Article

DecoStrat: Leveraging the Capabilities of Language Models in D2T Generation via Decoding Framework

Elias Lemuye Jimale [*] , Chen Wenyu [*] , Mugahed A. Al-antari [*] , Yeong Hyeon Gu , Victor Kwaku Agbesi , Wasif Feroze

Posted Date: 28 October 2024

doi: 10.20944/preprints202410.2163.v1

Keywords: decoding methods; data-to-text generation (D2T); language models (LM); natural language generation (NLG); natural language processing (NLP)

Article

DecoStrat: Leveraging the Capabilities of Language Models in D2T Generation via Decoding Framework

Elias Lemuye Jimale [1,2,*], Chen Wenyu [1,*], Mugahed A. Al-Antari [3], Yeong Hyeon Gu [3], Victor Kwaku Agbesi [1] and Feroze Wasif [1]

1 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
2 School of Electrical Engineering and Computing, Adama Science and Technology University, Adama, Ethiopia
3 Department of Artificial Intelligence and Data Science, College of AI Convergence, Daeyang AI Center, Sejong University, Seoul, Republic of Korea
* Correspondence: lemuye@gmail.com (E.L.J.); cwy@uestc.edu.cn (C.W.)

Abstract: Current language models have achieved remarkable success in NLP tasks. Nonetheless, individual decoding methods face difficulties in realizing the immense potential of these models. The challenge is primarily due to the lack of a decoding framework that can integrate language models and decoding methods. We introduce the DecoStrat which bridges the gap between language modeling and the decoding process in D2T generation. By leveraging language models, DecoStrat facilitates the exploration of alternative decoding methods tailored to specific tasks. We fine-tune the model on the MultiWOZ dataset to meet task-specific requirements and employ it to generate output(s) through multiple interactive modules of the framework. The Director module orchestrates the decoding processes, engaging the Generator to produce output(s) text based on the selected decoding method and input data. The Manager module enforces a selection strategy, integrating Ranker and Selector to identify the optimal result. Evaluations on the stated dataset show that DecoStrat effectively produces diverse and accurate output, with MBR variants consistently outperforming other methods. DecoStrat with the T5-small model surpasses some baseline. Generally, the findings highlight DecoStrat's potential for optimizing decoding methods in diverse real-world applications.

Keywords: decoding methods; data-to-text generation (D2T); language models (LM); natural language generation (NLG); natural language processing (NLP)

1. Introduction

The task of accurately generating diverse and contextually relevant text from structured data is crucial but challenging, with numerous applications such as dialogue systems [1], healthcare reporting [2,3], weather forecasting [4], and sports summarization [5]. For instance, leveraging D2T generation can empower chatbots to provide personalized responses to customer inquiries. However, this task requires language models to grasp complex relationships and contextual dependencies within the structured input data. Moreover, the input data can be in various formats, such as meaning representations [6], graphs [7], and tables [8], making the process difficult. Despite the importance of D2T generation, leveraging the immense capabilities of transformer models, such as T5 [9] and BART [10], remains a significant challenge. While these models have shown remarkable success in natural language processing (NLP) [11], generating contextually relevant text from structured data is a complex task that requires more than just powerful models [12]. Recent studies have highlighted the limitations of current approaches, including the need for more effective handling of contextual dependencies in more realistic settings [13–16].

One significant hurdle in harnessing the potential of language models for D2T generation lies in the lack of a unified and modular framework that can effortlessly integrate multiple decoding methods. The absence of this framework hampers the progress and seamless development, integration, and selection of optimal decoding strategies for specific D2T generation tasks, making it challenging to pinpoint the most effective decoding method for a particular task. Commonly used decoding methods in D2T generation, such as beam search and its variants [17–21], tend to favor high-probability words, resulting in limited diversity in the generated text [13,22]. While stochastic approaches, such as top-k [23] and top-p [13], have shown promise in addressing these limitations in open-ended text generation NLP tasks, their effectiveness in D2T generation remains unexplored. Furthermore, Minimum Bayes-risk (MBR) decoding, which has demonstrated better performance in machine translation [22,24,25], a task similar to the mathematical framework of D2T generation, its application has not been explored in D2T generation. Consequently, identifying the optimal decoding method remains a long-standing open research problem with significant implications for real-world applications where diverse and contextually relevant text is essential [13,20].

This study presents DecoStrat, a unified and modular framework that provides alternative decoding strategies to bridge the gap between language modeling and decoding processes. By integrating multiple decoding methods, DecoStrat enables the seamless exploration of optimal decoding strategies for specific tasks, as illustrated in Figure 1. We demonstrate its effectiveness on the MultiWOZ dataset, showcasing the effective utilization of language models in D2T generation. The DecoStrat's modular architecture enables the flexible development, integration, and optimization of various decoding methods, leading to advancements in D2T generation. The framework's main contributions include:

- **Modular Decoding Architecture**: DecoStrat features a modular design that integrates various decoding strategies through its interactive modules. This architecture allows for the flexible development and optimization of decoding methods tailored to specific tasks.
- **MBR Decoding** : The framework includes essential modules—Manager, Ranker, and Selector—specifically for MBR decoding. These components work together to determine candidate selection strategies and optimize the output sequence, enhancing the effectiveness of the decoding process.
- **Evaluation**: DecoStrat is designed to be domain-independent, making it applicable to a wide range of language generation tasks. Its effectiveness is showcased through evaluations on the MultiWOZ dataset, demonstrating its capability to produce contextually relevant text across various domains.

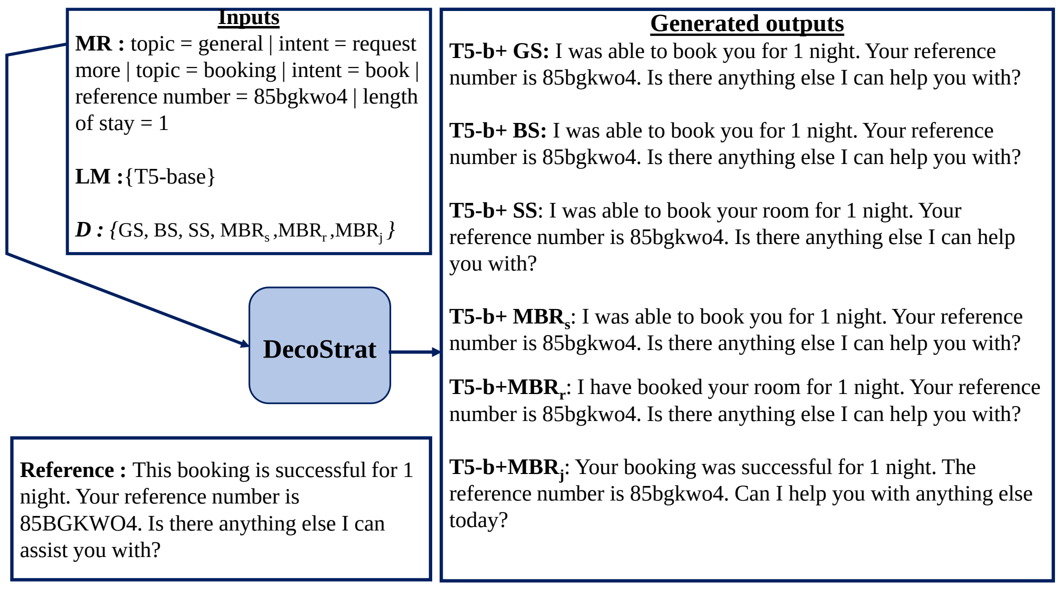
Inputs
MR : topic = general | intent = request more | topic = booking | intent = book | reference number = 85bgkwo4 | length of stay = 1
LM :{T5-base}
D : {GS, BS, SS, MBR_s, MBR_r, MBR_j }
DecoStrat
Reference : This booking is successful for 1 night. Your reference number is 85BGKWO4. Is there anything else I can assist you with?
Generated outputs
T5-b+ GS: I was able to book you for 1 night. Your reference number is 85bgkwo4. Is there anything else I can help you with?
T5-b+ BS: I was able to book you for 1 night. Your reference number is 85bgkwo4. Is there anything else I can help you with?
T5-b+ SS: I was able to book your room for 1 night. Your reference number is 85bgkwo4. Is there anything else I can help you with?
T5-b+ MBR_s: I was able to book you for 1 night. Your reference number is 85bgkwo4. Is there anything else I can help you with?
T5-b+MBR_r: I have booked your room for 1 night. Your reference number is 85bgkwo4. Is there anything else I can help you with?
T5-b+MBR_j: Your booking was successful for 1 night. The reference number is 85bgkwo4. Can I help you with anything else today?

Figure 1. The high-level operation of the DecoStrat framework that demonstrates DecoStrat utilizing a T5-base language model trained on the MultiWOZ dataset. When given an input data MR about a booking request, the trained model generates six alternative outputs, each corresponding to one of the six decoding methods $\mathcal{D}$. These outputs are evaluated against a reference text to determine the best output, indicating the optimal decoding method for the specific D2T generation task.

The remaining parts of this study are structured into the following sections. Section 2 provides a concise overview of recent related works. This is followed by an illustration and description of the research methodology in Section 3. The experimental setup and the details of experimental results are presented in Section 4. Section 5 presents the Results and Discussion. Finally, Section 6 summarizes the main findings and concludes the study.

2. Related Work

This section briefly presents the current state of neural language modeling and decoding methods in D2T generation, focusing on recent advancements in transformer models such as T5 and BART. Moreover, we examine deterministic, stochastic, and MBR decoding methods. Finally, we identify the gaps in existing approaches and show how our DecoStrat framework addresses these limitations, offering a unified and modular solution for D2T generation.

2.1. Neural Language Modeling

Neural language models have revolutionized NLP by learning the complex patterns in language and estimating the probability of the next word in a sequence [11]. By leveraging this approach, researchers have developed powerful language models and applied them to NLP tasks, including D2T generation. These models acquire knowledge through pretraining on large datasets, contributing to their powerful performance on various tasks.

The emergence of the transformer model [26] marked a significant milestone in neural language modeling, leading to notable improvements in the field. Variants of the transformer model, such as T5 [9] and BART [10], have achieved state-of-the-art results in various NLP tasks. However, despite these advancements, generating text from structured data remains challenging. One crucial aspect that has not kept pace with language modeling advancements is the decoding process, which plays a vital role in determining the output of language models [15,27,28]. This disparity highlights the need for further investigation into decoding methods and unlocking the immense potential of neural language models in D2T generation.

2.2. Deterministic Decoding Methods

In D2T generation, researchers often employ greedy search, beam search, and its variants. Greedy search constructs the output sequence by selecting the highest probability word at each position. Some notable works that utilized greedy search include [29], which investigated the effectiveness of the pre-train followed by fine-tune strategy, and [30] as an alternative method.

Beam search selects multiple tokens for a position based on conditional probability, considering a range of alternatives determined by the beam width. Considering the context of preceding words and their probabilities, beam search makes more informed decisions about the upcoming word in the sequence [31]. Several works have employed beam search, including [17], which introduced DataTuner D2T system for diverse domains, [32], who developed an end-to-end neural architecture that incorporates content selection and planning, and [19], who proposed an entity-centric neural architecture. Additionally, [30] applied beam search and introduced beam search variants that leverage cross-attention to extract meaningful information and figure out attributes mentioned in the generated text. Another variant of beam search is guided beam search, proposed by [21], which aimed to reduce omissions and hallucinations during the decoding process.

These works have contributed to the advancements of D2T generation. However, despite their contributions, deterministic methods can still result in suboptimal output, characterized by unnatural language and repetitive patterns, a problem considerably discussed in previous research [13,27,33].

2.3. Stochastic Decoding Methods

When generating sequences, the standard approach involves using the distributions predicted by a model without modifications. However, this can lead to issues with coherence and diversity in the generated text as low-probability tokens can dominate and decrease the quality of the text [13,28].To address this, researchers have explored alternative approaches that can modify the generation behavior of the model. One such approach is to use controlled sampling methods, which can help to improve the coherence and diversity of the generated text. The two popular methods that can achieve this are top-k and top-p (Nucleus) sampling methods.

Top-k sampling is a decoding method that involves selecting the top k most likely tokens from a predicted distribution, which is suggested to be effective in generating coherent and diverse text. [23] introduced top-k sampling in their work on hierarchical story generation, where a premise is first generated and then developed into a coherent text passage using model fusion and self-attention mechanisms. This approach maintains balanced coherence and diversity by choosing k tokens from the model's probability distribution while excluding the least likely ones. For example, consider a token probability distribution of 10 tokens, where k=5. In this case, the top 5 most probable tokens would be selected: A (0.25), B (0.20), C (0.15), D (0.10), and E (0.08), while the remaining tokens would be discarded. However, using a fixed value for K can be limiting, as it may only cover a small portion of the total probability in uniform distributions or include highly unlikely tokens in distributions with extreme peaks.

Another approach is to use top-p sampling, which involves selecting tokens from the dynamic nucleus of the probability distribution [13]. This approach demonstrated its effectiveness in improving the quality of machine-generated text by reducing the likelihood of bland and repetitive text. Specifically, [13] found that neural language models trained on likelihood objectives tend to generate bland and repetitive text due to distributional differences between human and machine text, and proposed the top-p method as a solution to this problem. The top-p sampling considers the top p% of the probability mass, providing a better balance between coherence and diversity. The model can generate more diverse and coherent text by selecting the top p% of the probability mass. For example, with the same token probability distribution and p=0.7, we select tokens until the cumulative probability reaches 0.7, which includes A (0.25), B (0.20), C (0.15), and D (0.10). This approach can be more effective than top-k sampling as it dynamically adapts to the shape of the probability distribution.

Combining these techniques can improve the coherence and diversity of generated text. Stochastic methods, which have proven effective in open-ended text generation, overcome the degeneration inherent in deterministic methods [20]. The DecoStrat framework leverages the combined approach to enhance diversity in the D2T domain, yielding promising results.

2.4. MBR Decoding

MBR decoding provides a systematic approach to introducing randomness into the decoding process to generate diverse outputs, each of which has a probability of being valid. This approach enables language models to explore different translation possibilities, providing a better understanding of the input data [25,34]. MBR decoding has demonstrated encouraging results in various tasks, including text generation and neural machine translation, with notable works including [35], who proposed MBR to address the trade-off between diversity and quality in text generation tasks; [24], who presented lattice MBR decoding approach over translation lattices to efficiently encode translation hypotheses and improve translation performance on different languages; [36], who proposed an approach to combining neural machine translation (NMT) with traditional statistical machine translation (SMT) by minimizing the Bayes-risk with respect to syntactic translation lattices; and [22], who argued that issues in neural machine translation systems are not inherent to the model or training algorithm but stem from using maximum a posteriori decoding, and proposed considering the full translation distribution over focusing solely on the highest-scoring translation. While MBR decoding can impose a significant computational burden, requiring careful decisions regarding the number of candidates generated and selecting a suitable utility function for hypothesis evaluation, its potential benefits make it a promising approach worth exploring further, particularly in novel applications such as the D2T generation domain, where we introduced three variants of MBR that seamlessly integrate with the modules of the DecoStrat framework.

The review of existing literature highlights several critical limitations in current decoding methods for data-to-text (D2T) generation. First, many individual approaches fail to fully leverage the capabilities of neural models, leading to suboptimal language generation. Deterministic methods often produce degenerate outputs, while stochastic techniques, although they enhance diversity, struggle to maintain coherence. Moreover, the effectiveness of Minimum Bayes Risk (MBR) methods in D2T generation remains unproven. Another significant issue is the lack of modularity in current decoding strategies, hindering the exploration and adaptation of decoding methods tailored to specific tasks. The proposed DecoStrat framework provides a unified and modular solution that integrates multiple decoding methods, including novel MBR variants. With its flexible design, the framework allows for seamless adaptation to diverse D2T generation tasks.

3. Research Methodology

This section identifies the essential elements required for D2T generation and introduces the innovative DecoStrat framework, designed to harness these components and unlock the potential of language models for D2T generation. Additionally, the visual representation of the DecoStrat framework illustrates its core components and their interactions, thereby offering a comprehensive understanding of the framework's architecture.

3.1. Neural Language Modeling for D2T Generation

D2T generation can be formulated as a sequence-to-sequence problem, where the goal is to learn a mapping from an input sequence X to a target sequence Y. Let $X = (x_1, x_2, \ldots, x_T)$ be the input sequence, and $Y = (y_1, y_2, \ldots, y_T)$ be the target sequence. The goal is to model the conditional probability $P(Y|X)$, which can be viewed as a next-word prediction problem, where the model predicts the next word in the target sequence given the context of the input sequence and the previous words in the target sequence.

A sequence-to-sequence model consists of an encoder and decoder components. Traditionally, researchers used Recurrent Neural Networks (RNNs) for encoder and decoder components. However, RNNs have significant limitations that make them less effective for modeling sequential data. Their sequential processing is slow and inefficient, and they suffer from the vanishing gradient problem, which hinders the training of deep RNNs. In contrast, transformer models such as T5 and BART have become the current state-of-the-art for sequence-to-sequence tasks. They utilize self-attention mechanisms to process input sequences in parallel and capture long-range dependencies [26,37], making them a superior choice for our purposes.

The encoder transforms the input sequence X into a continuous representation $\bar{X}$, and the decoder generates the target sequence Y based on $\bar{X}$ in an autoregressive manner, where each output token is generated based on the previous tokens in the target sequence. The encoder and decoder can be represented as functions f and g, respectively:

$$\bar{X} = f(X) = \text{Concat}(\bar{x}_1, \bar{x}_2, ..., \bar{x}_T) \tag{1}$$

where Concat denotes the concatenation operation, combining multiple vectors ($\bar{x}_1$, $\bar{x}_2$, ..., $\bar{x}_T$) into a single vector $\bar{X}$, to form a continuous representation of the input sequence X.

The probability of generating each token y_t in the target sequence is computed using the softmax function:

$$P(y_t|y_{<t}, \bar{X}) = \text{softmax}(g(y_{<t}, \bar{X})) \tag{2}$$

where $y_{<t}$ =$y_{1:t-1}$ =$\{y_1, y_2, \ldots, y_{t-1}\}$ is the sequence of previous target outputs . In the context of the decoder function g, the softmax function is applied to the output of g. Let's denote this vector

as $z = g(y_{<t}, \bar{X})$. The softmax function takes the output vector z from the decoder and returns a probability distribution over all possible tokens in the vocabulary. This is done by computing the exponential of each element in z, normalizing the resulting values, and returning a vector of probabilities that add up to 1.

Then, the softmax function can be defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{3}$$

where z_i is the i^{th} element of the vector z, N is the number of elements in the vector z, and e is a mathematical constant of approximately 2.718.

The overall probability of generating the target sequence Y given the input sequence X is:

$$P(Y|X) = \prod_{t=1}^{T} P(y_t|y_{<t}, \bar{X}) \tag{4}$$

The training objective is to maximize the likelihood of the target sequence Y given the input sequence X:

$$\mathcal{L}(\theta) = \max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p(Y_n|X_n; \theta) \tag{5}$$

where $\mathcal{L}$ is the likelihood of the target sequence given the input sequence, X_n and Y_n are the input and target sequences for the n-th example in the training set, N is the number of examples in the training set, and θ is the set of model parameters.

3.1.1. Decoding Methods

The decoder generates the target sequence Y based on the continuous representation $\bar{X}$ in an autoregressive manner, relying on its prediction history to construct the output sequence $\hat{y}_t$ during inference. The decoding method determines which token to choose at each time step, playing a crucial role in determining the quality of the generated output as it navigates the complex probability space defined by the language model [31]. We explore three distinct decoding scenarios in D2T generation: deterministic methods that produce a single output, stochastic methods that introduce randomness, and Minimum Bayes Risk (MBR) methods that select the best candidate output from the generated options.

Deterministic decoding methods: These methods involve selecting the most likely token at each time step, resulting in a single, deterministic output sequence. This is achieved by choosing the token with the highest probability from the vocabulary set V at each time step t, as in Equation (6).

$$\hat{y}_t = \arg\max_{y \in V} P(y_t|y_{1:t-1}, \bar{X}) \tag{6}$$

The Equation (6), can represent both Greedy Search and Beam Search, which differ in their decoding method d and their parameters d_θ. In Greedy Search, d iteratively selects the most likely token at each step, with d_θ including beam size $b = 1$ [31,38,39], whereas in Beam Search, d maintains a set of top-k hypotheses, with d_θ including the beam size $b \geq 2$ [31,40–43]. We can view both methods as different instantiations of the decoding method d and its parameters d_θ, used to determine the output sequence y_t .

Stochastic decoding methods: These methods involve randomly sampling from the probability distribution $P(y_t|y_{1:t-1}, \bar{X})$ at each time step t, resulting in a stochastic output sequence. This is achieved by introducing randomness in the output generation process, as shown in Equation (7).

$$\hat{y}_t \sim P(y_t|y_{1:t-1}, \bar{X}) \tag{7}$$

Equation (7), indicates that we can obtain the predicted output sequence y_t by sampling from the probability distribution $p(y_t|y_{1:t-1}, \bar{X})$, where the decoding method d and its parameters d_θ determine the specific sampling strategy by controlling how to carry out the sampling from the probability distribution. The two popular methods that fall under stochastic decoding are top-k samples from the top-k most probable words at each time step [23] and top-p that samples a subset of the most probable words with cumulative probability equal to or exceeding a threshold (p) at each step [13]. We can represent both as $\hat{y}_t \sim P(y_t|y_{1:t-1}, \bar{X})$.

MBR decoding methods: This method involves generating multiple candidate outputs and selecting the best one based on a utility function such as BLEURT [22,24,35]. This can be achieved by first generating a set of candidate outputs $\mathcal{Y}$ as shown in Equation (8).

$$\mathcal{Y} = \{y_1, y_2, ..., y_m\} \sim P(y_t|y_{1:t-1}, \bar{X}) \tag{8}$$

Then the predicted output sequence $\hat{y}_t$ is selected using the candidate selection strategies $\mathcal{S}$ as shown in Equation (9).

$$\hat{y}_t = \arg\max_{y \in \mathcal{Y}} \mathcal{S}(y, \mathcal{Y}) \tag{9}$$

We can evaluate the quality of the generated text $\hat{y}_t$ for all scenarios using conventional automatic metrics such as BLEU [44], ROUGE [45], METEOR [46], CIDEr [47] or any other metric that measures the similarity or relevance of the generated text $\hat{y}_t$ to the reference text y [48]. Moreover, we can apply human evaluation to examine the qualities of the generated text such as coherence, informativeness, relevance, and fluency [49,50].

3.2. Proposed Framework

The proposed DecoStrat framework integrates finetuned language models, alternative decoding methods, and candidate selection strategies $\mathcal{S}$ through the interacting modules. This section presents the DecoStrat system architecture and the details of each module.

3.2.1. DecoStrat System Architecture

The DecoStrat system architecture integrates finetuned language models followed by six interacting modules: Director, Generator, Manager, Ranker, Selector, and DecoDic. We designed this architecture to effectively process input data using trained language models and alternative decoding methods to produce the output sequences. The high-level operation of this system is as follows.

The model training process builds upon the work of [30]. We initialized the parameters of the pre-trained model checkpoints from the hugging face hub [51], which provides a standardized framework for conducting experiments with transformer-based models on NLP tasks. Moreover, the training and validation datasets were then loaded, and training parameters were employed. The experiment section provides the detailed implementation of the training procedure including the model specifications and training parameters. During the inference stage, the Director receives input data, a language model, and a decoding method and then instructs the Generator to produce outputs. The Manager subsequently processes the generated output(s), determining the decoding method category by applying a selection strategy $\mathcal{S}$ and returning the selected candidate sc. The interactions between these modules are crucial in determining the final output sequence. The DecoStrat system architecture visually depicts the main components and the interactions that comprise the framework as illustrated in Figure 2.

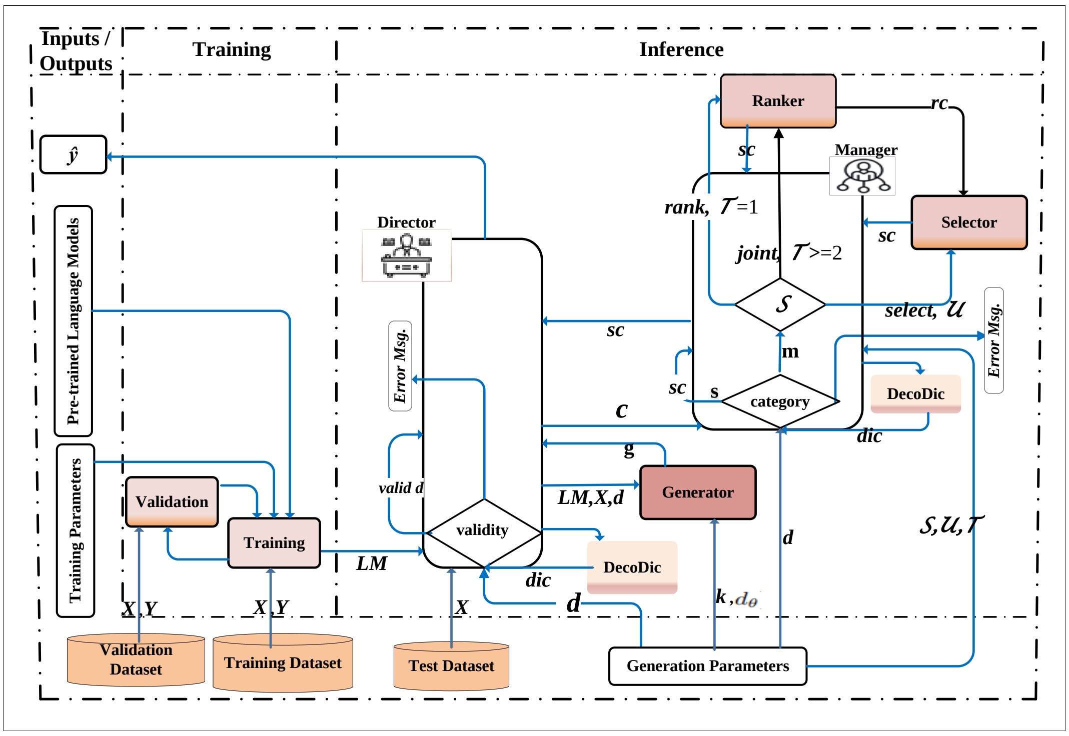

Figure 2. The DecoStrat system architecture illustrates the effective use of trained language model LM with the interactions of main modules in the inference stage to produce an output. The Director calls the Generator with input data X, the LM, and decoding method d and receives generated output(s) in return. After receiving the required inputs, the Manager determines the decoding method category to apply a selection strategy, sending the output directly to the Director for single-output method s or involving the Ranker and Selector modules for multi-output method m. The subsequent sections present a detailed explanation of each module's functionality and notations.

3.2.2. DecoDic

The DecoStrat framework comprises a crucial component, the DecoDic , a comprehensive dictionary of decoding methods that serves as a central hub for decoding methods and their corresponding parameters. The primary function of DecoDic is to validate and configure decoding methods employed by the Director and Manager modules, facilitating seamless integration and optimal performance in the D2T generation process.

DecoDic , as shown in Algorithm 1, is a dictionary of decoding methods and their corresponding parameters. The Director and Manager modules call the DecoDic function to retrieve a dictionary containing decoding methods and their parameters, categorized into two keys: s and m. The s key represents decoding methods, such as greedy search, that produce a single output for a given input sequence. In contrast, the m key corresponds to decoding methods, such as MBR decoding, that generate multiple intermediate outputs. We can represent each decoding method in DecoDic as a nested dictionary with method_name as the key and dictionaries of parameters (parameter_name:parameter_value) as the value. DecoDic plays a crucial role in the D2T generation task by providing a standardized way of representing decoding methods and their parameters. The Director module uses the DecoDic function to validate its chosen decoding method. Similarly, the Manager module applies this function to determine the category (s or m) of the selected decoding method, enabling informed decisions about how to proceed with the task. By providing this critical functionality, the DecoDic function ensures that the decoding methods used by the Director and Manager modules are valid and correctly configured for the D2T generation task.

Algorithm 1: Decoding methods dictionary.

```
Function DecoDic:
    begin
        return
            $s \ [ds_1, ds_2 \ldots, ds_n]$
            $m \ [dm_1, dm_2 \ldots, dm_n]$
            $ds_1, \ldots, ds_n \ \{\theta_{ds_i} : value_{ds_i}\}, i = 1, \ldots, n$
            $dm_1, \ldots, dm_n \ \{\theta_{dm_i} : value_{dm_i}\}, i = 1, \ldots, n$
    end
```

3.2.3. Director

The Director module is one of the main components of the DecoStrat framework that oversees the generation of output sequences from the input data using a fine-tuned language model and decoding method. It coordinates the interactions between the Generator and Manager modules to produce the final output.

Algorithm 2: Director module

```
Module Director($X$, $\mathcal{D}$, $LM$):
    candidates $\leftarrow$ []
    selected_candidate $\leftarrow$ []
    $\hat{y} \leftarrow$ []
    category $\leftarrow$ DecoDic()
    if $\mathcal{D} = d$ not in category then
        raise value error message.
    candidates $\leftarrow$ Generator($X, d, LM$) [3]
    selected_candidate $\leftarrow$ Manager(candidates) [4]
    $\hat{y} \leftarrow$ selected_candidate
    return $\hat{y}$
Input: $X$: Input sequence.
$\mathcal{D} = \{d_1, d_2, ..., d_n\}$: Decoding methods.
$LM$: D2T generation language model.
Output: $\hat{y}$: The predicted output sequence.
```

The Director module, shown in Algorithm 2, coordinates the generation of output sequences by validating the decoding method using the `DecoDic` function, generating output sequences using the `Generator` module, selecting the optimal candidate output sequence using the `Manager` module, and finally returning the predicted output sequence $\hat{y}$. The `Director` module acts as a high-level controller, orchestrating the interactions between the `Generator` and `Manager` modules to produce the predicted output $\hat{y}$. The Director module controls and efficiently generates output sequences by directing the D2T generation process.

3.2.4. Generator

The Generator module is a crucial lower-level component of the DecoStrat framework performing the actual D2T generation task. It takes input data, a language model, decoding methods, decoding parameters, and the number of candidate output(s) and produces a list containing the generated output sequence(s).

Algorithm 3: Generate output sequence(s). (2)

```
Module Generator( $X$, $\mathcal{D}$, $LM$):
    $X \leftarrow \{x_1, ..., x_T\}$, Encode the input data where $T$ is the length of the input sequence
    $g \leftarrow []$
    if $k = 1$ and $\mathcal{D}$=d is deterministic with parameters $d_\theta$ then
        for $t$ in 1...$T$ do
            $o_t \leftarrow LM(y_{1:t-1}, X)$ , Model Prediction
            $P(y_t|y_{1:t-1}, X) \leftarrow softmax(o_t)$
            $\hat{y}_t \leftarrow \arg\max_{y \in V} P(y_t|y_{1:t-1}, X)$ , Generate Text
            $g \leftarrow g$.append($\hat{y}_t$)
        end
    end
    else if $k = 1$ and $\mathcal{D}$=d is stochastic with parameters $d_\theta$ then
        for $t$ in 1...$T$ do
            $o_t \leftarrow LM(y_{1:t-1}, X)$
            $P(y_t|y_{1:t-1}, X) \leftarrow softmax(o_t)$
            $\hat{y}_t \sim P(y_t|y_{1:t-1}, X)$
            $g \leftarrow g$.append($\hat{y}_t$)
        end
    end
    else if $k \geq 2$ and $\mathcal{D}$=d is MBR with parameters $d_\theta$ then
        for $t$ in 1...$T$ do
            $o_t \leftarrow LM(y_{1:t-1}, X)$
            $P(y_t|y_{1:t-1}, X) \leftarrow softmax(o_t)$
            for $i$ in 1...$k$ do
                $y_{t_i} \sim p(y_t|y_{1:t-1}, X)$
                $g \leftarrow g$.append($y_{t_i}$)
            end
        end
    end
    return $g$
Input: $X$: Input data.
$LM$: Language model finetuned for D2T generation on the specific task.
$\mathcal{D} = \{d_1, d_2, ..., d_n\}$: Decoding methods.
$d_\theta$: Decoding parameters.
$k$: Number of candidate output(s).
Output: $g$: Generated output sequence(s).
```

The `Generator` module, shown in Algorithm 3 produces output sequence(s) based on the provided inputs and the corresponding processing algorithm. The algorithm accommodates three distinct categories of decoding methods: deterministic, stochastic, and MBR.

- **Deterministic decoding** : When $k = 1$ and $\mathcal{D}$ are deterministic, the algorithm generates a single output sequence by iteratively predicting the next token with the highest probability according to the language model LM.
- **Stochastic decoding**: When $k = 1$ and $\mathcal{D}$ is stochastic, the algorithm generates a single output sequence by sampling from the probability distribution predicted by LM.
- **MBR decoding**: When $k \geq 2$ and $\mathcal{D}$ is MBR, the algorithm produces k distinct output sequences by sampling from the probability distribution predicted by LM.

The output is a list containing the generated output sequence(s), a single sequence for deterministic and stochastic decoding, or k sequences for MBR decoding. Overall, the Generator module provides a flexible and efficient way to generate text from input data, supporting a range of decoding methods and parameters. Essentially, the Generator module offers a versatile and effective means of text generation from input data, accommodating an array of decoding methods and associated parameters. The singular output from the module represents the final predicted output, whereas multiple outputs are subject to subsequent processing by other components within the DecoStrat framework.

3.2.5. Manager

The `Manager` module is a decision-making algorithm that considers multiple possible scenarios to select a single candidate output from a set of candidates based on the specified decoding method and selection strategy.

Algorithm 4: : Manager module. (2)

```
Module Manager($\mathcal{Y}$):
    $category \leftarrow$ DecoDic();
    $sc \leftarrow []$;
    if $\mathcal{D} = d \in category[s]$ then
        $sc \leftarrow y$;
    end
    else if $\mathcal{D} = d \in category[m]$ then
        if $\mathcal{S} = rank$ then
            $rc \leftarrow []$;
            $rc \leftarrow$ (Ranker($\mathcal{Y}, \mathcal{T} = 1$));
            $sc \leftarrow rc$;
        end
        else if $\mathcal{S} = select$ then
            $sc \leftarrow$ Selector($\mathcal{Y}, \mathcal{U}$);
        end
        else if $\mathcal{S} = joint$ then
            $rc \leftarrow []$;
            $rc$.append (Ranker ($\mathcal{Y}, \mathcal{T} \geq 2$))
            $sc \leftarrow$ Selector($rc, \mathcal{U}$)
        end
        else
            raise Invalid selection strategy.
        end
    end
    else
        raise Invalid decoding method.
    end
    return $sc$
Input: $\mathcal{Y} = y_1, y_2, ..., y_m$ : Candidate outputs.
$\mathcal{D} = \{d_1, d_2, ..., d_n\}$ : Decoding methods.
$\mathcal{S} = \{rank, select, joint\}$: Selection strategies.
$\mathcal{U} = \{BLEURT\}$: Utility functions for candidate selection.
$\mathcal{T} = \{1, 2, ..., n\}$ : Top-n ranked candidate(s).
Output: $sc$: Selected candidate.
```

The `Manager` module, shown in Algorithm 4, checks the validity of the decoding method and selection strategy to determine the selection process. The process involves selecting a candidate using the `Selector` module, ranking candidates using `Ranker` module, or applying a joint ranking and selection approach. The `Manager` module integrates the `Ranker` and `Selector` modules to provide a flexible approach to choosing the most suitable candidate. The algorithm raises an error message if the selection strategy is not recognized. Finally, it returns the selected candidate output to the `Director`.

3.2.6. Ranker

The Ranker module is a ranking algorithm that adapts the PageRank [52] algorithm to rank candidates based on their similarity scores.

Algorithm 5: : Rank candidates.

```
Module Ranker(($\mathcal{Y}$, $\mathcal{T}$)):
    $M \leftarrow$ CSM($\mathcal{Y}$) ;
    $cs \leftarrow \frac{1}{|\mathcal{Y}|} \cdot \mathbf{1}$;
    for $i$ = 1 to $epoch$ do
        $prev\_cs \leftarrow$ copy($cs$);
        for $j = 1$ to $|\mathcal{Y}|$ do
            $cs[j] \leftarrow (1 - df) + df \cdot \sum_{k=1}^{|\mathcal{Y}|} \frac{M[j,k] \cdot cs[k]}{\sum_{l=1}^{|\mathcal{Y}|} M[j,l]}$;
        end
        if $\|cs - prev\_cs\| < threshold$ then
            break;
        end
    end
    $sorted\_indices \leftarrow$ argsort($-cs$);
    $sorted\_candidates \leftarrow \mathcal{Y}[sorted\_indices]$;
    $sc \leftarrow \{(cs[j]) \mid j \in$ sorted_indices[: $\mathcal{T}$]$\}$;
    return $sc$;
```

Input: $\mathcal{Y} = y_1, y_2, ..., y_m$: Candidates .
$\mathcal{T} = \{1, 2, ..., n\}$: Top n ranked candidate(s).
threshold: Convergence threshold.
df: Damping factor.
epoch: Number of Iteration.
Output: *sc*: Selected candidates by ranking.

3.2.7. Calculate Similarity Matrix

Algorithm 6: : Calculate Similarity Matrix.

```
Function CSM(($\mathcal{Y}$)):
    $n \leftarrow$ len($\mathcal{Y}$);
    $M \leftarrow zeros((n, n))$;
    for $i \leftarrow 0$ to $n - 1$ do
        for $j \leftarrow 0$ to $n - 1$ do
            $M[i, j] \leftarrow$ CSS($(y[i], y[j]) \in \mathcal{Y}$);
        end
    end
    return $M$
```

Input: $\mathcal{Y}$: Candidates.
Output: M: Similarity Matrix.

Algorithm 7: : Calculate Similarity Score.

```
Function CSS($y1, y2$):
    Tokenize ($y1$ and $y2$) using the $tokenizer$;
    Generate embeddings : $embedding1 \leftarrow E(y1)$ & $embedding2 \leftarrow E(y2)$ using the $model$ ;
    $score \leftarrow$ dotProduct ($embedding1, embedding2$);
    return $score$
Input: $y1$ & $y2$: Candidates.
$model$: Model such as BERT base.
$tokenizer$ : The corresponding tokenizer of the model.
Output: $score$: Similarity score
```

The Ranker module, outlined in Algorithm 5, takes in a set of candidate outputs, a threshold for convergence, a damping factor, the number of iterations, and a similarity matrix calculated by the Calculate Similarity Matrix (CSM) algorithm.

The Ranker algorithm initializes a score vector with uniform scores for each candidate to iteratively update the score vector based on the similarity matrix and the damping factor until convergence or it reaches a maximum number of iterations. The algorithm converges to a stable set of rankings based on the similarity matrix and the damping factor, then it returns ranked top-n denoted as $\mathcal{T}$ candidates. The Ranker operates in two modes, either directly selecting the best output based on its ranking when working individually or ranking the top $\mathcal{T}$ candidate outputs and passing the list to the Selector module when working jointly. We redesigned the algorithm to be well-suited for D2T generation tasks where the generated multiple outputs require ranking based on similarity.[Note df=0.85, threshold= 1e-5, epoch=100, and the BERT base model are used during implementation].

3.2.8. Selector

The Selector module selects an optimal output from a list of candidates based on pairwise similarity scores computed using a utility function.

Algorithm 8: : Select a candidate.

```
Module Selector($C, \mathcal{U}$):
    Initialize an $n \times n$ matrix $\mathbf{M}$ with zeroes ;
    for $i \leftarrow 1$ to $n$ do
        for $j \leftarrow 1$ to $n$ do
            score $\leftarrow \mathcal{U}(C[i], C[j])$ ;
            $\mathbf{M}[i][j] \leftarrow$ score ;
        end
    end
    for $i \leftarrow 1$ to $n$ do
        sum_scores $\leftarrow \sum_j \mathbf{M}[i][j]$ ;
    end
    $k \leftarrow \arg\max_i$ sum_scores ;
    $sc \leftarrow C[k]$ ;
    return $sc$
Input: $C$: Candidates.
$n$: Number of candidates.
$\mathcal{U}$: Utility functions.
Output: $sc$: The selected candidate.
```

The Selector module, as shown in Algorithm 8, initializes a matrix, calculates scores for each pair of candidates, and determines the sum of scores for each candidate. The algorithm then selects the output with the highest sum, indicating its relevance or most representative to the overall set.

The Selector can operate in two modes, either receiving candidates from the Generator and making the final selection of the best output based on utility scores when working individually, or receiving the ranked list of top n candidates from the Ranker and making the final selection based on utility scores when working jointly. The Ranker also receives candidates from the Generator, ranks them, and passes the top n candidates to the Selector. By doing so, the Ranker reduces the burden of the Selector, allowing it to focus on making the final selection from a reduced number of promising candidates. This collaborative approach enables the Ranker and Selector to work together efficiently, leveraging their strengths to produce the best possible output.

3.3. *Illustration on DecoStrat Module Interactions*

To demonstrate the flexibility and configurability of the DecoStrat framework, we present a scenario that showcases the seamless interaction of all its modules. This scenario highlights how the modules work together to produce the desired output. Using sample data from the MultiWOZ test dataset, we illustrate the operation of the DecoStrat framework in generating text from meaning representation (MR) input data. This example scenario requires the involvement of all modules, allowing us to observe their interactions and demonstrate the framework's capabilities.

Given the inputs, shown in Table 1, the DecoStrat framework modules start interacting provided they generate the output $\hat{y}$.The DecoStrat framework generates the output $\hat{y}$ as follows:

1. Director: The Director module takes in the linearized input data MR, decoding method $\mathcal{D}$, and language model LM. It checks if the decoding method $\mathcal{D}$ is in the category of supported decoding methods using the `DecoDic`() module. It initializes an empty list for the selected candidate and outputs $\hat{y}$.
2. Generator: The Director module calls the Generator, passing in the input data MR, decoding method $\mathcal{D}$, and language model LM. Moreover, the Generator module retrieves the number of candidates to be generated k and the decoding parameters d_θ, where d_θ consists of top-p and top-k. It generates k candidate outputs as shown in Table 2 for the given input data using the MBR decoding method with parameters top-p = 0.7 and top-k = 30.
3. Manager: The Director module calls the Manager, passing in the set of candidate outputs $\mathcal{Y}$. The Manager module also retrieves the decoding method $\mathcal{D}$, selection strategy $\mathcal{S}$, top N ranked candidates $\mathcal{T}$, and utility function $\mathcal{U}$. The Manager module initializes an empty list for the selected candidate sc and retrieves the category of decoding methods using the `DecoDic`() function. Since we set the selection strategy $\mathcal{S}$ to a "joint" value, the Manager module integrates the Ranker and Selector modules, provided that they can work together. The Ranker module takes the generated candidates, ranks them, and returns the top 3 candidates as shown in Table 3 to the Selector module. The Selector module selects the best candidate from the top 3 candidates as shown in Table 4 using the BLEURT utility function.
4. Output: The Director module returns the selected candidate as the output $\hat{y}$

Table 1. MultiWOZ sample data and Configurable Inputs to Illustrate the Interactions of DecoStrat Modules.

DecoStrat Inputs	Values
MR	topic = general | intent = request more | topic = booking | intent = book | reference number = 85bgkwo4 | length of stay = 1
LM	/checkpt/multiwoz/T5base/epoch_22_step_1749/
$\mathcal{D}$	MBR
top-p	0.7
top-k	30
k	10
$\mathcal{S}$	joint
$\mathcal{T}$	3
$\mathcal{U}$	BLEURT

Table 2. Sample generated candidate outputs using the Generator module.

Cand-ID	Candidate outputs
Cand-1	Your booking was successful. The reference number is 85bgkwo4. You'll be staying for 1 night. Is there anything else I can help you with?
Cand-2	Your booking for 1 night was successful. The reference number is 85bgkwo4. Do you need anything else?
Cand-3	I was able to book you for 1 night, your reference number is 85bgkwo4. Can I help you with anything else?
Cand-4	I have successfully booked your hotel for 1 night. Your reference number is 85bgkwo4. Can I help with anything else today?
Cand-5	I was able to book you for 1 night. Your reference number is 85bgkwo4. Can I help you with anything else?
Cand-6	Booking was successful for 1 night. Reference number is : 85bgkwo4. Is there anything else I can help you with?
Cand-7	Your booking was successful for 1 night. The reference number is 85bgkwo4. Can I help you with anything else today?
Cand-8	I have made that reservation for 1 night, your reference number is 85bgkwo4. Is there anything else I can help you with today?
Cand-9	Yes, I was able to book 1 night. The reference number is 85bgkwo4. Can I help you with anything else?
Cand-10	Booking was successful for 1 night. The table will be reserved for 15 minutes. Reference number is : 85bgkwo4. Anything else I can help with?

Table 3. Ranked candidate outputs using the Ranker module.

RC-ID	Ranked candidate outputs
RC-1	Your booking for 1 night was successful. The reference number is 85bgkwo4. Do you need anything else? [Cand-2]
RC-2	I was able to book you for 1 night. Your reference number is 85bgkwo4. Can I help you with anything else? [Cand-3]
RC-3	Your booking was successful for 1 night. The reference number is 85bgkwo4. Can I help you with anything else today? [Cand-4]

Table 4. Selected candidate output using the Selector module.

SC-ID	**Selected candidate output**
SC	Your booking was successful for 1 night. The reference number is 85bgkwo4. Can I help you with anything else today? [Cand-4]

In summary, the proposed framework enables effective utilization of language models in D2T generation. It provides a flexible architecture that accommodates various decoding strategies and language models. At its core, the framework offers three key features: flexibility, separation of concerns, and abstraction. These features enable the building and customization of NLG systems using our framework. DecoStrat's flexibility allows for the integration of finetuned language models, integration and customization of decoding methods, and candidate selection strategies, making it adaptable to diverse applications and evolving requirements. The separation of concerns within the framework allows researchers to focus on designing diverse decoding methods without concerning themselves with the intricate details of other components. DecoStrat abstracts complex implementation details using PyTorch framework, transformer-based language models, and utilities to make it simple and user-friendly. While DecoStrat has the potential to be a promising solution, its effectiveness relies on careful consideration of factors such as task-specific model training, decoding parameter tuning, and the choice of utility function. With DecoStrat, users can seamlessly integrate and optimize different decoding strategies with language models, achieving better results in D2T generation tasks. Next, we will experimentally evaluate the effectiveness of DecoStrat, considering task-specific model training, integration of proposed decoding methods, and parameter tuning.

4. Experimental Setup

This section presents the experimental data, baselines we compared with, and implementation details, including the system parameters for experimentation, model specifications, training parameters, and decoding strategies with associated parameters, thereby establishing a comprehensive setup for our investigation.

4.1. Experimental Data

The MultiWOZ 2.1 [1] dataset is a large-scale, multi-domain publicly available dialogue dataset consisting of 70,530 dialogues, divided into training (55,951), validation (7,286), and test (7,293) sets. It covers seven domains: attraction, hotel, taxi, restaurant, train, police, and general. In task-oriented dialog systems, the NLG module converts dialog acts represented in semantic form into natural language responses [53]. The dataset used in this work is an extracted version of MultiWOZ 2.1, provided by [30]. This extracted dataset follows the approach of [29,54]. Our goal is to train a model that takes in a Meaning Representation (MR) as input data and generates a system turn as output text, a process known as data-to-text (D2T) generation. Then, the trained model should accurately predict system responses across various domains and scenarios, conditioned on the input MR, and generate system responses that are contextually relevant and coherent.

Table 5 illustrates a sample of paired Meaning Representation (MR) and Reference (Ref) data from the MultiWOZ dataset, along with the preprocessed linearized MR. The table showcases the structured input data and its corresponding natural language output for a dialogue system. The MR contains semantic representations of user intents and preferences, such as selecting a restaurant, providing booking information, and specifying food choices. The linearized MR demonstrates a processed version of the MR in a linear format, separating different attributes using specific delimiters. For instance, when presented with an MR indicating a preference for three Chinese restaurants, the model is expected to respond appropriately by acknowledging the request, informing the available options, and suggesting a booking. The MR undergoes a preprocessing step described by [30], where the process separates individual slot-value pairs using "|" and delimited values from slot names using "=" symbols. The Reference (Ref) presents a human-readable response generated based on the given

MR, offering the user relevant information and prompting further interaction, such as booking a table at a suitable restaurant.

Table 5. MultiWOZ sample paired MR and Ref data along with linearized MR.

Input names	Sample data
MR	Restaurant-Select(), Booking-Inform(), Restaurant-Inform(Choice[three], Food[Chinese])
Linearized MR	topic = restaurant | intent = select | topic = booking | intent = inform | topic = restaurant | intent = inform | choice = three | food = chinese
Ref	I found three Chinese restaurants that meet your requests. Would you like for me to book a table at one of them?

4.2. Baselines

Our approach is compared with two recent state-of-the-art baseline studies in D2T generation tasks.

- The works by [30] examine the attention mechanism in pre-trained language models fine-tuned for D2T generation tasks. Their research revealed that encoder-decoder models with cross-attention can grasp semantic constraints that traditional decoding methods may not fully exploit. They introduced a methodology that extracts pertinent information from the cross-attention mechanism at each decoding step to identify accurately realized slots in the output. Then, they combine this approach with beam search and rescore beam hypotheses to select those with the fewest missing or incorrect slot mentions. Unlike our approach, the author's method relies on beam search.
- The second work by [29] explore T5 pre-training and fine-tuning for D2T tasks. DecoStrat framework employs the T5 and BART models with multiple decoding strategies, diverging from their singular use of T5 and greedy search. Despite this distinction, their observations offer valuable insights for our investigation, emphasizing the potential of pre-trained T5 models for D2T tasks.

4.3. System Parameters

The model training and DecoStrat evaluation is conducted on a Linux server equipped with eight NVIDIA GeForce RTX 3090 GPUs, each with 24 GB of memory, using the PyTorch framework and the Hugging Face Transformers [51]

4.4. Model Specifications and Training Parameters

The model training process builds upon the work of [30]. We initialized the parameters of the pre-trained T5 and BART model checkpoints sourced from the Huggingface hub [51], which provides a standardized framework for experimenting with models on various NLP tasks. Subsequently, we loaded the training and validation datasets and employed the AdamW optimizer [55] with a linear learning rate schedule and a warm-up period. During training, the loop iterated over epochs and batches, preparing and processing each batch, performing forward and backward passes, and updating the optimizer and learning rate scheduler. The objective was to maximize the log-likelihood of the target sequence given the input sequence [56], as formulated in Equation (5). The DecoStrat framework leverages pre-trained models and established training techniques to generate text from the input data. The detailed model specifications and training parameters used in our experiments are summarized in Table 6.

Table 6. Model specifications and the training parameters used for experiments.

Models	BART-base	T5-base	T5-small
Layers	6+6	12+12	6+6
Heads	12	12	8
Hidden State Size	768	768	512
Parameters	139M	220M	60M
Batch Size	32	64	64
Learning Rate	1×10^{-5}	3×10^{-5}	2×10^{-4}
Epochs	25	30	30

4.5. Best Model Checkpoints

We monitored the model's training performance on the validation set using the BLEU score and updated the best model checkpoints accordingly. The optimal model checkpoints achieved during the training process are presented in Table 7.

Table 7. Best model checkpoints achieved during the training process.

Model checkpoints	Validation BLEU	Epoch and step
BART-base	0.356	epoch 21, step 1749
T5-base	0.360	epoch 22, step 1749
T5-small	0.360	epoch 27, step 875

During inference, these model checkpoints are effectively utilized with the interaction of different modules to produce an output given the input data and alternative decoding methods as shown in Figure 2 and the detailed proposed methods.

4.6. DecoStrat Generation Parameters

Table 8 outlines decoding strategies and associated parameters used in our experiments. The study explores four decoding methods: Greedy Search (GS) with a beam size of 1, Beam Search (BS) with a beam size of 5, Stochastic Sampling (SS) with a top_p of 0.7 and top_k value of 30, and Minimum Bayes Risk (MBR) decoding with its three variants. In MBR decoding, we generate multiple candidates and choose the best one using a candidate selection strategy $\mathcal{S}$ operating in three alternative modes: select, rank, and joint. We refer to the MBR decoding method with the "select" strategy as MBR_s. In MBR_s, we employ the `Selector` module to choose the final candidate, utilizing BLEURT as the utility function denoted by $\mathcal{U}$. We denote the MBR decoding method with the "rank" selection strategy as MBR_r, which configures the `Ranker` module to use the "rank" strategy and top-n denoted as $\mathcal{T}$ value is set to 1. This approach ensures that we choose only the highest-ranked candidate from multiple options. We denote the MBR decoding method with the joint strategy as MBR_j, which configures the candidate selection strategy $\mathcal{S}$ to "joint" and $\mathcal{T}$ value to 3. This strategy involves a joint operation of the `Ranker` module and the `Selector` module to select the final candidate. We explored the effects of tuning decoding parameter values and set decoding parameter values that can help us attain optimal results. For more information on the candidate selection strategies, we refer the readers to Algorithm 4.

Table 8. DecoStrat decoding methods and the generation parameters used for experiments.

Parameters	GS	BS	SS	MBR_s	MBR_r	MBR_j
k	1	1	1	10	10	10
max_length	160	160	160	160	160	160
num_beams	1	5	NA	NA	NA	NA
top_p	NA	NA	0.7	0.7	0.7	0.7
top_k	NA	NA	30	30	30	30
$\mathcal{S}$	NA	NA	NA	select	rank	joint
$\mathcal{T}$	NA	NA	NA	NA	1	3
$\mathcal{U}$	NA	NA	NA	BLEURT	BLEURT	BLEURT

4.7. DecoStrat Performance Evaluation Metrics

The evaluation of the DecoStrat framework involved a comparison with the currently state-of-the-art baseline works using established automatic metrics: BLEU (B) [44], ROUGE (R) [45], METEOR (M) [46], CIDEr (C) [47] developed by the E2E NLG Challenge [57], and Slot Error Rate (SER) on the MultiWOZ test data. SER measures the proportion of instances where slot value from the original structured data (MR) is not mentioned or incorrectly mentioned in the predicted response. We calculate SER by checking if all slot values are present in the generated text, regardless of their order or phrasing [30].

5. Results and Discussion

This section presents the findings from our in-depth evaluation of the DecoStrat framework, providing a breakdown of the experimental outcomes and exploring their implications.

5.1. Results of DecoStrat with T5-Small Model

Our DecoStrat framework outperforms the baselines in METEOR with the highest score of 0.325 achieved by T5-s+MBR$_s$, indicating that it generates responses with relatively better coherence and semantic similarity to the reference texts. Moreover, DecoStrat with T5-small model and alternative decoding methods achieve competitive results, with the highest BLEU score of 0.356 achieved by T5-s+MBR$_j$, as shown in Table 9. Our framework achieves a score of 1.63 SER with T5-s+MBR$_s$, which is higher than the baseline SA's score of 0.63, indicating that our framework struggles to preserve slots compared to the baselines. Compared with the baselines, which have similar BLEU scores with the highest score of 0.360 achieved by SA and SG, our framework shows promising results, especially with the MBR variants. The MBR variants introduced controlled randomness using *top_k* and *top_p* to enable a more diverse and better exploration of the search spaces. However, the impact on the BLEU score was relatively marginal compared to the state-of-the-art baseline.

Table 9. DecoStrat with T5-small model and alternative decoding methods evaluated on MultiWOZ dataset and compared against K&R [29], beam search with slot aligner reranking (SA) and Semantically attention guided decoding (SG) [30] baselines.

Model	B	M	R	C	SER
K&R	0.346	NA	NA	NA	1.27
SA	**0.360**	0.323	NA	NA	**0.63**
SG	**0.360**	0.323	NA	NA	0.85
T5-s+GS	0.349	0.321	0.518	2.84	1.99
T5-s+BS	0.353	0.325	0.529	2.89	1.94
T5-s+SS	0.350	0.322	0.518	2.85	1.95
T5-s+MBR$_s$	0.353	**0.325**	0.526	2.84	1.63
T5-s+MBR$_r$	0.349	0.323	0.523	2.82	1.99
T5-s+MBR$_j$	0.356	**0.324**	0.523	2.83	1.91

The results of DecoStrat with the T5-small model and alternative decoding methods on the MultiWOZ dataset, shown in Table 9, demonstrate its effectiveness in dialogue response generation. Notably, DecoStrat surpasses the baseline K&R in the BLEU score, except in SER, across all decoding methods. DecoStrat framework outperforms the baselines in METEOR with the highest score of 0.325 achieved by T5-s+MBR$_s$, indicating that it generates responses with relatively better coherence and semantic similarity to the reference texts. Moreover, DecoStrat with T5-small model and alternative decoding methods achieve competitive results, with the highest BLEU score of 0.356 achieved by T5-s+MBR$_j$. Furthermore, DecoStrat is competitive with the SA and SG baselines, achieving relatively similar performance in BLEU scores. The MBR decoding methods, in particular, perform well in BLEU score and SER, with MBR_j achieving the highest BLEU score among all DecoStrat variants. The MBR variants introduced controlled randomness using top-k and top-p to enable a more diverse and better exploration of the search spaces. However, the impact on the BLEU score was relatively marginal compared to the state-of-the-art baseline. These results suggest that DecoStrat with alternative decoding methods can be a valuable approach for enhancing the performance of dialogue response generation models. Overall, the findings highlight the potential of DecoStrat as a viable strategy for improving the quality of generated responses.

5.2. Results of DecoStrat with T5-Base Model

DecoStrat framework with the T5-base model and alternative decoding methods achieves improved results, with the highest BLEU score of 0.354 achieved by T5-b+BS, as shown in Table 10. Moreover, our framework outperforms the baseline in terms of METEOR, with the highest score of 0.327 achieved by T5-b+BS. The DecoStrat framework also achieves lower SER scores of 1.28 with T5-b+MBRs, which is competitive to the baseline K&R's score of 1.27, indicating relatively equivalent performance in slots mentioned in their output text. The framework shows promising results, especially with the BS and MBR variants, which have implications for generating more accurate and informative output in data-to-text generation tasks.

Table 10. DecoStrat with T5-base model and alternative decoding methods evaluated on MultiWOZ dataset and compared against K&R [29] baseline.

Model	B	M	R	C	SER
K&R	0.351	NA	NA	NA	**1.27**
T5-b+GS	0.351	0.322	0.519	2.85	1.98
T5-b+BS	**0.354**	**0.327**	0.531	2.92	1.74
T5-b+SS	0.345	0.321	0.516	2.84	1.47
T5-b+MBR$_s$	0.351	0.324	0.519	2.85	1.28
T5-b+MBR$_r$	0.345	0.323	0.518	2.83	1.6
T5-b+MBR$_j$	0.348	0.324	0.522	2.86	1.47

5.3. Results of DecoStrat with BART-Base Model

As shown in Table 11, the DecoStrat framework with the BART-base model and alternative decoding methods achieves competitive results, with the highest BLEU score of 0.359 achieved by Bart+BS and Bart+SS. Additionally, the framework outperforms the baselines in METEOR, with the highest score of 0.329 achieved by Bart+SS. DecoStrat framework SER score of 1.34 with Bart+BS is higher than the baseline SA's score of 0.60, indicating that the baseline is better at preserving slots. Overall results demonstrate the potential of the DecoStrat framework with the BART-base model and alternative decoding methods for generating accurate output from a given input data.

Table 11. DecoStrat with BART-base model and alternative decoding methods evaluated on MultiWOZ dataset and compared against SA and SG [30] baselines.

Model	B	M	R	C	SER
SA	**0.364**	0.324	NA	NA	**0.60**
SG	0.363	0.323	NA	NA	0.72
Bart+GS	0.354	0.324	0.526	2.92	1.61
Bart+BS	0.359	0.328	0.536	2.96	1.34
Bart+SS	0.359	**0.329**	0.536	2.96	1.41
Bart+MBR$_s$	0.359	0.328	0.536	2.96	1.41
Bart+MBR$_r$	0.359	0.328	0.536	2.96	1.41
Bart+MBR$_j$	0.358	0.328	0.536	2.96	1.41

5.4. Implications of the Results

The DecoStrat framework performs competitively with baselines. The alternative decoding methods in the framework can generate text of comparable quality to the baselines. Among these methods, the MBR variants consistently outperform the others in METEOR and SER metrics. This performance indicates that the MBR variants excel the other alternative methods in generating diverse and contextually relevant text while maintaining high accuracy in slot preservation, meeting the pivotal objective of D2T generation tasks. Although the BS decoding method demonstrates competitive results in BLEU and METEOR, it exhibits higher SER scores than the MBR variants, implying difficulties in preserving slots effectively. The results also indicate that the T5-small model with MBR variants achieved relatively similar performance to the T5-base model, suggesting that the T5-small model may present a cost-effective option for D2T generation tasks. This finding aligns with previous studies, such as [29], which have also reported that model size does not always correlate with performance. Moreover, the BART-base model with MBR variants demonstrates competitive results compared to the T5 models, although with slightly lower BLEU scores. These results suggest that the BART-base model can produce text of comparable quality with the T5 models but may struggle to generate accurate text. Despite this, the difference in BLEU scores is relatively small, indicating that the BART-base model remains a viable choice for D2T generation tasks.

6. Conclusions

In this paper, we introduced DecoStrat, a unified and modular framework that bridges the gap between language modeling and decoding processes in D2T generation. By integrating multiple decoding methods, DecoStrat enables seamless exploration of optimal decoding strategies for specific tasks. Our experimental results on the MultiWOZ dataset demonstrate the effectiveness of DecoStrat in generating diverse and contextually relevant text from structured data. Notably, the framework with MBR variants, which have shown encouraging performance in machine translation, consistently outperforms other alternative decoding methods in METEOR and SER metrics, indicating their ability to generate diverse and contextually relevant text while maintaining high accuracy in slot preservation. Moreover, the BS decoding option with the language models also demonstrates competitive results but exhibits higher SER scores than the MBR variants, implying difficulties in preserving slots effectively. Overall, our findings suggest that DecoStrat has the potential to make a significant impact on D2T generation tasks. The framework's ability to explore different decoding strategies allows it to generate diverse and contextually relevant text from structured data, addressing a key challenge in D2T generation. Future work can build upon this framework to improve and explore its potential in real-world scenarios. Additionally, investigating alternative decoding methods, language models, learning-based evaluation metrics, and human evaluation could provide a more comprehensive understanding of the framework's performance.

Author Contributions: (1) Elias-Designed and built the DecoStrat framework, Conduct Experiment , analysis, Visualization, Writing. (2) Chen- Carried out Supervision and Resource allocation. (3)Al-Antari-Involve in Analysis, Review and Editing, Visualization, Funding acquisition. (4)Hyeon Gu-Involve in Analysis, Review and Editing, Funding acquisition. (5)Victor-Support result analysis and Editing. (6)Wasif-Support result analysis and Editing

Funding: This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2021-II210755, Dark data analysis technology for data scale and accuracy improvement). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2022-00166402 and RS-2023-00256517

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The datasets used in this study are publicly available and freely accessible from the link https://github.com/jjuraska/data2text-nlg/tree/master/data/multiwoz

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study.

References

1. Eric, M.; Goel, R.; Paul, S.; Sethi, A.; Agarwal, S.; Gao, S.; Kumar, A.; Goyal, A.; Ku, P.; Hakkani-Tur, D. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. Proceedings of the Twelfth Language Resources and Evaluation Conference; Calzolari, N.; Béchet, F.; Blache, P.; Choukri, K.; Cieri, C.; Declerck, T.; Goggi, S.; Isahara, H.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; Piperidis, S., Eds.; European Language Resources Association: Marseille, France, 2020; pp. 422–428.
2. Portet, F.; Reiter, E.; Gatt, A.; Hunter, J.; Sripada, S.; Freer, Y.; Sykes, C. Automatic generation of textual summaries from neonatal intensive care data. *Artif. Intell.* **2009**, *173*, 789–816. doi:10.1016/j.artint.2008.12.002.
3. Pauws, S.; Gatt, A.; Krahmer, E.; Reiter, E., Making Effective Use of Healthcare Data Using Data-to-Text Technology. In *Data Science for Healthcare: Methodologies and Applications*; Consoli, S.; Reforgiato Recupero, D.; Petković, M., Eds.; Springer International Publishing: Cham, 2019; pp. 119–145. doi:10.1007/978-3-030-05249-2_4.
4. Belz, A. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.* **2008**, *14*, 431–455. doi:10.1017/S1351324907004664.
5. Barzilay, R.; Lapata, M. Collective Content Selection for Concept-to-Text Generation. Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing; Mooney, R.; Brew, C.; Chien, L.F.; Kirchhoff, K., Eds.; Association for Computational Linguistics: Vancouver, British Columbia, Canada, 2005; pp. 331–338.
6. Juraska, J.; Bowden, K.; Walker, M. ViGGO: A Video Game Corpus for Data-To-Text Generation in Open-Domain Conversation. Proceedings of the 12th International Conference on Natural Language Generation; van Deemter, K.; Lin, C.; Takamura, H., Eds.; Association for Computational Linguistics: Tokyo, Japan, 2019; pp. 164–172. doi:10.18653/v1/W19-8623.
7. Gardent, C.; Shimorina, A.; Narayan, S.; Perez-Beltrachini, L. Creating Training Corpora for NLG Micro-Planners. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Barzilay, R.; Kan, M.Y., Eds.; Association for Computational Linguistics: Vancouver, Canada, 2017; pp. 179–188. doi:10.18653/v1/P17-1017.
8. Parikh, A.; Wang, X.; Gehrmann, S.; Faruqui, M.; Dhingra, B.; Yang, D.; Das, D. ToTTo: A Controlled Table-To-Text Generation Dataset. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP); Webber, B.; Cohn, T.; He, Y.; Liu, Y., Eds.; Association for Computational Linguistics: Online, 2020; pp. 1173–1186. doi:10.18653/v1/2020.emnlp-main.89.
9. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*.
10. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics;

Jurafsky, D.; Chai, J.; Schluter, N.; Tetreault, J., Eds.; Association for Computational Linguistics: Online, 2020; pp. 7871–7880. doi:10.18653/v1/2020.acl-main.703.

11. Yu, F.; Xiu, X.; Li, Y. A Survey on Deep Transfer Learning and Beyond. *Mathematics* **2022**, *10*. doi:10.3390/math10193619.
12. Lee, M. A Mathematical Interpretation of Autoregressive Generative Pre-Trained Transformer and Self-Supervised Learning. *Mathematics* **2023**, *11*. doi:10.3390/math11112451.
13. Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; Choi, Y. The Curious Case of Neural Text Degeneration. 8th International Conference on Learning Representations, ICLR 2020, 2020.
14. Wiher, G.; Meister, C.; Cotterell, R. On Decoding Strategies for Neural Text Generators. *Transactions of the Association for Computational Linguistics* **2022**, *10*, 997–1012. doi:10.1162/tacl_a_00502.
15. Zarrieß, S.; Voigt, H.; Schüz, S. Decoding Methods in Neural Language Generation: A Survey. *Information* **2021**, *12*. doi:10.3390/info12090355.
16. Welleck, S.; Kulikov, I.; Kim, J.; Pang, R.Y.; Cho, K. Consistency of a Recurrent Language Model With Respect to Incomplete Decoding. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP); Webber, B.; Cohn, T.; He, Y.; Liu, Y., Eds.; Association for Computational Linguistics: Online, 2020; pp. 5553–5568. doi:10.18653/v1/2020.emnlp-main.448.
17. Harkous, H.; Groves, I.; Saffari, A. Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity. Proceedings of the 28th International Conference on Computational Linguistics; Scott, D.; Bel, N.; Zong, C., Eds.; International Committee on Computational Linguistics: Barcelona, Spain (Online), 2020; pp. 2410–2424. doi:10.18653/v1/2020.coling-main.218.
18. Puduppully, R.; Dong, L.; Lapata, M. Data-to-text generation with content selection and planning. Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI Press, 2019, AAAI'19/IAAI'19/EAAI'19. doi:10.1609/aaai.v33i01.33016908.
19. Puduppully, R.; Dong, L.; Lapata, M. Data-to-text Generation with Entity Modeling. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics; Korhonen, A.; Traum, D.; Màrquez, L., Eds.; Association for Computational Linguistics: Florence, Italy, 2019; pp. 2023–2035. doi:10.18653/v1/P19-1195.
20. Welleck, S.; Kulikov, I.; Roller, S.; Dinan, E.; Cho, K.; Weston, J. Neural Text Generation with Unlikelihood Training, 2019, [arXiv:cs.LG/1908.04319].
21. Garneau, N.; Lamontagne, L. Guided Beam Search to Improve Generalization in Low-Resource Data-to-Text Generation. Proceedings of the 16th International Natural Language Generation Conference; Keet, C.M.; Lee, H.Y.; Zarrieß, S., Eds.; Association for Computational Linguistics: Prague, Czechia, 2023; pp. 1–14. doi:10.18653/v1/2023.inlg-main.1.
22. Eikema, B.; Aziz, W. Is MAP Decoding All You Need? The Inadequacy of the Mode in Neural Machine Translation. Proceedings of the 28th International Conference on Computational Linguistics; Scott, D.; Bel, N.; Zong, C., Eds.; International Committee on Computational Linguistics: Barcelona, Spain (Online), 2020; pp. 4506–4520. doi:10.18653/v1/2020.coling-main.398.
23. Fan, A.; Lewis, M.; Dauphin, Y. Hierarchical Neural Story Generation. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Gurevych, I.; Miyao, Y., Eds.; Association for Computational Linguistics: Melbourne, Australia, 2018; pp. 889–898. doi:10.18653/v1/P18-1082.
24. Tromble, R.; Kumar, S.; Och, F.; Macherey, W. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing; Lapata, M.; Ng, H.T., Eds.; Association for Computational Linguistics: Honolulu, Hawaii, 2008; pp. 620–629.
25. Müller, M.; Sennrich, R. Understanding the Properties of Minimum Bayes Risk Decoding in Neural Machine Translation. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers); Zong, C.; Xia, F.; Li, W.; Navigli, R., Eds.; Association for Computational Linguistics: Online, 2021; pp. 259–272. doi:10.18653/v1/2021.acl-long.22.
26. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. Advances in Neural Information Processing Systems; Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., Eds. Curran Associates, Inc., 2017, Vol. 30.

27. Gong, N.; Yao, N. A generalized decoding method for neural text generation. *Computer Speech & Language* **2023**, *81*, 101503. doi:https://doi.org/10.1016/j.csl.2023.101503.
28. Ippolito, D.; Kriz, R.; Sedoc, J.; Kustikova, M.; Callison-Burch, C. Comparison of Diverse Decoding Methods from Conditional Language Models. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics; Korhonen, A.; Traum, D.; Màrquez, L., Eds.; Association for Computational Linguistics: Florence, Italy, 2019; pp. 3752–3762. doi:10.18653/v1/P19-1365.
29. Kale, M.; Rastogi, A. Text-to-Text Pre-Training for Data-to-Text Tasks. Proceedings of the 13th International Conference on Natural Language Generation; Davis, B.; Graham, Y.; Kelleher, J.; Sripada, Y., Eds.; Association for Computational Linguistics: Dublin, Ireland, 2020; pp. 97–102. doi:10.18653/v1/2020.inlg-1.14.
30. Juraska, J.; Walker, M. Attention Is Indeed All You Need: Semantically Attention-Guided Decoding for Data-to-Text NLG. Proceedings of the 14th International Conference on Natural Language Generation; Belz, A.; Fan, A.; Reiter, E.; Sripada, Y., Eds.; Association for Computational Linguistics: Aberdeen, Scotland, UK, 2021; pp. 416–431. doi:10.18653/v1/2021.inlg-1.45.
31. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach (4th Edition)*; Pearson, 2020.
32. Puduppully, R.; Dong, L.; Lapata, M. Data-to-Text Generation with Content Selection and Planning. *Proceedings of the AAAI Conference on Artificial Intelligence* **2019**, *33*, 6908–6915. doi:10.1609/aaai.v33i01.33016908.
33. Schulz, P.; Aziz, W.; Cohn, T. A Stochastic Decoder for Neural Machine Translation. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Gurevych, I.; Miyao, Y., Eds.; Association for Computational Linguistics: Melbourne, Australia, 2018; pp. 1243–1252. doi:10.18653/v1/P18-1115.
34. Kumar, S.; Byrne, W. Minimum Bayes-Risk Decoding for Statistical Machine Translation. Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004; Association for Computational Linguistics: Boston, Massachusetts, USA, 2004; pp. 169–176.
35. Suzgun, M.; Melas-Kyriazi, L.; Jurafsky, D. Follow the Wisdom of the Crowd: Effective Text Generation via Minimum Bayes Risk Decoding. Findings of the Association for Computational Linguistics: ACL 2023; Rogers, A.; Boyd-Graber, J.; Okazaki, N., Eds.; Association for Computational Linguistics: Toronto, Canada, 2023; pp. 4265–4293. doi:10.18653/v1/2023.findings-acl.262.
36. Stahlberg, F.; de Gispert, A.; Hasler, E.; Byrne, B. Neural Machine Translation by Minimising the Bayes-risk with Respect to Syntactic Translation Lattices. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers; Lapata, M.; Blunsom, P.; Koller, A., Eds.; Association for Computational Linguistics: Valencia, Spain, 2017; pp. 362–368.
37. Zeng, H.; Liu, J.; Wang, M.; Wei, B. A sequence to sequence model for dialogue generation with gated mixture of topics. *Neurocomputing* **2021**, *437*, 282–288. doi:https://doi.org/10.1016/j.neucom.2021.01.014.
38. Germann, U.; Jahr, M.; Knight, K.; Marcu, D.; Yamada, K. Fast Decoding and Optimal Decoding for Machine Translation. Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics; Association for Computational Linguistics: Toulouse, France, 2001; pp. 228–235. doi:10.3115/1073012.1073042.
39. Germann, U. Greedy decoding for statistical machine translation in almost linear time. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1; Association for Computational Linguistics: USA, 2003; NAACL ’03, p. 1–8. doi:10.3115/1073445.1073455.
40. Moore, B.; Quirk, C. Faster Beam-Search Decoding for Phrasal Statistical Machine Translation. Proceedings of MT Summit XI, Proceedings of MT Summit XI ed. European Association for Machine Translation, 2007.
41. Cohen, E.; Beck, C. Empirical Analysis of Beam Search Performance Degradation in Neural Sequence Models. Proceedings of the 36th International Conference on Machine Learning; Chaudhuri, K.; Salakhutdinov, R., Eds. PMLR, 2019, Vol. 97, *Proceedings of Machine Learning Research*, pp. 1290–1299.
42. Yang, Y.; Huang, L.; Ma, M. Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; Riloff, E.; Chiang, D.; Hockenmaier, J.; Tsujii, J., Eds.; Association for Computational Linguistics: Brussels, Belgium, 2018; pp. 3054–3059. doi:10.18653/v1/D18-1342.
43. Meister, C.; Vieira, T.; Cotterell, R. Best-First Beam Search. *Transactions of the Association for Computational Linguistics* **2020**, *8*, 795–809. doi:10.1162/tacl_a_00346.

44. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. Bleu: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics; Isabelle, P.; Charniak, E.; Lin, D., Eds.; Association for Computational Linguistics: Philadelphia, Pennsylvania, USA, 2002; pp. 311–318. doi:10.3115/1073083.1073135.
45. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. Text Summarization Branches Out; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 74–81.
46. Banerjee, S.; Lavie, A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization; Goldstein, J.; Lavie, A.; Lin, C.Y.; Voss, C., Eds.; Association for Computational Linguistics: Ann Arbor, Michigan, 2005; pp. 65–72.
47. Vedantam, R.; Zitnick, C.L.; Parikh, D. CIDEr: Consensus-based image description evaluation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4566–4575. doi:10.1109/CVPR.2015.7299087.
48. Li, J.; Lan, Y.; Guo, J.; Cheng, X. On the relation between quality-diversity evaluation and distribution-fitting goal in text generation. Proceedings of the 37th International Conference on Machine Learning. JMLR.org, 2020, ICML'20.
49. van der Lee, C.; Gatt, A.; van Miltenburg, E.; Krahmer, E. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language* **2021**, *67*, 101151. doi:https://doi.org/10.1016/j.csl.2020.101151.
50. van der Lee, C.; Gatt, A.; van Miltenburg, E.; Wubben, S.; Krahmer, E. Best practices for the human evaluation of automatically generated text. Proceedings of the 12th International Conference on Natural Language Generation; van Deemter, K.; Lin, C.; Takamura, H., Eds.; Association for Computational Linguistics: Tokyo, Japan, 2019; pp. 355–368. doi:10.18653/v1/W19-8643.
51. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; Rush, A. Transformers: State-of-the-Art Natural Language Processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations; Liu, Q.; Schlangen, D., Eds.; Association for Computational Linguistics: Online, 2020; pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.
52. Page, L.; Brin, S.; Motwani, R.; Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, 1999. Previous number = SIDL-WP-1999-0120.
53. Wang, W.; Zhang, Z.; Guo, J.; Dai, Y.; Chen, B.; Luo, W. Task-Oriented Dialogue System as Natural Language Generation. Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval; Association for Computing Machinery: New York, NY, USA, 2022; SIGIR '22, p. 2698–2703. doi:10.1145/3477495.3531920.
54. Peng, B.; Zhu, C.; Li, C.; Li, X.; Li, J.; Zeng, M.; Gao, J. Few-shot Natural Language Generation for Task-Oriented Dialog. Findings of the Association for Computational Linguistics: EMNLP 2020; Cohn, T.; He, Y.; Liu, Y., Eds.; Association for Computational Linguistics: Online, 2020; pp. 172–182. doi:10.18653/v1/2020.findings-emnlp.17.
55. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; Bengio, Y.; LeCun, Y., Eds., 2015.
56. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); Moschitti, A.; Pang, B.; Daelemans, W., Eds.; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734. doi:10.3115/v1/D14-1179.
57. Dušek, O.; Novikova, J.; Rieser, V. Findings of the E2E NLG Challenge. Proceedings of the 11th International Conference on Natural Language Generation; Krahmer, E.; Gatt, A.; Goudbeek, M., Eds.; Association for Computational Linguistics: Tilburg University, The Netherlands, 2018; pp. 322–328. doi:10.18653/v1/W18-6539.