A Decoding Algorithm for Length-Control Summarization Based on Directed Acyclic Transformers

Chenyang Huang[*1], **Hao Zhou**[†2], **Cameron Jen**[1], **Kangjie Zheng**[2], **Osmar R. Zaïane**[‡1], **Lili Mou**[‡1]

[1]Dept. of Computing Science, Alberta Machine Intelligence Institute (Amii), University of Alberta
[2]Institute for AI Industry Research (AIR), Tsinghua University

chenyangh@ualberta.ca zhouhao@air.tsinghua.edu.cn cjen@ualberta.ca
kangjie.zheng@gmail.com zaiane@ualberta.ca doublepower.mou@gmail.com

Abstract

Length-control summarization aims to condense long texts into a short one within a certain length limit. Previous approaches often use autoregressive (AR) models and treat the length requirement as a soft constraint, which may not always be satisfied. In this study, we propose a novel length-control decoding algorithm based on the Directed Acyclic Transformer (DAT). Our approach allows for multiple plausible sequence fragments and predicts a *path* to connect them. In addition, we propose a Sequence Maximum a Posteriori (SeqMAP) decoding algorithm that marginalizes different possible paths and finds the most probable summary satisfying the length budget. Our algorithm is based on beam search, which further facilitates a reranker for performance improvement. Experimental results on the Gigaword and DUC2004 datasets demonstrate our state-of-the-art performance for length-control summarization.[1]

1 Introduction

Summarization systems aim to condense lengthy text into a shorter form, while preserving key information (Nenkova et al., 2011; Rush et al., 2015; Schumann et al., 2020; Tsvigun et al., 2022). Recent studies underscore the importance of length-control summarization (Liu et al., 2018; Takase and Okazaki, 2019; Liu et al., 2022a). For example, the Google search engine limits webpage title displays to 63 characters, tweets have a 280-character cap, and scientific paper abstracts are typically restricted to a few hundred words.

Most previous text summarization methods rely on autoregressive (AR) models (Sutskever et al., 2014; Vaswani et al., 2017), which generate a summary word by word. For length control, AR models typically treat the summary length as an additional signal during both training and inference (Liu et al., 2018; Takase and Okazaki, 2019). However, such methods do not strictly control the generation length, and thus fall into the *soft-constraint* category.

Recently, non-autoregressive (NAR) models (Gu et al., 2018; Lee et al., 2018) have been applied to text summarization (Su et al., 2021; Liu et al., 2022a). Unlike its autoregressive counterpart, an NAR model predicts target words independently so that the length-control text generation problem can be decomposed into shared sub-problems, facilitating dynamic programming for efficient computation (Liu et al., 2022a,b). Such methods fall into the *hard-constraint* category.

Specifically, Liu et al. (2022a,b) apply the Connectionist Temporal Classification (CTC) model (Graves et al., 2006), which allows placeholders and repetitions during generation but removes them in a post hoc manner. However, these CTC-based methods tend to produce word-level extractive summaries due to their restricted modeling capacity. As a result, the generated summaries contain very few new words and mostly preserve the original word order (Chuang et al., 2021; Shao and Feng, 2022).

In this work, our first insight is to apply the Directed Acyclic Transformer (DAT, Huang et al., 2022b) for length-control summarization. DAT, originally introduced for machine translation, is a powerful NAR approach that expands its canvas to allow multiple plausible text fragments and predicts *links* to connect them (the links along with the predicted words are called a *path*). In this way, DAT is more flexible in predicting words and determining their order, resulting in better generation quality than CTC (Huang et al., 2022b, 2023b).

For length-control decoding based on DAT, we propose a Sequence Maximum a Posteriori

[*]Work partially done during an internship at AIR
[†]Corresponding author
[‡]Canada CIFAR AI Chair
[1]Our code, output, and data are available at https://github.com/MANGA-UOFA/DAT-LenC

Findings of the Association for Computational Linguistics: EACL 2024, pages 11572–11583

(SeqMAP) decoding objective, which marginalizes possible linked steps and seeks the most probable sequence satisfying the length budget. Our SeqMAP is different from the existing MAP decoding objective (Shao et al., 2022) that seeks the most probable path (links and words), which we refer to as PathMAP. Since DAT training considers the marginalization of all possible links for a groundtruth sequence, our SeqMAP aligns with the training objective better (as it also performs marginalization), and is expected to surpass PathMAP in length-control summarization.

Compared with PathMAP, our SeqMAP is a more challenging decoding objective. The traditional PathMAP performs $\arg\max$ for both links and words, which can be reorganized according to prediction time steps and accomplished by dynamic programming in one pass. On the contrary, our SeqMAP performs $\arg\max$ for word selection after summing over all possible linked steps, but $\max$ and sum operations cannot be directly swapped to decompose the overall problem by time steps, breaking down the dynamic programming algorithm.

To this end, we propose an approximate algorithm to address SeqMAP, where we nevertheless decompose the objective by time steps. For each step, we make a greedy selection for the $\max$ operation, where we only consider a beam of high-probability sequences for the sum operation. Further, we may apply a reranker by a pretrained BERT model (Zhuang et al., 2021) to select the best sequence in the beam for performance improvement.

We perform extensive experiments using the Gigaword (Graff et al., 2003) and DUC2004 (Bommasani and Cardie, 2020) datasets, following previous work in non-autoregressive length-control summarization (Liu et al., 2022a). Results show that both SeqMAP and PathMAP outperform existing models based on CTC, which justifies our choice of the DAT model. Further, our approximate algorithm for SeqMAP outperforms PathMAP even without the reranker, suggesting the superiority of the SeqMAP objective. Our reranker further improves the summarization quality consistently, demonstrating its effectiveness.

2 Methodology

In this section, we first present the Directed Acyclic Transformer (DAT) and the existing PathMAP decoding method (Subsection 2.1). Then, we introduce our SeqMAP decoding objective, and develop a beam search-based dynamic programming algorithm to approach it (Subsection 2.2). Finally, we design a reranking process to select the best beam candidate (Subsection 2.3).

2.1 Length Control with DAT

DAT Model. Our first contribution is the adaptation of the Directed Acyclic Transformer (DAT, Huang et al., 2022b) to the length-control summarization task. DAT is a non-autoregressive (NAR) model, capable of generating multiple plausible output segments, which are then selected and connected via *links* to form the final output. DAT provides more flexibility in word selection and ordering, compared with the existing NAR summarization system (Liu et al., 2022a), which is based on the Connectionist Temporal Classification (CTC) model (Graves et al., 2006).

Consider the source text $\mathbf{x} = (\mathrm{x}_1, \cdots, \mathrm{x}_{T_\mathrm{x}})$ and the corresponding groundtruth summary $\mathbf{y} = (\mathrm{y}_1, \cdots, \mathrm{y}_{T_\mathrm{y}})$, where T_x and T_y denote their lengths. To allow multiple plausible output segments, DAT expands its generation canvas by having S prediction steps, typically $S \geq T_\mathrm{y}$.

At each step $s \in \{1, \cdots, S\}$, DAT makes a word prediction $p^{(s)}_{\text{word}}(\cdot)$ and a link prediction $p^{(s)}_{\text{link}}(\cdot)$.

In particular, the word prediction is given by

$$p^{(s)}_{\text{word}}(\cdot|\mathbf{x}) = \mathrm{softmax}(\mathbf{W}_{\text{word}} \boldsymbol{h}_s) \quad (1)$$

where $\mathbf{W}_{\text{word}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a learnable matrix and $\boldsymbol{h}_s \in \mathbb{R}^d$ is the decoder's hidden state at step s, with d being the dimension and $|\mathcal{V}|$ representing the vocabulary size.

Link prediction forecasts the step of the subsequent word that follows the sth step:

$$p^{(s)}_{\text{link}}(\cdot|\mathbf{x}) = \mathrm{softmax}([\boldsymbol{k}_s^\top \boldsymbol{q}_{s+1}; \cdots; \boldsymbol{k}_s^\top \boldsymbol{q}_S]) \quad (2)$$

where $\boldsymbol{k}_s = \mathbf{W}_\mathrm{k} \boldsymbol{h}_s$ and $\boldsymbol{q}_s = \mathbf{W}_\mathrm{q} \boldsymbol{h}_s$ are transformations of the hidden state, both $\mathbf{W}_\mathrm{k}$ and $\mathbf{W}_\mathrm{q}$ being learnable matrices in $\mathbb{R}^{d \times d}$. The $[;]$ operation concatenates scalars into a column vector.

Given a reference summary $\mathbf{y}$ of length T_y from the training set, the DAT model predicts links to connect T_y-many steps among a total of S generation steps. We denote the linked steps by $\boldsymbol{a} = (a_1, \cdots, a_{T_\mathrm{y}})$, where $1 = a_1 < \cdots < a_{T_\mathrm{y}} = S$. Following Huang et al. (2022b), we refer to the linked steps and corresponding words as a *path*.[2]

[2] In practice, two special tokens, ⟨bos⟩ and ⟨eos⟩, are added at the beginning and end of $\mathbf{y}$, respectively. Therefore, every path has ⟨bos⟩ at step 1 and ⟨eos⟩ at step S.

The probability of a path—having the linked steps $\boldsymbol{a}$ and yielding the target summary $\mathbf{y}$—is given by

$$p(\mathbf{y}, \boldsymbol{a}|\mathbf{x}) = \prod_{t=2}^{T_\mathrm{y}} p_{\text{link}}^{(a_{t-1})}(a_t) \prod_{t=1}^{T_\mathrm{y}} p_{\text{word}}^{(a_t)}(\mathrm{y}_t) \quad (3)$$

where $p_{\text{link}}^{(a_{t-1})}(a_t)$ is the probability of linking the a_{t-1}th generation step to the a_tth, and $p_{\text{word}}^{(a_t)}(\mathrm{y}_t)$ is the probability of predicting the word y_t at the a_tth step. Conditioning on $\mathbf{x}$ is omitted in p_{word} and p_{link} for brevity. Further denoting them by l_{a_{t-1},a_t} and w_{a_t,y_t}, we rewrite Eqn. (3) as

$$p(\mathbf{y}, \boldsymbol{a}|\mathbf{x}) = \prod_{t=2}^{T_\mathrm{y}} l_{a_{t-1},a_t} \prod_{t=1}^{T_\mathrm{y}} w_{a_t,\mathrm{y}_t} \quad (4)$$

$$= w_{1,\mathrm{y}_1} \prod_{t=2}^{T_\mathrm{y}} l_{a_{t-1},a_t} w_{a_t,\mathrm{y}_t} \quad (5)$$

For DAT, obtaining the probability of generating a target summary $\mathbf{y}$ requires marginalizing over all possible sequences of linked steps, given by

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\boldsymbol{a} \in \Gamma_{T_\mathrm{y},S}} p(\mathbf{y}, \boldsymbol{a}|\mathbf{x}) \quad (6)$$

$$= \sum_{\boldsymbol{a} \in \Gamma_{T_\mathrm{y},S}} w_{1,\mathrm{y}_1} \prod_{t=2}^{T_\mathrm{y}} l_{a_{t-1},a_t} w_{a_t,\mathrm{y}_t} \quad (7)$$

where $\Gamma_{T_\mathrm{y},S} = \{\boldsymbol{a} = (a_1, \cdots, a_{T_\mathrm{y}}) | 1 = a_1 < \cdots < a_{T_\mathrm{y}} = S\}$ is the set of all possible T_y-many linked steps among S-many generation steps.

Although direct enumeration over $\Gamma_{T_\mathrm{y},S}$ is intractable, a dynamic programming algorithm can efficiently compute the marginalization for training (Huang et al., 2022b).

PathMAP Decoding. Recently, Shao et al. (2022) propose a DAT-based decoding algorithm (referred to as PathMAP) that finds the most probable path of words and linked steps, given a specific length.

Formally, we consider a length-T path of linked steps $\boldsymbol{a} \in \Gamma_{T,s}$ and predicted words $\mathbf{v}_{\boldsymbol{a}} = (\mathrm{v}_{a_1}, \cdots, \mathrm{v}_{a_T})$, where $\mathrm{v}_{a_t} \in \mathcal{V}$ is the predicted word at the a_tth step and $s \in \{T, \cdots, S\}$ is any valid prediction step allowing T words. The most probable length-T path is obtained by maximizing the joint probability of the linked steps and word predictions at any valid prediction step, given by

$$\max_{s \in \{T,\cdots,S\}} \max_{\substack{\boldsymbol{a} \in \Gamma_{T,s} \\ \mathbf{v}_{\boldsymbol{a}} \in \mathcal{V}^T}} w_{1,\mathrm{v}_1} \prod_{t=2}^{T} l_{a_{t-1},a_t} w_{a_t,\mathrm{v}_{a_t}} \quad (8)$$

$$= \max_{s \in \{T,\cdots,S\}} \max_{\boldsymbol{a} \in \Gamma_{T,s}} w_{1,\mathrm{v}_1^*} \prod_{t=2}^{T} l_{a_{t-1},a_t} w_{a_t,\mathrm{v}_{a_t}^*} \quad (9)$$

$$= \max_{s \in \{T,\cdots,S\}} w_{s,\mathrm{v}_s^*} \max_{s' \in \{T-1,\cdots,s-1\}} \Bigg\{ l_{s',S} \max_{\boldsymbol{a} \in \Gamma_{T-1,s'}} \left(w_{1,\mathrm{v}_1^*} \prod_{t=2}^{T-1} l_{a_{t-1},a_t}\, w_{a_t,\mathrm{v}_{a_t}^*} \right) \Bigg\} \quad (10)$$

In Eqn. (9), we choose word predictions greedily because the max operation of a word is independent of the max operations over other words and linked steps. Further, Eqn. (9) can be decomposed into Eqn. (10) in a recursive fashion, allowing for efficient dynamic programming.

2.2 Our SeqMAP Decoding

A Limitation of PathMAP. As seen, the PathMAP objective described in Eqn. (8) performs max for both links and words, which is different from DAT's marginalization training objective. This is not ideal, as a discrepancy between training and inference often leads to performance degradation (Bengio et al., 2015; Zhang et al., 2019).

SeqMAP Objective. To this end, we propose a novel Sequence Maximum a Posteriori (SeqMAP) objective that marginalizes all possible linked steps to find the most probable sequence of length T. This is given by

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{V}^T} \sum_{s \in \{T,\cdots,S\}} \sum_{\boldsymbol{a} \in \Gamma_{T,s}} w_{1,\mathrm{y}_1} \prod_{t=2}^{T} l_{a_{t-1},a_t} w_{a_t,\mathrm{y}_t} \quad (11)$$

However, solving Eqn. (11) is challenging. This is because the arg max and summation cannot be swapped to decompose the overall objective based on the sentence length, making it infeasible to design a SeqMAP-like dynamic programming algorithm.

Decoding Algorithm for SeqMAP. We propose an approximate algorithm to maximize our SeqMAP objective. The general idea is to perform dynamic programming (DP) by re-organizing arg max and summation operations based on the output length anyway, and further improve the efficiency of the summation with beam search.

Let $\mathcal{A}_{t,s}$ be the (approximated) top-K length-t sequences that are generated *at or before* DAT's sth step, denoted by

$$\mathcal{A}_{t,s} = \{\mathbf{b}^{(k)}\}_{k=1}^{K} \tag{12}$$

In addition, we need to store the probability of generating $\mathbf{b}^{(k)}$ ending at step s'; we denote it by $u_{s'}(\mathbf{b}^{(k)})$, for $t \leq s' \leq s$. This is because a sequence $\mathbf{b}^{(k)}$ can be generated at different steps before the sth, and tracking their probabilities is helpful for marginalization.

The initialization of $\mathcal{A}_{t,s}$ fills in the DP table for $t = 0$, where each $\mathcal{A}_{0,s}$ (for $s > 0$) contains a special $\langle\text{bos}\rangle$ token, indicating the beginning of a sentence. Additionally, the score for each $u_s(\langle\text{bos}\rangle)$ is initialized to 1.

The DP recursion iteratively computes $\mathcal{A}_{t,s}$ for every $t > 0$ and every s such that $t \leq s \leq S$. We observe that each $\mathcal{A}_{t,s}$ can be obtained by 1) expanding the length-$(t-1)$ sequences in $\mathcal{A}_{t-1,s-1}$ with words from the sth step, which results in sequences ending at step s, and 2) further merging them with the length-t sequences in $\mathcal{A}_{t,s-1}$, which approximates the best length-t sequences ending before step s. Therefore, we design a recursive step with an EXPAND operation and a MERGE operation.

EXPAND. Let $\mathcal{B}_{t,s}$ be the approximated top-K length-t sequences that are generated *exactly at* the sth step. Intuitively, we can approximate $\mathcal{B}_{t,s}$ by expanding sequences in $\mathcal{A}_{t-1,s-1}$ with the word predictions at step s, given by

$$\mathcal{B}_{t,s} = \text{top-}K\Big\{\mathbf{b} \oplus \text{v} \;\Big|\; \mathbf{b} \in \mathcal{A}_{t-1,s-1}, \text{v} \in \mathcal{V}\Big\} \tag{13}$$

When expanding a partial sequence with $\text{v} \in \mathcal{V}$, our implementation only considers the top-V words based on the predicted word probability in Eqn. (1) to improve efficiency.

For each expanded sequence in $\mathcal{B}_{t,s}$, the corresponding scores are given by

$$u_s(\mathbf{b} \oplus \text{v}) = w_{s,\text{v}} \sum_{s'=t-1}^{s-1} u_{s'}(\mathbf{b}) \cdot l_{s',s} \tag{14}$$

Here, we marginalize out different previous linked steps that may generate $\mathbf{b}$, which follows the spirit of the marginalization in our SeqMAP objective (11).

MERGE. By definition, $\mathcal{A}_{t,s}$ approximates the best length-t sequences, which may be generated at the sth step (approximated by $\mathcal{B}_{t,s}$) or before the sth step (approximated by $\mathcal{A}_{t,s-1}$). Therefore, we can obtain $\mathcal{A}_{t,s}$ by merging $\mathcal{B}_{t,s}$ and $\mathcal{A}_{t,s-1}$:

$$\mathcal{A}_{t,s} = \text{top-}K\Big\{\mathcal{A}_{t,s-1} \cup \mathcal{B}_{t,s}\Big\} \tag{15}$$

where the top-K operation ranks the sequences by their total scores at and before the sth step, given by $\sum_{s'=t}^{s} u_{s'}(\mathbf{b})$. This ensures that our approximation algorithm considers the marginalization over s in Eqn. (11).

Appendix A provides the pseudocode of our decoding algorithm.

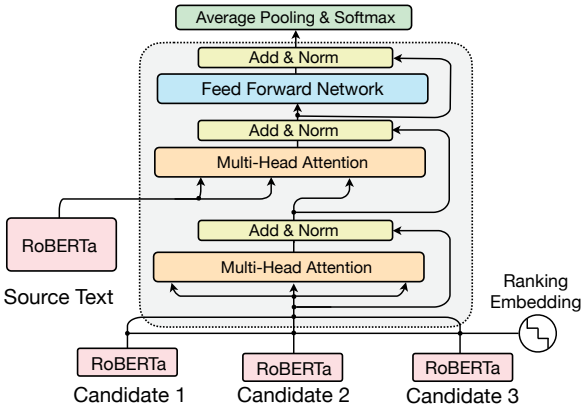

Figure 1: The neural architecture of our reranker. This example assumes a beam size of $K = 3$.

2.3 Reranker

Our SeqMAP beam search finds several high-probability summary candidates, providing an opportunity to further improve the generation quality using a reranker (Och et al., 2004; Lee et al., 2021; Ravaut et al., 2022). In particular, we train the reranker by predicting which beam candidate has the most overlapped words compared with the groundtruth summary.

Neural Architecture. Figure 1 depicts the overall architecture of our reranker.

First, we use the pretrained RoBERTa model (Zhuang et al., 2021) to represent the source text $\mathbf{x}$ and each of the summary candidates $\mathbf{b}^{(k)}$, for $1 \leq k \leq K$, individually.

Then, we build a one-layer Transformer block to predict the best summary for $\mathbf{x}$. We start by proposing a novel approach of rank embeddings (initialized randomly and trained by backpropagation) to capture the rank of a candidate given by the DAT model. The rank embedding has the same dimension as RoBERTa, and the kth rank's embedding is added to the RoBERTa representation of every

word in the kth candidate. After that, self-attention is performed among the words in different candidates, and cross-attention is performed to fetch information from the source text. Finally, we pool the representations of all the time steps (including all the words in different candidates) and apply a K-way softmax to predict the best candidate.

Training Objective. The target of the classification is the beam candidate $\mathbf{b}^{(k)}$ having the highest number of overlapped words compared with the groundtruth summary $\mathbf{y}$ of a training sample. We use the standard cross-entropy loss for training, and apply label smoothing (Szegedy et al., 2016) to improve generalization.

3 Experiments

3.1 Setup

Our experimental setups generally follow the previous non-autoregressive length-control studies (Liu et al., 2022a,b).

Datasets. We evaluated the proposed method on the Gigaword headline generation dataset (Graff et al., 2003) and the DUC2004 dataset (Bommasani and Cardie, 2020). The Gigaword dataset pairs news articles with their headlines; it contains 3.0M training pairs, 189K validation pairs, and 1951 test pairs. The DUC2004 dataset contains 500 samples, and is only used as a test set. The performance in the DUC2024 experiment is obtained by the model trained on Gigaword.

Metrics. We used ROUGE scores (Lin, 2004) as the main metric for the quality evaluation of generated summaries. Specifically, we reported ROUGE-n, for $n \in \{1, 2\}$, to measure the n-gram overlaps between the generated summary and the ground truth, and reported ROUGE-L to measure the length of the longest common subsequence.

In addition, we used a large language model (LLM) as a surrogate for human evaluation. We performed pairwise comparisons between the generations of our SeqMAP and baseline models. We reported the win, loss, and tie percentages.

Implementation Details. For the neural architecture of our DAT models, we used the Transformer-base (Vaswani et al., 2017) as the backbone. To train the models, we used a batch size of 64K tokens, with a maximum of 100K updates. For regularization, we set the dropout rate to 0.1 and the label smoothing factor to 0.1. For decoding, our SeqMAP had a beam size of $K = 20$ and a vocabulary exploration of $V = 5$, as mentioned after Eqn. (13). More details can be found in our GitHub repository (Footnote 1), where the configurations of training and inference are included.

3.2 Results and Analyses

Main Results on Gigaword. Table 1 presents our main results on the Gigaword dataset. For thorough comparison, we evaluate the approaches in three scenarios: setting the summary lengths to 20%, 25%, and 30% of the source text length. For non-strict length-control methods, the output will be truncated if the length exceeds the limit.

We first include two baseline methods that do not have length control: 1) the standard autoregressive Transformer (AT, Vaswani et al., 2017), and 2) BERT-CRF (Su et al., 2021), which is a non-autoregressive method that uses BERT as the encoder (Devlin et al., 2019) and applies a conditional random field (CRF, Lafferty et al., 2001) for decoding. As seen in Table 1, a large portion of the generated sentences from AT and BERT-CRF require truncation to meet the length requirement (Rows 1 and 3), resulting in incomplete summaries. Also, their ROUGE scores are low in general.

We further consider a soft length-control method: AT-LenInfo (Liu et al., 2018), which integrates a length embedding to softly guide the output length of the autoregressive Transformer. As we can see, AT-LenInfo largely reduces the truncation ratio, while achieving similar ROUGE scores to AT. However, AT-LenInfo still has 1.03–4.66% of sentences truncated, suggesting that such a soft-constraint method is inadequate for length control.

On the other hand, the CTC (Liu et al., 2022a) and our DAT methods perform exact length control by dynamic programming; thus, no truncation is required (Rows 4–7). Among these models, our DAT (with any inference method) consistently outperforms CTC, verifying that DAT is a more powerful non-autoregressive model.

We also observe that our SeqMAP method, with or without the reranker, is better than PathMAP, with only one exception (the R-2 score in the 25% setting); our reranker further improves the total ROUGE scores (R-Sum) by 0.49–1.32 points. The results verify the superiority of our SeqMAP objective and the effectiveness of our decoding algorithms.

Main Results on DUC2004. We report the performance on the DUC2004 dataset in Table 2. As seen, the trends mirror those observed on the Gigaword dataset. In particular, our SeqMAP con-

Ratio	#	Model	%Truncate	R-1	R-2	R-L	R-Sum	Length
20%	1	AT w/ Truncate	60.28%	33.16	15.02	30.91	79.09	6.1
	2	AT-LenInfo w/ Truncate	1.03%	33.53	14.32	31.21	79.06	5.7
	3	BERT-CRF w/ Truncate	39.98%	31.94	13.67	30.09	75.70	5.8
	4	CTC Length Control	0.00%	34.14	13.61	31.96	79.71	6.9
	5	DAT PathMAP	0.00%	35.19	16.95	32.84	84.98	6.9
	6	DAT SeqMAP (Ours)	0.00%	**36.34**	**17.29**	**33.81**	**87.44**	6.9
	7	w/o Reranking	0.00%	35.63	17.24	33.28	86.15	6.8
25%	1	AT w/ Truncate	33.52%	34.52	15.78	31.90	82.20	6.8
	2	AT-LenInfo w/ Truncate	1.54%	35.12	16.19	32.60	83.91	7.3
	3	BERT-CRF w/ Truncate	18.93%	32.85	14.23	30.86	77.94	6.2
	4	CTC Length Control	0.00%	34.51	13.89	32.08	80.48	8.5
	5	DAT PathMAP	0.00%	36.11	17.43	33.41	86.95	8.5
	6	DAT SeqMAP (Ours)	0.00%	**36.74**	17.22	**33.87**	**87.83**	8.4
	7	w/o Reranker	0.00%	36.30	**17.40**	33.64	87.34	8.4
30%	1	AT w/ Truncate	20.04%	35.07	16.00	32.35	83.42	7.1
	2	AT-LenInfo w/ Truncate	4.66%	35.29	16.15	32.44	83.88	8.8
	3	BERT-CRF w/ Truncate	9.65%	33.08	14.44	31.08	78.60	6.4
	4	CTC Length Control	0.00%	34.04	13.63	31.49	79.16	10.0
	5	DAT PathMAP	0.00%	35.57	16.65	32.88	85.10	9.9
	6	DAT SeqMAP (Ours)	0.00%	**36.24**	16.74	**33.34**	**86.32**	9.9
	7	w/o Reranker	0.00%	35.80	**16.76**	33.04	85.60	9.9

Table 1: Results on the Gigaword dataset, where we set the length constraint to be 20%, 25%, and 30% of the source length. %Truncate is the percentage of sentences that require truncation to meet the length requirement. R-1, R-2, R-L, and R-Sum denote the ROUGE-1, ROUGE-2, ROUGE-L, and the sum of the three ROUGE scores, respectively.

	R-1	R-2	R-L	R-Sum
AT w/ Truncate	27.03	8.92	24.20	60.15
AT-LenInfo w/ Truncate	27.88	9.28	25.58	62.74
BERT-CRF w/ Truncate	23.69	7.16	21.64	52.49
CTC Length Control	28.67	8.29	26.34	63.30
DAT PathMAP	28.77	9.98	26.48	65.23
DAT SeqMAP (Ours)	**30.11**	**10.29**	**27.22**	**67.62**
w/o Reranker	29.17	10.27	26.66	66.10

Table 2: Results on the DUC 2004 dataset. The summary length is set to 20% of the source length.

Given the text: **[Source]**
Consider two summaries of the text:
Summary **[ID1]**: **[Summary1]**
Summary **[ID2]**: **[Summary2]**
A good summary is a shorter piece of text that has the essence of the original and adheres to coherence, faithfulness, relevance, and overall quality as defined above. Which summary is better?
Answer:

Table 3: Our prompt template for LLM-based pairwise evaluation. Here, "**Source**" is the text to be summarized. The choices of IDs are "1" and "2"; "**Summary1**" and "**Summary2**" are substituted with model-generated text. Since LLM is not robust to candidate ID and order (Zheng et al., 2023; Shen et al., 2023), we enumerate different combinations for a given case, resulting in four LLM queries.

sistently outperforms other baselines in all metrics. Since the DUC2004 results are obtained from models trained on Gigaword, we conclude that our approach is transferable to different testing scenarios.

LLM Evaluation. In addition to ROUGE scores, we prompt a large language model (LLM), in particular, `gpt-4-0125-preview`, serving as a surrogate for human evaluation. Concretely, we perform pairwise comparison between the outputs of our SeqMAP and baseline models on the Gigaword test set. For each comparison, we query an LLM four times by enumerating the order and IDs of the candidates for better robustness; the exact prompt is presented in Table 3.

Table 4 shows the results of LLM evaluation. We first observe that our SeqMAP dominates AT, AT-LenInfo, and CTC baselines with a winning rate of 68.80–72.40%. Further, our SeqMAP has an 11.73% higher winning rate than PathMAP with the same DAT backbone. Overall, our pairwise LLM evaluation is consistent with the comparison based on ROUGE scores (Tables 1 and 2), fur-

Pairwise Comparison	Win	Loss	Tie
Ours vs. AT w/ Trunc.	**68.70%**	0%	31.30%
Ours vs. AT-LenInfo w/ Trunc.	**69.45%**	0%	30.55%
Ours vs. CTC Length Control	**72.40%**	20.00%	7.60%
Ours vs. DAT PathMAP	**43.77%**	32.04%	24.18%

Table 4: LLM pairwise comparison between DAT SeqMAP and baseline methods. For each sample, we determine the outcome of "win," "loss," or "tie" based on the votes of the four LLM queries (by varying candidate order and ID tokens). We report the ratios of wins, losses, and ties of our model.

#	Model	Word Novelty	Reordering Degree
1	CTC	18.41%	8.69%
2	DAT PathMAP	21.80%	**9.09%**
3	DAT SeqMAP (Ours)	**22.48%**	9.01%
4	Human Reference	50.51%	8.05%

Table 5: The word novelty and reordering ratio of CTC, DAT PathMAP, DAT SeqMAP, and the human reference. The results are based on the Gigaword test set.

ther verifying the effectiveness of our SeqMAP approach.

Generation Novelty. In this work, we hypothesize that the Directed Acyclic Transformer (DAT, Huang et al., 2022b) is more flexible in predicting words and determining their order, compared with the Connectionist Temporal Classification (CTC) model (Graves et al., 2006), which is used in Liu et al. (2022a) for length-control summarization. To verify this hypothesis, we adopt two metrics, namely *word novelty* and the *reordering degree*, to compare the generated summaries by DAT and CTC.

In particular, word novelty measures the percentage of the summary's words that are not in the source text, indicating the method's flexibility in word choice. The reordering degree assesses the flexibility in generation order. It is computed by the following steps: 1) Align the words in the summary to the source text,[3] 2) Enumerate every pair of word positions (i, j) such that $i < j$ in the source, and 3) Count the fraction of the order being reversed in the output.

Table 5 presents the results on the Gigaword dataset. Interestingly, humans tend to use novel words but preserve the word order. The DAT model has the highest reordering degree, which, although not agreeing with humans, verifies our hypothesis that DAT is more flexible than CTC in word ordering. In addition, DAT also yields more novel words than CTC.

[3]We use FastAlign (Dyer et al., 2013), a tool for word alignment, to obtain the target-to-source alignments.

#	Model	Sentences/s	Words/s
1	AT w/ Trunc.	10.59	90.99
2	CTC Length Control	25.61	219.71
3	DAT PathMAP	**61.30**	**520.64**
4	DAT SeqMAP w/ Reranker	10.97	104.08
5	DAT SeqMAP w/o Reranker	20.75	177.00

Table 6: Inference speed of different methods.

Inference Speed. Non-autoregressive (NAR) models are originally designed to improve inference efficiency for text generation (Gu et al., 2018; Gu and Kong, 2021; Huang et al., 2022a). In this work, we compare the inference speed following the convention of NAR research, where the batch size is set to 1, mimicking real-world scenarios where user queries come one after another.

As seen from Table 6, NAR-based methods, except DAT SeqMAP with the reranker (Row 4), are faster than the autoregressive baseline (Rows 2, 3, and 5 versus Row 1). It is understandable that our DAT SeqMAP with the reranker is not as efficient as other NAR models because it requires a dynamic programming-based beam search and an external neural module for reranking. Nevertheless, our method is still faster than the autoregressive model, showing its practical value in application.

It is worth mentioning that our SeqMAP implementation has not been optimized for parallelism yet, as opposed to the PathMAP implementation (Shao et al., 2022). Therefore, there is room to further improve the efficiency of our SeqMAP.

Decoding Hyperparameters. Our SeqMAP decoding method has two hyperparameters controlling its search scope:

- K: the beam size for the candidate set $\mathcal{A}_{s,t}$ as in Eqn. (12), and
- V: the number of words explored at each step, as mentioned after Eqn. (13).

We analyzed their impact by varying K (with fixed $V = 5$) and varying V (with fixed $K = 20$). The results are listed in Tables 7 and 8, respectively.

First, we observe that SeqMAP's performance generally improves as K or V increases, with or without the reranker. This shows that our approximation algorithm for SeqMAP benefits from increasing its search scope.

When K or V is increased beyond a certain value ($K = 20$ or $V = 5$), the performance stops increasing, or even slightly decreases in some met-

	Model	K	R-1	R-2	R-L	R-Sum
	PathMAP	-	36.11	17.43	33.41	86.95
SeqMAP	w/o Reranker	10	36.27	17.3	33.58	87.15
		15	36.27	17.34	33.59	87.20
		20	36.30	17.40	33.64	87.34
		25	36.27	17.36	33.6	87.23
	w/ Reranker	10	36.61	17.27	33.8	87.68
		15	36.65	**17.41**	33.80	87.86
		20	**36.74**	17.22	**33.87**	87.83
		25	**36.74**	17.32	**33.87**	**87.93**

Table 7: Reranker performance of different K. Here, $V = 5$, and the length ratio is 0.25.

		V	R-1	R-2	R-L	R-Sum
	PathMAP	-	36.11	17.43	33.41	86.95
SeqMAP	w/o Reranker	1	36.23	17.37	33.59	87.19
		3	36.24	17.34	33.58	87.16
		5	36.30	17.40	33.64	87.34
		7	36.24	17.34	33.59	87.17
		9	36.24	17.34	33.59	87.17
	w/ Reranker	1	36.45	16.92	33.62	86.99
		3	36.73	17.16	33.84	87.73
		5	**36.74**	**17.22**	**33.87**	**87.83**
		7	**36.74**	17.20	33.85	87.79
		9	**36.74**	17.18	33.85	87.77

Table 8: Reranker performance of different K. Here, $V = 20$, and the length ratio is 0.25.

rics such as R-2. This is consistent with autoregressive beam search, where a moderately sized beam works the best (Stahlberg and Byrne, 2019; Meister et al., 2020; Wen et al., 2024).

Analysis of the Reranker. As mentioned in Section 2.3, our neural reranker for summarization consists of several key components: 1) *Label smoothing*, which is used for regularization (Szegedy et al., 2016), 2) *Rank Embedding*, which captures the rank of a candidate given by DAT, and 3) *Pretraining*, where we use RoBERTa-base (Zhuang et al., 2021) for sentence representation. We perform an ablation study on the impact of the three components, and present the results in Table 9.

As seen, removing label smoothing results in a 0.27 decrease in R-Sum (Rows 1 and 2). This is because the difference between beam candidates can be minimal, and lowering the confidence of the reranker in training leads to better generalization.

When removing the rank embedding, we observe a 0.6 decrease in R-SUM (Rows 1 and 3). This confirms our intuition that incorporating the DAT model's ranking as prior knowledge is beneficial.

#	LS	RankEmb	Pretrain	R-1	R-2	R-L	R-Sum
1	✓	✓	✓	**36.61**	17.27	**33.80**	**87.68**
2		✓	✓	36.50	17.18	33.73	87.41
3	✓		✓	36.23	17.28	33.57	87.08
4	✓	✓		36.38	**17.34**	33.68	87.41

Table 9: Ablation study on the effect of label smoothing, positional embedding, and pretraining. To save the training time, we set K to 10 and V to 5.

Additionally, we observe that having Rank Embedding leads to faster training convergence.

Without using the pretrained RoBERTa model (i.e., randomly initializing the weights), R-SUM decreases by 0.27. This drop is anticipated since the pretrained model enhances text understanding.

4 Related Work

Non-autoregressive (NAR) models predict words independently, and are initially developed to increase the inference speed of neural machine translation (Gu et al., 2018; Lee et al., 2018; Qian et al., 2021; Gu and Kong, 2021; Huang et al., 2023a). Recently, NAR models have been adapted for length-control summarization in our previous work (Liu et al., 2022a,b), where we find that NAR's independent word predictions allow the length-control tasks to be divided into several independent sub-tasks, resulting in an efficient exploration of the NAR output space. Therefore, we have developed dynamic programming algorithms based on the Connectionist Temporal Classification (CTC) model (Graves et al., 2006).

Our work introduces a novel decoding algorithm that leverages the Directed Acyclic Transformer (DAT, Huang et al., 2022b). Unlike CTC, which preserves the order of the source sequence (Chuang et al., 2021; Shao and Feng, 2022), DAT offers greater flexibility in word selection and generation order. While recognizing the value of existing DAT-based decoding methods (Shao et al., 2022) for managing length, we identify their limitations and propose a new SeqMAP approach.

Our proposed reranker is inspired by the reranking methods in machine translation (Och et al., 2004; Lee et al., 2021) and summarization (Ravaut et al., 2022), where a list of n-best sequences are presented to an external model for scoring.

5 Conclusion

This work proposes a novel SeqMAP decoding method based on the Directed Acyclic Transformer

(DAT) for length-control summarization. Our SeqMAP is a beam search-based dynamic programming algorithm, which bridges the gap between the training and inference of DAT. Experimental results on the Gigaword and DUC2004 datasets demonstrate the effectiveness of our SeqMAP approach.

6 Limitations

One potential limitation of this work is that the proposed algorithm for SeqMAP does not guarantee to find the best sequence in DAT's decoding space. As discussed, an exact algorithm may not exist due to the computational complexity of the Maximum A Posteriori (MAP) problems (Koller and Friedman, 2009; de Campos, 2011). Nevertheless, we propose an approximate algorithm for SeqMAP and empirically show that it is consistently better than PathMAP.

We also notice a slight performance decrease with a large scope of our beam search (controlled by K and V mentioned in Section 2.2). This phenomenon, previously observed in autoregressive text generation models (Stahlberg and Byrne, 2019; Meister et al., 2020), might stem from the label bias issue (Lafferty et al., 2001; Huang et al., 2021). We leave the investigation of this phenomenon as future work.

Acknowledgments

We would like to thank all reviewers and chairs for their comments. This research was supported in part by the Natural Science Foundation of China under Grant No. 62376133. This research was also supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant Nos. RGPIN-2020-04440 and RGPIN-2020-04465, the Amii Fellow Program, the Canada CIFAR AI Chair Program, the Alberta Innovates Program, the Digital Research Alliance of Canada (alliancecan.ca), and a donation from DeepMind.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Rishi Bommasani and Claire Cardie. 2020. Intrinsic evaluation of summarization datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8075–8096.

Shun-Po Chuang, Yung-Sung Chuang, Chih-Chiang Chang, and Hung-yi Lee. 2021. Investigating the reordering capability in CTC-based non-autoregressive end-to-end speech translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1068–1077.

Cassio P de Campos. 2011. New complexity results for map in bayesian networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2100–2106.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English Gigaword. *Linguistic Data Consortium*, 4:34.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine learning*, pages 369–376.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133.

Chenyang Huang, Fei Huang, Zaixiang Zheng, Osmar Zaïane, Hao Zhou, and Lili Mou. 2023a. Multilingual non-autoregressive machine translation without knowledge distillation. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023*, pages 161–170.

Chenyang Huang, Wei Yang, Yanshuai Cao, Osmar Zaïane, and Lili Mou. 2021. A globally normalized neural model for semantic parsing. In *Proceedings of the 5th Workshop on Structured Prediction for NLP*, pages 61–66.

Chenyang Huang, Hao Zhou, Osmar R Zaïane, Lili Mou, and Lei Li. 2022a. Non-autoregressive translation with layer-wise prediction and deep supervision. In

Proceedings of the AAAI Conference on Artificial Intelligence, pages 10776–10784.

Fei Huang, Pei Ke, and Minlie Huang. 2023b. Directed acyclic transformer pre-training for high-quality non-autoregressive text generation. *Transactions of the Association for Computational Linguistics*, 11:941–959.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. 2022b. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 9410–9428.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, page 282–289.

Ann Lee, Michael Auli, and Marc'Aurelio Ranzato. 2021. Discriminative reranking for neural machine translation. In *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)*, pages 7250–7264.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Puyuan Liu, Chenyang Huang, and Lili Mou. 2022a. Learning non-autoregressive models from search for unsupervised sentence summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7916–7929.

Puyuan Liu, Xiang Zhang, and Lili Mou. 2022b. A character-level length-control algorithm for non-autoregressive sentence summarization. In *Advances in Neural Information Processing Systems*, pages 29101–29112.

Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4110–4119.

Clara Meister, Ryan Cotterell, and Tim Vieira. 2020. If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2185.

Ani Nenkova, Sameer Maskey, and Yang Liu. 2011. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, page 3.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 161–168.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 1993–2003.

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. 2022. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Raphael Schumann, Lili Mou, Yao Lu, Olga Vechtomova, and Katja Markert. 2020. Discrete optimization for unsupervised sentence summarization with word-level extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5032–5042.

Chenze Shao and Yang Feng. 2022. Non-monotonic latent alignments for CTC-based non-autoregressive machine translation. In *Advances in Neural Information Processing Systems*, pages 8159–8173.

Chenze Shao, Zhengrui Ma, and Yang Feng. 2022. Viterbi decoding of directed acyclic transformer for non-autoregressive machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4390–4397.

Chenhui Shen, Liying Cheng, Xuan-Phi Nguyen, Yang You, and Lidong Bing. 2023. Large language models are not yet human-level evaluators for abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4215–4233.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing, pages 3356–3362.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 234–243.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3999–4004.

Akim Tsvigun, Ivan Lysenko, Danila Sedashov, Ivan Lazichny, Eldar Damirov, Vladimir Karlov, Artemy Belousov, Leonid Sanochkin, Maxim Panov, Alexander Panchenko, Mikhail Burtsev, and Artem Shelmanov. 2022. Active learning for abstractive text summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5128–5152.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 6000–6010.

Yuqiao Wen, Behzad Shayegh, Chenyang Huang, Yanshuai Cao, and Lili Mou. 2024. EBBS: An ensemble with bi-level beam search for zero-shot machine translation. *arXiv preprint arXiv:2403.00144*.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023. Large language models are not robust multiple choice selectors. In *International Conference on Learning Representations*.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227.

A The Pseudocode for Our Algorithm

We present in Algorithm 1 the pseudocode of our beam search-based dynamic programming algorithm for the SeqMAP objective.

As a recap of Section 2.2, $\mathcal{A}_{t,s}$ is the approximated top-K length-t sequences that are generated *at or before* DAT's sth step, whereas $\mathcal{B}_{t,s}$ is the approximated top-K length-t sequences generated *exactly at* the sth step.

In Algorithm 1, Lines 2–4 are the initialization of $\mathcal{A}_{0,s}$, which is set to be the special token $\langle\text{bos}\rangle$ indicating the beginning of a sentence, and the score for the token is 1.

Lines 5–14 are the iterations for computing the dynamic programming (DP) table for each $t \in \{1, \cdots, T\}$ and $s \in \{1, \cdots, S\}$. Within each iteration,

- Lines 7–12 correspond to our EXPAND operation, which approximates $\mathcal{B}_{t,s}$ by expanding sequences in $\mathcal{A}_{t-1,s-1}$. Specifically, Line 10 corresponds to the sequence expansion as described in Eqn. (13), and Line 11 computes the scores of newly generated sequences as described in Eqn. (14). Then, the top-K operation in Line 12 prevents the beam from growing exponentially.

- Lines 13–14 correspond to the MERGE operation, which combines the newly formed sequences in $\mathcal{B}_{t,s}$ and the inherited sequences from $\mathcal{A}_{t,s-1}$. Again, we have the top-K operation to keep the beam search tractable, where the ranking is given by summing the probabilities over different steps, as explained after Eqn. (15).

Our DP algorithm terminates when we have computed $\mathcal{A}_{T,S}$, which contains a few length-T sequences of high probability given by DAT. Finally, Line 15 applies a reranker (as described in Section 2.3) to determine the best summary in $\mathcal{A}_{T,S}$.

Algorithm 1: Our beam search-based dynamic programming algorithm for the SeqMAP objective

Input: S: total prediction steps in DAT; $\mathcal{V}$: vocabulary; $l_{i,j}$: link probability for $i, j \in \{1, \cdots, S\}$; $w_{s,\mathrm{v}}$: word probability for $s \in \{1, \cdots, S\}$ and $\mathrm{v} \in \mathbf{V}$; K: beam size; V: number of words to be expanded; T: the desired length

▷ `Initialization`

for $s := 1, \cdots, S$ **do**

  $\mathcal{A}_{0,s} := \{\langle \text{bos} \rangle\}$ ▷ $\langle \text{bos} \rangle$ `is the starting token`

  $u_s(\langle \text{bos} \rangle) := 1$ ▷ $\boldsymbol{u}_s(\mathbf{b})$ `is the score of generating` $\mathbf{b}$ `at step` s

▷ `Recursive steps`

for $t := 1, \cdots, T$ **do**

  for $s := t, \cdots, S$ **do**

    ▷ `Expand` $\mathcal{A}_{t-1,s-1}$ `to obtain` $\mathcal{B}_{t,s}$

    for each $\mathbf{b}$ in $\mathcal{A}_{t-1,s-1}$ **do**

      $B_{t,s} := \{\}$ ▷ `Initialization`

      for each top-V most probable predicted words $\mathrm{v}' \in \mathcal{V}$ at step s **do**

        $\mathcal{B}_{t,s} := \mathcal{B}_{t,s} \cup \{\mathbf{b} \oplus \mathrm{v}'\}$ ▷ $\oplus$ `denotes string concatenation`

        $u_s(\mathbf{b} \oplus \mathrm{v}') := w_{s,\mathrm{v}'} \sum_{s'=t-1}^{s-1} u_{s'}(\mathbf{b}) \cdot l_{s',s}$ ▷ `Marginalization`

      $\mathcal{B}_{t,s} := \text{top-}K \left\{\mathcal{B}_{t,s}\right\}$ ▷ `Ranking is based on` $u_s(\mathbf{b})$

    ▷ `Merge` $\mathcal{B}_{t,s}$ `and` $\mathcal{A}_{t,s-1}$

    if $s > t$ **then**

      $\mathcal{A}_{t,s} := \text{top-}K\left\{\mathcal{B}_{t,s} \cup \mathcal{A}_{t,s-1}\right\}$ ▷ `Ranking is based on` $\sum_{s'=t}^{s} u_{s'}(\mathbf{b})$

▷ `Reranking`

return the top reranked $\mathbf{b}$ in $\mathcal{A}_{T,S}$ according to our reranker (Section 2.3)