ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

THE SUMMARY IMITATION GAME: IMPROVING ABSTRACTIVE SUMMARIZERS VIA NEURAL-APPROXIMATED DECODING STRATEGIES

Elaborato in
Programmazione di Applicazioni Data Intensive

Relatore
Prof. Gianluca Moro

Co-relatori
Dott. Giacomo Frisoni

Presentata da
Benedetta Pacilli

Seconda Sessione di Laurea
Anno Accademico 2022 – 2023

KEYWORDS

Natural Language Processing
Language Models
Decoding Strategies
Natural Language Generation
Transformer-based Models

To define is to limit.
-Oscar Wilde

Ai miei genitori

Sommario

Negli ultimi anni, la *abstractive summarization* ha subito notevoli progressi grazie all'emergere di nuovi modelli linguistici neurali, architetture basate sui *transformers*, l'utilizzo di spazi ad alta dimensionalità, la disponibilità di ampi set di dati e *task* innovative di pre-addestramento. In questo panorama in evoluzione, le strategie di *decoding* giocano un ruolo cruciale nel trasformare le distribuzioni di probabilità generate da questi modelli in testi coerenti, seguendo un approccio autoregressivo.
I sistemi di *summarization* prendono numerose decisioni sulle proprietà di sintesi durante l'inferenza, come il grado di copiatura, la specificità e la lunghezza degli output, ecc.
Goyal et al. nel 2021 [1] hanno introdotto Hydrasum, un modello di *summarization* con molteplici *decoder*. La peculiarità di Hydrasum risiede nella sua architettura, che comprende più canali di decodifica. HydraSum fornisce un semplice meccanismo per ottenere riassunti stilisticamente diversi campionando da singoli *decoder* o da loro miscele attraverso un meccanismo di *gating*.
Questa tesi propone un'architettura che comprende due *large language model* in grado di replicare diverse strategie di *decoding*. Data la non differenziabilità intrinseca delle strategie di *decoding*, viene impiegato un approccio innovativo in cui l'emulazione di queste strategie è ottenuta attraverso l'integrazione di un secondo *language model*, piuttosto che affidarsi a *layer* differenziabili per ciascuna strategia. In questa configurazione, al *language model* iniziale viene fornito un testo in *input* destinato alla sintesi, ma l'obiettivo primario non è la generazione della sintesi stessa. Invece, il primo *language model* calcola la distribuzione di probabilità di ciascun *token* per il testo in *input*. Successivamente, queste distribuzioni di probabilità vengono utilizzate come *input* per il secondo *language model*, responsabile di produrre il *token* che corrisponde alla selezione che verrebbe effettuata da una specifica strategia di *decoding*.

Abstract

In recent years, abstract summarization has undergone significant advancements driven by the emergence of novel neural language models, transformer-based architectures, the utilization of high-dimensional spaces, the availability of extensive datasets, and innovative pre-training tasks. Within this evolving landscape, decoding strategies play a crucial role in transforming the probability distributions generated by these models into coherent text, following an autoregressive approach.
Summarization systems make numerous *decisions* about summary properties during inference, e.g. degree of copying, specificity, and length of outputs, etc. However, these are implicitly encoded within model parameters and specific styles cannot be enforced.
Goyal et al. in 2021 [1] introduced Hydrasum, a summarization model with multiple decoders. The peculiarity of Hydrasum relies upon its architecture, which includes multiple decoding channels. HydraSum provides a simple mechanism to obtain stylistically diverse summaries by sampling from either individual decoders or their mixtures through a gating mechanism.
The thesis presents an architecture comprising two Large Language Models that replicate distinct decoding strategies. Given the inherent non-differentiability of decoding strategies, an innovative approach is employed wherein the emulation of these strategies is achieved through the integration of a second language model, rather than relying on differentiable layers for each strategy. In this configuration, the initial language model is provided with input text intended for summarization, but the primary objective is not the generation of the summary itself. Instead, the first language model computes the probability distribution of each token for the given input text. Subsequently, these probability distributions are utilized as input for the second language model, responsible for producing the token that corresponds to the selection made by a specific decoding strategy.

Introduzione

Motivazioni e contributi

Negli ultimi anni, il campo del *Natural Language Processing* (NLP) ha visto notevoli progressi, grazie all'avvento di nuovi modelli linguistici neurali e di architetture basate su trasformatori. Tra le varie applicazioni dell'NLP, una che ha registrato progressi significativi è l' *Abstractive Summarization.*
L'*Abstractive Summarization* è un'attività di NLP che consiste nel condensare un testo più lungo in un riassunto più breve e coerente, preservandone il significato essenziale e il contesto. Il raggiungimento di questo obiettivo necessita di più sforzi rispetto ai metodi estrattivi tradizionali, in quanto richiede la generazione di frasi nuove che catturino le idee fondamentali del testo di partenza.

Il percorso verso la creazione di modelli efficaci di *Abstractive Summarization* è stato favorito da diversi fattori chiave.
Innanzitutto, l'ascesa delle architetture basate sui *transformer*, in particolare di modelli come *GPT-3*, ha portato a una rivoluzione nella comprensione e generazione del linguaggio. Questi modelli, grazie a un ampio pre-addestramento su vaste raccolte di testo, sono in grado di cogliere le sfumature più complesse del linguaggio, fondamentali per produrre riassunti di alta qualità.
In secondo luogo, l'utilizzo di spazi ad alta dimensionalità ha permesso a questi modelli di rappresentare il linguaggio in modo più significativo e consapevole del contesto. Ciò consente di cogliere le sottigliezze del contenuto e dello stile, che sono cruciali per generare riassunti non solo informativi, ma anche stilisticamente coerenti.
Inoltre, la disponibilità di ampie serie di dati ha svolto un ruolo fondamentale nell'addestramento e nella messa a punto dei modelli per i compiti di sintesi. Questi set di dati forniscono i campioni di addestramento necessari ai modelli per apprendere le complessità della sintesi.
Innovative attività di pre-addestramento hanno ulteriormente affinato le capacità di questi modelli, consentendo loro di comprendere meglio la struttura semantica del testo e rendendoli più abili nel generare riassunti concisi e signifi-

cativi.

Tuttavia, in mezzo a questi notevoli progressi, persiste una sfida critica nel campo della *Abstractive Summarization. Come imporre stili e proprietà specifiche nei riassunti generati?*
Sebbene i moderni *language model* siano in grado di generare riassunti, spesso mancano di un controllo a grana fine su aspetti quali il grado di copiatura dal testo di partenza, il livello di specificità o la lunghezza desiderata dell'output. Queste proprietà sono implicitamente codificate nei parametri del modello, rendendo difficile la generazione di riassunti con stili o caratteristiche specifiche.

Il contributo principale di questa tesi consiste nell'introduzione di una architettura che combina due *language model*, ottenendo come risultato finale l'emulazione di diverse strategie di *decoding*.

Questa tesi è suddivisa nei seguenti capitoli:

- **Capitolo 1** - Un *background* teorico sulle tecnologie necessarie che hanno portato allo sviluppo di questo lavoro, comprendendo l'architettura del *transformer*, le strategie di *decoding* prevalenti e le tecniche di *Abstractive Summarization*.
- **Capitolo 2** - Un'esplorazione delle ricerche precedenti relative al campo dell'*Abstractive Summarization*.
- **Capitolo 3** - Una descrizione della soluzione proposta, inclusi i dati e le tecnologie impiegate per la sua implementazione.
- **Capitolo 4** - Vengono mostrati il setup sperimentale e i risultati.
- **Capitolo 5** - Conclusioni e lavori futuri.

Introduction

Motivation and contribution

In recent years, the field of Natural Language Processing (NLP) has witnessed remarkable advancements, thanks to the advent of novel neural language models and transformer-based architectures. Among the various applications of NLP, one that has seen significant progress is abstractive summarization.
Abstractive summarization is a task in NLP that entails condensing a longer piece of text into a shorter, coherent summary while preserving its essential meaning and context. Achieving this requires more than traditional extractive methods, as it demands the generation of novel sentences that capture the core ideas of the source text.

The journey to creating effective abstractive summarization models has been significantly propelled by several key factors.
First and foremost, the rise of transformer-based architectures, particularly models like *GPT-3*, has brought about a revolution in language understanding and generation. These models, through their extensive pre-training on vast corpora of text, have the ability to grasp intricate nuances of language, which is vital for producing high-quality summaries.
Secondly, the utilization of high-dimensional spaces has enabled these models to represent language in a more meaningful and context-aware manner. This allows them to capture the subtleties of content and style, which are crucial for generating summaries that are not only informative but also stylistically coherent.
Additionally, the availability of extensive datasets has played a pivotal role in training and fine-tuning models for summarization tasks. These datasets provide the necessary training samples for models to learn the intricacies of summarization.
Innovative pre-training tasks have further refined the capabilities of these models, enabling them to better understand the semantic structure of text, and making them more adept at generating concise and meaningful summaries.

However, amidst these remarkable advancements, a critical challenge persists in the field of abstractive summarization. *How to enforce specific styles and properties in generated summaries?*
While modern language models are capable of generating summaries, they often lack fine-grained control over aspects such as the degree of copying from the source text, the level of specificity, or the desired length of the output. These properties are implicitly encoded within the model parameters, making it challenging to generate summaries with specific styles or characteristics.

The primary contribution of this thesis lies in introducing an architectural framework that combines two language models, ultimately resulting in the emulation of distinct decoding strategies.

This thesis is divided into the following chapters:

- **Chapter 1** - A theoretical background on the necessary technologies that led to the development of this work, encompassing the transformer architecture, prevailing decoding strategies, and abstractive summarization techniques.
- **Chapter 2** - An exploration of prior research related to the field of abstractive summarization.
- **Chapter 3** - Description of the proposed solution, inclusive of the data and technologies employed in its implementation.
- **Chapter 4** - Experimental setup and results are shown.
- **Chapter 5** - Conclusion and future work.

Contents

List of Figures

List of Tables

Chapter 1

Theoretical Framework

This chapter discusses natural language processing, natural language generation, and transformers, with particular emphasis on decoder architecture and various decoding techniques.

1.1 NLP and NLG

Natural Language Processing (NLP) and Natural Language Generation (NLG) are pivotal disciplines within computer science and linguistics that converge to unravel the intricate interactions between human language and computational systems. NLP can be defined as the comprehensive exploration of the linguistic facets that govern human-human and human-machine communication [4].
NLP is an interdisciplinary field that encompasses the development of models elucidating linguistic competence and performance, which are subsequently translated into computational frameworks embodying these models. Iterative enhancement mechanisms are then devised to refine these processes and models, while methodologies are crafted to scrutinize and evaluate the efficacy of the resultant systems. NLG, on the other hand, focuses on the inverse process of NLP, generating human-like language from structured data or information [5]. These two fields, NLP and NLG, synergistically culminate in establishing advanced natural language systems with multifaceted implications across a spectrum of applications, ranging from information retrieval and text classification to automatic summarization and human-computer dialogue.

The core facets of NLP are divided into four primary dimensions: formalization, algorithmization, programming, and realization [6]. The first phase necessitates the formalization of linguistic quandaries through mathematical constructs, thus affording a meticulous representation of language in a rigorous

mathematical framework. Thereafter, algorithmization transpires, involving the translation of these mathematical descriptions into implementable algorithms. After algorithm design, programming ensues, wherein computer programs are developed and executed, catalyzing the inception of a variety of practical NLP systems. Finally, realization encapsulates the incessant evaluation and enhancement of NLP systems to harmonize with users' requisites.

Linguistic knowledge contributes substantively to NLP, spanning acoustics, prosody, phonology, morphology, lexicology, syntax, semantics, discourse analysis, pragmatics, and common-sense encyclopedia knowledge [6]. This comprehensive spectrum of linguistic knowledge serves as the cornerstone upon which NLP systems are built, allowing them to bridge the gap between human communication and computational analysis with a level of depth and sophistication that continues to shape the boundaries of artificial intelligence.

In addition, NLP extends its reach to a wide range of disciplines. Computer science furnishes techniques that underlie model representation, algorithm design, and system implementation, while mathematics offers formal models and methods for NLP. Psychology augments NLP with models of human speech behavior, while philosophy supplies theories encompassing human thought and language. Statistics provides tools for predictive analytics based on data, and electronic engineering imparts insight into information theory and language signal processing. Biology bridges NLP with the neural underpinnings of human language behavior [6].

There is no doubt that the revolutionary breakthrough offered by transformer architecture has led to the rise of NLP in an era of unprecedented advancements. While NLP has historically harnessed linguistic theories, statistical models, and machine learning algorithms to navigate the complexities of human language, it is the transformer that stands as the paramount milestone, redefining the contours of this domain. With its inception, transformer architecture has guided a new paradigm marked by its profound ability to capture long-range contextual dependencies and intricate relationships within language data.
The next chapter will explore the inner workings of transformers.

1.2 Transformers

In recent years, the field of NLP has seen a remarkable revolution that has transformed the way machines comprehend and generate human language. Among the key breakthroughs that have catalyzed this revolution is the advent

of transformers, a class of deep learning models that have shown unparalleled performance in a wide range of NLP tasks. Originally introduced by Vaswani et al. in 2017 [7], transformers have since become the cornerstone of modern NLP, driving state-of-the-art advances in language understanding, translation, generation, and more.
The first transformer architecture 1.1, the model that inspired all the following transformers, consists mainly of two components: an **encoder** and a **decoder**. The role of the encoder is to transform an input sequence of symbolic representations $(x_1, ..., x_n)$ into a series of continuous representations denoted as $z = (z_1, ..., z_n)$. Subsequently, this continuous representation is channeled into the decoder, which in turn generates an output sequence $(y_1, ..., y_m)$ composed of symbols, generated successively. In each iterative step, the model operates in an auto-regressive manner, signifying that the recently generated output serves as a supplementary input for the subsequent step in the sequence.

The encoder configuration consists of an assemblage of $N = 6$ identical layers. Each layer encompasses two distinct substructures. The initial component entails a multi-head self-attention mechanism, while the subsequent unit entails a straightforward, position-wise, and fully connected feed-forward network. To preserve the flow of information, a residual connection encircles both substructures, followed by the application of layer normalization. Specifically, the resultant output of each substructure can be indicated as

$$LayerNorm(x + Sublayer(x))$$

where $Sublayer(x)$ denotes the inherent functionality of the respective substructure. To facilitate these residual connections, the dimensions of the outputs generated by all the substructures within this model, together with the embedding layers, are standardized to $d_{model} = 512$.

The decoder is similarly composed of an array of $N = 6$ identical layers. Beyond the pair of substructures present in each encoder layer, the decoder introduces an additional third substructure, responsible for executing multi-head attention across the results yielded by the encoder array. Analogous to the encoder framework, we integrate residual interconnections around each substructure within the decoder architecture, followed by the application of layer normalization. Additionally, adjustments are made to the self-attention substructure within the decoder layers to prevent the consideration of subsequent positional elements during the attention process. This *masking*, coupled with the deliberate displacement of the output embeddings by a single position, ensures that predictions related to a specific position i are exclusively based on observable outputs preceding position i.

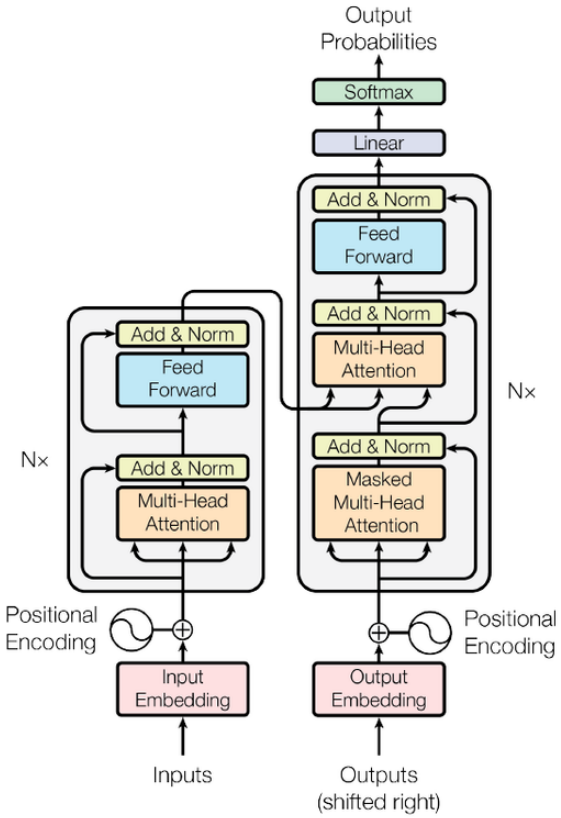

Figure 1.1: The transformer architecture.
From [7]

Preceding the advent of transformer architecture, alternative paradigms within machine learning had incorporated an encoder-decoder framework. Among these, a prominent exemplar is the Sequence-to-Sequence (Seq2Seq) model [8], a pioneering architecture aimed at tasks that require the manipulation of sequential data, encompassing domains such as machine translation, textual generation, and analogous realms. What engenders the transformer architecture as a noteworthy advancement beyond the confines of the Seq2Seq model lies in the assimilation of the self-attention mechanism, a conceptual innovation endowing it with heightened efficacy in navigating the intricacies of assorted sequential tasks. The concept of self-attention in the transformer provides an environment that supports parallel processing while also allowing for identifying connections between elements in the sequence, regardless of their order. This characteristic helps overcome limitations present in previous models, such as the Seq2Seq model, which relied on recurrent neural networks.

The attention mechanisms hinge upon the intricate manipulation of vectors denoted as Q (query), K (keys), and V (values), as shown in Figure 1.2. Collectively, these vectors contribute to underpin the creation of an informative output. Within this framework, a given query interacts with a set of key-value pairs, and through a compatibility function, the relationship between the query and each key is assessed. This function yields weights, effectively demarcating the pertinence of individual values relative to the query. Consequently, the derivation of the output materializes through a weighted sum of the values, where the values' weighted contributions combine to shape the outcome. The use of such a mechanism enables models to focus on designated informational constituents, thereby amplifying their significance in tasks that require nuances and contextually informed outcomes.
Referencing [7], the attention mechanism, coined as *Scaled Dot-Product Attention* is delineated by the subsequent mathematical expression:

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

Embedded within this framework, the input configuration encompasses queries and keys distinguished by a dimensionality of d_k, alongside values characterized by a dimension of d_v. The process involves calculating the dot products between the query and all keys, and a quotient established through division by $\sqrt{d_k}$. Subsequently, a softmax function is applied, returning a systematic assignment of weights to the values.
The softmax function, often employed in probability distributions, serves to transform an input sequence into a probabilistic distribution. This function exponentiates the values of the input sequence and normalizes them, resulting in non-negative values that collectively sum up to 1. In the context of attention mechanisms, the softmax function operates on the computed weights derived from the compatibility scores between queries and keys. This ensures that the weights assume a probabilistic nature, signifying the relative importance of corresponding values concerning a specific query.

The attention mechanism employed within transformers is commonly referred to as *Multi-head Attention*, 1.1 illustrates this concept. Instead of performing a singular attention operation involving keys, values, and queries of dimensions d_{model}, it is advantageous to subject the queries, keys, and values to a series of h distinct linear projections, each acquired through individual learned linear transformations leading to dimensions d_k, d_k, and d_v, respectively. The subsequent procedure involves performing parallel attention operations on these transformed versions of queries, keys, and values, thereby generating output values of dimension d_v. Subsequently, these resulting values are concatenated

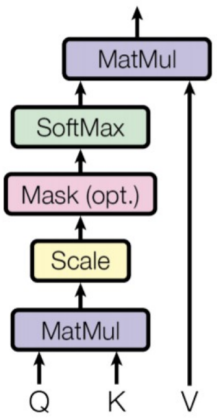

Figure 1.2: Scaled dot-product Attention.
From [7]

once again projected, as shown in Figure 1.3.
Multi-head attention facilitates the model's capacity to simultaneously engage with diverse information residing within disparate representation sub-spaces and across distinct positions. Contrastingly, a single attention head usually undermines such potential by promoting an averaging effect. Mathematically, the Multi-head attention operation can be formulated as follows:

$$MultiHead(Q, K, V) = Concat(head_1, head_h)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^K)$. Here, each $head_i$ signifies an individual attention head and is computed through the function

$$Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where the projection matrices W_i^Q, W_i^K, W_i^V, and W^O are defined as follows: $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$, and $W^O \in R^{hd_v \times d_{model}}$. This configuration ensures that the attention mechanism operates effectively while preserving a manageable level of complexity.

1.3 Decoding strategies and the Decoder

The primary function of the decoder (shown on the right side of Figure 1.1) involves the autoregressive generation of output sequences, relying on the encoded information from the encoder and previously generated tokens [9]. Essentially, the decoder transforms abstract representations into coherent and contextually relevant textual output.

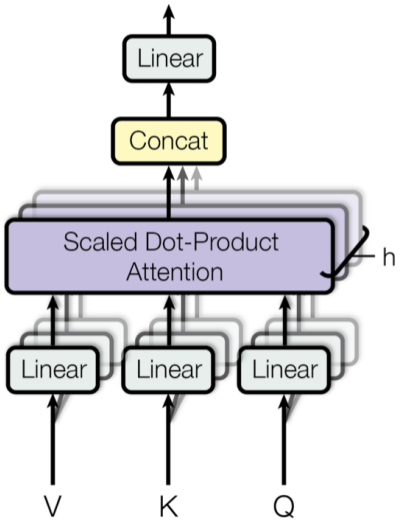

Figure 1.3: Multi head Attention.
From [7]

Decoding strategies assume a central role in shaping the characteristics of the output sequences generated by the decoder. These strategies are essential to improve the fluency, coherence, and relevance of the generated text in various NLP tasks.
The selection of an appropriate decoding strategy depends on the task's nature and the desired goals of text generation. For example, diversity-centric strategies may be favored in creative writing contexts, while in news article generation, approaches that emphasize relevance and coherence may be better suited.
The following sections present and explain the most important and distinguished decoding strategies.

1.3.1 Beam search

Beam search constitutes an evolved instantiation of the *best-first search* algorithm, where only a predetermined number of paths are kept as candidates. [1] It works by narrowing down the search area in a controlled way. This happens by keeping just a certain number of paths, the *beam*, which serves as a pool of candidate solutions during the search process. The beam width, denoted as B, prescribes the number of concurrently considered paths, ensuring computational feasibility and memory conservation.

[10]Initiating the sequence generation, for example, in language translation,

[1] http://foldoc.org/beam+search

the beam embraces the B most plausible word options. Each word accrues a score predicated upon its anticipated likelihood, establishing initial candidates for the sequence's outset. For subsequent sequence positions, the incumbent beam members individually guide the model's anticipations for forthcoming words. This strategic conditioning grants heterogeneity in predictions, thus diversifying word choices between candidates.
Throughout the search, candidate hypotheses accumulate scores that amalgamate predicted word probabilities and preceding cumulative scores. This collective score holistically quantifies the sequence's verisimilitude. Termination occurs upon a candidate hypothesis that produces the end-of-sentence marker, indicative of a valid sequence.
The results of the beam search crystallize into a hypothesis graph, each path beginning with an initial *start of sentence* $< s >$ token and culminating in an *end of sentence* $< /s >$ token 1.4. This graphical representation embodies the spectrum of potential translations, and the optimal translation manifests itself as the path linked to the highest cumulative score.
Opting for comparison and selection amidst the beam members, the convention involves utilizing the product of predicted word probabilities as the evaluation metric. Yet, to forestall favoritism towards shorter sequences, a common practice involves normalizing scores by the sequence length. This equitable treatment of sequences with varying lengths upholds fairness in comparative assessments.
The process of beam search can be described mathematically using the following equation:

$$Y_{[t]} = \underset{y_{1,[t]},\ldots,y_{B,[t]} \in \mathcal{Y}_t y_{i,[t]} \neq y_{j,[t]}}{\operatorname{argmax}} \sum_{b \in [B]} \Theta(y_{b,[t]}) \tag{1.1}$$

where

- $Y_{[t-1]} = y_{1,[t-1]}, ..., y_{B,[t-1]}$ is the set of B solutions held by Beam Search at the start of time t
- $\mathcal{Y}_t = Y_{[t-1]} \times \mathcal{V}$ is the set that gives all possible single token extensions of the beams.

1.3.2 Sampling Search

In its elementary form, the sampling process involves the stochastic selection of the following word, denoted as w_t, according to its conditional probability distribution [11]:

$$w_t \sim P(w|w_{1:t-1})$$

This methodology introduces a non-deterministic dimension into language generation.

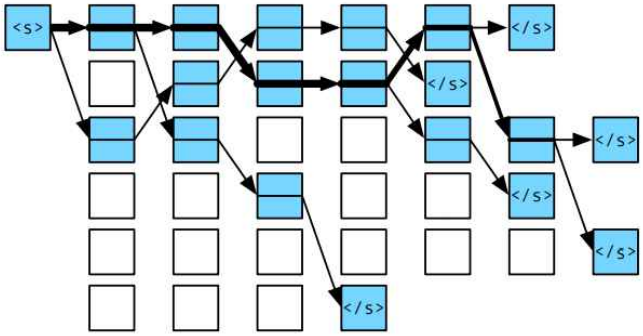

Figure 1.4: Beam search graph.
From [10]

Illustrated in Figure 1.5, the illustration showcases the practical execution of Sampling Search. The term *car* is sampled from the conditioned probability distribution $P(w|"The")$, subsequently succeeded by the sampling of *drives* from $P(w|"The","car")$.

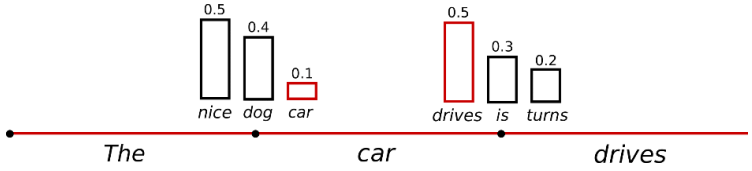

Figure 1.5: Sampling Search Functionality Example. [2]

Of significance, Ari Holtzman et al. (2019) [11] observe that the pure sampling strategy, which involves direct extraction from the predicted probabilities of the model, frequently yields text that lacks coherence and contextual relevance. This phenomenon is ascribed to the presence of the *unreliable tail*, encompassing an array of candidate tokens characterized by relatively modest probabilities. This group of tokens, while collectively overrepresented, plays a pivotal role in causing the observed incoherence.

1.3.3 Beam Sampling

Beam sampling is a decoding strategy that introduces an element of randomness into the process, differentiating it from conventional Beam Search [12]. In conventional Beam Search, the top-k hypotheses are consistently chosen at each decoding step based solely on their likelihood scores. In contrast, beam sampling involves stochastically selecting hypotheses from the distribution of

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

their scores, leading to a more probabilistic approach. This incorporation of randomness can generate a broader range of outputs, fostering diversity and creativity in the generated sequences. This contrasts the deterministic nature of standard Beam Search, which tends to produce more predictable results.

1.3.4 Greedy Search

[13] In contrast to the beam search algorithm, which expansively explores all possible subsequent steps and retains the most promising candidates within a designated beam size B, the greedy decoding approach involves selecting the word with the highest individual probability 1.6 at each decoding step t:

$$w_t = \underset{w}{\text{argmax}} P(w|w_{1:t-1})$$

Greedy decoding, characterized by its computational efficiency, necessitates a singular traversal of the underlying decoding process. In comparison, beam search mandates multiple iterations, which entails substantial supplementary overhead in terms of data management. However, this efficiency comes at the cost of potentially foregoing more globally optimal solutions, thus embodying the exploration-exploitation trade-off [14]. Beam search addresses this limitation by considering a range of possibilities, although it requires augmented computational resources and time.
Moreover, beam search can introduce diversity in generated sequences, a crucial attribute for applications such as language generation and machine translation [14]. Recent advances in decoding strategies have proposed nuanced techniques such as *nucleus sampling* and *top-k sampling*, aimed at refining the quality of outputs generated by greedy decoding. The choice between these decoding methodologies hinges upon the specific context. Greedy decoding proves advantageous when expeditious responses are imperative and minor output imperfections can be accommodated. In contrast, beam search finds its niche in tasks demanding refined, linguistically coherent, and varied outputs.

1.3.5 Diverse Beam Search

The primary limitation of conventional Beam Search lies in its tendency to generate sequences that exhibit minimal variation from each other. This practice not only incurs unnecessary computational overhead but also fails to effectively capture the inherent complexities and ambiguities inherent to intricate AI tasks.

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

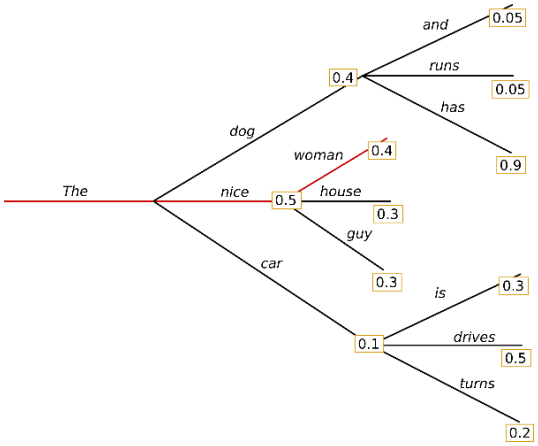

Figure 1.6: Greedy Search functioning. [3]

Addressing this concern, Vijayakumar et al. (2018) [15] introduced an innovative technique termed *"Diverse Beam Search"* as an alternative to the traditional Beam Search approach. This method facilitates the generation of diverse output sequences by incorporating a diversity-centric objective into the decoding process.

In this context, the authors proposed an augmentation of the Beam Search objective, as outlined in Equation 1.1, with an additional dissimilarity term denoted as $\Delta(Y[t])$. This term quantifies the diversity between candidate sequences under consideration. To enhance diversity, the beam budget, denoted as B, is partitioned into distinct groups labeled as G, with each group being optimized independently through a beam search procedure. Notably, the optimization of subsequent groups is carried out while preserving the fixed outputs of the preceding groups.
The final form of the proposed equation is expressed as:

$$\begin{cases} Y^g_{[t]} = \underset{y^g_{1,[t]},\dots,y^g_{B',[t]} \in \mathcal{Y}^g_t}{\operatorname{argmax}} & \sum_{b\in[B']} \Theta(y^g_{b,[t]}) + \sum_{h=1}^{g-1} \lambda_g \Delta(y^g_{b,[t]}, Y^h_{[t]}) \\ & \text{s.t. } y^g_{i,[t]} \neq y^g_{j,[t]},\ \lambda_g \geq 0 \end{cases}$$

Here, $\Delta(y_{[t]}, Y^g_{[t]})$ represents the dissimilarity term, quantifying the distinction between a given sequence $y_{[t]}$ and the sequence group $Y^g_{[t]}$. Mathematically, the dissimilarity term $\Delta(y_{[t]}, Y^g_{[t]})$ is defined as:

$$\Delta(y_{[t]}, Y^g_{[t]}) = \sum_{b=1}^{B'} \delta(y_{[t]}, Y^g_{[t]})$$

The function $\delta(y_{[t]}, Y^g_{[t]})$ serves as a measure of sequence dissimilarity, which could involve elements such as negative costs for co-occurring n-grams within two sentences or distances between distributed sentence representations.

1.3.6 Top-K sampling

In top-K sampling, a pivotal role is played by identifying the most probable K subsequent words [16, 17]. These selected words effectively constitute a restricted pool, and then the probability mass is redistributed among only those chosen words. An example of this approach can be found in the GPT-2 design [18], where it played a key role in improving the model's ability to create stories.

However, it should be noted that an inherent limitation of the top-K sampling methodology is the absence of an adaptive mechanism for the dynamic adjustment of the count of words subject to filtration of the distribution governing prospective word generation, denoted as

$$P(w|w_{1:t-1})$$

This particular characteristic engenders certain concerns, particularly related to the probabilistic nature of word occurrence. This concern is underscored by the nuanced interplay between distribution shapes, wherein certain words might emanate from sharply peaked distributions (as illustrated in the distribution on the right in Figure 1.7), while others are drawn from distributions characterized by a more diffuse profile (as depicted in the distribution on the left in Figure 1.7).
This constraint, imposed by the fixed sample size K, could result in two different outcomes with potential consequences 1.8. First, for words originating from distributions evincing pronounced concentration, the deployment of top-K sampling might engender an elevated susceptibility to the generation of incoherent or non-sensical language constructs. Second, in the context of distributions evincing a more uniform nature, the prescribed sample pool confinement could negatively impact the model's creativity, potentially resulting in a deficiency of linguistic diversity. Addressing these issues led to the development of *top-p*, or *nucleus*, sampling [11].

1.3.7 Top-p Sampling or Nucleus Sampling

[11]Top-p sampling, an alternative approach to word sampling, is distinguished by its dynamic selection of words based on cumulative probabilities.

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

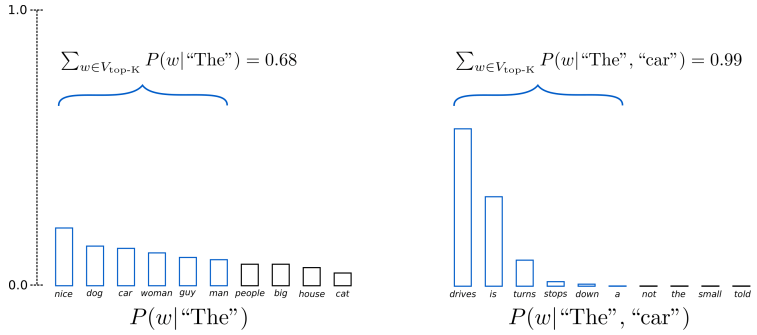

Figure 1.7: Top-K Sampling range of words and distribution. [4]

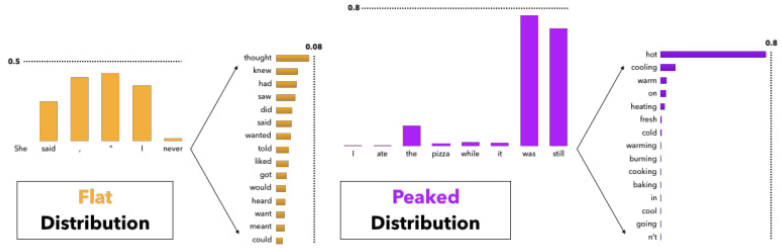

Figure 1.8: Flat distributions lead to many moderately probable tokens, while peaked distributions concentrate most probability mass into just a few tokens. The presence of flat distributions makes the use of a small k in top-k sampling problematic, while the presence of peaked distributions makes large k's problematic.

From [11]

Instead of limiting the selection to a fixed count of the most probable K words, top-p sampling operates by considering the smallest set of words whose cumulative probability surpasses a designated threshold, denoted as p:

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p', & \text{if } x \in V^{(p)} \\ 0, & \text{otherwise} \end{cases}$$

where p' represents the sum of probabilities for words in the set $V^{(p)}$[5]:

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1})$$

This adaptive method ensures that the size of the candidate word set fluctuates dynamically, guided by the probability distribution of the subsequent word. In particular, the magnitude of p significantly impacts the subset of vocabulary,

[5] $V^{(p)}$ contains the minimum set of words that contribute to the cumulative probability mass reaching or surpassing p

often referred to as the *nucleus* [11] of the distribution.
Top-p sampling presents distinct advantages over conventional methods. By striking a balance between randomness and determinism, it facilitates the generation of diverse and coherent text. This contrasts with other strategies like pure random or deterministic greedy sampling.

1.3.8 Contrastive Decoding

In their recent work, Li et al. [19] introduced an innovative search-based technique known as contrastive decoding. This decoding strategy, presented as a proficient method, demonstrates the ability to produce coherent and linguistically varied text while maintaining high fluency.
The essence of contrastive decoding lies in exploiting the inherent distinctions between large and small language models. This is accomplished by strategically selecting tokens that optimize the disparity in log-likelihoods between these models.

Illustrated in Figure 1.9, the Contrastive Decoding process produces exceptional quality text, capitalizing on the favorable attributes of a robust large model while mitigating the less desirable characteristics exhibited by a smaller counterpart.

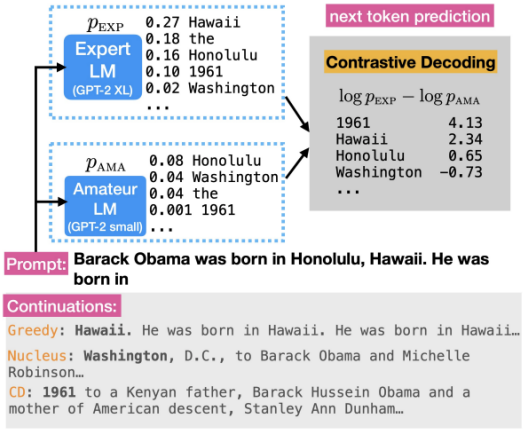

Figure 1.9: Contrastive decoding results compared to other decoding strategies. From [19]

At the heart of this approach lies the insight put forth by Li et al. in their study. They consider a concise prompt of length n, denoted as $x_{\text{pre}} = x_1, ..., x_n$, with each x_i representing a token drawn from the vocabulary V. The challenge is then to extend this prompt into a continuation of length m, denoted

as $x_{\text{cont}} = x_{n+1}, ..., x_{n+m}$. To achieve this, the authors employ a pre-trained autoregressive language model denoted as p_{LM}.

During the decoding process, tokens are generated iteratively, one at a time, taking into account the preceding context. This is mathematically formulated as:

$$p_{\text{LM}}(x_{\text{cont}}|x_{\text{pre}}) = \prod_{i=n+1}^{n+m} p_{\text{LM}}(x_i|x_{<i})$$

Here, $p_{\text{LM}}(x_i|x_{<i})$ signifies the distribution of the next token given the preceding tokens.
Different language models are distinctly referred to:

- p_{AMA} is representative of an amateur language model, denoting a smaller language model.
- p_{EXP} is used to represent an expert language model, indicating a larger and more sophisticated language model.

The fundamental objective of contrastive decoding is to segregate unfavorable behaviors highlighted by amateur language models and harness the generative power of expert language models to produce text of high quality. To operationalize this notion, the authors propose a contrastive objective.

$$\log p_{\text{EXP}}(x_{\text{cont}}|x_{\text{pre}}) - \log p_{\text{AMA}}(x_{\text{cont}}|x_{\text{pre}})$$

This contrastive decoding objective effectively promotes text patterns that align with the preferences of large expert language models, while simultaneously discouraging patterns favored by smaller amateur language models. It should be noted that amateur language models are not always incorrect; indiscriminately penalizing all of their behaviors would lead to undesired outcomes, penalizing valid outputs (false negatives) and rewarding implausible tokens (false positives). To address this challenge, the authors introduce the concept of plausibility constraint.
In the end, the complete formulation of the contrastive decoding approach can be expressed as:

$$\begin{cases} \max\limits_{x_{\text{cont}}} \log p_{\text{EXP}}(x_{\text{cont}}|x_{\text{pre}}) - \log p_{\text{AMA}}(x_{\text{cont}}|x_{\text{pre}}) \\ \text{subject to} \quad x_i \in V_{\text{head}}(x_{<i}), \forall x_i \in x_{\text{cont}} \end{cases}$$

1.3.9 η Sampling

In 2022, John Hewitt, Christopher D. Manning, and Percy Liang introduced η Sampling in their paper titled *Truncation Sampling as Language Model*

Desmoothing [20]. η Sampling is a technique that truncates words below an entropy-dependent probability threshold. Compared to previous algorithms, this method exhibits several advantages, including the generation of more plausible long English documents as perceived by human readers, enhanced capability to break out of repetitive patterns, and more rational behavior across various test distributions.
The authors derived the principles of truncation from an explicit smoothing model that formalizes the following intuitions:

- Words with high probability should not be truncated.
- When all words in the distribution have a low probability, only words with low probability relative to the rest should be truncated.

Interestingly, their investigation revealed that state-of-the-art truncation sampling algorithms, such as top-p, deviate from these principles, as top-p tends to unnecessarily truncate high-probability words.

The authors considered a conditional distribution $P_\theta(X|x_{<i})$ with entropy $h_{\theta,x_{<i}}$. The probability of a word in the uniform distribution of entropy $h_{\theta,x_{<i}}$ is $\exp(-h_{\theta,x_{<i}})$. To establish an entropy-dependent threshold, they introduced α, a hyperparameter within the range [0,1]. This threshold is defined as $\alpha \exp(-h_{\theta,x_{<i}})$, in accordance with the Absolute Probability Principle.
The Absolute Probability Principle proposes a straightforward truncation algorithm: for a given hyperparameter threshold ϵ, any word with a probability greater than ϵ is retained. This principle ensures that words with reasonably high probabilities are preserved, contributing to the overall coherence of the generated text.
By integrating this principle with the entropy-dependent threshold, we obtain:

$$A_{x_{<i}} = \{x \in \mathcal{V} : P_\Theta(x|x_{<i}) > \eta$$
$$\eta = \min(\epsilon, \alpha \exp(-h_{\theta,x_{<i}}))$$

The results presented by the authors are visually depicted in Figure 1.10. They conducted experiments using two low-entropy prompts, namely *My name...* and *Donald...*, demonstrating that top-p decoding often leads to single-word continuations, while η Sampling suggests a wider range of possibilities. Similarly, for repetitive prompts like *The feeling!*, top-p tends to replicate patterns, whereas η Sampling provides more diverse outputs. In scenarios where the prompt requires specifying the capitals of countries, top-p typically provides only the correct capital name, whereas η Sampling offers alternative continuations that do not rigidly adhere to the in-context trend.

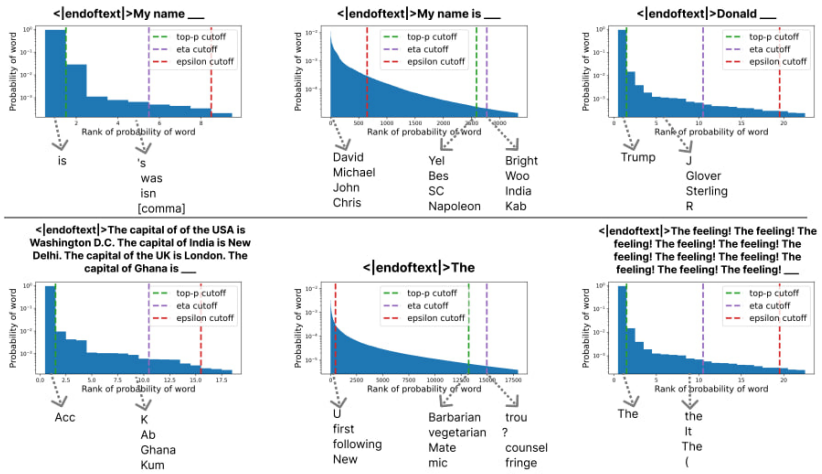

Figure 1.10: Unit tests of the truncation behavior of top-p, ϵ, and η Sampling. From [20]

A summary of the decoding strategies presented in this chapter is shown in table 1.1.

1.4 Abstractive summarization

Abstract summarization is an essential task in NLP that addresses the challenge of distilling fundamental information from extensive textual documents. The elementary premise of abstractive summarization revolves around the transformation of input documents into concise, coherent, and human-like summaries. This task, characterized by the need for a nuanced understanding of the semantic context, has gained significant attention in the fast-paced digital landscape of today's information-rich internet era. As individuals are inundated with copious textual content daily, the ability to efficiently extract meaningful insights from this deluge of data has become increasingly vital.
To set the stage, it is essential to define the problem formally. In the context of abstractive summarization, the input is typically represented as a document $\mathcal{X}$, where each element $x_i \in \mathcal{X}$ is a token. The nature of this input can vary depending on the specific subfield within abstractive summarization.
Summarizing documents can be divided into three main categories:

- **Single Document Summarization**: in Single Document Summarization, $\mathcal{X}$ typically constitutes a relatively short text, usually containing fewer than 1024 tokens. This limitation arises from the quadratic complexity constraints imposed by transformer-based language models, as demonstrated in prior research efforts [29, 30].

- **Multi-Document Summarization**: in Multi-Document Summarization, $\mathcal{X}$ typically comprises a cluster of documents related to a specific topic. To address this, researchers often concatenate these documents into a single extended input, akin to the LDS setup [31, 32, 33, 34].
- **Long Document Summarization**: Long Document Summarization deals with input documents that can be substantially larger, sometimes exceeding 10,000 tokens. To tackle this scenario, researchers have devised efficient transformer architectures with linear complexity, enabling them to process documents with up to 16,384 tokens [35].

Abstractive summarization, as one of the two primary categories in text summarization, stands in contrast to extractive summarization. While extractive summarization entails the selection and extraction of pivotal sentences or passages from the source text to form a summary, abstractive summarization takes a more creative approach.
In abstractive summarization, the goal is not just to extract but to generate a summary that may include new sentences not present in the source text. This generation process aims to capture the underlying meaning of the original text, mimicking human-like language generation.

Table 1.1: A summary of the decoding strategies benchmarked in this abstractive summarization study.

From [2]

DECODING STRATEGIES[*],[†]		
Greedy Search	**Contrastive Search** [21]	**Beam Search** [22]
It selects the most probable token at each t (locally-optimal decision): $y_t = \underset{y \in \bar{\mathcal{V}}}{\text{argmax}} \log p_\theta(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ (for $t > 0$)	It extends Greedy Search by jointly considering the model confidence (i.e., semantic coherence) and the similarity w.r.t. the previous context: $y_t = \underset{y \in \bar{\mathcal{V}}^{(k)}}{\text{argmax}} \{(1-\alpha) \times p_\theta(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ $-\alpha \times (\max\{s(h_y, h_{y_j}) : 1 \leq j \leq t-1\})\}$ (for $t > 0$) $\bar{\mathcal{V}}^{(k)}$ is the set of top-k predictions from the LM's probability distribution ($k \in \mathbb{Z}_+$). The second term is a degeneration penalty governed by $\alpha \in [0, 1]$; it is computed with the cosine similarity $s(\cdot, \cdot)$ between the candidate representation h_y and the prior tokens. When $\alpha = 0$, Contrastive Search degenerates to Greedy Search.	A pruned breadth-first search algorithm that expands the hypotheses in a greedy left-to-right way, retaining the top-b candidates at each t: $Y_t = \underset{\substack{Y' \subseteq \mathcal{B}_t, \\ \lvert Y' \rvert = b}}{\text{argmax}} \mathcal{S}(Y'_t)$ (for $t > 0$) $\mathcal{B}_t$ denotes all possible roll-out extensions $\{\boldsymbol{y}_{t-1} \circ y \mid y \in \bar{\mathcal{V}}$ **and** $\boldsymbol{y}_{t-1} \in Y_{t-1}\}$. $\mathcal{S} : \mathcal{Y} \to \mathbb{R}$ is a scoring function on $Y \subseteq \mathcal{Y}$. By default, $\mathcal{S}(Y) = \sum_{\boldsymbol{y} \in Y} \log p_\theta(\boldsymbol{y} \mid \boldsymbol{x})$. $\boldsymbol{y}*$ is chosen from the final set Y_T.
Diverse Beam Search [23]	**Sampling**	**Top-k Sampling** [24]
In vanilla Beam Search, as t and b increase, the candidate summaries of a single beam gradually occupy the top positions. This inner functioning results in a high overlap among hypotheses that often only differ by punctuation and slight morphological variations [25], indicating poor search space coverage. Diverse Beam Search splits a beam into sub-groups sequentially optimized and adds a penalty in $\mathcal{S}$ to encourage inter-group dissimilarity: $\mathcal{S}(Y_t^{(g)}) = \sum_{\boldsymbol{y} \in Y} \log p_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ $-\lambda \sum_{g' < g} \Delta\left(\boldsymbol{y}, Y_t^{(g')}\right)$ $\Delta(\boldsymbol{y}, Y_t^{(g')})$ is a diversity measure whose strength is controlled by λ, and g is a sub-group.	Instead of approximating $\boldsymbol{y}^*$, it makes a random selection at each t, thus giving a non-zero chance to every token following the model distribution ($\sim$): $y_t \sim p_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ (for $t > 0$) Softmax can be re-calibrated through a temperature parameter $\tau \in (0, 1]$. By decreasing τ (1 = no effect), we skew the distribution towards likely tokens and reduce the mass in the unreliable tail: $p_\theta(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) = \frac{\exp(\frac{u}{\tau})}{\sum_j \exp(\frac{u_j}{\tau})}$ where $u \in U$ are logits.	It limits the sampling space to the top-k probable tokens, regardless of the distribution shape. Here, the decoding maximizes: $\sum_{y \in \bar{\mathcal{V}}^{(k)}} p_\theta(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ where $\bar{V}^{(k)} \subseteq \bar{V}$.
Top-p (Nucleus) Sampling [26]	**Beam Sampling** [27]	**η Sampling** [28]
Instead of assuming a fixed-sized shortlist, it dynamically expands and contracts the candidate pool according to the shape of the distribution (e.g., fewer contenders if sharp). When many next tokens are plausible, the allowed set reflects that. Formally, it samples from the smallest subset of tokens whose cumulative probability mass exceeds a chosen threshold $p \in (0, 1]$: $\sum_{y \in \bar{\mathcal{V}}^{(p)}} p_\theta(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) \geq p$	It is a variant of Beam Search where, at each t, the b next tokens for constructing $\mathcal{B}$ are sampled conditioned on the current hypotheses (no local optimum).	It samples tokens above an entropy-dependent probability threshold to avoid unnecessary cutting-offs. Eligible tokens come from: $\{y \in \bar{\mathcal{V}} \mid p_\theta(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) > \eta\}$ $\eta = \min(\epsilon, \alpha \exp(-e_{\boldsymbol{x}, \boldsymbol{y}_{<t}}))$ where $\exp(-e_{\boldsymbol{x}, \boldsymbol{y}_{<t}})$ is the expected next token probability given the entropy $e_{\boldsymbol{x}, \boldsymbol{y}_{<t}}$, scaled by a constant α. To expose a single hyperparameter, α is set to $\sqrt{\epsilon}$. The general principle is to only truncate tokens whose probabilities are low relative to the rest of the distribution or an absolute threshold.

[*] Basic notation. $\boldsymbol{x}$ = document(s), $\boldsymbol{y}$ = summary, θ = model weights, t = decoding step, $\bar{\mathcal{V}} := \mathcal{V} \cup \{\text{EOS}\}$, p_θ = model probability distribution, $\mathcal{Y}$ = hypothetical summaries, b = beam size, $\boldsymbol{y}*$ = best summary.

[†] ● = deterministic, ● = stochastic.

Chapter 2

Related Work

The purpose of this chapter is to clarify the context and summarize prior work related to this project.

2.1 Overview

In recent times, the introduction of the transformer model has revolutionized NLP.
T-PTLM, which stands for *Transformative Pre-trained Language Model*, is a contemporary advancement in Natural Language Processing and machine learning. T-PTLMs are an evolution of pre-trained language models like GPT [36] (Generative Pre-trained Transformer) and BERT [37] (Bidirectional Encoder Representations from Transformers).
T-PTLMs build upon the foundational idea of pre-training a neural network on a massive corpus of text data to learn the intricacies of language and world knowledge. However, what sets them apart is their focus on *transformative* capabilities. These models are designed to go beyond traditional NLP tasks, such as text generation or classification, and aim to foster creativity, problem-solving, and adaptive language understanding.
The term *transformative* implies that T-PTLMs are engineered to excel in tasks that require dynamic adaptation and contextual understanding. These transformer-based pre-trained models have emerged as an essential technological advancement, wielding the remarkable capacity to acquire universal language representations from vast expanses of unannotated text data and then seamlessly transfer this acquired linguistic knowledge to a diverse array of downstream NLP tasks.
As the field of NLP continues to evolve, these T-PTLMs have been continually refined and adapted. Researchers have worked to enhance both the training strategies and architectural components of these models. This evolution served

multiple purposes: improving the readability and coherence of the models, expanding their capacity to process larger inputs, and generating more comprehensive and considerable outputs.
Diverse research groups have dedicated their efforts to various aspects of T-PTLM development [38]. Some have focused on training models tailored to specific NLP tasks, such as text summarization, question-answering, or code interpretation. Others have trained transformers in one or more languages or diverse datasets to broaden their linguistic capabilities.
As a result of all these years of research, a wide variety of different T-PTLM have been developed, as shown in Figure 2.1, each with its own unique characteristics.

Taking this context into account, the conceptual foundation of this study can be outlined. The area of decoding structure and algorithmic adaptations is one of many fascinating areas of Natural Language Processing. Essentially, the goal is to exert control over the decoder to produce different types of outputs, in accordance with specific rules.

2.2 Hydrasum

The inspiration for this work comes from research conducted by Goyal et al. (2021), Hydrasum[1].
Hydrasum comprises an encoder network designed to accept the document input denoted as x. The decoder network has been adapted to incorporate $k(> 1)$ individual decoders, represented as $\phi_1, \phi_2, \ldots, \phi_k$, as depicted in Figure 2.2. At each time step i, each decoder generates a probability distribution $P_{\phi_k}(y_i|x, y_{<i})$ over the vocabulary, reflecting the probabilities of the next token. The ultimate output probability $P(y_i|x, y_{<i})$ is computed as a combination of these k probability distributions, with the mixing coefficients determined by a gating mechanism designated as G.
The gating mechanism G is employed to amalgamate the output distributions of the k decoders. At time step i, h_i^m represents the hidden state output of the m^{th} decoder layer. This hidden state representation h_i^m is passed through a feed-forward layer W with dimensions $(|h_i^m|, k)$, followed by a softmax layer. This process yields a probability distribution g_i, which is subsequently utilized to compute the overall next-token output probability as follows:

$$P(y_i|x, y_{<i}) = \sum_{j=1}^{k} g_i^j \cdot P_{\phi_k}(y_i|x, y_{<i})$$

Here, g_i^j denotes the probability of selecting the j^{th} decoder at time step i.
The authors of this study demonstrate that Hydrasum's incorporation of mul-

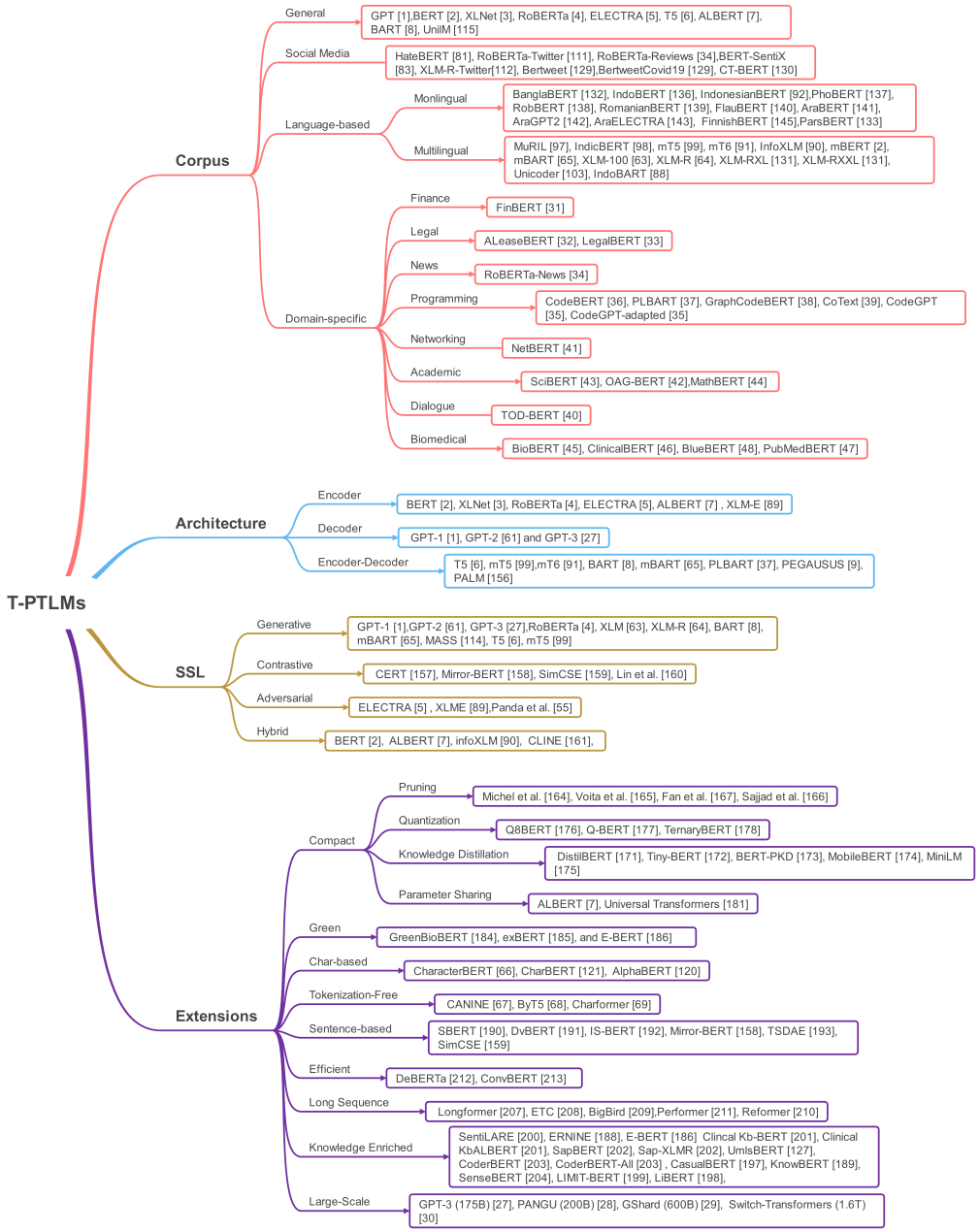

Figure 2.1: Taxonomy of T-PTLMs.
From [38]

tiple decoders autonomously learns distinct summary styles during training without the need for additional supervision. Experimental evaluations were conducted on three summarization datasets (Cnn [39, 40], Newsroom [41], and XSum [42]). The results indicate that Hydrasum offers a straightforward

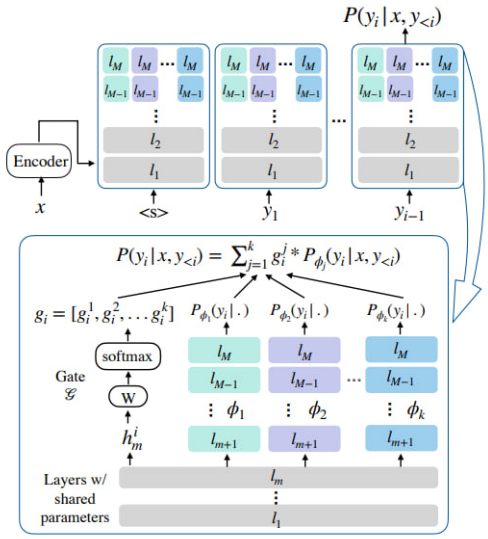

Figure 2.2: Architecture of Hydrasum. The decoder network of conventional models is modified to include multiple decoders.

From [1]

approach to acquiring stylistically diverse summaries by sampling from either individual decoders or their combinations, surpassing the performance of baseline models.

In the context of the Hydrasum architecture, particularly within the decoding canal, the authors introduce a distinctive feature wherein multiple decoding channels are established without imposing any prescriptive constraints or control over them. On the contrary, decoding strategies exhibit discernible attributes tailored to exercise control over the resultant text, reflecting their deliberate design to govern text generation properties. As a matter of fact, their behavior is sought after, and controlled, and it does not occur by chance. Additionally, the scalability implications of this approach remain a topic of uncertainty. Evaluating whether the observed behaviors outlined in the paper persist when transitioning to different models or datasets not employed by the authors, emerges as a critical consideration.
Hence, the idea proposed in this thesis is based on the incorporation of additional *channels* into the final segment of the network. However, it is important to note that these channels are not mere collections of uncontrolled weights or devoid of specific training objectives tied to text quality attributes. Rather, each channel embodies a distinct decoding strategy, characterized by well-understood behavior. Furthermore, the gating mechanism found in Hydrasum can be retained, allowing the network to autonomously determine whether

to employ one strategy over another or blend them. This approach not only enhances control, thereby augmenting interpretability by elucidating the network's decision-making logic but also addresses a notable and still unresolved gap in current literature: the identification of the most suitable decoding strategy for specific scenarios. Currently, this aspect remains largely overlooked, given the multitude of decoding strategies, each with its unique parameter set. Consequently, a comprehensive exploration of all potential strategies and combinations is impractical, and to date, no single strategy has been universally recognized as superior to others.

2.3 Differentiable Decoding Strategies

Categorical variables serve as a natural means to represent discrete structures within the real world. Nonetheless, stochastic neural networks have traditionally refrained from employing categorical latent variables due to their inherent challenge in facilitating backpropagation through sampled data points. In a seminal paper by Jang et al. [43], an innovative gradient estimation approach is introduced. This method substitutes the non-differentiable sampling process from a categorical distribution with a differentiable alternative derived from a novel construct known as the *Gumbel-Softmax* distribution. Remarkably, the Gumbel-Softmax distribution possesses a crucial characteristic: it can be progressively transformed into a categorical distribution while maintaining smoothness. The authors demonstrate that the Gumbel-Softmax estimator surpasses existing gradient estimation techniques in tasks involving structured output prediction, and unsupervised generative modeling with categorical latent variables, and even yields substantial computational accelerations in semi-supervised classification scenarios. To introduce the Gumbel-Softmax distribution, we begin by considering a categorical variable, denoted as z, with associated class probabilities $\pi_1, \pi_2, \ldots, \pi_k$. The Gumbel-Max trick [44, 45], provides an elegant and efficient method for generating samples from this categorical distribution, where z is obtained as:

$$z = \text{one\_hot}\left(\underset{i}{\text{argmax}}[g_i + \log \pi_i]\right)$$

Here, g_i through g_k represent independent and identically distributed samples drawn from the $Gumbel(0, 1)$ distribution [1].
In practice, the authors leverage the softmax function as a continuous and differentiable approximation to the $argmax$ operation. This allows them to

[1] $Gumbel(0, 1)$ samples can be generated using inverse transform sampling by drawing $u \sim \text{Uniform}(0, 1)$ and computing $g = -\log(-\log(u))$.

generate k-dimensional sample vectors, denoted as y, residing in the simplex Δ^{k-1}, defined as follows:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \quad \text{for } i = 1, \ldots, k$$

The probability density function of the Gumbel-Softmax distribution is subsequently defined as:

$$p_{\pi,\tau}(y_1, \ldots, y_k) = \Gamma(k)\tau^{k-1} \left(\sum_{i=1}^{k} \frac{\pi_i}{y_i^{\tau}} \right)^{-k} \prod_{i=1}^{k} \left(\frac{\pi_i}{y_i^{\tau+1}} \right)$$

As the softmax temperature parameter, τ, approaches zero, samples drawn from the Gumbel-Softmax distribution tend to become one-hot, rendering the distribution practically indistinguishable from the categorical distribution $p(z)$. This transition is visually depicted in Figure 2.3. Consequently, the selection of an appropriate temperature parameter can pose a challenge, as excessively high values make the distribution nearly uniform, impeding the model's ability to make meaningful categorical choices. On the other hand, overly low temperatures drive the distribution towards a hard categorical choice, which may hinder exploration in reinforcement learning or lead to mode collapse in generative models.

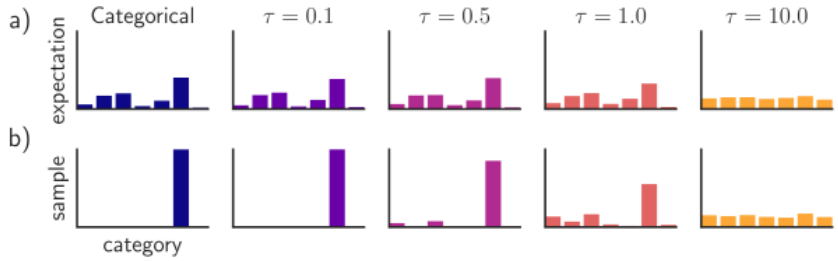

Figure 2.3: a) For low temperatures ($\tau = 0.1$,$\tau = 0.5$), the expected value of a Gumbel-Softmax random variable approaches the expected value of a categorical random variable with the same logits. As the temperature increases ($\tau = 1.0$, $\tau = 10.0$), the expected value converges to a uniform distribution over the categories. (b) Samples from GumbelSoftmax distributions are identical to samples from a categorical distribution as $\tau \to 0$. At higher temperatures, Gumbel-Softmax samples are no longer one-hot and become uniform as $\tau \to$ inf
From [43]

Moreover, it is essential to note that while the Gumbel-Softmax distribution facilitates gradient-based optimization, the process of sampling from it is not entirely differentiable. Consequently, users incorporating this distribution into their models may encounter challenges when applying certain optimization algorithms or loss functions that rely on smooth gradients.

2.3.1 Differentiable Beam Search

In 2019 Collobert et al. introduced a fully differentiable Beam Search decoder [46]. In this paper the authors introduce a novel fully differentiable Beam Search decoder, offering a versatile optimization approach during the training phase through the inference procedure. This decoder facilitates the integration of models operating at varying levels of granularity, such as acoustic and language models, even when the target sequences do not align perfectly with the input sequences, considering all feasible alignments between the two. This end-to-end system ensures that gradients permeate throughout the architecture from the word-level transcriptions. While recent research has highlighted the efficacy of deep neural networks with attention-based mechanisms for training acoustic models while implicitly learning a language model, this paper diverges by demonstrating the discriminative training of an acoustic model in conjunction with an explicit, and potentially pre-trained, language model.
It is worth highlighting that the code of this research is not open-source and it is based on dated architectures.

2.4 Abstractive Summarization

Extracting accurate and contextually relevant information from the source text is a non-trivial task. Various methodologies have been explored, from non-neural methods [47] to neural network-based techniques [40, 48]. A significant leap in the field occurred with the emergence of large pre-trained language models based on transformer architectures. These models, initially trained on vast corpora of text, are subsequently fine-tuned for summarization tasks.
The latest trends in abstractive summarization revolve around structural properties in neural document modeling [49] and knowledge injection [50, 51]. Reinforcement learning with sequence-level rewards has also emerged as an alternative approach [50, 52]. Nevertheless, traditional encoder-decoder models trained under maximum likelihood regimes remain widely employed in the field.
The foundation of modern abstractive summarization largely relies on sequence-to-sequence architectures built upon self-supervised pre-trained transformers [53, 54, 29]. These architectures have demonstrated remarkable capabilities, even in challenging multi-document and low-resource settings [55]. However, they are not immune to challenges such as content hallucination [56, 57] or reliance on extraction-based approaches [58].

Chapter 3

Method

This chapter describes the architecture created and the structure of the dataset used.

3.1 Architecture

When it comes to document summarization, the conventional approach typically involves the utilization of a single language model. For instance, an extensively employed model, currently recognized as state-of-the-art in document summarization, is Bart [29]. Therefore, the standard procedure involves employing Bart to process input text and generate corresponding summaries. The challenge arises initially in training and in model prediction, where the primary objective is to enhance the likelihood of accurately predicting the subsequent token. This imperative arises from the autoregressive nature of the models, as the summary evolves token by token. The aim is to train these models in a manner that maximizes the probability of the token with the highest likelihood aligning with the corresponding token in the target summary. This approach aligns with the broader practice observed in generative language models, wherein the output entails a probability distribution for the next token at each step. This distribution encompasses probabilities assigned to all vocabulary tokens associated with the model. Importantly, this process does not commence with a *blank slate* Bart, devoid of any preset weights, but rather begins with a pre-trained Bart that possesses a predefined vocabulary containing a selection of specific tokens.
The outcome of a generative autoregressive language model does not yield a summary; instead, it yields a sequence of probability distributions. Consequently, the process of transitioning from these probability distributions to text, and ultimately to a summary, involves the application of decoding strategies. These strategies encompass various heuristics that guide the selection of tokens from

the predicted probability distribution in a step-by-step manner. As previously elucidated in Section 1.3, there exist several decoding strategies, each endowed with distinctive characteristics and operational methodologies.
Decoding strategies, in general, possess highly specific attributes that have been rigorously scrutinized by the scientific community. They play a crucial role in regulating the generation process, ensuring a deliberate and controlled outcome. These strategies are designed to steer the text generation towards a predetermined desired behavior, rather than leaving it to chance.
In the latter portion of the network architecture, we introduce distinct "terminals," where each terminal does not represent an inscrutable ensemble of uncontrolled weightings, nor does it impose any specific task related to text quality properties during training. Instead, each "channel" or "terminal" embodies a predefined decoding strategy, characterized by a well-understood behavior. Furthermore, a gating mechanism, such as the one employed in Hydrasum[1], remains an option, instructing the network on when to autonomously employ one strategy over another, or even potentially blending them. This approach not only affords greater control and interpretability, allowing for a deeper comprehension of the network's decision-making rationale but also addresses a significant ongoing challenge in the literature: the determination of the most suitable decoding strategy for a given context. Currently, this aspect remains somewhat neglected, given the multitude of available decoding strategies, each accompanied by its own set of parameters, necessitating exhaustive and time-consuming exploration of numerous combinations.
At this point, however, a significant challenge has surfaced, from an implementation perspective. Decoding strategies pose a considerable obstacle due to their non-differentiable nature. These strategies involve intricate algorithms that necessitate discrete decision-making processes, which are not agreeable to conventional backpropagation. Consequently, the approach I previously outlined faces feasibility constraints.
Subsequently, another concept emerged: rather than attempting to make the layers differentiable through the direct encoding of decoding strategies, we pursued a different approach, involving their approximation. In essence, we aimed to replicate their functionality using an alternative language model. In this context, the transformation process entails taking an input and generating an output that differs from the traditional input-document to output-summary paradigm. However, pertinent questions arise when adapting models like Bart or T5, not for document summarization, but for the emulation of decoding strategies. The primary concern centers around determining the appropriate input and output. In this context, the input no longer consisted of the extensive document, and the output was no longer the expected summary. Instead, the input now corresponds to that of a decoding strategy, namely, the probability

distributions predicted by a language model, as better explained in Figure 3.1. Usually, over these output probability distributions, a specific strategy

```python
# Tokenize the source documents
documents_ids = self.tokenizer.encode(
    self.documents[idx],     # documents to tokenize
    return_tensors="pt",     # return a PyTorch tensor
    max_length=512,          # maximum length of the sequence
    truncation=True,         # truncate to max_length if longer
    padding="max_length"     # pad to max_length if shorter
).to(device)                 # move the tensor to the specified device

# Generate vocabulary probability distributions for the entire output sequence
with torch.no_grad():
    outputs = self.model(documents_ids, return_dict=True)
    logits = outputs.logits # unnormalized
probability_distribution = torch.softmax(logits[0], dim=1)
```

Figure 3.1: This code snippet takes the input documents from the dataset (in our case, the CNN dataset) and derives the document IDs, used to retrieve the probability distributions of the input text. The model mentioned in the code is the first model, in our experimental runs it is the Bart large pre-trained on CNN.

is applied to derive a sequence of tokens. This resultant sequence represents the summary, although not the target, or *Gold*, summary originally desired, but rather the output generated by the network, often referred to as the *Silver* summary.

In summary, our final goal was to train a neural network to perform a task wherein, at each step, the input consists of probability distributions and the output represents the chosen token. However, another issue emerges at this point. How to adapt language models like Bart or T5, which are inherently designed as text-to-text models? In our context, the text component is present in the output, but in the input, we encounter a distinct representation, the probability distribution.

It is pertinent to inspect the dimensions of this distribution. The number of probabilities in the distribution is one for each token in the vocabulary. If we denote the vocabulary size with V and consider it to have L distributions, the shape of the probability distribution is a matrix, LxV.

Within a language model, following the tokenization process, an initial phase involves input embedding. In the embedding input layer, the shape is VxH, denoting the vocabulary size by V and the embedding dimension by H. It is therefore possible to perform a matrix multiplication operation involving LxV and VxH matrices, resulting in a LxH matrix, a probability distribution that can be directly utilized as input for the embedding input layers.

The concept underlying this approach involves taking a language model, pre-trained on a specific dataset. The output probability distributions generated

by this language model are used for the training of the second component, the second language model, as depicted in Figure 3.3 and shown in the code in Figure 3.2.

```python
# Get the target text (summary) to tokenize
target_texts = self.summaries['predictions'][self.index % len(self.summaries)][idx]
self.index += 1  # Increment the index for the next iteration

# Tokenize the silver summaries
summary_ids = self.tokenizer.encode(
    target_texts,       # The text to tokenize
    return_tensors="pt",    # Return a PyTorch tensor
    max_length=512,         # Maximum length of the sequence
    truncation=True,        # Truncate to max_length if longer
    padding="max_length"    # Pad to max_length if shorter
).to(device)                # Move the tensor to the specified device
```

Figure 3.2: This code snippet obtains target texts from the dataset presented in Section 3.2 and tokenizes them, preparing them for further processing. As mentioned earlier, the target text represents the Silver summary, which is generated using a specific decoding strategy. In our experimental runs, we employed the Beam Search strategy to produce these summaries.

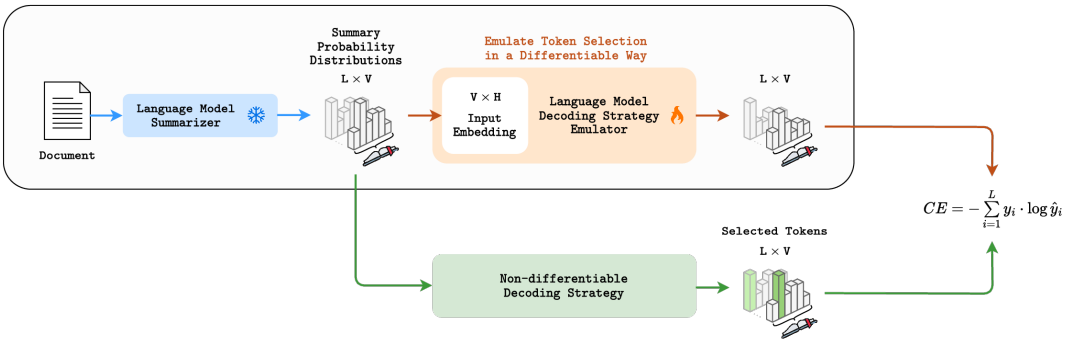

Figure 3.3: Model architecture.

Another important consideration pertains to the choice of loss function. Traditionally, in document summarization tasks, the loss function is based on cross-entropy, which aims to maximize the likelihood of the model's generated tokens aligning with those in the reference, the *Gold* summary.
However, in this adapted scenario where the objective is to emulate a decoding strategy, the notion of the reference undergoes a shift. The conventional *Gold* summary is no longer the reference point. Instead, the loss calculation involves ensuring that the probability assigned to the token corresponding to the chosen strategy is maximized during training, as shown in Figure 3.4.
In essence, by presenting references that are indicative of particular strategies, the network is compelled to adapt and emulate these strategies. This approach

```python
for epoch in range(epochs):
    ...
    for batch_idx, batch in enumerate(train_dataloader):
        ...
        # Cross-entropy loss
        # Meausures the difference between the most probable token predictions and the
        ↪  target token IDs from the decoding strategy to emulate
        loss = torch.nn.functional.cross_entropy(
            output.logits.view(-1, model.config.vocab_size),
            target_ids.view(-1),
            ignore_index=tokenizer.pad_token_id, # ignore padding tokens
        )
        # Backpropagation and gradient computation
        loss.backward()
        ...
    # Compute the average loss over all training batches for the current epoch
    avg_loss = total_loss / len(train_dataloader)
```

Figure 3.4: In the training loop for multiple epochs, this code calculates the cross-entropy loss. The loss measures the dissimilarity between the model's most probable token predictions and the target token IDs from the decoding strategy being emulated. This adaptation in loss calculation aims to maximize the probability assigned to the token associated with the selected strategy during training. The average loss over all training batches for the current epoch is computed as an indicator of training progress.

can be seen as a form of probability maximization, wherein the model learns to prioritize tokens in accordance with predefined strategies.

3.2 Data

Within the domain of natural language generation, Abstractive Summarization has witnessed a notable transformation driven by transformer-based language models. However, the significance of decoding strategies is often overlooked, despite their profound influence on the generated summaries. In light of the myriad token selection heuristics and their associated hyperparameters, it becomes essential to provide guidance to the research community to facilitate well-informed decision-making, based on specific task requirements and target metrics.
To address this existing issue, there has been a study undertaking a comprehensive comparative analysis of the efficacy and efficiency of decoding-time techniques in the context of short, long, and multi-document AS. A rigorous exploration, encompassing over 2500 permutations involving three extensively utilized million-scale autoregressive encoder-decoder models, six distinct datasets, and nine diverse decoding configurations. As a result of this study, important insights are gained into the field, demonstrating that judiciously

Table 3.1: Hyperparameters for each decoding strategy.

HYPERPARAMS*,†	Deterministic				Stochastic				
	Greedy	Contrastive	Beam Search	Diverse Beam Search	Sampling	Top-k Sampling	Top-p Sampling	Beam Sampling	η Sampling
`no_rep_ngram_size`					{2, 3, 4, 5}				
`early_stopping`			✓	✓				✓	
`diversity_penalty`				0.{2, 4, 6, 8}, 1.0					
`num_beams`			[2, 3, …, 10]	[2, 3, …, 10]				[2, 3, …, 10]	
`temperature`					0.{8, 9}, 1.0	0.{8, 9}, 1.0	0.{8, 9}, 1.0		
`top_k`		[20, 30, …, 60]				[20, 30, …, 60]	{/, 20, …, 60}		
`top_p`							0.{4, 6, 8}	0.9	
`penalty_alpha`		0.{2, 4, 6, 8}, 1.0							
`eta_cutoff`									$\{4, 2\}\times10^{-3}$, $\{9, 6, 3\}\times10^{-4}$
`do_sample`					✓	✓	✓	✓	✓
# Runs	16	400	144	720	48	240	864	144	80

* {…} = sets, […] = series, ✓ = true, / = none.

† orange and cyan initialization schemes are tested on 2 and 4 datasets, respectively; all other values are equally applied to the 6 surveyed datasets.

optimizing decoding choices can result in significant performance improvements. In addition to human evaluation, the effects of these decoding strategies are quantitatively assessed using a battery of ten automatic metrics, encompassing dimensions such as semantic similarity, factuality, compression, redundancy, and carbon footprint. This study introduced an innovative and first-of-its-king dataset that pairs Abstractive Summarization gold input-output examples with language models' predictions generated under a wide spectrum of decoding options.

3.2.1 Approach

A series of generative experiments, denoted as $(S_{\mathcal{H}}, D, M)$, have been devised for the purpose of this dataset study. Herein, $S_{\mathcal{H}}$ represents a decoding strategy characterized by fixed hyperparameters, D signifies the dataset employed, which introduces distinct challenges and length constraints associated with Abstractive Summarization, and M denotes the autoregressive language model under consideration.

The entirety of the experimental undertaking encompasses a total of 2656 individual runs, as delineated in Table 3.1, necessitating a computational effort spanning 73 GPU days. It is worth noting that, to the best of our knowledge, this endeavor constitutes the most finely-grained hyperparameter sweep to date within the realm of NLG literature.

To achieve this level of granularity, a grid search methodology has been employed, enabling the concurrent adjustment of generative variables associated with each heuristic. This approach was guided by careful consideration of the range of values, informed by insights gleaned from prior research studies in the field.

3.2.2 Datasets

To create this dataset six notable English-language datasets that are distinguished by their domain and are publicly accessible for use as testbeds have been selected. Specifically, two datasets for each Abstract Summarization family were chosen.

- Short Document Summarization (SDS). **XSum** [42], which comprises a collection of BBC articles spanning the years 2010 to 2017, and is tailored for the generation of highly abstractive one-sentence extreme summaries. Additionally, **CNN/DM** [40] was also included. **CNN/DM** is a dataset featuring articles from various news sources, accompanied by concise sentence-level highlights.
- Long Document Summarization (LDS). **PubMed** and **arXiv** [59]. Both of these two datasets consist of extensive and structured scientific papers along with their corresponding abstracts.
- Multi-Document Summarization (MDS). The selection includes **Multi-News** [60], a dataset containing news articles and human-authored summaries sourced from `newser.com`, which aggregates content from numerous U.S. and international news outlets. **Multi-LexSum** [61] is also present as a compilation of expert-written summaries for legal documents derived from federal civil rights lawsuits.[1]

3.2.3 Models

To ensure impartiality, three state-of-the-art transformer-based models in their large configurations are employed in this study. These models have already undergone fine-tuning on the datasets described in Section 3.2.2.
The first model, **Bart** [29], boasts 406 million parameters and exhibits quadratic memory complexity in relation to input size, thus limiting its capacity to handle intricate sequences of up to 1024 tokens.
The second model, **Led** [31] (LDS), utilizes sparse attention mechanisms to provide **Bart** with linear input scaling capabilities, enabling it to process sequences of up to 16,384 tokens.
The third model, **Primera** [35] (MDS), extends the functionality of **Led** by adapting it to multi-input scenarios through a pretraining objective that is specific to summarization tasks. This entails concatenating input sources with a designated separator token, resulting in a single input sequence of up to 4096 tokens.

[1] In light of the multi-granular nature of summaries within Multi-LexSum, the focus was on the D $\rightarrow$ S task, specifically, the synthesis of source documents into concise *short* summaries.

Table 3.2: Evaluation benchmarks after sampling. Top: SDS. Center: LDS. Bottom: MDS. Statistics are yielded with NLTK [3] and include sample size, number of sources per instance, and total words in source and target texts. All values are averaged except "# Samples."

			Source		Target
Dataset	**Domain**	**# Samples**	# Docs	# Words	# Words
XSum	News	1134	1	414.1	22.8
Cnn/Dm	News	1148	1	753.8	58.0
PubMed	Biomedical	667	1	3095.5	207.2
arXiv	Scientific	644	1	5770.7	160.6
Multi-News	News	555	2.8	1947.9	248.0
Multi-LexSum	Legal	616	10.3	88,122.9	126.5

3.2.4 Data Sampling

In light of the extensive expanse of research within the domains of decoding and evaluation, it becomes infeasible to engage with complete datasets, primarily attributable to constraints imposed by temporal and resource limitations. Consequently, it was opted for the utilization of representative subsets from each dataset, with the exception of the succinct Multi-LexSum corpus. The determination of an appropriate sample size for the purpose of discerning statistically significant metric effects is accomplished through the application of power analysis, as explained in Section 3.2.5. Subsequently, a 10% dataset proportion was employed, selected via the implementation of proportional stratified random sampling without replacement, thereby ensuring a robust representation of distinct subgroups. Notably, stratification is carried out based on document and summary length, with the delineation of classes into $\{\text{short}, \text{medium}, \text{large}\}$ categories via the computation of tertiles. Furthermore, for the context of Multi-Document Summarization, due consideration is also accorded to the number of source documents involved in the summarization process. An exhaustive compilation of the datasets used in the study is presented in Table 3.2.

3.2.5 Power Analysis

A prudent preliminary step involves the determination of the necessary sample size to ensure adequate statistical power for detecting potentially meaningful metric effects across decoding runs if they were to exist. In pursuit of this objective, a power analysis for the t-tests is conducted employing the `statsmodels` Python library (version 0.14.0). Technically, consideration is given to several key factors, which include:

- *Power level*: This parameter denotes the probability of correctly identifying a true effect, should it indeed be present. Higher power levels signify

an enhanced capacity for detecting effects. In this analysis, a widely endorsed value of 0.8 is adopted, corresponding to an 80% likelihood of correctly detecting a genuine effect.

- *Significance level*: The significance level serves as the threshold utilized to determine statistical significance. In this study, a significance level of 0.05 is chosen, equating to a 5% probability of incorrectly rejecting the null hypothesis.

- *Effect size*: Effect size represents the magnitude of the anticipated difference among various decoding strategies. Cohen's d [62] is employed as the measure of effect size, computed by taking the difference in means between two groups and dividing it by the pooled standard deviation, i.e., $(\bar{x_1}-\bar{x_2})/s$. Based on preliminary experiments, an informed estimate is made regarding the aggregate effect size across all NLG metrics. Specifically, $\bar{x_1} - \bar{x_2}$ is set at 0.01, with s assigned a value of 0.05.

The resultant required sample size is calculated to be 393.4.

3.2.6 Decoding Hyperparameters

For the Contrastive Search method, the parameters α and k were manipulated, selecting values from the sets $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $\{20, 30, 40, 50, 60\}$, respectively. For the Beam Search approach, the beam size b was varied within the range of $[2, 10]$, providing flexibility for a trade-off between quality and computational performance. In the case of Diverse Beam Search, the same beam size b was maintained while implementing the diversity parameter Δ using Hamming distance, as recommended by Vijayakumar et al. (2018) [23], diversity penalty values λ was considered from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$; the number of subgroups is set equal to b. The exploration range for b remains consistent when employing Beam Sampling. Additionally, for Top-k Sampling, k values were selected from $\{20, 30, 40, 50, 60\}$, while for Top-p Sampling, based on the findings of DeLucia et al. (2021) [63], the value p eas determined from the set $\{0.4, 0.6, 0.8\}$. In the case of η Sampling, a search over the η-cutoff parameter was performed, considering values from the set $\{0.004, 0.002, 0.0009, 0.0006, 0.0003\}$, as suggested by Hewitt et al. (2022) [28]. Furthermore, for Pure (ancestral), Top-k, and Top-p Sampling, wan ablation study on the temperature hyperparameter was conducted, exploring values of τ from the set $\{0.8, 0.9, 1.0\}$, guided by the recommendations of Holtzman et al. (2020) [26] and Pasunuru et al. (2021) [64]. Default hyperparameters are retained unless otherwise specified, depending on the model configuration.
In the realm of artificial summarization, redundancy is a known challenge [65].

Therefore, the application of non-negligible word-level n-grams penalties was investigated, inspired by Klein et al. (2017) [66] and Paulus et al. (2018) [67], particularly in the context of Abstractive Summarization. To mitigate redundancy, constraints were enforced on the decoder, ensuring that it did not generate the same sequence of words more than once during testing. This is achieved by manually setting the probability of already hypothesized n-grams to 0. Moreover experiments with different values of n drawn from $\{3, 4, 5\}$ were conducted for the PubMed, arXiv, Multi-News, and Multi-LexSum datasets, and from $\{2, 3\}$ for XSum (one-line output) and Cnn/Dm. This experimental setup results in a minimum of 16 runs for each decoding strategy.
As is customary in abstractive summarization, minimum and maximum summary length constraints were established for each dataset, as detailed in Table 3.3. These constraints are determined by analyzing the summary lengths in the validation tests, and they prevent the generation of an EOS token before or after a specified threshold.

Table 3.3: List of the utilized datasets and relative length constraints.

Dataset	**URL**	**Input**	**Output**	
		Max Len	Min Len	Max Len
XSum	https://huggingface.co/datasets/xsum	1024	20	100
Cnn/Dm	https://huggingface.co/datasets/cnn_dailymail	1024	20	100
PubMed	https://huggingface.co/datasets/ccdv/pubmed-summarization	4096	100	256
arXiv	https://huggingface.co/datasets/ccdv/arxiv-summarization	4096	100	256
Multi-News	https://huggingface.co/datasets/multi_news	4096	100	256
Multi-LexSum	https://huggingface.co/datasets/allenai/multi_lexsum	4096	50	256

3.2.7 Decoding runs

The present state of research reveals a conspicuous scarcity of studies pertaining to generative outcomes influenced by the prevailing next-token selection strategies. As of our current understanding, only two systematic benchmarks have been identified that specifically scrutinize cross-task behaviors and the diversification of outputs Ippolito et al. (2019) [68] and Meister et al. (2022) [69]. Comparative investigations primarily emanate from scholarly works that introduce innovative decoding techniques, guided by the objectives of furnishing either theoretical validation or empirical substantiation of their efficacy. It is widely acknowledged that the choice of decoding strategy is inextricably tied to the nature of the task at hand; however, the predominant focus within the research community has been on open-ended generation, thereby leaving a multitude of unresolved inquiries, especially within applied domains such as Abstractive Summarization. In this context, Beam Search is frequently accorded the status of a de-facto standard, with relatively limited exploration of alternative decoding methodologies [31, 29, 35]. Notwithstanding the collective

superficiality and recent heuristic developments, it is imperative to exercise critical scrutiny and reevaluate this widely accepted assumption. Furthermore, research endeavors that do incorporate Abstractive Summarization in their investigations, often derive task-level inferences from the analysis of a singular dataset and a restricted research purview, frequently resorting to the utilization of SDS for the sake of experimental expediency [69, 21]. It is notable that efficiency metrics related to runtime and CO_2 emissions are seldom included in these examinations. In comparison to prior works, this task-specific inquiry distinguishes itself through the comprehensive exploration of diverse heuristics and their configurations, affording a nuanced elucidation of the attributes characterizing the SDS/LDS/MDS family. To the best of our knowledge, this constitutes the most expansive study of decoding strategies, also encompassing a broad array of Abstractive Sumamrization models under evaluation. For instance, preceding investigations by Meister et al. (2022) [69] and Su et al. (2022) [21] exclusively considered a single Abstractive Summarization model each, namely BART [29] and GPT-2 [18], respectively.

Chapter 4

Experiments

4.1 Experimental Setup

This chapter explains in detail the technologies used for the experiments' runs, along with the results achieved.

4.1.1 Bart

Our architectural framework remains adaptable to various language models without being bound only to a certain kind. However, for the purpose of conducting experiments, we opted to employ Bart [29], specifically a Bart large model pre-trained on the CNN dataset.
BART (Bidirectional and AutoRegressive Transformer) [29] stands as a prominent model within the realm of NLP. Derived from the robust transformer architecture, renowned for its exceptional capability in capturing extended dependencies and contextual nuances in sequential data, BART introduces several notable advancements. Noteworthy among these is BART's utilization of a distinctive bidirectional and autoregressive training paradigm, effectively amalgamating the strengths inherent in bidirectional models like BERT [37] and autoregressive models akin to GPT [70]. This innovative synthesis empowers BART to adeptly capture information flow from both directions, resulting in the generation of sequences characterized by a high degree of quality and coherence. BART has effectively showcased its prowess across a spectrum of NLP tasks, with particular distinction in the domain of text summarization, thereby solidifying its status as a pivotal model within the field.
Throughout the pretraining phase, BART undergoes rigorous training on extensive corpora, employing a denoising autoencoder objective. Its training objective involves the acquisition of the capability to reconstruct original documents from intentionally corrupted versions, with optimization based on the

reconstruction loss between the decoder's output and the source document. What distinguishes BART from its counterparts is its exceptional adaptability in addressing document corruption. In contrast to models constrained by specific noising methodologies, BART accommodates various forms of document corruption. The work by Lewis et al. [29] introduces a multitude of experimental noising schemes, as illustrated in Figure 4.1, which includes techniques such as text filling and sentence permutation, both of which have demonstrated highly favorable outcomes. In text filling, segments of the text are obscured by a single [MASK] token, thereby presenting a challenge to the model to reconstruct the original sentence. Conversely, sentence permutation involves the random reshuffling of sentences within a document. The BART model is

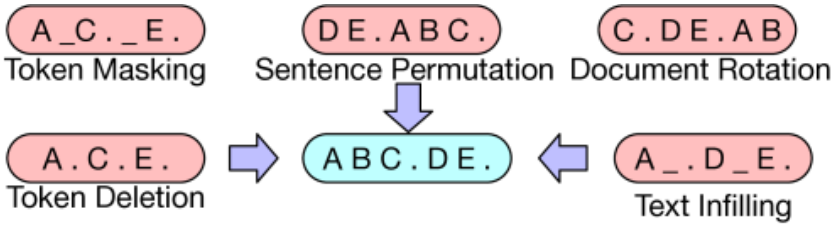

Figure 4.1: Transformations for noising the input that was used in BART. From [29]

available in two versions: BART-base, characterized by six layers in both the encoder and decoder components with a hidden size of 768 and BART-large, featuring twelve layers in each component and a hidden size of 1024.
Following the pretraining phase, BART can be subjected to fine-tuning for various downstream tasks encompassing sequence classification, sequence generation, and machine translation. Exceptional performance on benchmark datasets has been demonstrated by the model, surpassing previous state-of-the-art models. In terms of summarization, BART has exhibited superior performance in comparison to alternative approaches. Notably, it has outperformed previous state-of-the-art models on well-established benchmark datasets such as CNN/Daily Mail and XSum, thus validating its efficacy in producing high-quality summaries. Furthermore, BART has demonstrated significant enhancements in other tasks, including dialogue response generation and abstractive question answering, thereby further establishing its versatility and proficiency across diverse domains within natural language processing.
In conclusion, BART represents a potent and versatile model that harnesses bidirectional and autoregressive training methods to generate sequences of high quality. Its adaptability in managing various forms of document corruption and its remarkable performance across a spectrum of tasks bestow upon it substantial value as a tool within the realm of natural language processing.

4.1.2 Training Data and implementation details

For the purposes of our analysis, we focused on a specific subset of experiments within the extensive dataset described in Section 3.2. Specifically, we isolated experiments pertaining to the beam search decoding strategy and selectively considered those instances where the dataset column was labeled as "CNN" and the model column indicated "Bart". Consequently, our dataset exclusively encompasses runs generated by a substantial Bart model pre-trained on CNN data. [1] Each of these runs comprises the model's predictions on the CNN dataset, alongside their respective hyperparameters, results, and associated metrics.
The runs regarding the training of our model were performed on a Nvidia GeForce RTX3090 GPU with 24∼GB of dedicated memory, 64∼GB VRAM, and an Intel® Core™ i9-10900X1080 CPU @ 3.70GHz.
The code is based on the following libraries:

- We employed the **PyTorch** library [71] version 2.0.0, leveraging the power of the **CUDA** toolkit (version 11.8) [2] for GPU acceleration. This combination provided essential computational resources and GPU support necessary for our experiments.

- To implement our natural language processing tasks, we relied on the **HuggingFace Transformers** library (version 4.33.1). It allowed us to efficiently manipulate pre-trained language models and perform various text generation tasks.

- For evaluating our model's performance, we integrated the **HuggingFace Evaluate**[3] library (version 0.3.0). This library proved invaluable for comparing the text outputs generated by our second model with the *Silver* summaries taken from our dataset. We measured the quality and coherence of our model's generated text against the reference summaries, using the ROUGE metric [72].

4.1.3 Dataset implementation details

Inference runs of the dataset presented in Section 3.2 were performed on a workstation with 4 Nvidia GeForce RTX3090 GPUs with 24∼GB of dedicated memory each, 64∼GB VRAM, and an Intel® Core™ i9-10900X1080 CPU @ 3.70GHz.

[1] https://huggingface.co/facebook/bart-large-cnn
[2] https://docs.nvidia.com/cuda/archive/11.8.0/
[3] https://huggingface.co/docs/evaluate/index

The code is founded on PyTorch 1.10.2 [71], the HuggingFace Transformers [73] and Datasets [74] libraries.
Table 3.3 and Table 4.1 enumerate datastore references and model checkpoints fine-tuned on them.
Sticking to [35], the input for PRIMERA is the concatenation of documents within the clusters (in the same order), where truncation is applied to each document based on the input length limit divided by the size of the cluster, thus guaranteeing the representation of all sources.
All decoding runs for autoregressive summary generations are subject to the method `generate()`.
We set the global seed to 42 for reproducibility.

Table 4.1: List of the utilized models.

Model	Dataset	URL
BART-large	XSUM	`https://huggingface.co/facebook/bart-large-xsum`
	CNN/DM	`https://huggingface.co/facebook/bart-large-cnn`
LED-large	PUBMED	`https://huggingface.co/patrickvonplaten/led-large-16384-pubmed`
	ARXIV	`https://huggingface.co/allenai/led-large-16384-arxiv`
PRIMERA-large	MULTI-NEWS	`https://huggingface.co/allenai/PRIMERA-multinews`
	MULTI-LEXSUM	`https://huggingface.co/allenai/primera-multi_lexsum-source-short`

4.1.4 Experiment tracking

All of the runs were tracked with Weights & Biases.[4] CO_2 emissions were monitored with the CodeCarbon library.[5]

4.2 Results

Here, we present the results obtained from the training and evaluation of our model. All experiments were conducted on a Nvidia GeForce RTX 3090 GPU with 24 GB of dedicated memory, 64 GB of VRAM, and an Intel® Core™ i9-10900X1080 CPU @ 3.70GHz.
Our primary objective was to assess the model's training progress by monitoring the loss reduction over time. We are pleased to report that the loss consistently decreased during the training process, as illustrated in Figure 4.2, signifying the successful acquisition of the decoding strategy we aimed to replicate. In our experiments, we focused on emulating the Beam Search decoding strategy.

[4] `https://wandb.ai`
[5] `https://github.com/mlco2/codecarbon`

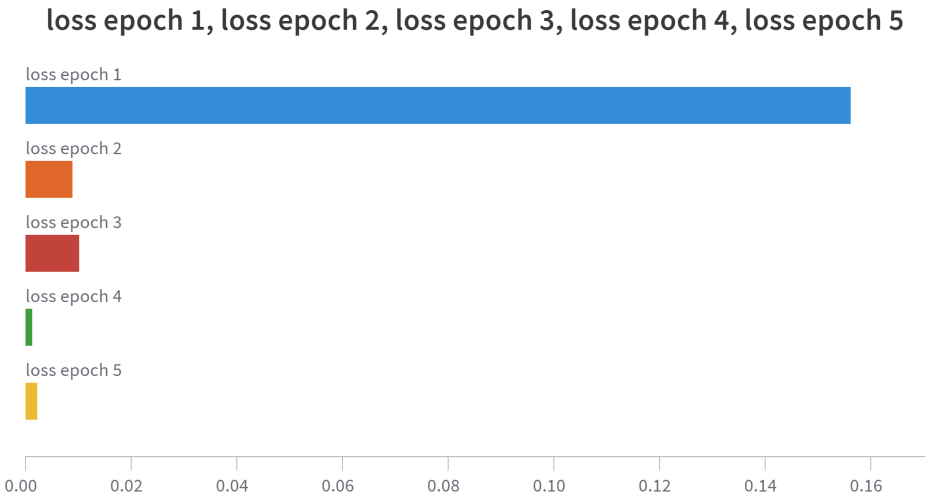

Figure 4.2: Progression of loss during training epochs.

The following parameters were utilized for the experiments:

- **Batch size**: 4
- **Learning Rate**: 1×10^{-4}
- **Number of epochs**: 5

The average total carbon footprint associated with our computations amounted to 0.0628 tCO2e, while the average execution time per experiment was approximately 30 minutes.

Conclusioni e Sviluppi Futuri

In conclusione, questa tesi ha esplorato il panorama, in continua evoluzione, dell'*Abstractive Summarization*, evidenziando i progressi significativi guidati da nuovi *language model* neurali, architetture basate su i *transformer* e strategie di *decoding* innovative. L'obiettivo centrale di questo lavoro è stato lo sviluppo di un'architettura unica che affronta la sfida della codifica e della replica delle strategie di *decoding* nel contesto della sintesi di documenti.
Fino ad ora, la sintesi dei documenti si è affidata a modelli mono-linguistici, come Bart, per generare sintesi basate su *language modeling* autoregressivo. Tuttavia, la non-differenziabilità intrinseca delle strategie di *decoding* ha rappresentato un ostacolo sostanziale. Per superare questa sfida, abbiamo introdotto un approccio innovativo che prevede l'integrazione di un secondo *language model*, per approssimare l'operatività svolta delle strategie di *decoding*.
Poiché il campo dell'elaborazione del linguaggio naturale continua a evolversi, le intuizioni e le metodologie presentate in questa tesi promettono ulteriori progressi e sfide. In primo luogo, l'integrazione di altre strategie di *decoding* oltre a Beam Search, che è la strategia utilizzata durante i nostri esperimenti. Un'altra direzione interessante per la ricerca futura è l'esplorazione di *dataset* differenti, compresi quelli che coinvolgono documenti per la *long-document summarization*, per quella *multi-document* e per il riassunti di dialoghi.
Inoltre, un'altra sfida interessante consiste nel modificare la *attention mask* in modo da non dare importanza a *token* ritenuti irrilevanti, come il *token* indicante la fine della frase. Questa modifica ai meccanismi di *attention* potrebbe potenzialmente migliorare l'efficienza e l'efficacia del modello.
Infine, sarebbe interessante addestrare una rete di tipo *sequence-to-sequence*, simile a *Optformer* [75], per ottimizzare autonomamente i valori degli iperparametri associati a specifiche strategie di decodifica. Attualmente, gli utenti devono spesso regolare manualmente questi iperparametri per ottenere i risultati di sintesi desiderati. Tuttavia, impiegando tecniche avanzate di *deep learning* e di *reinforcement learning*, possiamo puntare a un sistema che impari ad adattare e ottimizzare questi parametri automaticamente. Questa ottimizzazione autonoma degli iperparametri potrebbe semplificare in modo significativo il processo di sintesi, rendendolo più accessibile a una più ampia gamma di

utenti e riducendo la necessità di competenze specialistiche per impostare correttamente questi parametri.
Optformer sfrutta dati di ottimizzazione su larga scala con rappresentazioni basate sul testo. A differenza dei metodi precedenti, non si basa su vincoli numerici per il meta-apprendimento, rendendolo flessibile per vari spazi di ricerca iperparametrici. Rappresenta tutti i dati dello studio, compresi i metadati basati sul testo, come una sequenza di token. OptFormer utilizza queste informazioni per generare nuovi iperparametri, prevedere l'accuratezza del compito e perfezionare in modo adattivo gli iperparametri nel corso di più iterazioni. È in grado di imitare il comportamento di vari algoritmi di ottimizzazione e di prevedere i valori oggettivi con elevata precisione.

Conclusions and Future Challenges

In conclusion, this thesis has explored the evolving landscape of Abstractive Summarization, highlighting the significant advancements driven by novel neural language models, transformer-based architectures, and innovative decoding strategies. The central focus of this work has been the development of a unique model architecture that addresses the challenge of encoding and replicating decoding strategies in the context of document summarization.
Traditionally, document summarization has relied on single-language models, such as Bart, to generate summaries based on autoregressive language modeling. However, the inherent non-differentiability of decoding strategies presented a substantial obstacle. To overcome this challenge, we introduced an innovative approach involving the integration of a second language model to approximate the functionality of decoding strategies.
As the field of natural language processing continues to evolve, the insights and methodologies presented in this thesis hold promise for further advancements and challenges. First and foremost, the integration of other decoding strategies besides Beam Search, the strategy we employed during our experiments.
Another interesting direction for future research is to explore various different datasets, including those involving long documents, multi-document sources, and dialogues, for summarization tasks.
Moreover, a pressing challenge lies in modifying the attention mask to give no importance to tokens that are deemed irrelevant, such as the End of Sentence (EOS) tokens. This adjustment to attention mechanisms could potentially enhance the efficiency and effectiveness of the model.
Lastly, it would be interesting to train a sequence-to-sequence network, similar to *Optformer* [75], to autonomously optimize the values of hyperparameters associated with specific decoding strategies. Currently, users are often required to manually fine-tune these hyperparameters to achieve the desired summarization results. However, by employing advanced techniques in deep learning and reinforcement learning, we can strive towards a system that learns to adapt and optimize these parameters automatically. This autonomous hyperparameter optimization could significantly simplify the summarization process, making it more accessible to a wider range of users and reducing the need for specialized

expertise in setting these parameters correctly.
Optformer leverages large-scale optimization data with text-based representations. Unlike previous methods, it doesn't rely on numerical constraints for meta-learning, making it flexible for various hyperparameter search spaces. It represents all study data, including text-based metadata, as a sequence of tokens. The OptFormer uses this information to generate new hyperparameters, predict task accuracy, and adaptively refine hyperparameters over multiple iterations. It can imitate the behavior of various optimization algorithms and predict objective values with high accuracy.

Ringraziamenti

Il mio primo ringraziamento va al relatore, il Prof. Gianluca Moro, per l'opportunità datami e per aver stimolato il mio interesse verso questa disciplina. Voglio poi ringraziare il co-relatore, il Dottor Giacomo Frisoni, per tutto l'aiuto, le spiegazioni e il supporto che mi ha dato in questi ultimi mesi.

Ringrazio dal profondo del mio cuore mamma e papà, ai quali sarò per sempre grata, per gli insegnamenti trasmessi, per il sostegno e l'amore che mi hanno dato, per averci sempre creduto e aver sempre fatto il possibile affinché qualunque mio desiderio si avverasse.
Ci tengo a ringraziare anche degli amici che sono stati fondamentali per il mio percorso e ricordargli che ovunque saranno nel mondo in futuro, saranno anche sempre nel mio cuore e nei miei pensieri.

- Grazie Valentina e Salvatore per essere stati i miei compagni di triennale. Grazie per tutti i pomeriggi di studio, tutti i progetti fatti insieme, tutti i momenti di disperazione e quelli di incoraggiamento. Grazie per aver condiviso con me le lacrime e i sorrisi.

- Grazie Serena, Carlo, Gabriele e Luca per essere i migliori amici che potessi mai desiderare. Grazie per essere ancora miei amici e sopportarmi dopo tutti questi anni. Grazie per l'amore e la compagnia che mi avete fatto percepire nonostante la distanza.

Benedetta
5 Ottobre 2023

Acknowledgments

My first thanks go to my supervisor, Professor Gianluca Moro, for the opportunity he gave me and for sparking my interest in this field.
I would also like to thank my co-supervisor, Dr. Giacomo Frisoni, for all the help, the explanations, and the support he provided me over these past months.

From the bottom of my heart, I want to thank my mom and dad, to whom I will be forever grateful, for the lessons they imparted, for the support and love they gave me, for always believing in me, and for always doing their best to make any of my wishes come true.
I also want to thank some friends who have been crucial to my journey and remind them that wherever they may be in the world in the future, they will always be in my heart and thoughts.

- Thank you, Valentina and Salvatore, for being my undergraduate companions. Thanks for all the afternoons of study, all the projects we did together, all the moments of despair, and those of encouragement. Thank you for sharing tears and smiles with me.

- Thank you, Serena, Carlo, Gabriele, and Luca, for being the best friends I could ever wish for. Thanks for still being my friends and putting up with me after all these years. Thanks for the love and company you made me feel despite the distance.

Benedetta
October 5, 2023

Bibliography

[1] Tanya Goyal, Nazneen Rajani, Wenhao Liu, and Wojciech Kryscinski. Hydrasum: Disentangling style features in text summarization with multi-decoder models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 464–479. Association for Computational Linguistics, 2022.

[2] Giacomo Frisoni, Luca Ragazzi, David Cohen, Gianluca Moro, Antonella Carbonaro, and Claudio Sartori. Abstractive summarization through the prism of decoding strategies. *Submitted to the Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[3] Steven Bird. NLTK: the natural language toolkit. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics, 2006.

[4] Bill Z. Manaris. Natural language processing: A human-computer interaction perspective. *Adv. Comput.*, 47:1–66, 1998.

[5] Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. A survey of natural language generation. *ACM Comput. Surv.*, 55(8):173:1–173:38, 2023.

[6] Zhiwei Feng. *Formal Analysis for Natural Language Processing: A Handbook*. Springer, 2023.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[9] Gian Wiher, Clara Meister, and Ryan Cotterell. On decoding strategies for neural text generators, 2022.

[10] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009.

[11] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

[12] Daniel Jurafsky and James H. Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson, 2009.

[13] Yun Chen, Kyunghyun Cho, Samuel R. Bowman, and Victor O.K. Li. Stable and effective trainable greedy decoding for sequence to sequence learning, 2018.

[14] Jiatao Gu, Kyunghyun Cho, and Victor O. K. Li. Trainable greedy decoding for neural machine translation. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1968–1978. Association for Computational Linguistics, 2017.

[15] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *CoRR*, abs/1610.02424, 2016.

[16] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 426–434, New York, NY, USA, 2008. Association for Computing Machinery.

[17] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. On sampling top-k recommendation evaluation. In *Proceedings of the 26th ACM SIGKDD*

International Conference on Knowledge Discovery & Data Mining, KDD '20, page 2114–2124, New York, NY, USA, 2020. Association for Computing Machinery.

[18] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[19] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada, July 2023. Association for Computational Linguistics.

[20] John Hewitt, Christopher D. Manning, and Percy Liang. Truncation sampling as language model desmoothing, 2022.

[21] Xi'ao Su, Ran Wang, and Xinyu Dai. Contrastive learning-enhanced nearest neighbor mechanism for multi-label text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 672–679, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[22] Bruce T Lowerre. *The harpy speech recognition system.* Carnegie Mellon University, 1976.

[23] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. In *AAAI*, pages 7371–7379. AAAI Press, 2018.

[24] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[25] Xu-Wang Han, Hai-Tao Zheng, Jin-Yuan Chen, and Cong-Zhi Zhao. Diverse decoding for abstractive document summarization. *Applied Sciences*, 9(3), 2019.

[26] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*. OpenReview.net, 2020.

[27] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. In *ICLR*. OpenReview.net, 2020.

[28] John Hewitt, Christopher Manning, and Percy Liang. Truncation sampling as language model desmoothing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[29] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[30] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR, 2020.

[31] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

[32] 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. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States, July 2022. Association for Computational Linguistics.

[33] Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, et al. Efficient attentions for long document summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1419–1436. Association for Computational Linguistics, 2021.

[34] Jason Phang, Yao Zhao, and Peter J. Liu. Investigating efficiently extending transformers for long input summarization. *CoRR*, abs/2208.04347, 2022.

[35] Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263, Dublin, Ireland, May 2022. Association for Computational Linguistics.

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

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[38] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. AMMUS : A survey of transformer-based pretrained models in natural language processing. *CoRR*, abs/2108.05542, 2021.

[39] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[40] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics.

[41] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

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

for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[43] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.

[44] E.J. Gumbel. *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*. Applied mathematics series. U.S. Government Printing Office, 1954.

[45] Chris J Maddison, Daniel Tarlow, and Tom Minka. Ast sampling. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[46] Ronan Collobert, Awni Y. Hannun, and Gabriel Synnaeve. A fully differentiable beam search decoder. *CoRR*, abs/1902.06022, 2019.

[47] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, 2002.

[48] Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[49] Shuyang Cao and Lu Wang. HIBRIDS: Attention with hierarchical biases for structure-aware long document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–807, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[50] Giacomo Frisoni, Paolo Italiani, Stefano Salvatori, and Gianluca Moro. Cogito ergo summ: Abstractive summarization of biomedical papers via semantic parsing graphs and consistency rewards. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 12781–12789. AAAI Press, 2023.

[51] Giacomo Frisoni, Paolo Italiani, Francesco Boschi, and Gianluca Moro. Enhancing biomedical scientific reviews summarization with graph-based factual evidence extracted from papers. In Alfredo Cuzzocrea, Oleg Gusikhin, Wil M. P. van der Aalst, and Slimane Hammoudi, editors, *Proceedings of the 11th International Conference on Data Science, Technology and Applications, DATA 2022, Lisbon, Portugal, July 11-13, 2022*, pages 168–179. SCITEPRESS, 2022.

[52] Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *CoRR*, abs/2210.01241, 2022.

[53] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics.

[54] David Wan and Mohit Bansal. Factpegasus: Factuality-aware pre-training and fine-tuning for abstractive summarization, 2022.

[55] Gianluca Moro, Luca Ragazzi, Lorenzo Valgimigli, and Davide Freddi. Discriminative marginalized probabilistic neural method for multi-document summarization of medical literature. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 180–189, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[56] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact aware neural abstractive summarization. *CoRR*, abs/1711.04434, 2017.

[57] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online, July 2020. Association for Computational Linguistics.

[58] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[59] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[60] Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy, July 2019. Association for Computational Linguistics.

[61] Zejiang Shen, Kyle Lo, Lauren Yu, Nathan Dahlberg, Margo Schlanger, and Doug Downey. Multi-lexsum: Real-world summaries of civil rights lawsuits at multiple granularities, 2022.

[62] Jacob Cohen. Statistical power analysis. *Current Directions in Psychological Science*, 1(3):98–101, 1992.

[63] Alexandra DeLucia, Aaron Mueller, Xiang Lisa Li, and João Sedoc. Decoding methods for neural narrative generation. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 166–185, Online, August 2021. Association for Computational Linguistics.

[64] Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. Efficiently summarizing text and graph encodings of multi-document clusters. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4768–4779, Online, June 2021. Association for Computational Linguistics.

[65] Wen Xiao and Giuseppe Carenini. Systematically exploring redundancy reduction in summarizing long documents. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 516–528, Suzhou, China, December 2020. Association for Computational Linguistics.

[66] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[67] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017.

[68] Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy, July 2019. Association for Computational Linguistics.

[69] Clara Meister, Gian Wiher, and Ryan Cotterell. On decoding strategies for neural text generators. *Trans. Assoc. Comput. Linguistics*, 10:997–1012, 2022.

[70] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[71] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, 2020.

[72] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[73] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[74] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan

Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[75] Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers, 2022.