

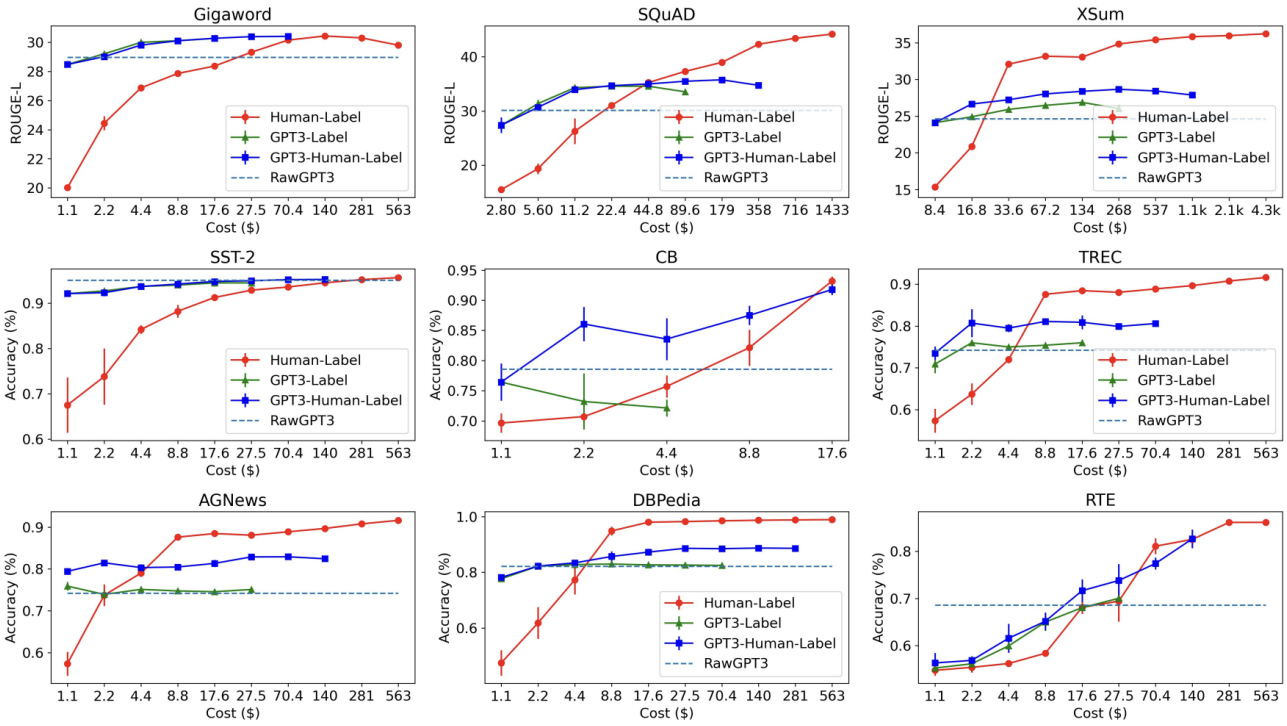


Figure 1.9: Comparison of labeling strategies' effectiveness against their associated expenses across nine NLG and NLU datasets. The horizontal axis represents the estimated cost according to OpenAI's pricing guidelines and crowd-sourced annotation rates. Every data point reflects the mean outcome from three separate trials of two distinct models, each utilizing three unique label sets, with the variation indicated by the standard deviation.

1.5 Semi-Automated Labeling

Employing GPT-3 or similar LLMs for generating labels on unlabeled datasets does not fall under the traditional umbrella of data augmentation. More precisely, this method can be identified as semi-automated labeling [7].

In a semi-automated labeling paradigm the predictive model (the machine labeler) not only assists the human when deciding on which labels to apply, but is also capable of automating some portion of the labeling task itself.

Data augmentation covers strategies that enhance the variety of a dataset used in the training process, without the need for new data acquisition. Such strategies modify current data, still remaining true to potential real-world variations. For instance, augmentation in image processing can include image manipulations like flipping, rotating, resizing, or color adjustments.

Conversely, utilizing a language model like GPT-3 to annotate a dataset involves generating labels. The precision of the labels produced by the language model is pivotal to the efficacy of the consequent predictive model.

1.6 Summarization

Text summarization is a key task in NLP which involves the condensation of large volumes of text into shorter, coherent summaries while retaining the essential information. There are two main approaches to text summarization: abstractive and extractive summarization [4].

- **Abstractive summarization**

  Abstractive summarization involves generating a concise summary that may contain words, phrases, or sentences not present in the source text. This approach requires the model to understand the context and generate human-like language to convey the central ideas. Abstractive summarization methods commonly employ advanced language models, such as LLMs, to rewrite and rephrase content in a more concise form.

- **Extractive summarization**:

Extractive summarization, conversely, is focused on identifying and selecting the most significant sentences or phrases directly from the source text to create the summary. It does not involve rephrasing or generating new sentences. Techniques used in extractive summarization include sentence scoring and ranking to pinpoint and extract the most important content.

It is important to note that while abstractive summarization can produce more "human-like" summaries, it may also introduce errors in understanding or interpretation.

Extractive summarization, while potentially more accurate and easier to implement, may result in overly simplistic summaries and lack readability due to issues such as redundancy.

The choice between these two approaches depends on the specific requirements of the summarization task and the desired trade-offs between informativeness, fluency, and conciseness in the final summary.

2. Models

We now transition from the foundational theory laid out in the preceding section to an in-depth examination of the various models that constitute the core of our research.

2.1 Generative Pre-trained Transformer (GPT)

The Generative Pre-trained Transformer (GPT), particularly its advanced version GPT-3, is a large language model released by OpenAI in 2020. It garnered significant global attention with its integration into the well-known ChatGPT application.

At the core of their approach lies language modeling. Language modeling is usually framed as unsupervised distribution estimation from a set of examples $(x_1, x_2, ..., x_n)$ each composed of variable length sequences of symbols $(s_1, s_2, ..., s_n)$.

Since language has a natural sequential ordering, it is common to factorize the joint probabilities over symbols as the product of conditional probabilities [19].

$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, ..., s_{n-1}) \tag{2.1}$$

GPT-3, which is an autoregressive language model boasting 175 billion parameters, a tenfold increase over any prior non-sparse language model, is known for its strong few-shot performance on many NLP tasks [5].

2.1.1 Architecture

In this study, we utilize the GPT-3.5 model, which is built upon the previous GPT frameworks. This section will provide a concise examination of each upgrade of this structure.

GPT

The model largely follows the original transformer work [24]. It employs a 12-layer decoder-only transformer with masked self-attention heads and a 3072 dimensional inner state for the position-wise feed-forward networks [18].

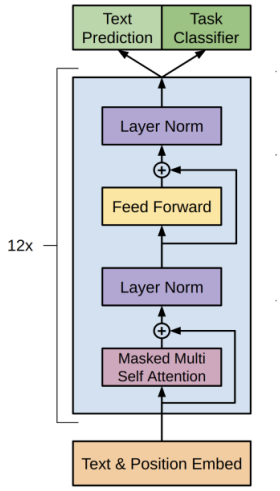


Figure 2.1: GPT architecture and training objectives

GPT-2

This next iteration of the GPT architecture introduces minor adjustments to some of its components [19].

Layer normalization was moved to the input of each sub-block, similar to a pre-activation residual network, and an additional normalization layer was added after the final self-attention block.

They also scaled the weights of residual layers at initialization by a factor of $\frac{1}{\sqrt{N}}$, where N is the number of residual layers, and expanded the vocabulary to size 50,257.

The context size was also increased from 512 to 1024 tokens and a larger batch size of 512 is used.

GPT-3

The architecture almost identically follows the structure of the previous GPT-2 iteration, with the exception that they alternated dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [5].

2.1.2 Energy Usage

The process of pre-training models on a massive scale, such as GPT-3 with 175 billion parameters, entails considerable computational demands, leading to substantial energy consumption. Specifically, the pre-training phase of GPT-3 involved the usage of several thousand petaflop-days, displaying the intense computation required.

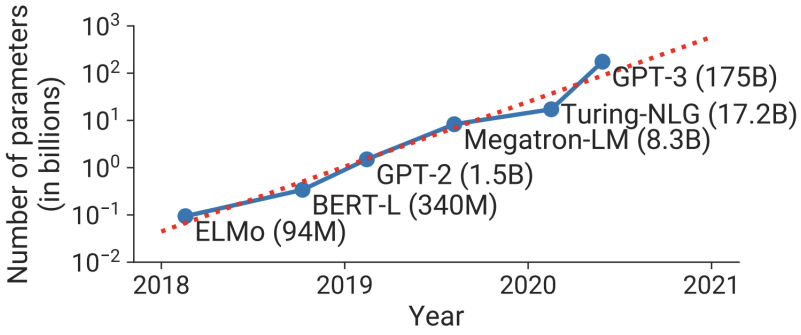


Figure 2.2: Trend of sizes of state-of-the-art NLP models with time [17]

However, it is crucial to assess the energy efficiency of such models from a broader perspective, taking into account not just the immediate energy input during the training phase but also the extent to which these resources are distributed over the model's operational lifespan. Post-training, the energy efficiency of GPT-3 can be quite notable. For instance, generating a hundred pages of content using the fully trained GPT-3 model needs an energy cost of approximately 0.4 kilowatt-hours, which translates to just a few cents worth of electricity.

Nevertheless, the significant memory requirements to accommodate the vast number of parameters of GPT-3 must also be considered. Storing 175 billion parameters, with each parameter occupying 2 bytes of space, necessitates around 350 gigabytes of memory storage.

To manage this substantial memory demand without exceeding available memory resources, OpenAI employed a combination of model parallelism strategies. These strategies involved distributing the model's computation both within individual matrix multiplications and across the layers of the neural network.

The entirety of the training was conducted on multiple NVIDIA V100 GPUs as part of a high-bandwidth computing cluster supplied by Microsoft.

2.2 Text-to-Text Transfer Transformer (T5)

Transfer learning has emerged as a powerful technique in NLP, giving rise to a diversity of approaches, methodology, and practices. Building on this concept, the Text-to-Text Transfer Transformer (T5) was designed, introducing a unified framework that converts all text-based language tasks into a text-to-text format.

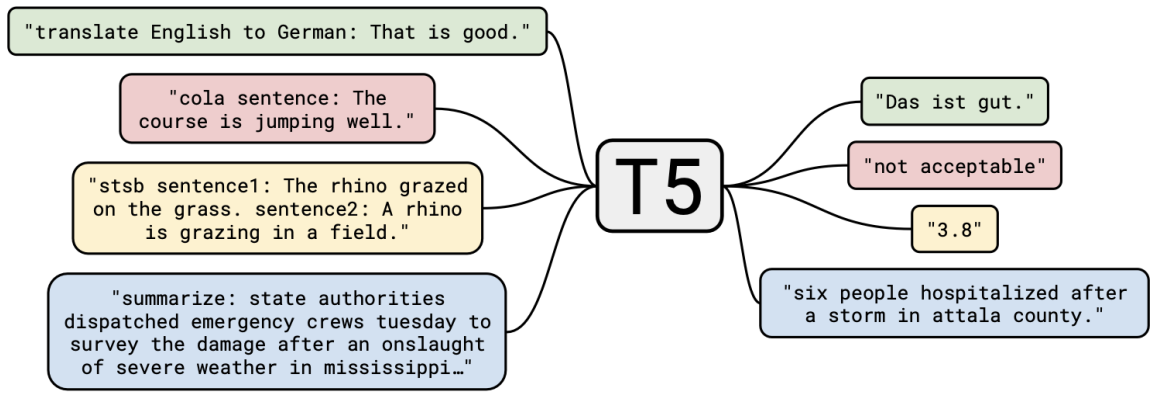


Figure 2.3: Diagram of the T5 text-to-text framework.

Training a machine learning model to perform NLP tasks requires the model to process text in a way that is amenable to downstream learning. This can be

viewed as developing a general-purpose knowledge base that allows the model to "understand" text, ranging from low-level to high-level knowledge.

The trend of pre-training a model extensively on a data-rich task has gained popularity, making the model abilities and knowledge more general-purpose oriented, allowing them to be transferred to the desired downstream tasks. Most recent SOTA models in NLP now frequently adopt this method, leveraging unsupervised learning on vast amounts of unlabeled data during the pre-training phase.

As previously anticipated, the key idea underlying the T5 model is to treat every text processing problem as a "text-to-text" problem. By doing so, the same model architecture, objectives, training procedure, and decoding process can be uniformly applied across a wide array of downstream tasks. This introduces a high level of adaptability, allowing for seamless transitions between different types of language processing assignment [20].

2.2.1 Architecture

Overall, the T5 encoder-decoder transformer implementation closely follows the original transformer design [24].

First, an input sequence of tokens is mapped to a sequence of embeddings, which is then passed into the encoder. The encoder consists of a stack of "blocks" with its classic sub-components. Minor difference is the use of a simplified version of the normalization layer, having the activations only rescaled with no additive bias applied.

The decoder structure follows the original design.

Additional variation to the model is the use of a simplified form of position embedding, where each "embedding" is a scalar that is added to the corresponding logit for the attention weights. For efficiency, they decided to share the position embedding parameters across all layers in the model.

To summarize, the T5 model is almost equivalent to the original transformer design, with the exception of the removal of the Layer Norm bias, the shift of the layer normalization outside the residual path, and the use of a different position embedding scheme.

2.2.2 Input Format

In order to train a single model on a diverse set of tasks, T5 cast all of the possible tasks into a unified "text-to-text" format. This framework provides a consistent training objective both for pre-training and fine-tuning.

To specify which task the model should perform, they decided to add a task-specific prefix (text prefix) to the original input sequence before feeding it to the model. As an example, to ask the model to translate the sentence "That is good." from English to German, the model would be fed the sequence "translate English to German: That is good." and would be trained to output "Das ist gut."

The choice of a prefix for a given task is essentially a hyperparameter; they also found that changing the exact wording of the prefix had limited impact.

2.2.3 Training & Objectives

In order to make the model learn the text-to-text objective, they chose to provide the model with the concatenation of the input sequence and its target.

The model would be trained on next-step prediction over the concatenated input sequence "translate English to German: That is good. target: Das ist gut.".

To obtain the model's prediction for this example, the model would be fed as input "translate English to German: That is good. target:" and would be asked to generate the remainder of the sequence autoregressively.

A frequently cited drawback of using a language model under the text-to-text setting is that causal masking forces the model's representation of the i-th entry of the input sequence to only depend on the entries up until i.

To see why this is potentially disadvantageous, lets consider the text-to-text framework where the model is provided with a prefix/context before being asked to make predictions.

With fully causal masking, the model's representation of a prefix state can only depend on prior entries of the prefix. So, when predicting an entry of the output, the model will attend to a representation of the prefix that is unnecessarily limited.

This issue can be avoided in a transformer-based language model simply by changing the masking pattern, for instance, instead of using a causal mask they

utilized fully-visible masking during the prefix portion of the sequence.

In the English to German translation example mentioned before, fully-visible masking would be applied to the prefix "translate English to German: That is good. target:".

The primary goal of the T5 model is unsupervised denoising for text-to-text tasks.

2.2.4 Model Variations

There are five T5 variants with varying parameters and model sizes:

1. **Base**: Comparable to that of BERT_base. Baseline model with 222 million parameters.
2. **Small**: Scaled down version of the Base model. It only has 60 million parameters with only 6 layers of encoders and decoders.
3. **Large**: Scaled up version of the base with 770 million parameters.
4. **3B**: Scaled up version of the base with 3 Billion parameters.
5. **11B**: Scaled up version of the base with 11 Billion parameters

2.3 Long T5

The initial T5 design featured a maximum input length of 512 tokens, typically sufficient for many NLP tasks. However, challenges arise for tasks requiring longer inputs.

While the model technically can handle inputs beyond the 512 token limit, it is not recommended due to the T5 architecture's utilization of the conventional self-attention mechanism, resulting in memory consumption that scales quadratically (n^2) with the input length.

In addition to find a solution for the aforementioned problem, recent work has shown that increasing the input length or increasing the model size can improve the performance of transformer-based neural models.

Under these consideration, the Long T5 model was conceived, exploring the effects of scaling both input length and model size at the same time [10].

2.3.1 Architecture

Architecturally, the main difference between T5 and Long T5 lies in the attention mechanism.

They experimented with both Local Attention and Transient Global Attention (TGlobal).

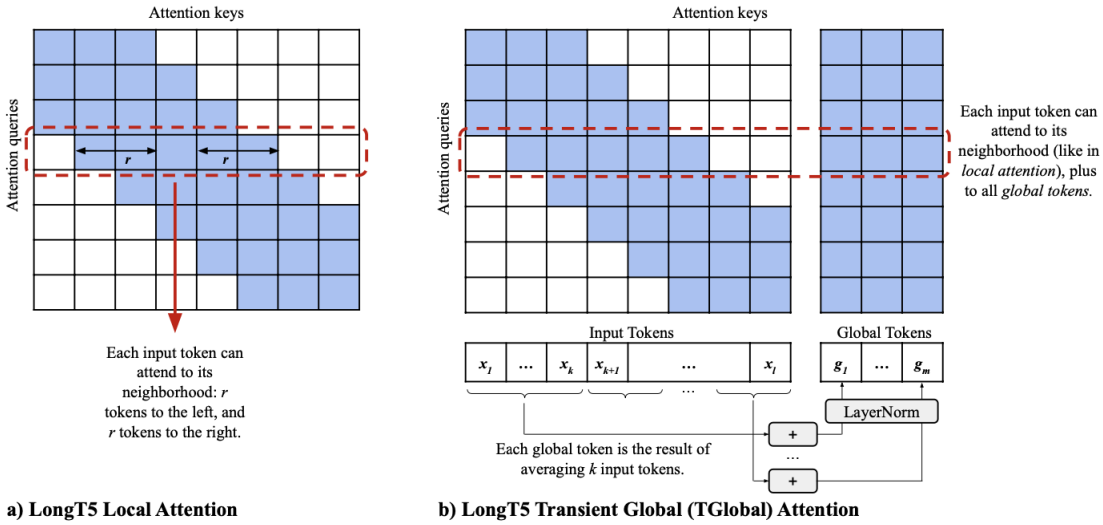


Figure 2.4: Illustration of the two attention mechanisms used in Long T5.

Local Attention

The self-attention utilized in the encoder part of the T5 model is replaced with a sparse sliding-window local attention operation.

Specifically, for a given local radius r, this formulation only allows each token to attend r tokens to the left and right of it (figure 2.4.a).

Local Attention does not introduce any new parameters and the complexity is linear in input sequence length $l : O(l \times r)$.

Transient Global Attention (TGlobal)

To allow input tokens to interact with each other in each layer of the encoder at a longer range than Local Attention's allowed radius, Long T5 introduce Transient Global Attention in a "fixed blocks" pattern.

The input sequence is divided into blocks of k tokens, where for each block they compute a global token, summing (and then normalizing) the embeddings of every token in the block (figure 2.4.b).

Now when computing attention, each input token is allowed to attend not only to nearby tokens like in Local Attention, but also to every global token.

Their approach is defined as transient since these global tokens are dynamically constructed (and subsequently discarded) within each attention operation, removing any requirement for deciding which input tokens should be treated as "global".

The complexity of this modified approach, given a input length of l and a block size k, is $O(l(r + \frac{l}{k}))$.

2.3.2 Model Variations

There are three Long T5 variants with varying parameters and model sizes:

1. **Base**: Baseline model with 250 million parameters.
2. **Large**: Scaled up version of the base with 780 million parameters.
3. **xl**: Scaled up version of the base with 3 Billion parameters.

3. Approach and Relevant Researches

After delving into the analysis of models at the core of our thesis, this chapter serves to present the pertinent research papers within the literature that have influenced the approach adopted in our study.

3.1 Chain-of-Thought

The contemporary NLP landscape has experienced, as we have previously noted, an important shift with the emergence of LLMs. These models have demonstrated that increasing the size yields various advantages, including improved performance and sample efficiency.

However, scaling up model size alone has not proved sufficient for achieving high performance on challenging tasks such as arithmetic, commonsense, and symbolic reasoning.

To augment the performance of LLMs further, it has been shown that making the model generate the series of intermediate reasoning steps significantly improves the ability of LLM to perform complex reasoning. This decomposed and more granular reasoning process is called Chain-of-Thought (CoT) [27].

3.1.1 Chain-of-Thought Prompting

Consider one's own thought process when solving a complicated reasoning task such as a multi-step math word problem. It is typical to decompose the problem into intermediate steps and solve each before giving the final answer.

The paper explore the ability of LLMs to perform few-shot prompting for reasoning tasks, given a prompt that consists of triples: ⟨input, *chain of thought*, output⟩.

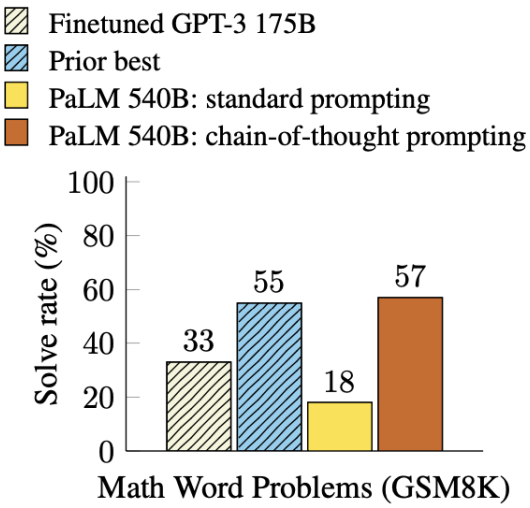


Figure 3.1: Comparison of LLMs performance on math world problems.

A CoT is a series of intermediate natural language reasoning steps that lead to the final output, and thus this approach is referred as CoT prompting

CoT prompting, as an approach for facilitating reasoning in language models, has several properties:

1. The approach allows the breakdown of complex, multi-step problems into smaller, intermediate passage. This modularization enables the allocation of additional computational resources to tasks that demand higher reasoning and analysis, thereby enhancing the problem-solving capabilities of the models.

2. CoT provides an interpretable glimpse into the behavior of the model, suggesting how it might have arrived at a particular solution and providing opportunities to debug where the reasoning process went wrong (although fully characterizing a model's computations that support an answer remains an open question).

3. This reasoning approach can be used for tasks such as math word problems, commonsense reasoning, symbolic manipulation, and is potentially applicable (at least in principle) to any task that humans can solve via language.

4. CoT reasoning can be readily elicited in sufficiently large off-the-shelf language models simply by including examples of chain of thought sequences into the provided few-shot prompts.

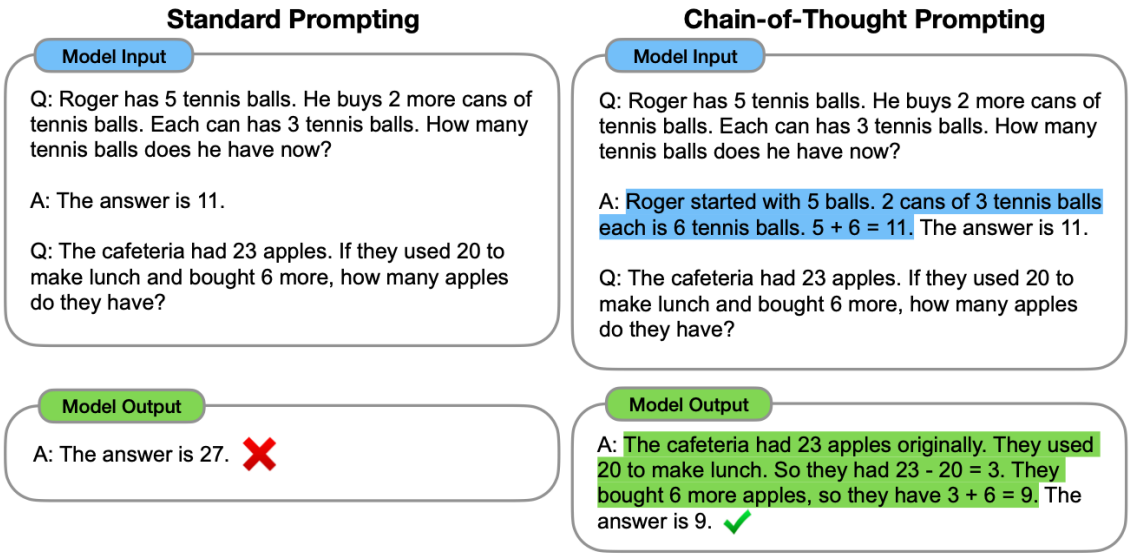


Figure 3.2: Chain-of-thought prompting examples. The Chain-of-thought reasoning processes are highlighted.

3.2 Distilling Step-by-Step

Despite the impressive few-shot ability offered by LLMs, these models are challenging to deploy in real world applications due to their sheer size. Serving a single 175 billion LLM requires at least 350GB GPU memory using specialized infrastructure. To make matters worse, today's SOTA LLMs are composed of over 500B parameters, requiring significantly more memory and computational power.

Such computational requirements are beyond affordable for most product teams, especially for applications that require low latency performance.

As a solution to this limitation, researchers train smaller task-specific models either by fine-tuning with human labels or distilling using LLM-generated labels.

However, fine-tuning and distillation require large amounts of training data to achieve comparable performance to LLMs.

Introduced in this study they propose Distilling step-by-step, a training paradigm that aims to train smaller models to outperform LLMs and achieves so by leveraging less training data needed by fine-tuning or distillation [13].

Core to their approach is the change of perspective from viewing LLMs as a source of noisy labels to active agents able to reason: LLMs can produce natural

language rationales justifying their predicted labels.

For example, when asked "Jesse's room is 11 feet long and 15 feet wide. If she already has 16 square feet of carpet. How much more carpet does she need to cover the whole floor?", an LLM can be prompted by CoT technique to provide intermediate rationales such as "Area = length x width. Jesse's room has 11 x 15 square feet." that better connects the input to the final answer "(11 x 15) - 16".

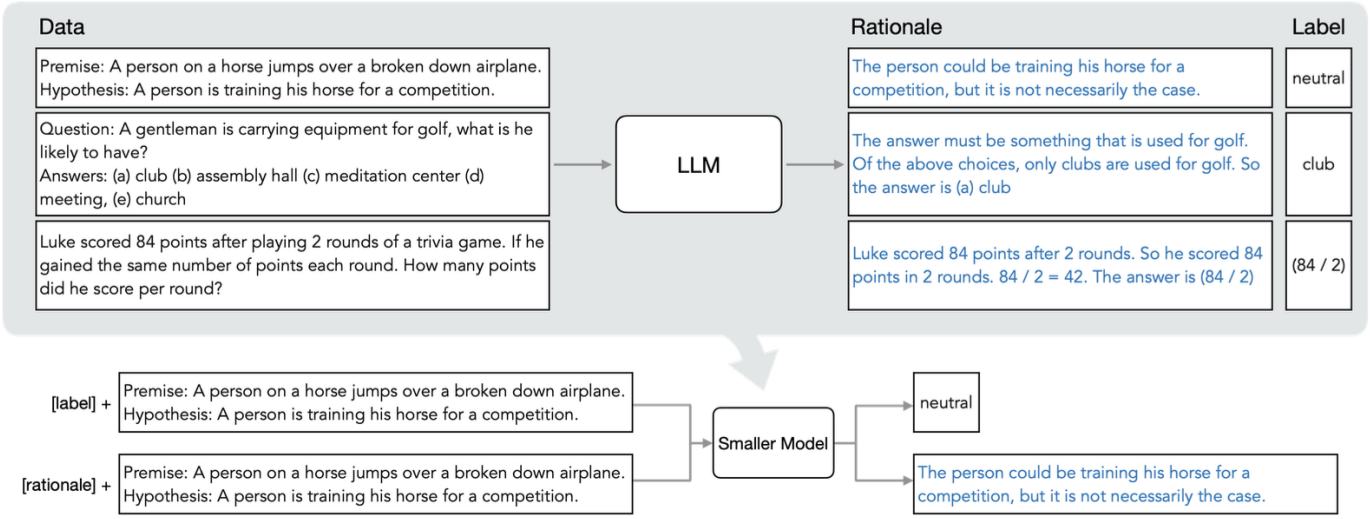


Figure 3.3: Overview on Distilling step-by-step

These rationales can contain relevant task knowledge that would have required a larger amount of data for small task-specific models to learn the same concept.

The idea is thus to utilize these extracted rationales as additional, richer information, to train small models in a multi-task training setup, providing both label and rationale prediction tasks.

The primary models employed to assess performance in the step-by-step distillation process included a range of implementations of different sizes of the T5 model, along with the utilization of PaLM as main LLM reference.

3.2.1 Related Work

Knowledge distillation from large models

Knowledge distillation has been successfully used to transfer knowledge from larger, highly trained teacher models into smaller student models affordable for practical applications.

The one limitation that knowledge distillation often faces is its reliance on large amounts of unlabelled data required to create a useful noisy training dataset.

Learning with LLM generated rationales

Exploring the use of rationales generated by LLMs presents an exciting frontier for research, yet it stands in contrast to the traditional use of human-generated rationales. However, the latter come with considerable cost implications.

Today's LLMs are capable of explaining their predictions by generating high-quality reasoning steps. These reasoning steps have been used to augment input prompts to LLMs, improving their few-shot or zero-shot performance.

Modern approaches leverage generated rationales as informative supervision to train smaller task-specific models, i.e. models that can be deployed without incurring large computation or memory costs.

3.2.2 Training smaller models with rationales

Given an unlabeled dataset $x_i \in D$, they curated each prompt template p, obtaining the needed information for the step-by-step training.

Each entry became a triplet (x_i, r_i, y_i), where x_i is an example input, y_i is its corresponding label and r_i is a the provided LLM rationale that explains why x_i can be categorized as y_i.

As mentioned earlier, the revised training objective now involves a multi-task learning approach integrated with rationales. The core idea is to utilize the information that the model has to produce in order to extract the rationale $\hat{r_i}$, to emphasize and create a more explicit connection between x_i and $\hat{y_i}$.

There are several ways to incorporate rationales into the downstream model's

training process. One straightforward approach is to feed r_i as an additional input, aiming for $f(x_i, r_i) \rightarrow \hat{y}_i$.

$$L = \frac{1}{N} \sum_{i=1}^{N} l(f(x_i, r_i), y_i) \tag{3.1}$$

Unfortunately, this design requires an LLM to generate a rationale before the model can make a prediction, bounding also at deployment the need of an LLM, thus limiting its deployability.

Instead of using rationales as additional model inputs, they framed learning with rationales as a multi-task problem. Specifically, they train the model $f(x_i) \rightarrow (\hat{r}_i, \hat{y}_i)$, not only predicting the task labels but also generating the corresponding rationales given the text inputs.

The new multi-task loss becomes:

$$L = L_{label} + \lambda L_{rationale} \tag{3.2}$$

Where:

$$L_{label} = \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i) \tag{3.3}$$

$$L_{rationale} = \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), r_i) \tag{3.4}$$

The rationale generation loss enables the model to learn to generate the intermediate reasoning steps for the prediction, and could therefore guide the model in better predicting the resultant label.

Given that they employed the T5 architecture as their primary smaller model, it was needed to correctly translate the aforementioned structure to one fitted for the T5 input format.

To do so, they prepend "task prefixes" ([label], [rationale]) to the input examples and train the smaller model to output $\hat{y}_i$ when [label] is provided and to produce $\hat{r}_i$ with [rationale] provided.

3.2.3 Results

They conducted experiments on four popular benchmark datasets across three different NLP tasks:

1. e-SNLI and ANLI for natural language inference.
2. CQA for commonsense question answering.
3. SVAMP for arithmetic math word problems.

Distilling step-by-step outperforms standard fine-tuning with much less labeled examples

They discovered that distilling step-by- step can achieve the same performance as standard fine-tuning with much less labeled examples.

In particular, by using only 12.5% of the full e-SNLI dataset, distilling step-by-step can outperform standard fine-tuning trained with 100% of the full dataset.

Similarly, they achieved 75%, 25%, and 20% reduction in training examples required to outperform standard fine-tuning on ANLI, CQA, and SVAMP respectively.

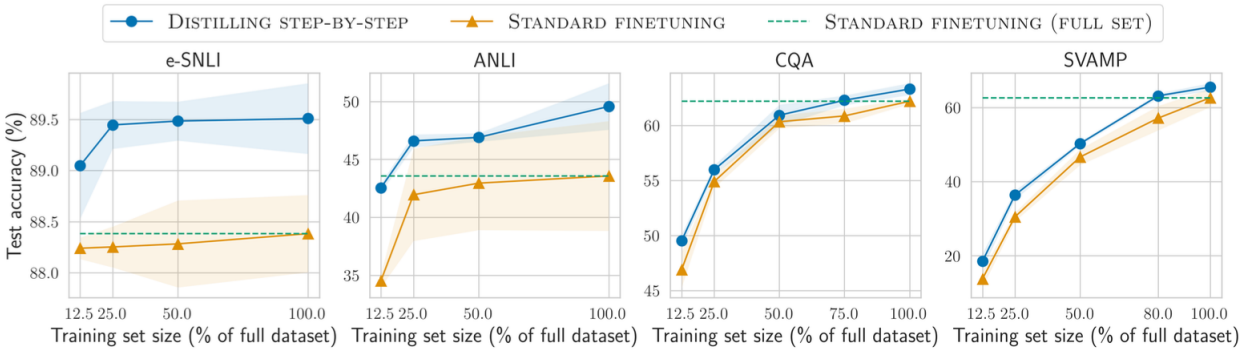


Figure 3.4: Distilling step-by-step and standard fine-tuning using 220M T5 models on varying sizes of human-labeled datasets.

Distilling step-by-step outperforms standard distillation with much less unlabeled examples

When only unlabeled data is available, they compared distilling step-by-step to standard task distillation.

It can be seen that distilling step-by-step outperforms standard task distillation on all four datasets under different numbers of unlabeled data used.

Furthermore, distilling step-by-step requires much less unlabeled data to outperform standard task distillation. For instance, they needed only 12.5% of the full unlabeled dataset to outperform the performance achieved by standard task distillation using 100% of the training examples on e-SNLI dataset.

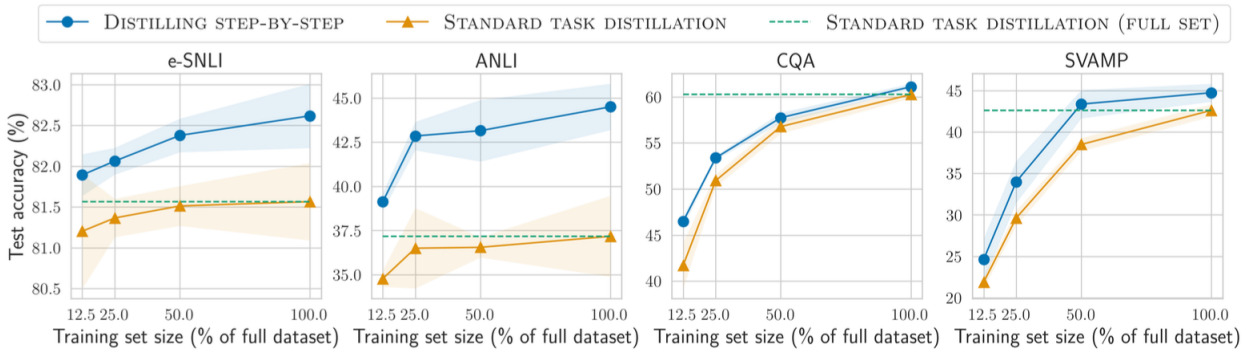


Figure 3.5: Distilling step-by-step and standard knowledge distillation using 220M T5 models on varying sizes of unlabeled datasets.

Distilling step-by-step outperforms LLMs with much smaller models by using less data

An additional remarkable finding was that distilling step-by-step outperforms PaLM's Few-shot CoT with much smaller T5 models using only a subset of the available training examples.

Specifically, on e-SNLI, distilling step-by-step can achieve better performance than Few-shot CoT with a model over 2000 times smaller (220M T5) and only 0.1% of the full dataset.

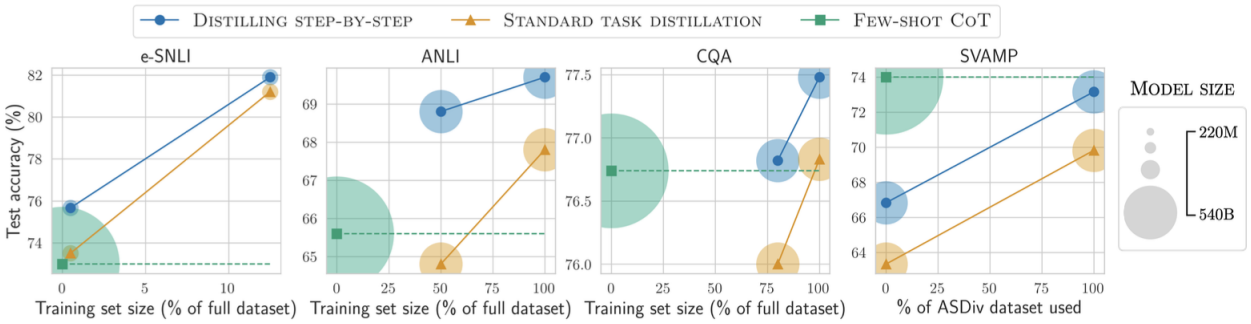


Figure 3.6: Minimum size of T5 models trained using only unlabeled example can outperform LLM's Few-shot CoT by a coarse-grained search.

3.3 Element-aware Summarization and Chain-of-Thought Method

Commonly utilized datasets like CNN/DailyMail and BBC XSum are prevalent choices for evaluating performance within the news sub-domain. Nevertheless, it has become evident that the reference summaries provided by these datasets often contain inaccuracies and redundant information.

Furthermore, experiments have shown that reference summaries in these standard datasets perform poorly on human assessment dimensions, especially coherence, consistency, and relevance.

To address this challenge, they first produce an annotated expert-writing Element-aware test sets following the "Lasswell Communication Model" as introduced by Lasswell in 1948, and later evolved into the "5W1H" paradigm.

This improvement enables reference summaries to concentrate on a finer granularity of news elements in an objective and comprehensive manner.

This approach states that effective summaries encompass four fundamental components: **Entity**, **Date**, **Event**, and **Result**.

A valuable contribution to our research was found in the section of this paper where the authors introduced a technique called the Summary Chain-of-Thought (SumCoT) method.

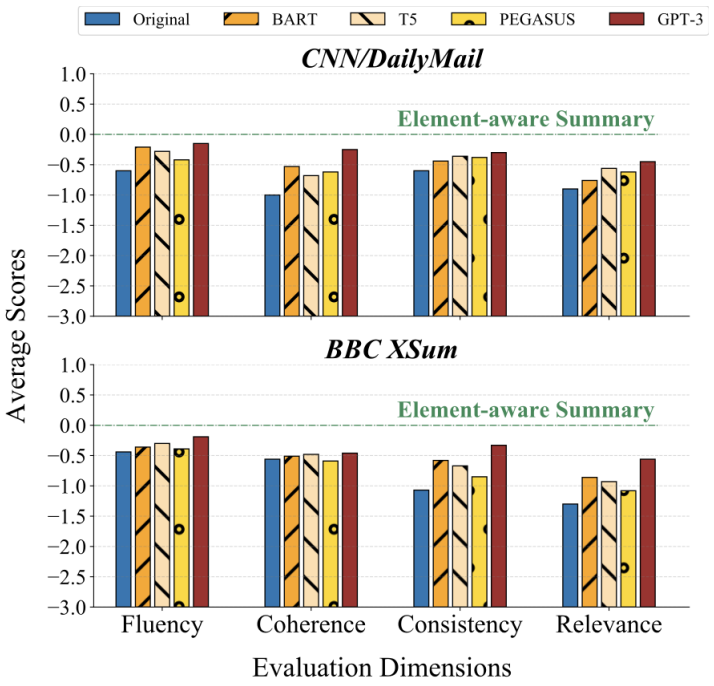


Figure 3.7: Human evaluation scores of four dimensions about summary quality on the 50-shot CNN/DailyMail (the upper part) and BBC XSum (the lower part) datasets.

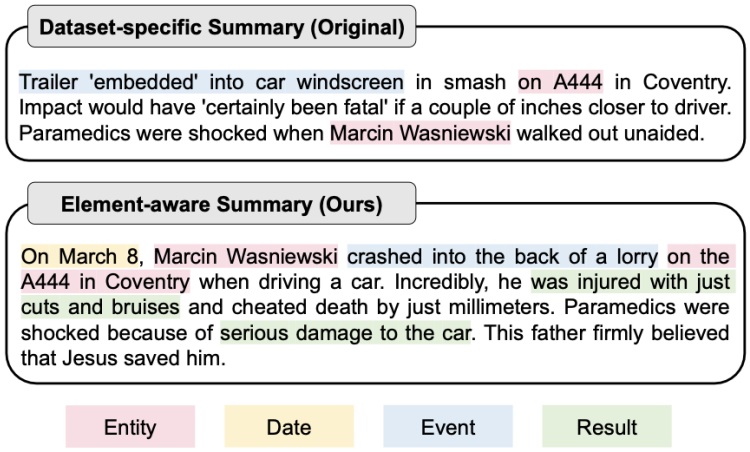


Figure 3.8: Comparison between the element-aware summary and original dataset-specific summary.

This technique elicit LLMs to generate summaries step by step, which helps them integrate more fine-grained details of source documents into the final summaries that correlate with the human writing mindset [26].

3.3.1 Implementation Pipeline

Expanding upon the LLM SumCoT segment of the paper, the authors show that their approach centers on a two-step pipeline outlined as follows:

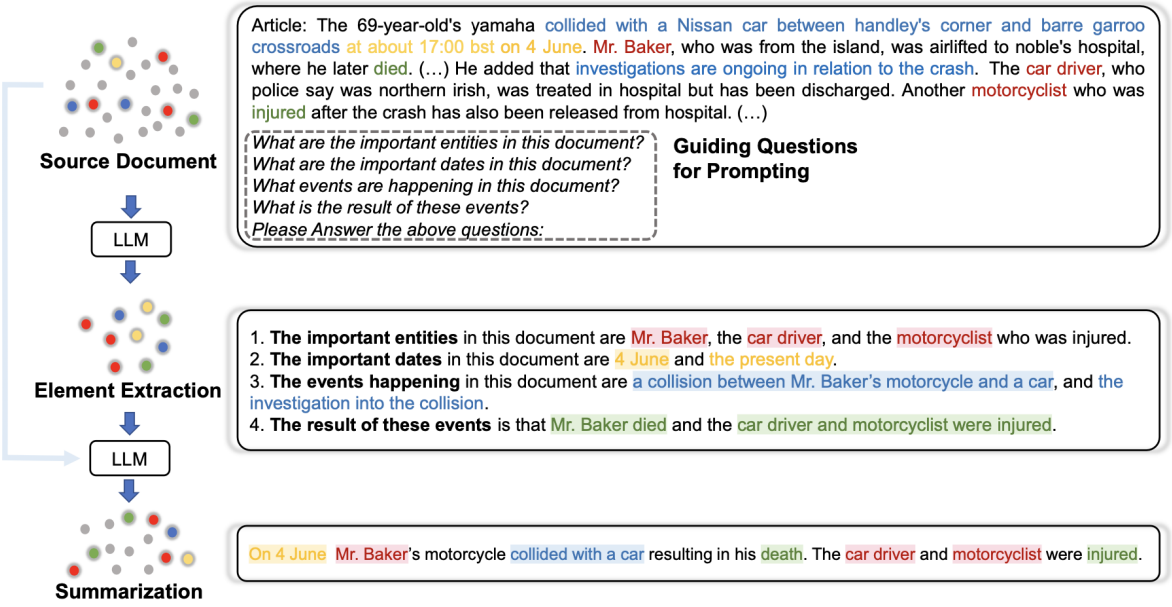


Figure 3.9: Full pipeline and example of The SumCoT method.

1. **Core element extraction**: They create guiding-question prompts to elicit the LLM to extract the four core elements: Entity, Date, Event, Result.

   For the i-th element, they set a simple question q_i to guide the model in its extraction process. Then, they concatenated these questions into $Q = [q_1, q_2, q_3, q_4]$.

   Let the source document be S, then the LLM input in this stage is formulated as $[S; Q]$.

2. **Multiple information integration and summarization**: With the obtained information, they integrated the extracted elements with the original

input article to obtain a more detailed information about the source document.

They then concatenated the source document, questions, answer, and a simple prompt $[p']$ = "Let's integrate the above information and summarize the article:" to prompt the LLM for summary generation.

The input in this stage is formulated as $[S; Q; A; [p']]$, and the output is the final summary.

Compared with the standard GPT-3, and previous SOTA, GPT-3 with their SumCoT technique obtains salient improvement in all ROUGE metrics when compared with the element-aware summaries.

Furthermore, they conducted human studies to compare summaries of GPT-3 without and with SumCoT. Results indicate that the SumCoT technique further improves the performance of the standard zero-shot paradigm in all dimensions, particularly coherence and relevance.

4. Data

The initial phase in our distillation/fine-tuning process focuses on selecting the appropriate training data.

As previously outlined, our objective is to distill the knowledge embedded within summaries generated by GPT, ultimately crafting a more compact model capable of summary creation as efficiently as GPT itself.

The choice to utilize summaries generated by GPT, as opposed to established datasets like CNN/DailyMail [15] or XSum [16], is motivated by an elementary decision: GPT-generated summaries exhibit better writing quality and align more closely with human summary expectations.

As highlighted in the SumCoT paper [26], summaries from these traditional datasets tend to suffer from issues of inaccuracy and redundancy.

Moreover, GPT serves as the baseline for many PwC internal services, and so there was a clear preference to adopt GPT-generated summaries as benchmark for accuracy.

With our objective in mind, we had to decide whether to create our dataset using the GPT API or to use a pre-existing collection of GPT-generated summaries.

The idea of generating our own dataset was quickly dismissed due to the complexity and scrupulosity required. Crafting a dataset via GPT API calls requires a well-designed prompt, able to ensure the consistent production of high-quality summaries, even on large dataset-scale.

Moreover, the time investment for such task was prohibitive. For instance, just to process the training portion of the CNN/DailyMail dataset, which contains 287,000 articles, considering an API response time of 10 seconds per entry, it would take over 33 days to complete. Not to mention the financial cost of performing this

many API requests.

Consequently, we opted to work with pre-existing GPT-generated summaries.

The only component we generated for our dataset was the CoT rationales, crucial for our step-by-step training.

It's crucial to note a section in the terms of use provided by OpenAI, which explicitly prohibits the utilization of generated outputs for the development of models that directly compete with OpenAI.

In light of this, it's important to emphasize that the work conducted here is strictly for research purposes.

4.1 GPT Created Datasets

The dataset we employed is a combination of two datasets, the details of which we'll delve into shortly.

The dataset comprises a total of 2097 entries, with each entry containing the reference article along with its corresponding GPT-generated summary.

	article	gpt3
0	(CNN) Two CNN Heroes are among the earthquake ...	Two CNN heroes who are working in Nepal after ...
1	Seoul (CNN) South Korea's Prime Minister Lee W...	Prime Minister Lee Wan-koo offered to resign o...
2	Sao Paulo, Brazil (CNN) Brazilian police have ...	Brazilian police have arrested the treasurer o...
3	(CNN) Most kids want to go out and play when t...	Zuriel Oduwole is a 12-year-old filmmaker from...
4	(CNN) A mammoth fire broke out Friday morning ...	A fire broke out at the General Electric Appli...
...	...	...
2092	A three-month-old baby boy was found dead afte...	Angela Williams found her 3-month-old son, Bob...
2093	(CNN) -- New guidelines for the management of ...	The Journal of the American Medical Associatio...
2094	A father-of-five has been stabbed to death at ...	Peter Kelly, father of five and wrestling coac...
2095	By . Bianca London . PUBLISHED: . 18:48 EST, 1...	In a unique collaboration, Disney and Harrods ...
2096	New Delhi (CNN) -- More than 200,000 people we...	Cyclone Hudhud triggers mass evacuations of ov...

Figure 4.1: The dataset utilized in this study.

A crucial point to note regarding the entries is that the length of the articles exceeds 512 tokens for 731 entries. Among these entries, 600 surpass the 1024 input length. The longest article has 4075 tokens.

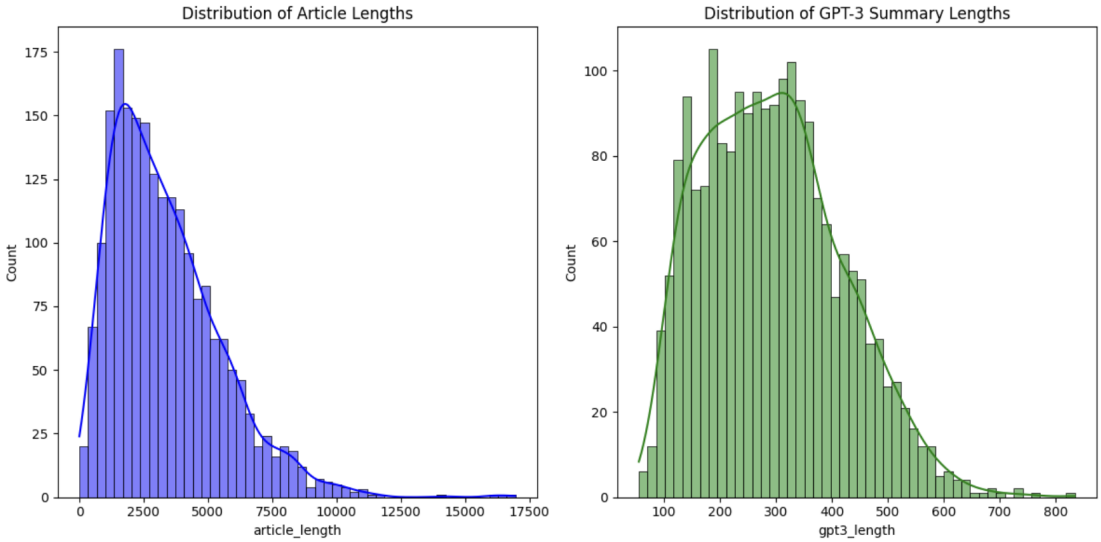


Figure 4.2: Character distribution of the articles and summaries.

4.1.1 News Summarization and Evaluation in the Era of GPT-3

This study [8] provided free access to their dataset, and thus to the GPT-generated output summaries of news articles. Specifically, researchers utilized OpenAI's *text-davinci-002* GPT-3 model to generate the summaries.

A combination of datasets, including CNN/DailyMail, XSum, and Newsroom, was utilized as the principal resource of articles for their summarization purposes.

4.1.2 GPT Chain-of-Density

The following dataset was created to investigate the potential of the Chain-of-Density (CoD) methodology applied to the most recent release of the OpenAI model, GPT-4 [1].

Considering the nature of the CoD approach, details of which we will not delve into, we have opted to adopt the third iteration of the generated summaries. This decision stems from the observation that these summaries align more closely with PwC's ideal "ground truth" summary.

4.2 CoT Rationales

The sole portion of the dataset that we had to craft was the one related to the CoT rationales.

Following the "Lasswell Communication Model" described in the SumCot paper [26], we developed four different rationales corresponding to the four fundamental questions employed for assessing the primary information of a summary:

- What are the important entities in this article?
- What are the important dates in this article?
- What events are happening in this article?
- What is the result of these events?

To construct the rationales for our training dataset, comprising 1781 entries, we employed the following prompt:

```
You are an article analyst and your duty is to answer questions
about articles.
Given the following article delimited by triple quotes:"""
{article}
"""
Please generate the following JSON answering its questions.
It's crucial that each answer must be an elaborated summary.
Each JSON value must be a single string between double quotes.
Do not forget the ',' after each entry.
{
"What are the important entities in this article?": ... ,
"What are the important dates in this article?": ... ,
"What events are happening in this article?": ... ,
"What is the result of these events?": ... ,
}"
```

The total duration required to produce the CoT section amounted to 1 hour and 38 minutes.

The overall cost amounted to €2.99, comprising €2.25 for prompt costs (based on 1,497,535 tokens sent) and €0.74 for completion costs (based on 371,457 tokens received).

4.3 Quality Remarks, Limitations and Ethic Statements

It is important to emphasize that utilizing LLM-generated entries for training purposes remains an area of study that requires careful consideration. Despite the promising outcomes observed, it's crucial to acknowledge that LLM's have inherent limitations in their reasoning capabilities, particularly when faced with more complex tasks. Even with the most meticulously crafted prompts, the results obtained may still fall short of optimal expectations.

In our limited case study, we encountered challenges with the CoT entries GPT had to generated for the dataset. As notable in the prompt, we had to ensure that ChatGPT adhered to the prescribed JSON structure for its outputs, as it frequently deviated from the specified format.

Within the 1781 training entries utilized for the CoT generation process, two of the resulting CoT outputs exhibited formatting errors. These issues arose from GPT's tendency to incorrectly interpret single and double quotation marks in the JSON values, usually resulting in a confusion between these marks and proper names within the text.

```
"What are the important entities in this article?": "New York state
authorities, Gov. Andrew Cuomo, synthetic marijuana, 'spice' or "K2",
New York State Health Commissioner Dr. Howard Zucker,
New York State Department of Health, New York Alcohol
and Substance Abuse Services Commissioner Arlene Gonzalez Sanchez,
Alabama Department of Public Health, Mobile County,
Mississippi Poison Control Center",
```

Additionally, it is important to recognize that the behavior of student models is influenced by biases inherited from the teacher model. These biases can inadvertently shape the output and must be taken into account when interpreting the results.

5. Pipeline & Practical Insight

In this section, we will explore the strategies employed in our project to address the challenges encountered and the overall structure of our approach.

Additionally, we will delve into the analysis of the primary supportive ecosystem of our project: HuggingFace.

5.1 HuggingFace

HuggingFace is a renowned platform and community-driven library for NLP tasks. It offers a comprehensive suite of tools, pre-trained model checkpoints, and datasets that significantly aid and enhance NLP research and development.

One of the key strengths of HuggingFace is its vast collection of pre-trained transformer models, which have defined various NLP applications, enabling a simple and powerful access to their implementation.

The extensive suite of tools and libraries offered by HuggingFace serves as a comprehensive resource for training and utilizing these models, positioning itself as a focal point within the AI community.

5.2 General Approach

Our methodology draws upon a combination of principles from two key research papers: the Distilling Step-by-Step! [13] and the SumCoT [26] paper.

We sought to explore the potential of applying the CoT methodology to enhance the distillation of knowledge from LLMs into more compact yet effective models.

The Distilling Step-by-Step approach served as the core template for structuring

our methodology, outlining the primary framework we aimed to adopt.

Additionally, we delved into the SumCoT paper to deepen our understanding and gather insights into the fields of summarization applied to CoT. This literature provided valuable guidance and practical knowledge essential for our research.

By incorporating both proposed methodologies, we composed a unified approach. Our goal was to leverage the CoT procedure within the domain of summarization, aiming to condense the knowledge encapsulated within larger LLMs into more manageable and efficient ones.

5.3 Implemented Model

Following the Distilling Step-by-Step approach, we opted to employ a T5 model, specifically opting for the Long T5 variant.

This choice stemmed from the recognition that the 512 maximum token input length posed overly stringent limitations. Truncating long input articles might lead to the loss of crucial content, such as the conclusion.

Furthermore, the decision to adopt T5 as our primary architecture was underpinned by its exemplary performance across various tasks. T5 stands out for its versatility and robustness, delivering high-quality results in many NLP tasks, making it an ideal choice for our multi-task needs.

We deployed two variations of the Long T5 model for our project: one served as our baseline (base_model), while the other was tailored for our CoT approach (cot_model).

The key distinction between the two models lies in the type of input they receive. Beyond this difference, both models are optimized using the AdamW optimizer alongside a linear learning rate scheduler.

Each model underwent training using hyperparameters deemed most optimal for its respective variation. This approach ensured that both models were fine-tuned to perform optimally within their designated contexts.

5.4 Preprocessing & Tokenization Process

When it came to preprocessing, we made a joint decision with our internal supervisors to refrain from conducting any preprocessing operation. This choice was motivated by two primary reasons: the preservation of data integrity and the prevention of data loss.

The tokenization process was executed using the library-provided HuggingFace tokenizer.

For each input sequence, the process followed a consistent pattern: prefix + sentence.

In detail, the input articles were pre-pended with the prefix *"summarize: "* while the rationales were constructed with the sequence *"question: "* + CoT question + *" context: "* + article.

This pattern mirrors the prefix structure employed in the T5 model, allowing us to maximize the model's performance.

We set a maximum token length of 2000 for the input and 512 for the output in our tokenization process. This decision was influenced by the fact that we wanted to maintain as much of the original article as possible. However, due to this input limit, 61 of our 1781 training samples had to be truncated.

Despite our intention to retain as much of the input article as possible, the step-by-step CoT approach applied to summarization presented its initial limitation.

Given that we were effectively addressing a multi-task problem, we adopted and implemented the loss function defined in the formula 5.3. Consequently, our cot_model needed to execute a total of 5 forward passes (one for the summary input and four for the rationale inputs) before proceeding with the backward pass.

This resulted in the duplication of the same, typically lengthy, article five times for a single entry, imposing significant demands on GPU memory.

In light of these considerations, we made the decision to cap the maximum number of tokens at 2000 to strike a balance between preserving the maximum possible article tokens and avoiding out-of-memory issues.

As result, each component of the process produced a triplet comprising its input IDs, its attention mask, and its label, forming the input for our model.

5.5 Training Pipeline

Now we will analyse the training pipelines employed in both the base_model and the cot_model.

For the base_model, the training process was relatively straightforward. We provided the model with the reference article along with its corresponding summary as label. The model then made a prediction based on the input article, and computed the cross-entropy loss comparing the prediction to the actual summary.

On the other hand, for the cot_model we had to repeat, as mentioned earlier, the forward pass multiple times, each time providing the model with the different input needed. We then aggregated the losses from each input into a single multi-task loss.

Specifically, in our CoT step-by-step approach to summarization, we defined the rationale loss $L_{rationale}$ as follows:

$$L_{rationale} = L_{rationale_1} + L_{rationale_2} + L_{rationale_3} + L_{rationale_4} \tag{5.1}$$

Where:

$$L_{rationale_n} = \ l(f(x_i^n), r_i^n) \tag{5.2}$$

Following the Distilling Step-by-Step approach [13], we introduced a multiplicative factor λ to down-weight the importance of the rationale losses. This allowed the model to focus more on the summarization task.

$$L = L_{label} + \lambda L_{rationale} \tag{5.3}$$

Due to the memory limitations we mentioned earlier, we were unable to perform

training using batch sizes larger than one. To work around this issue and perform batched training to prevent overfitting and ensure generalization, we employed a technique called gradient accumulation.

With gradient accumulation, we were able to simulate the behavior of larger batches, improving the model's ability to learn from a wider variety of inputs and making it more robust to our limited input data.

Ultimately, it is important to note that during both the process of determining validation loss and in the ultimate assessment of the models, we solely focused on the performance of the generated summaries, without directly evaluating the generated CoT responses.

6. Experiments Run

6.1 Training Models

To achieve our objective of creating a CoT step-by-step distilled summarization model, we conducted multiple training sessions with our models, adjusting hyperparameters to find the most effective configuration.

Below is a breakdown of our training configuration and variables:

- **Hyperparameters**

  Only the models with the best performance were retained, and due to the challenges of assessing a good summarization model, we opted to retain two variations of each model. These two variations include: the best performing model based on training performance (specifically, the validation loss), and the best performing model based on the ROUGE score.

  The base_model_book model appears only once since it was both the best performing model both training and rouge wise.

  The hyperparameters chosen for training are provided in Table 7.1.

- **Datasets, Splits and Repeatability**

  All models are consistently trained on the exact same dataset, with the simple exception of the cot_model utilizing the expanded version with the CoT answers.

  The train/test split has also been the exact same for each trial: 1781 entries for the training data, 115 entries for the validation, and 200 entries for the test set.

Each set generation and each trial has been performed with a fixed seed of '42'.

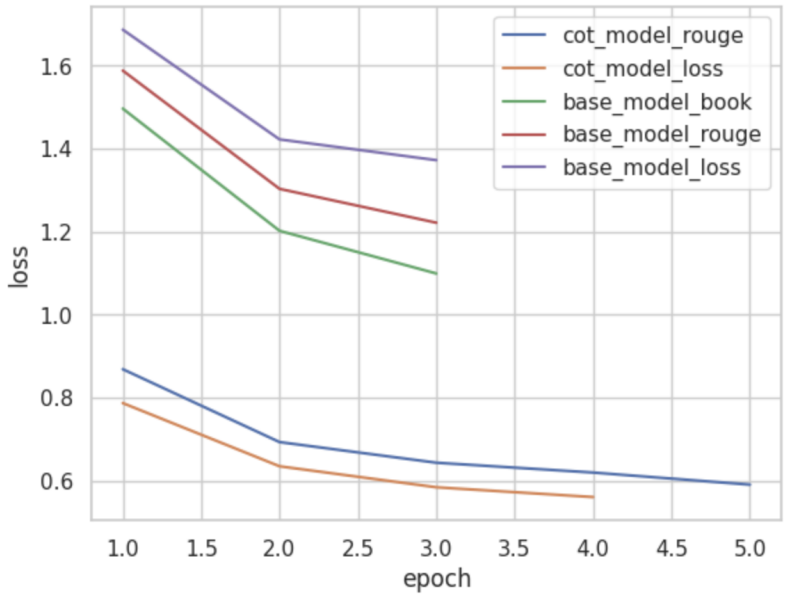
- **Hardware Specifications**

  The training procedures of the base_model were performed utilizing four NVIDIA T4 16GB.

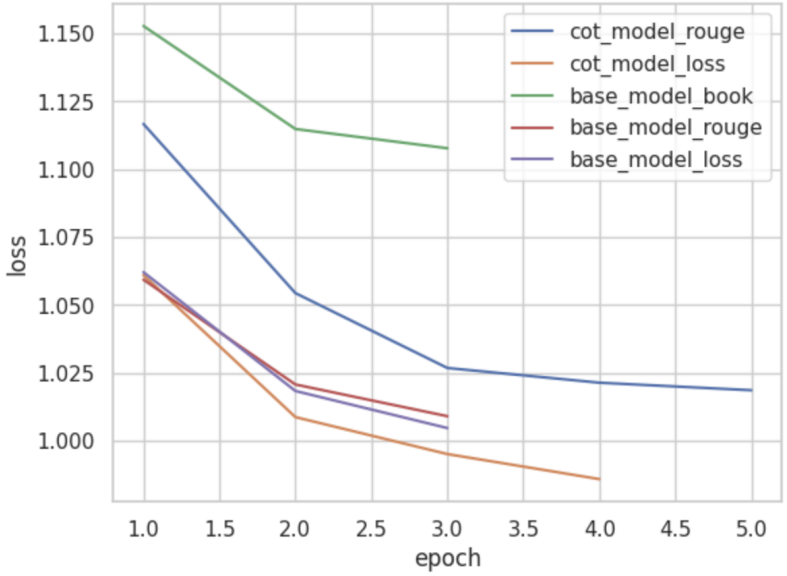
  The training procedure of the cot_model were performed utilizing a NVIDIA A100 40GB.

Model Name	Model Checkpoint	No. Epochs	Batch Size	Learning Rate	Lambda
base_model_rouge	long-t5-global-base	3	4	1e-4	
base_model_loss	long-t5-global-base	4	4	3e-5	
base_model_book	pszemraj/long-t5-tglobal-base-16384-book-summary	3	4	1e-4	
cot_model_rouge	long-t5-global-base	5	Gradient accumulation 8	5e-5	0.7
cot_model_loss	long-t5-global-base	4	Gradient accumulation 8	3e-5	0.7

Table 6.1: Hyperparameters of the best model variations.



(a) Training losses



(b) Training validation losses

Figure 6.1: Comprehensive comparison between losses and validation losses

6.1.1 Training base_model

We conducted several training sessions utilizing two model checkpoints to assess whether utilizing a checkpoint that had already been fine-tuned would enhance performance.

Unfortunately, we were unable to find a Long T5 model with global attention that had been specifically fine-tuned for news summarization tasks. As a result, we experimented with a model that had been trained for book summarization.

In total, we carried out 10 training sessions. Of these, 7 sessions were conducted using the standard 'long-t5-global-base' checkpoint, and the remaining 3 utilized the fine-tuned checkpoint.

Each training session required approximately one hour to complete. The cost per hour was $1.238, and considering the entire process of training and evaluation, the cumulative cost amounted to $14.17.

6.1.2 Training cot_model

We conducted several training sessions utilizing the 'long-t5-global-base' checkpoint, for a total of 9 training trials.

Each training session required approximately two hours to complete. The cost per hour was $3.354 , and considering the entire process of training and evaluation, the cumulative cost amounted to $36.89.

6.2 Analyzing Model Outputs

In order to generate the best summaries from the model, we conducted a grid search of the hyperparameters for generation, retaining those that consistently produced a higher ROUGE score on average between models.

It is also interesting to delve into the variations among the different summaries generated. In this particular segment, we will present the original Newsroom summary and GPT summary as a reference, along with the three summaries produced by our models.

Reference database summary:

```
It’s the latest sign the company is rethinking its menu
to keep up with changing tastes.
```

Reference GPT summary:

```
McDonald's is changing its grilled chicken recipe to remove
ingredients that people might not recognize, in an effort to
appeal to customers who are looking for more wholesome options.
The company is also trying to freshen up its image and shake
perceptions that it serves junk food.
```

base_model_rouge summary:

```
McDonald's is simplifying its grilled chicken recipe to remove
ingredients people might not recognize, marking the latest sign
the company is rethinking its menu to keep up with changing tastes.
The new "Artisan Grilled Chicken" will be in its more than
14,300 US stores by the end of next week, in products including a
new sandwich, as well as existing sandwiches, wraps, and salads.
```

base_model_book summary:

```
The article discusses McDonald's new Artisan grilled chicken recipe,
which will be available in 14,300 US store stores by the next week.
The company is reducing its ingredients list to remove ingredients
that might not be recognized, and is hoping to change perceptions
of the food.
```

cot_model_rouge summary:

```
McDonald’s is simplifying its grilled chicken recipe to remove
ingredients people might not recognize, marking the latest sign
the company is rethinking its menu to keep up with changing tastes.
```

```
The change comes as McDonald's fights to hold onto customers amid
the growing popularity of places like Chipotle that position
themselves as more wholesome alternatives to traditional fast-food.
```

Based on the analysis of the original article, several key concepts emerge that the summary must encapsulate: the change of the recipe, the effort to meet customer expectations and the need to revitalize the company's image.

We can now delve in a closer examination of the provided summaries:

- The GPT summary used as new ground truth, as verified in the evaluation of GPT-generated summaries [8], stands out for its enhanced human-like quality compared to the reference dataset summary.
- Although the base_model's summary encapsulates all primary concepts, a minor semantic error is noticeable in its conclusion (i.e. "... in products including a new sandwich, as well as existing sandwiches ...").
- The cot_model, on the other hand, surpasses the base_model's summary by delivering a well-articulated summary that not only encapsulates all concepts but also incorporates an additional detail (i.e "... growing popularity of places like Chipotle that position themselves as more wholesome alternatives to traditional fast-food."), effectively accentuating the importance of refreshing the company's image.
- Noteworthy is the commendable performance of the base_model_book, which delivers a finely articulated summary with all main concepts. Additional peculiarity is its introduction, employing the phrase "The article discusses", thereby establishing a more "human" tone probably inherited from its previous training process.

7. Results

In this chapter, we present the outcomes of our study, along with the standards and comparisons utilized to assest the quality of the generated summaries.

Our primary objective was to create a model capable of producing summaries that are of high quality, coherent, informative, and, most importantly, more "human-like".

However, the assessment of the summary quality poses a significant challenge.

Ideally, the most optimal approach to evaluate the effectiveness of a summarization system relies on human judgment. Here, human annotators would evaluate the quality of summaries based on several criteria, such as coherence or informativeness. While this method provides valuable insights and incorporates real human feedback, it is time-consuming and costly. Moreover, human judgment can vary from one annotator to another, rendering it difficult to achieve consistent evaluations.

To streamline the evaluation process, numerous studies have proposed automated evaluation metrics.

The most prevalent automatic evaluation metrics for summarization are reference-based methods, which directly compare the similarity of the generated summaries to dataset-specific summaries. However, these conventional text overlap-based metrics have their limitations.

For instance, they cannot estimate how much information the summaries have in common. Additionally, in our specific case study, evaluating prompt-based summaries not trained to mimic the original ground truth using these standard metrics may lead to suboptimal evaluation, even if the summaries are not necessarily bad.

Research has also explored newer evaluation methodologies, such as reference-

free metrics. These metrics aim to evaluate the quality of a summary by utilizing specifically trained language models to approach the evaluation process as a language understanding task.

Unfortunately, all the methods discussed above have been shown to be completely ineffective at evaluating GPT-3 summaries[8].

Dataset	BRIO		T0		GPT3	
	Best ↑	Worst ↓	Best ↑	Worst ↓	Best ↑	Worst ↓
CNN	36	24	8	67	58	9
BBC	20	56	30	29	57	15

Figure 7.1: From the News Summarization and Evaluation in the Era of GPT-3 paper [8], the percentage of times a summarization system is selected as the best or worst according to majority vote. Human annotators have a clear preference for GPT3-D2 for both CNN and BBC style summaries.

Dataset	Model	Overlap-Based			Similarity-Based		QAEval	
		ROUGE(1/2/L)	METEOR	BLEU	BERTScore	MoverScore	EM	F1
CNN	PEGASUS	34.85/14.62/28.23	.24	7.1	.858	.229	.105	.160
	BRIO	38.49/17.08/31.44	.31	6.6	.864	.261	.137	.211
	T0	35.06/13.84/28.46	.25	5.9	.859	.238	.099	.163
	GPT3-D2	31.86/11.31/24.71	.25	3.8	.858	.216	.098	.159
DailyMail	PEGASUS	45.77/23.00/36.65	.33	12.2	.865	.308	.159	.229
	BRIO	49.27/24.76/39.21	.37	11.7	.871	.331	.175	.259
	T0	42.97/19.04/33.95	.28	8.9	.863	.290	.121	.184
	GPT3-D2	38.68/14.24/28.08	.26	6.6	.859	.248	.101	.159
XSum	PEGASUS	47.97/24.82/39.63	.36	9.8	.901	.362	.145	.221
	BRIO	49.66/25.97/41.04	.39	10.6	.901	.372	.139	.224
	T0	44.20/20.72/35.84	.34	8.0	.896	.340	.125	.208
	GPT3-D2	28.78/7.64/20.60	.19	2.2	.869	.197	.066	.119
Newsroom	PEGASUS	39.21/27.73/35.68	.39	.14	.873	.272	0.182	0.253
	BRIO	-	-	-	-	-	-	-
	T0	25.64/9.49/21.41	.20	.04	.849	.145	.080	0.125
	GPT3-D2	27.44/10.67/22.18	.22	.05	.859	.159	.089	0.142

Figure 7.2: From the News Summarization and Evaluation in the Era of GPT-3 paper [8], the performance of different summarization methods measured using reference-based automatic metrics. It can be observed that automatic metrics report substantially worse results for GPT-3-D2 summaries compared to fine-tuned models. This directly contradicts the human preference results from the previous figure, demonstrating that these reference-based metrics cannot reliably compare the quality of prompt-based summaries against fine-tuned summaries

7.1 Evaluation Metrics

In our research, we chose to employ three primary metrics to assess the quality of our model.

The following are the metrics that were used in our evaluation process

7.1.1 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

ROUGE is a set of metrics commonly used in text summarization and machine translation. ROUGE measures the quality of a generated summary or translation by comparing it with one or more reference summaries or translations.

The most commonly used ROUGE metrics are:

- **ROUGE-N**: Measures the n-gram overlap between the generated and reference texts. The value of N determines the length of the n-grams considered (e.g., ROUGE-1 considers unigrams, ROUGE-2 considers bigrams, etc.).
- **ROUGE-L**: Measures the longest common subsequence between the generated and reference texts. It takes into account the order of the words. ROUGE-L calculates precision, recall, and F1-score.
- **ROUGELSUM**: It's an extension of ROUGE-L that also considers unigram overlap. It measures the similarity of the generated output to the reference output, considering word order and unigram overlap

7.1.2 BiLingual ANnotation Consistency (BLANC)

While originally intended for the evaluation of machine translation, the BLANC metric has been repurposed to assess the quality of text summaries. Two variations of this metric, BLANC-Help and BLANC-Tune, were introduced for this purpose.

Although we only employed BLANC-Help in our study due to time constraints, it is worth noting that these two metrics are highly correlated, with a correlation coefficient of 90%, suggesting that either one could be applied in most cases [2].

Behind BLANC lies the idea that an effective summary should enable a reader to grasp the main points of a document without the need of a full read-through. To evaluate this, the metric employs a fill-in-the-blank task to gauge how well a summary aids in comprehending the source material [22].

In our analysis we utilized the standard implementation of BLANC-Help which, with the gap value set to 2, has been demonstrated to have the strongest correlation with human scores [23].

The BLANC values can range from -1 to 1, but it has been shown that, typically, results are between 0 (summary is useless) and 0.3 (summary provides 30% help under the masked token task).

7.1.3 Human Evaluation

Finally, we also decided to conduct a human assessment ourselves to obtain a deeper understanding of how the summaries were created.

We opted to use 5 samples from the CNN/DailyMail dataset for each model and compare them against the dataset's ground truth representation.

In this evaluation approach, our criteria aimed to award points to models based on their ability to generate summaries that excelled in three key aspects we identified:

1. **Accuracy of the Summary**: This criterion gauged how effectively the model summarized the content, aligning with the principles of the Lasswell Communication Model.

2. **Faithfulness to the Article**: We assessed the extent to which the generated summary accurately reflected the content and objectives of the original article and summary.

3. **Human-like Production**: This aspect measured the degree to which the generated summary exhibited characteristics akin to human-produced summaries, which could resonate with human readers.

7.2 Results

7.2.1 ROUGE Scores

ROUGE				
Model Name	rouge1	rouge2	rougeL	rougeLsum
base_model_loss	0.404	0.211	0.318	0.318
base_model_rouge	0.417	0.210	0.323	0.322
base_model_book	0.379	0.131	0.277	0.276
green cot_model_loss	0.417	0.219	0.328	0.327
cot_model_rouge	**0.421**	**0.220**	**0.333**	**0.332**

Table 7.1: ROUGE scores evaluated on our dataset.

ROUGE				
Model Name	rouge1	rouge2	rougeL	rougeLsum
base_model_loss	0.314	0.114	0.222	0.263
base_model_rouge	0.314	0.113	0.222	0.261
base_model_book	0.288	0.067	0.192	0.231
cot_model_loss	**0.322**	**0.118**	**0.227**	**0.267**
cot_model_rouge	0.320	0.115	0.225	0.266
T5-11B (SOTA on the test set)	0.435	0.216	0.407	

Table 7.2: ROUGE scores on the CNN/DailyMail dataset over 500 entries.

It is evident that the cot_model stands out as the most effective model when evaluated using the ROUGE text-overlap metric, demonstrating improved performance on both the original dataset and the one generated by GPT.

This observation underscores its advanced ability to extract key concepts for summarization, suggesting a more profound understanding on how to distill essential information from the given articles and thus producing better "overlapping" results with the corresponding ground truths.

7.2.2 BLANC-Help Scores

BLACN_help	
Model Name	Mean Score
CNN/DailyMail Ground Truth	0.076
base_model_loss	0.106
base_model_rouge	0.105
base_model_book	0.077
cot_model_loss	0.113
cot_model_rouge	**0.121**

Table 7.3: BLANC_help mean scores on the CNN/DailyMail dataset over 200 entries.

Even when evaluated against a more informative-oriented metric like BLANC-Help, our cot_model consistently outperforms all other models. This indicates that the summary generated by our cot_model significantly enhances the performance of a pre-trained language model when it comes to understanding the original article under a masked token task.

Furthermore, we conducted a baseline evaluation by utilizing the ground truth original summaries from CNN/DailyMail the dataset. Surprisingly, the ground truth summaries, which one would expect to perform well, actually exhibited the poorest performance according to the BLANC-Help metric.

7.2.3 Human Evaluation Scores

Human Evaluation

Model Name	Votes
base_model_book	**2**
base_model_loss	1
base_model_rouge	1
cot_model_loss	1
cot_model_rouge	1

Table 7.4: Human evaluation performed on our dataset over 5 entries.

Human Evaluation

Model Name	Votes
base_model_book	2
base_model_loss	0
base_model_rouge	0
cot_model_loss	**3**
cot_model_rouge	**3**

Table 7.5: Human evaluation performed on the CNN/Dailymail dataset over 5 entries.

The limited selection of articles and corresponding summaries that were reviewed in our human evaluation unfortunately do not offer a robust foundation for drawing definitive conclusions. However, they did yield intriguing insights into the quality of the generated summaries which we will further delve into in the next section.

8. Discussion

8.1 Results Analysis

Based on the findings, we observed that our CoT step-by-step approach for summarization led to a point of improvements in almost every metric.

These improvements were manifold and of various nature.

Initially, we noted an increase in the average quality of summaries generated by our cot_model compared to the quality provided by the base_model. Although there were instances where the cot_model's summaries did not deviate significantly from those of the base_model, we noted that the cot_model generally delivered more precise and articulated summaries with regards to the article's subjects.

Another significant enhancement was the ability to perform longer training sessions, thanks to the multi-task objective. This mitigated the immediate risk of overfitting on our limited dataset. While the base_model typically reached a maximum of four training epochs before exhibiting signs of overfitting, the cot_model could sustain seven epochs under identical training conditions.

Unexpectedly, the base_model_book, despite being the least performing model according to the evaluation metrics, displayed commendable results creating even, to our belief, the most "human-like" summaries.

Unfortunately, we were unable to verify its performance using our CoT step-by-step approach for summarization.

The overall results indicate that the cot_model outperformed all other models, demonstrating that the additional knowledge integrated into its training process was indeed beneficial.

However, it is important to approach these results with caution, as there is considerable uncertainty regarding the correlation between higher metric results and the actual quality of the summaries. The base_model_book serves as a prime example of this uncertainty.

Additionally, a more comprehensive evaluation could have been achieved through human evaluation to further analyze the differences between the loss and rouge based models, but time constraints prevented us from doing so.

8.2 Feasibility & Sustainability

The primary objective of our research was to develop a more compact and eco-friendly model that still holds results comparable to larger LLMs in the downstream task of summarization. We are confident that we are moving in the right direction.

Although our results do not yet match nor surpass the capabilities of services like ChatGPT, we firmly believe that the potential enhancements as well as the decrease in computational demands and environmental costs of training are commendable.

In this study, we utilized the base version of Long T5, with 250 million parameters. This represents a 700-fold decrease in parameter size compared to GPT-3's 175 billion parameters.

To put this into perspective, the storage space required for our entire CoT model is just 944.7 MB, while GPT-3's parameters alone require 350 GB. Our model can comfortably fit on a commonly used NVIDIA T4 16GB GPU, whereas an in-house LLM of GPT-3's scale would necessitate at least 22 of these GPUs.

The training of our CoT model is the only computationally intensive aspect, necessitating at least an NVIDIA A100 40GB for its inputs. Nevertheless, even a future, more advanced version of our model would require similar resources, still falling short of the base demands of GPT-3.

8.3 Licenses

The next concern to address is the licensing and restrictions associated with the components we utilized in our research, assuming a future product release.

Regarding the model, the T5 model, developed by Google Research, is released under the Apache License 2.0. This license permits users to utilize, modify, and distribute the software at no cost, with limited restrictions. However, it does require that any derivative work or redistributed code must include a copy of the original license and copyright notice.

Therefore, ideally, this model could still be the preferred choice for an actual implementation.

The primary limitation we encountered, as described in the Data section of this study, was the construction of the dataset.

While a larger and more curated dataset would likely have yielded superior results, even aside from the time and financial resources required to generate such a quantity of prompts for the ChatGPT API, it would still not be possible to utilize the resulting dataset for the creation of a product for use and sale, due to OpenAI's Terms of Service (ToS).

One potential solution to this issue could be to employ open-source LLMs, such as LLaMA 2 or the newly released Gemma.

9. Conclusion

This thesis aimed to construct a commercially licensed summarization model as a component of an internal framework. Despite the strict dataset requirements and still to improve results, we made remarkable advancements with our CoT method, showing the potentiality of this technique for future progress.

Among the models we used, our CoT model exhibited the best performance, showing improvements across all metrics and enhancing the overall quality of generated summaries.

Our step-by-step CoT knowledge distillation method seemed to surpass the traditional knowledge distillation by fine-tune approach, elevating the quality of the results the student model can produce.

Additionally, using semi-automated labeling opened up a promising frontier. Correctly supervised, this technique could enhance the quality of training datasets and ultimately lead to better models.

By combining these techniques, we have contributed valuable insights to the ongoing exploration of this novel approach.

9.1 Future Work

We now present an overview of potential future research and improvements in our work. These include experiments we could not pursue due to time and resource limitations and ideas that may contribute to improving the utilized approach.

- **Dataset curation**:

  We believe that a larger and carefully curated dataset could improve the

training of our CoT model. A more precise and well defined standard of target summaries would facilitate learning and improve base performance, which could be further enhanced using the rationales.

- **Longer training sessions**:

  With a larger dataset, we could conduct longer and more accurate training sessions, potentially enhancing performance, generalizability and results.

- **Advanced hyperparameter research**:

  Due to our constraints, we could not explore a wider range of hyperparameter combinations. In future implementations, a more extensive analysis of the λ hyperparameter, in addition to the classic variations of learning rate, number of epochs, and optimizer, could yield interesting results.

- **Distillation on larger models**:

  It would be interesting to conduct the same training with larger iteration of the Long T5 model, such as the large checkpoint with 780 million parameters. This model would still be small enough to ensure energy and storage efficiency while potentially leading to further performance improvements.

- **Distillation on previously fine-tuned checkpoints**:

  As noted with the book summarization checkpoint, further research on this kind of transfer learning could lead to improved performance and generalizability. However, caution should be used, as an improvement in performance does not necessarily translate to improved metric performance.

- **Comprehensive metric analysis**:

  Existing text overlap-based and reference-free metrics for evaluating summary quality do not fully capture human evaluation. It would be valuable to further analyze other metrics and their correlation with produced summaries.

- **Adaptation for longer inputs**:

  Our CoT method is constrained by its requirement for multiple input sentences, which frequently leads to GPU memory saturation. To overcome this challenge and facilitate summary generation for longer input documents, a

hybrid approach combining extractive summarization with our abstractive summarization technique could be interesting to explore.

In conclusion, we are excited about the significant progress being made in the field of Artificial Intelligence, with particular interest in the field of Natural Language Processing.

The environment we examined in this thesis, though constrained, yielded intriguing and promising outcomes. We look forward to continuing this journey of exploration and discovery.

Bibliography

[1] ivanleomk/gpt4-chain-of-density. https://huggingface.co/datasets/ivanleomk/gpt4-chain-of-density?row=13.

[2] Primerai/blanc. https://github.com/PrimerAI/blanc?tab=readme-ov-file.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[4] Lochan Basyal and Mihir Sanghvi. Text summarization using large language models: A comparative study of mpt-7b-instruct, falcon-7b-instruct, and openai chat-gpt models, 2023.

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

[6] Krishna Teja Chitty-Venkata, Sparsh Mittal, Murali Emani, Venkatram Vishwanath, and Arun K. Somani. A survey of techniques for optimizing transformer inference, 2023.

[7] Michael Desmond, Evelyn Duesterwald, Kristina Brimijoin, Michelle Brachman, and Qian Pan. Semi-automated data labeling. In Hugo Jair Escalante and Katja Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition*

and Demonstration Track, volume 133 of *Proceedings of Machine Learning Research*, pages 156–169. PMLR, 06–12 Dec 2021.

[8] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. News summarization and evaluation in the era of gpt-3, 2023.

[9] Alex Graves. Sequence transduction with recurrent neural networks, 2012.

[10] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences, 2022.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Thi Kieu Khanh Ho and Jeonghwan Gwak. Utilizing knowledge distillation in deep learning for classification of chest x-ray abnormalities. *IEEE Access*, 8:160749–160761, 2020.

[13] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023.

[14] Chengqiang Lu, Jianwei Zhang, Yunfei Chu, Zhengyu Chen, Jingren Zhou, Fei Wu, Haiqing Chen, and Hongxia Yang. Knowledge distillation of transformer-based language models revisited, 2022.

[15] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond, 2016.

[16] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018.

[17] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi

Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021.

[18] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[21] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. Ul2: Unifying language learning paradigms, 2023.

[22] Oleg Vasilyev, Vedant Dharnidharka, and John Bohannon. Fill in the BLANC: Human-free quality estimation of document summaries. In Steffen Eger, Yang Gao, Maxime Peyrard, Wei Zhao, and Eduard Hovy, editors, *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 11–20, Online, November 2020. Association for Computational Linguistics.

[23] Oleg Vasilyev, Vedant Dharnidharka, Nicholas Egan, Charlene Chambliss, and John Bohannon. Sensitivity of blanc to human-scored qualities of text summaries, 2020.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[25] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help, 2021.

[26] Yiming Wang, Zhuosheng Zhang, and Rui Wang. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method, 2023.

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[28] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.