EEL: Efficiently Encoding Lattices for Reranking

Prasann Singhal[◇] **Jiacheng Xu**[♠] **Xi Ye**[◇] **Greg Durrett**[◇]
[◇]The University of Texas at Austin, [♠]Salesforce AI
`{prasanns, xiye, gdurrett}@cs.utexas.edu, jiacheng.xu@salesforce.com`

Abstract

Standard decoding approaches for conditional text generation tasks typically search for an output hypothesis with high model probability, but this may not yield the best hypothesis according to human judgments of quality. Reranking to optimize for *downstream* metrics can better optimize for quality, but many metrics of interest are computed with pre-trained language models, which are slow to apply to large numbers of hypotheses. We explore an approach for reranking hypotheses by using Transformers to efficiently encode lattices of generated outputs, a method we call EEL. With a single Transformer pass over the entire lattice, we can approximately compute a contextualized representation of each token as if it were only part of a single hypothesis in isolation. We combine this approach with a new class of *token-factored rerankers* (TFRs) that allow for efficient extraction of high reranker-scoring hypotheses from the lattice. Empirically, our approach incurs minimal degradation error compared to the exponentially slower approach of encoding each hypothesis individually. When applying EEL with TFRs across three text generation tasks, our results show both substantial speedup compared to naive reranking and often better performance on downstream metrics than comparable approaches.[1]

1 Introduction

Part of the progress in natural language generation over the past few years has been driven by a proliferation of decoding techniques, from beam search to sampling approaches like nucleus sampling (Holtzman et al., 2020), typical decoding (Meister et al., 2022), and contrastive decoding (Li et al., 2022). These techniques, however, only optimize for probabilistic objectives, rather than alignment with human judgments, which is typically better encapsulated by *downstream metrics*

[1]Code available at `https://github.com/PrasannS/eel-reranking`.

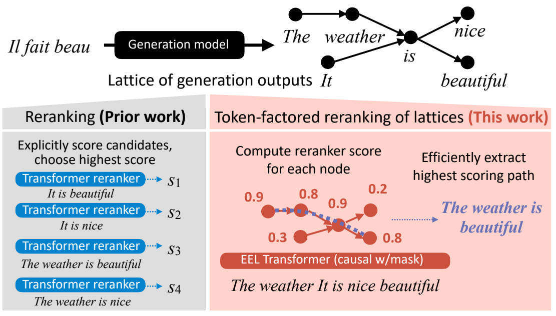


Figure 1: Overview of our approach. For a task such as translation, we can generate a lattice of plausible model outputs. Reranking these outputs with Transformers can be slow; we can achieve speedups by efficiently encoding the lattice in a single Transformer pass and using a token-factored reranker to efficiently find the best hypothesis in the lattice.

(Zhang et al., 2019b; Dhingra et al., 2019; Sellam et al., 2020; Rei et al., 2020) that specifically estimate human preference. Transformer (Vaswani et al., 2017) based *rerankers*, that assign estimated downstream scores to generation candidates, have recently made inroads in translation (Lee et al., 2021; Bhattacharyya et al., 2021; Rei et al., 2021; Freitag et al., 2022; Fernandes et al., 2022), open-ended generation (Krishna et al., 2022), and summarization (Ravaut et al., 2022; Song et al., 2021).

However, using rerankers poses several practical challenges. Rerankers work best over a large number of candidates, but generating large sets through beam search is slow. Recent work (Xu et al., 2022) has demonstrated the potential to derive and represent large candidate sets in directed acyclic graphs called *lattices*, but the problem remains that naively reranking these large sets is infeasible: scoring each candidate requires one, or even multiple (Fernandes et al., 2022) Transformer inference calls. Classic approaches for searching in lattices effectively (Koehn, 2004; Dyer and Resnik, 2010, inter alia) do not apply to Transformer rerankers, and there is no previously known approach for efficiently extracting good candidates from lattices. This paper proposes an approach to do exactly that, even on lattices encoding thousands of candidates.

We first propose a new class of reranker, the *token-factored reranker* (TFR), that allows efficient inference over a lattice by enforcing a causal mask and decomposing metric scores to the token level, allowing for flexible and efficient scoring while still performing at the same level as standard rerankers. We then show that lattices of generated hypotheses can be efficiently encoded by a Transformer in a single pass by using a custom attention mask and modified position encodings. We call this technique *EEL: Efficient Encoding of Lattices*. EEL enables fast TFR encoding of a large set of generation outputs at once, specifically enabling rapid extraction of the hypothesis with the highest TFR score; see Figure 1.

We evaluate our approach on lattices constructed from beam search as well as lattice decoding. Across three generation tasks (machine translation, summarization, and table-to-text generation), with several downstream metrics, we show that EEL is able to find optimal candidates with respect to the TFR with minimal degradation compared to an exhaustive approach. That is, the highest-scoring candidate from the efficient encoding is nearly as high quality as the highest scoring candidate from naively encoding all candidates independently. Moreover, we show that our approach is efficient and leads to gains on downstream metrics compared to naive reranking approaches. We further propose a method to diversely sample multiple varying hypotheses while reranking with respect to scoring metrics, allowing us to identify different optimal "modes" within an input lattice. Our approaches are particularly effective when used with lattice decoding, though we also demonstrate substantial speedups when reranking beam search outputs.

Our Contributions: (1) We introduce a new class of reranker, the token-factored reranker (TFR), that can support efficient inference over lattices. (2) We propose a method for encoding a lattice with a Transformer (EEL) that enables efficient reranking with TFRs with minimal degradation compared to exhaustive search. (3) We compare beam search, lattice decoding, and multiple reranking strategies across three NLG problems.

2 Setting

Our approach centers on reranking output from conditional text generation with neural models (Sutskever et al., 2014; Bahdanau et al., 2015). Such models place distributions over target output $\mathbf{y} = (y_1, \dots, y_n)$ given input sequence $\mathbf{x}$ via a text generation model θ: $p(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{k=1}^{n} p(y_k \mid \mathbf{y}_{<k}, \mathbf{x}; \theta)$. In trying to optimize this model probability $p(\mathbf{y} \mid \mathbf{x}; \theta)$ by finding the most likely $\mathbf{x}$, decoding algorithms often produce **candidate sets** H of highly likely outputs under the model for our reranking stage (e.g. for beam search, the candidate set is the N completions for all beams).

Our goal is to rerank candidate sets to optimize for a downstream objective $T(\mathbf{x}, \hat{\mathbf{y}})$. We assume our reranker $S : (\mathbf{x}, \hat{\mathbf{y}}) \rightarrow \mathbb{R}$ scores an (input, hypothesis) pair and returns a real-valued approximation of T using a Transformer model. Reranking can be divided into two steps: (1) the generation of a candidate set $H = \{\hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}, \dots, \hat{\mathbf{y}}^{(M)}\}$, and (2) the extraction of the highest scoring candidate $\mathbf{y}_{\text{best}} = \arg\max_{\mathbf{y} \in H} S(\mathbf{x}, \mathbf{y})$. The end result thus depends on the quality of the candidate set generated by a decoding method as well as how well S approximates T. Note, in this paper, we use *reranker* to refer to the model S that assigns scores to hypotheses, which is distinct from the actual reranking procedure itself.

In our setting, H is represented by a *lattice* which encodes the candidates (further detail in Section 4). This can either be a packed representation of beam search candidates or can be a more complex lattice generated natively by alternate decoding approaches. Specifically, we use the approach of Xu et al. (2022), which expands paths with a modified depth first search and merges similar paths in a recombination step to focus search budget on new paths, allowing for generation of larger sets of hypotheses with greater diversity. Notably, these lattices are not tree-structured and contain reentrancies.

Compared to complete candidate sets of normal text sequences, *lattices can exponentially reduce the number of tokens needed to encode large candidate sets*, enabling strong speedups if leveraged correctly. Thus, step (2) is the focus of this paper: given a scoring model S and a lattice encoding H, **can we encode a lattice and still select the highest scoring candidate encoded in H?** Our solution will specialize S to be a *token-factored* reranker, which we define below, and H to be encoded in a lattice; we show that these assumptions hold for strong rerankers that can be applied to real-world generation problems, even when encoding thousands of candidates in as little as a single Transformer pass. We attempt to minimize the error in our selected candidate versus the oracle best candidate $\mathbf{y}_{\text{best}}$ (defined above), which we refer to as *degradation*.

3 Reranking Generation Outputs

3.1 Token-factored Rerankers

A key idea in this work is that we can efficiently rerank a lattice encoding of a candidate set given a certain reranker structure. Specifically, if we can decompose the underlying reranking score to be a linear function of tokens in the lattice, we can extract hypotheses efficiently (Section 3.2). We thus propose the *token-factored reranker* (TFR).

Assume a reranker model $S(\mathbf{x}, \hat{\mathbf{y}})$ that, given a candidate, generates a score evaluating for some downstream objective T (e.g. quality, formality, etc.). Assume moreover that S involves a Transformer $f : (\mathbf{x}, \hat{\mathbf{y}}) \to \mathbf{h}$ that produces contextualized embeddings h_i for each output token $\hat{y}_i$. These can condition on $\mathbf{x}$, so f can be either an encoder, decoder, or encoder-decoder model. This work primarily examines causally constrained TFRs, defined as follows:

Definition 1 (token-factored reranker)**.** *Let S be a **token-factored reranker** (TFR) if it takes the form $S(\mathbf{x}, \hat{\mathbf{y}}) = \sum_{k=1}^{n} s(f_c(\mathbf{x}, \hat{\mathbf{y}}_{\leq k})_k)$ where s is some linear function and f_c is a* causal *contextualized model that only depends on tokens up to and including y_k.*

We specialize f_c to be a causal model because we found this important to improve the quality of our approximation. However, theoretically we can also use **bidirectional token-factored rerankers (bTFRs)** where instead $S(\mathbf{x}, \hat{\mathbf{y}}) = \sum_{k=1}^{n} s(f(\mathbf{x}, \hat{\mathbf{y}})_k)$ for non-causal f (e.g., BERT).

For the aggregation function, we found $s(x) = \frac{x}{n}$, averaging, to work well.

Generality TFRs can be trained straightforwardly on any dataset of $(\mathbf{x}, \hat{\mathbf{y}})$ pairs labeled with quality scores. Furthermore, a range of existing architectures fall into or near a TFR-oriented framework. For example, decoding time token-level *model score*, the negative log of the probability of a selected token at a given decoding step, is a TFR.

Algorithm 1 Extract Best Path in Lattice

```
Input: Topologically sorted list flat, list ends with all ending nodes $v \in V$, the set of all paths in lattice $P = (v_1, \ldots, v_k)$, $\text{par}(v_i) \subseteq V$ returns set of parents of $v_i$
Output: highest scoring path returned
1: best: $V \mapsto (P, \mathbb{R})$
2: for $c \in$ flat do
3:     $\text{s}, \hat{p} \leftarrow \arg\max_{p \in \text{par}(c)} \text{score}(\text{best}(p))$
4:     $\text{best}(c) \leftarrow (\hat{p} \cup (c), \text{s} + s(f(c)))$ // extend hypothesis
5:     $i \leftarrow i + 1$
6: end for
7: return $\arg\max_{e \in ends} \text{best}(e)$ // return best end state; can
   extract path via backpointers
```

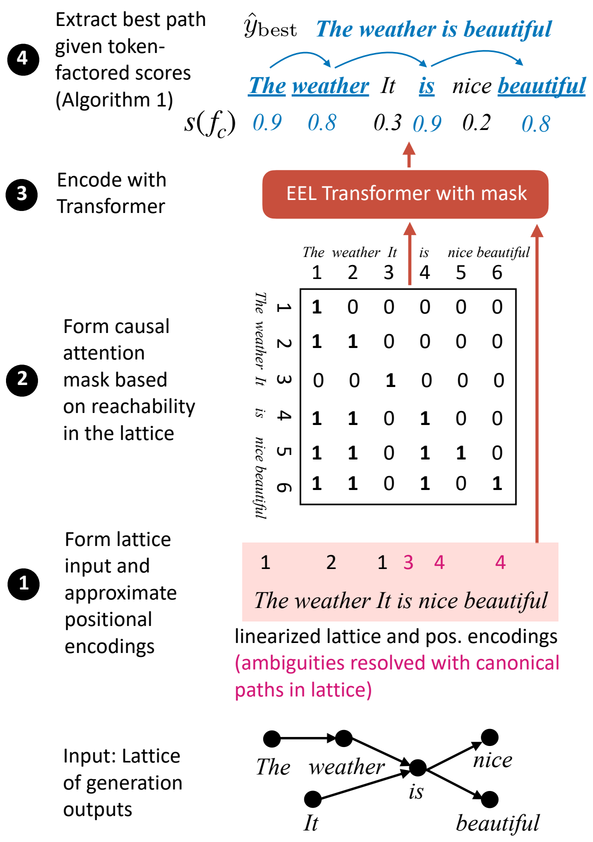
On-the-fly reranking approaches like RankGen (Krishna et al., 2022) can also factor into a TFR. Furthermore, the sums of two TFRs (or TFRs) will also be usable in our pipeline, allowing us to combine multiple TFRs or use a weighted composition of other token-level scores.

Definition 2 (ensemble TFR; E-TFR)**.** *Let S be a token-factored reranker. Let M be another TFR where $M(\mathbf{x}, \hat{\mathbf{y}}) = \sum_{k=1}^{n} \log p(\hat{y}_k \mid \mathbf{x}, \hat{\mathbf{y}}_{<k})$, where p is the probability under the base model. Define the **ensemble TFR** $S_e(\mathbf{x}, \hat{\mathbf{y}}) = S(\mathbf{x}, \hat{\mathbf{y}}) + \lambda M(\mathbf{x}, \hat{\mathbf{y}})$.*

E-TFR ensembles the TFR with model score for better performance in some settings. In this paper, E-TFR specifically refers to this combination ($\lambda = 0.75$ in this work), but note that TFRs can be defined or ensembled from any set of models or functions that meet Definition 1.

3.2 Reranking Lattices

If we assume that efficient computation of f_c is possible then it becomes easy to optimize S over H when S is a TFR. We can use dynamic programming to extract $\mathbf{y}_{\text{best}}$ from the input lattice. Algorithm 1 describes the procedure, which is essentially the Viterbi algorithm with no transition scores. We start with lists *flat* which contains all $v \in V$, sorted topologically, and *ends*, which contains the indices in flat of all nodes in V with no next nodes. We iterate through *flat*, choosing the highest scoring path leading to each node based on whichever previous node has the highest scoring path. Finally, we return the path ending with the highest overall score. In practice it may be necessary to normalize ending scores depending on the function s, for example we divide each path score by the length of the path before choosing the best scoring path.



Extract best path given token-factored scores (Algorithm 1)
$\hat{y}_{\text{best}}$ The weather is beautiful
The weather It is nice beautiful
$s(f_c)$ 0.9 0.8 0.3 0.9 0.2 0.8
Encode with Transformer
EEL Transformer with mask
Form causal attention mask based on reachability in the lattice
Form lattice input and approximate positional encodings
1 2 1 3 4 4
The weather It is nice beautiful
linearized lattice and pos. encodings
(ambiguities resolved with canonical paths in lattice)
Input: Lattice of generation outputs
The weather It is nice beautiful

	The 1	weather 2	It 3	is 4	nice 5	beautiful 6
The 1	1	0	0	0	0	0
weather 2	1	1	0	0	0	0
It 3	0	0	1	0	0	0
is 4	1	1	0	1	0	0
nice 5	1	1	0	1	1	0
beautiful 6	1	1	0	1	0	1

Figure 2: Detailed overview of EEL, starting from bottom, using a single context mask and causal reachability.

3.3 Diverse Path Selection

In addition to estimating a best path, our approach also supports being able to extract a diverse set of k paths while still optimizing for $S(\hat{\mathbf{y}})$. This can be accomplished by running Algorithm 1 k times, and at the end of each round, modifying the score of each node $s(y_i) = s(y_i) - w \cdot o(y_i)$, where w is a diversity weight hyper-parameter, and $o(y_i)$ corresponds to the number of times a given token has occurred in previous *bestpath* results.

4 Efficient Encoding of Lattices (EEL)

In this section, we discuss the computation of f_c: how can we efficiently compute Transformer encodings for a lattice? Across decoding methods, as the total number of lattice nodes is often exponentially fewer than the total number of tokens in H, being able to rerank all the candidates by only encoding each lattice token once can lead to a huge reduction of computation. Thus, our goal with EEL is to compute embeddings for all nodes in a set of hypotheses H represented by a directed graph (lattice) $G = (V, E)$ encoding all candidates in H. V is the set of all expanded nodes $v_i \in V$, and E the set of directed edges $e_{i,j}$ which represents $v_i \in V$ preceding $v_j \in V$ in some candidate in H (we'll use v_i interchangeably with its associated token y_i).

Figure 2 shows the overall steps of our solution, which we now describe in detail.

4.1 Constructing Inputs

Transformers primarily take three inputs: 1. **Token Ids**, laid out consecutively in an input canvas; 2. **Position Ids**, corresponding to each token's position in text; 3. **Attention Mask**, a mask that dictates which tokens can attend to each other.

To encode a lattice consistently with individual Transformer passes, the tokens each token attends to and the position ids of those tokens should be the same as if just part of a single sequence. As Transformers can only encode one canvas (context window) of tokens at a time, we accordingly need to lay the lattice tokens onto a single token id canvas. For position ids, the default format in most pre-trained Transformers, such as GPT-3, is **absolute** position ids, where the index of a token t in a sequence of inputs is simply its token id, corresponding directly to its location in the input.

In order to make lattice position embeddings compatible, we use a **canonical** position id strategy, where we do a depth-first traversal of all nodes in G with respect to node model probability. Assume v_i is the first node to precede v_k in the traversal and edge $e_{i,k}$ exists; then $pos(v_k) = pos(v_i) + 1$ (see Step 1 of Figure 2). Alternate position id schemes include computing the longest or shortest path from the start (Sperber et al., 2019), or a random depth-first traversal, though we generally don't note any empirical differences between these approaches. Our method gives slightly better outcomes in certain settings when a lattice doesn't fully fit within a canvas, as it serves as an implicit truncation of low model probability paths.

4.2 Masking

To give individual tokens information about their context, Transformer encoders typically allow all input tokens attend to each other. Since our TFR S is trained on normal text data, where a token expects to "see" each other position id exactly once, simply passing in linearized lattice inputs leads to large degradation, as tokens that don't share paths can still attend to each other. We formulate all masks as $n \times n$ matrices where $n = |V|$ and each index corresponds to a node in V based on position in the canonical canvas. The non-zero values of row i indicate which tokens y_i can attend to. We below walk step-by-step through several mask types to reach a strategy with the lowest degradation (ablations in Table 3):

Causal Reachability: We first construct an $n \times n$ adjacency matrix A such $A_{ij} = 1$ if $e_{i,j} \in E$, else 0. We can then obtain a **causal reachability** mask using the following reachability equation:

$$C = \min(I_n + \sum_{i=1}^{l} (A^T)^i, \mathbf{1}) \quad (1)$$

Where I_n is an identity matrix, l is the length of the longest hypothesis in H, and the min operation causes all values to be either 0 or 1. Such a mask prevents tokens that aren't in any common hypotheses from attending to each other, but connects a token to all tokens that come before it (we'll call these *contexts*) for all paths in H. Note that A, and thus C are one-directional to match the causal formulation of TFRs.

We can obtain a mask for a bidirectional TFR by replacing A^T in Equation 1 with $A + A^T$. However, we empirically found that reachability in both directions results in more degradation in the TFR, resulting in our decision to use causal TFRs. Causal TFRs enable lossless encoding for a lattice with no reentrancies. There can be degradation in graphs with reentrancies such as those produced by lattice decoding, due to multiple contexts or mismatched canonical position ids.

Single Context To constrain sources of degradation even further, we can limit A to only have a single randomly selected 1 per row, which would translate to each token only being able to look back at a single path: in other words a **single context** mask C^*. This strategy is equivalent to reranking all hypotheses encoded within a random subgraph $G^* = (V, E^*)$, where $E^* \subseteq E$ such that only one directed edge $e_{i,j}$ exists from any node v_i. Thus, remaining degradation is limited to the paths in G not encoded in G^*. In Figure 2, this manifests as *beautiful* not attending to *It*.

Few Mask We can improve our approximation with higher computational cost using a **few-mask** variant of EEL. With the same input and position ids, we run the EEL pipeline with "single context" with m different starting random adjacency A instances. This allows us to then obtain m different extracted best paths based on initialization, from which we can then choose an overall best scoring path based on the normalized path scores of the m best paths. This allows more potential paths in the lattice to be explored without suffering context-related degradation. Note that while this approach leads to the best performance, single context masking is the best in terms of efficiency.

5 Experimental Setup

5.1 Settings

To demonstrate the robustness of our approach, we run experiments on a variety of different base tasks, with lattices generated in different conditions, and with 3 different reranking models that fit our criterion. Our base tasks are as follows:

Machine translation We generate lattices (using an mBART model (Liu et al., 2020)) in 3 machine translation settings from WMT-2019: English to German (EN-DE), English to Russian (EN-RU), and French to English (FR-EN) (Barrault et al., 2019).

Table-to-text We use generated candidate sets from a BART model from Ribeiro et al. (2021) on examples from the WebNLG 2017 challenge (Gardent et al., 2017).

Document summarization We also generate a set on document summarization, using a BART model (Lewis et al., 2020) and XSum data (Narayan et al., 2018).

We experiment with 3 downstream objectives: COMET (Rei et al., 2020) quality estimation of machine translations, PARENT score (Dhingra et al., 2019) precision for quality estimation of table-to-text generation, and number of unique nouns[2] using a part-of-speech tagger. We train TFRs for each respectively; see Section 5.2 for model details.

Implementation Details We generate sets of 500 lattices each with different generation conditions, across 5 settings (3 MT settings, summarization, table-to-text), parallel for 3 decoding methods:

- **lattice decoding** (LATT), with a beam width of 4[3] and RCB recombination (based on n-gram matching during decoding) (Xu et al., 2022).

[2]We choose this somewhat synthetic setting to provide a reranking setting for summarization and to show that EEL works on diverse types of downstream metrics, not just continuous scores from 0-1.

[3]Xu et al. (2022) call this an "equivalent beam size." We run with a lower value than our beam search because their method is slower than beam search due to poorer parallelism.

- beam search, with **beam-width 12** (B-12), as a low generation cost baseline
- beam search, with **beam-width 50** (B-50), which we find to have a comparable wall clock generation time to LATT

For beam search, we use the Hugging Face generate() API to generate candidates, and we use the corresponding model probabilities returned alongside generation candidates as model scores.

5.2 TFR Training

We fine-tune three TFR (Section 3.1) models.

MT-TFR Downstream objective: COMET score, for reference-based machine translation quality estimation. MT-TFR uses an XLM-RoBERTa-Base (Conneau et al., 2020) encoder to encode both source and hypothesis sentences, and is fine-tuned on COMET scores generated on the multi-lingual WMT17-21 direct assessment sets (Barrault et al., 2021). Note that we are estimating a reference-based metric in a reference-free way similar to COMET-QE (Rei et al., 2021).

TAB-TFR Downstream objective: PARENT precision, for reference-based table-to-text generation quality estimation. We generate 16 candidates using a T5-large generation model (Wang et al., 2021) for each examples in the WebNLG 2017 training set, and obtain PARENT precision scores for each of these to acquire our training set labels. For the TFR encoder, we use a BART-Base encoder-decoder model, using the encoder to encode the source, and the decoder to encode the candidates.

NOUN-TFR Downstream objective: number of unique nouns. The model is trained using an XLM-RoBERTa-Base (Conneau et al., 2020) encoder on a set of 100,000 English sentences from the news-test-14 dataset. We fine-tune it to predict how many unique tokens with the NN, NNS, or NNP POS tags are passed into the candidate sentence, using an NLTK (Bird et al., 2009) POS tagger as our gold labeler (we normalize this by dividing by 10). Note that, while we believe Noun-TFR correlates with diversity and informativeness, as more nouns may indicate more specific content around entities, we are not setting it up as a gold standard for summarization quality; it's designed more as a testbed for our approach.

5.3 TFRs vs Non-TFR Rerankers

To confirm that token-factored rerankers do not have worse downstream performance, we validate TFR models against COMET-QE (Rei et al., 2021), a non-token factored reranking metric. When refactoring COMET-QE to be token-factored, and fine-tuning, we're able to reproduce reranking performance when ensembled with model score (see Section 7): 0.699 (French to English), 0.598 (English to German), and 0.650 (English to Russian) respectively (see more in A.3). With a TFR model, we find downstream performance to be 0.698 (French to English), 0.576 (English to German), and 0.614 (English to Russian); on average, only 0.02 COMET score worse. In other words, **TFRs don't significantly fall behind other Transformer-based reranking models**.

5.4 Evaluating Efficiency

Wall clock time: The main costs in a reranking system are the *generation time* (GEN) of the decoding method, and the *reranking time* (RRK) to process and rerank the input. We report these in the form of wall clock time as a practical heuristic.

Candidates / Nodes: To control for differences in architecture, parallelism, and other factors that influence wall-clock time, we also report a measure we call *candidates per nodes* (C/N). Specifically, for a given approach, we measure how many candidates it encodes, and the number of nodes (tokens) the TFR needs to rerank the candidate set.

6 Intrinsic Evaluation: Finding Hypotheses with High Reranker Scores

We first evaluate how closely the TFR scores of EEL selections come to matching the quality of the oracle top-1 hypotheses $\mathbf{y}_{\text{best}}$ with respect to TFR model $S(\mathbf{x}, \hat{\mathbf{y}})$. We compute an upper bound for this by calculating TFR scores on **exploded** lattice candidate sets: enumerating the exponentially-large set of complete paths and scoring each individually. Our EXHAUSTIVE numbers are the average top-1 TFR score (not the downstream metric score) across the entire exploded set.

As baselines, we rerank on a randomly sampled sets of 1 (RAND), 8 (TFR-8-SAMP), and 32 (TFR-32-SAMP) from exploded sets. For EEL approaches, we test a single mask (EEL 1-MASK) and few-mask (EEL 8-MASK) approach. We report these results across decoding methods in Table 1 alongside efficiency measurements on the MT-TFR FR-EN

	Method	MT-TFR reranker score			NOUN-TFR score		TAB-TFR score	Efficiency (Fr-En) ratio↑		sec ↓
		FR-EN	EN-DE	EN-RU	FR-EN	XSUM	WEBNLG	C/N	RRK	GEN
LATT	RAND	.605	.582	.631	.765	.831	.511	.020	.051	4.135±1.595
	TFR-8-SAMP	.690	.690	.792	.863	1.022	.589	.020	.167	
	TFR-32-SAMP	.716	.719	.838	.903	1.097	.627	.020	.695	
	EEL 1-MASK	.695	.700	.836	.922	1.118	.653	**3.304**	**.091**	
	EEL 8-MASK	**.720**	**.731**	**.862**	**.934**	**1.142**	**.657**	0.413	.252	
	EXHAUSTIVE	.743	.748	.883	.945	1.178	.692	.025	17.950	
B-12	RAND	.629	.616	.678	.751	.734	.582	.024	0	1.280±.260
	EEL 1-MASK	.687	.684	.783	.812	.848	.641	**.064**	**.078**	
	EXHAUSTIVE	.687	.684	.783	.812	.848	.641	.024	.248	
B-50	RAND	.618	.611	.640	.752	.733	.581	.025	0	3.670±.960
	EEL 1-MASK	.700	.707	.805	.845	.908	.651	**.075**	**.120**	
	EXHAUSTIVE	.706	.710	.810	.850	.909	.653	.025	1.051	

Table 1: Base task results, grouped by decoding method of input lattices. EEL 1-MASK and EEL 8-MASK strongly improve candidate/node (C/N) efficiency and demonstrate notable speedups compared to baselines. Across settings EEL 8-MASK comes close to matching the much slower LATT exhaustive, even outperforming B-50 exhaustive, demonstrating that EEL comes with low degradation for high efficiency gain.

setting, which we noted to be representative of performance across settings.

EEL universally provides strong efficiency boosts: While the lattice EXHAUSTIVE reranking always performs the best, it's incredibly slow, taking 17.95s on average to rerank a single candidate set. EEL 1-MASK, meanwhile, encodes the same candidate set roughly **200 times faster**, at only .091s, and even EEL 8-MASK only takes .252s. The candidate / node ratio of 3.304 (EEL 1-MASK) and .413 EEL 1-MASK, compared to baseline C/N efficiencies of .025, further demonstrates how much computation EEL saves on large lattices. Even on B-50 and B-12, we see strong reranking efficiency improvements, with 3x and 2.67x better C/N efficiency, and notable rerank time (RRK) boosts.

EEL selections nearly match ORACLE: While EEL, especially with large input lattices, can improve encoding efficiency by orders of magnitude, it does so while on average still coming very close to matching oracle top-1 candidates (see Appendix B for degradation analysis). We find EEL-8-MASK on LATT to be the most effective overall for efficiently approximating lattice decoding lattice TFR score, as outside of the unfeasible LATT EXHAUSTIVE, **EEL 8-MASK obtains the highest TFR scores in every setting**, outperforming baselines and even B-50 EXHAUSTIVE. Furthermore, EEL, applied on B-12, and B-50 comes with zero and near-zero degradation respectively.

Method	COMET			NOUN	PRT-P
	FR-EN	EN-DE	EN-RU	XSUM	WNLG
RAND-B50	.654	.541	.482	7.01	.573
RAND-LATT	.598	.419	.445	8.07	.452
B50-PROB	.680	.564	.518	–	.623
LATT-PROB	.660	.493	.545	–	.654
EEL-W 1-MASK	.673	.541	.592*	10.80*	.667
EEL-W 8-MASK	.674	.545	**.593***	**11.05***	**.670**
B50-E-TFR	**.689**	**.576**	.562	8.63	.664
Oracle values					
LATT-E-TFR	.698	.574	.614	11.24	.691
B50-ORACLE	.761	.664	.702	8.74	.778
LATT-ORACLE	.789	.677	.775	11.40	.825

Table 2: Downstream score results, grouped by systems which have comparable computational requirements. We find that our best method (EEL-W 8-MASK) achieves strong performance. The bottom rows report oracle reranking approaches, showing that lattices have much higher oracle values and that our rerankers can sometimes come close (especially on NOUN-XSum). * indicates statistically significant improvement over B50-E-TFR using a paired bootstrap test, $p < 0.05$.

7 Downstream Evaluation

While EEL's algorithmic objective is to find the best TFR scoring candidate in a lattice, the high-level goal, assuming a good TFR reranker, is to *find a high downstream scoring candidate*. To measure downstream success, we compute results with respect to the original metrics that we distill our TFR models on (COMET, NLTK POS-Tagger, PARENT Precision).

Specifically, we assess whether EEL can enable lattice decoding lattices to efficiently outperform comparable beam search (B50) reranking approaches. For the COMET and PRT-P settings, we use E-TFR, a weighted sum of TFR and token-level

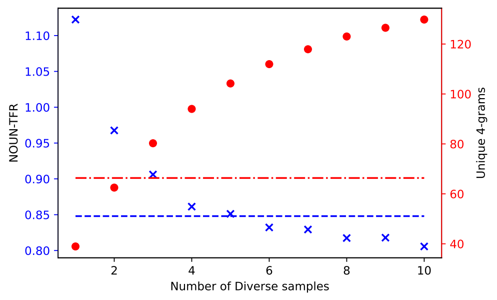


Figure 3: Scatter plot, showing, for XSUM NOUN-TFR (blue) and Unique 4-grams (red) for successive diverse samples. Red (Unique 4-grams) and Blue ORACLE NOUN-TFR are B-12 baselines. Within 3 diverse samples, EEL outperforms the baselines in both NOUN-TFR and Unique 4-grams.

	Method	MT-TFR FR-EN	TAB-TFR XSUM
	RANDOM	.605	.831
	ORACLE	.743	1.178
	EEL FULL CONTEXT	.695	1.120
	EEL DEFAULT POS	.655	.989
MULTI	EEL 1-MASK	.695	1.118
MULTI	EEL 8-MASK	.720	1.142
MULTI	EEL 16-MASK	.724	1.148

Table 3: Ablation that validates several choices in EEL pipeline, including posids, contexts, and more masks.

model score (PROB) (see Section 3.1).

Table 2 reports our results. In addition to E-TFR EEL on lattice decoding (EEL-W 1-MASK, EEL-W 8-MASK), we report model score reranking baselines (B50-PROB, LATT-PROB), a weighted naive TFR (B50-E-TFR, LATT-E-TFR) upper bound, and downstream ORACLE results. Note, ORACLE results that need references (COMET and PRT-P) assume access to human annotation and are unrealistic upper bounds.

EEL and TFRs can enable LATT to efficiently outperform B-50: Across settings, the gap between downstream E-TFR performance and EEL is minimal: for EEL 8-MASK, only .024 (FR-EN), .029 (FR-EN), .019 (EN-RU), .019 (XSUM), and .021 (WNLG), compared to much larger gaps from random baselines. In other words, EEL *is always quite close to gold TFR reranking selections even on downstream metrics*. In settings where the reranking capabilities of lattice decoding (LATT-ORACLE) strongly outperforms ORACLE capabilities of B50-ORACLE, such as EN-RU (.073), XSUM (2.66), and WNLG(.047), where LATT-E-TFR strongly outperforms B50-E-TFR, **EEL on LATT likewise outperforms the reranking capabilities of beam search**. Note that, on settings such as FR-EN where the oracle gap is a more narrow .028, EEL still outperforms LATT-PROB, but not B50. Recall from Table 1, however, that this isn't apples-to-apples: EEL approaches achieve comparable / best performance while being several times faster than exhaustive reranking alternatives.

8 Analysis

Diversity We further explore whether EEL can sample a diverse set of candidates (see Section 3.3) that still optimize for TFR metrics. We examine this trade-off in Figure 3, which shows diversity and NOUN-TFR of diverse samples taken at n steps (x-axis) aggregated across several examples on XSUM. While overall diversity increases rapidly, this comes with a trade-off of lower TFR scores, eventually selecting candidates with roughly average NOUN-TFR score (the random baseline is .831).

That said, we find **diverse sampling works to produce several diverse samples**. Our approach is able to rapidly obtain a highly diverse set of candidates before the NOUN-TFR score of samples eventually converges with the random baseline. BEAM-12, across all 12 candidates, has an average diversity of 66.35 4-grams, and .848 ORACLE NOUN-TFR. at 3 samples, we have good diversity and good quality, while the first 5 diverse samples all outperform the B-12 NOUN-TFR and diversity baselines. It's worth noting that the diversity weight is a hyperparameter, and can be adjusted to allow for a slower trade-off.

Ablations Table 3 shows ablations of different parts of our method. We validate that canonical position ids, and single-context masks are necessary for the success of our approach. We further note that while allowing a token to see all valid contexts has similar performance to EEL 1-MASK, it doesn't scale to multi-mask versions, and comes with correctness downsides. We also find there to be diminishing returns past 8 mask runs.

9 Related Work

Reranking and efficiency There is substantial past work on reranking, including significant recent work using Transformers for it (Rei et al., 2021). Some models can be expressed as TFRs (Krishna et al., 2022); however, others like Rei et al. (2021) use approaches like pooling of representations of the hypothesis before additional preprocessing. We do not have an approximation for such techniques; we show that TFRs can be successfully trained in this setting, but other approximations beyond TFRs would be a useful contribution of future research.

Older machine translation work takes advantage of the structure of n-gram language models to do "reranking" on the fly, either in hierarchical models (Li and Khudanpur, 2012) or phrase-based decoding (Koehn, 2004). However, recent work on both evaluation (Sellam et al., 2020; Rei et al., 2020) and reranking (Rei et al., 2021) show how effective Transformers are, suggesting that approximating these is more promising than devising rerankers with inherently more restrictive factorizations.

Background: Lattice Encoding Work in the speech recognition community (Pandey et al., 2021; Li et al., 2021) encodes lattices of speech recognition model decoding outputs with LSTMs to decode better candidates in second pass. Other work examines the ability to encode lattices with self-attentional models (Sperber et al., 2019), for the purpose of augmented text generation (Zhang et al., 2019a; Lai et al., 2021), albeit using models trained on datasets of lattices. As these approaches often require lattice-specific training, and often don't account for degradation, due to alternate, coarser task formulations, they are much more constrained, and not suitable for our setting.

Controlled generation While we focus on reranking for better generation "quality", our work relates to tasks where one may want to re-rank with respect to some control attribute. Prior work examines the ability to adjust output logits specific to controls (Yang and Klein, 2021), MCTS (Leblond et al., 2021) as well as control classifier guided decoding (Dathathri et al., 2020).

10 Conclusion

Across a variety of settings, downstream objectives, and decoding methods, we consistently find EEL to provide high quality selections with low degradation and substantial speedup compared to exhaustive top-1 reranking output. We further demonstrate the capability of our approach to select multiple diverse outputs while still optimizing for TFR scores. By proposing a method that can efficiently encode lattices of large candidate sets, through the combination of EEL and TFR models, we thus demonstrate the capability of reranking to be more effective and efficient than what was previously possible with naïve approaches.

Limitations

While our approach is designed to be as broadly applicable as possible, an inherent limitation of our work is that it depends on the usage of causal TFR-style models, which, though architecturally similar to many existing pre-trained models, require hyperparameter search and fine-tuning to replace non-TFRs on downstream tasks. While we find evidence that such models are as capable as other rerankers, and we believe TFRs can be a default design choice going forward with little loss, this extra requirement may still be a barrier for the broad adoption of our approach.

More broadly, our approach is designed for settings where *some* reranker is available. If it is not possible to train a robust reranker, for example in a low data setting or a setting where evaluation relies entirely on human judgments that cannot be reproduced by models, our approach cannot be applied. However, we believe that the growth in learned functions of human judgments as part of RL from human feedback loops provides a promising avenue to roll out our approach to new settings.

Our experiments were carefully chosen to represent the capabilities of our models with several base tasks and several reranking objectives. We didn't, however, explore certain domains involving attribute control such as formality or simplicity, choosing instead to explore more quality-based downstream exploration. We showed the applicability of our approach when reranking outputs on languages other than English, but further results on languages with different typological characteristics may show different trends.

While our work already provides strong speedups both with candidate sets from lattice and beam search decoding, these speedups become even more valuable for approaches that combine multiple rerankers, which have been shown to potentially lead to further improvements in reranking (Fernandes et al., 2022). While we explore this partially in the form of ensembled EEL with model probabilities, more exploration on EEL for multiple rerankers may be valuable.

Acknowledgments

This work was supported by NSF CAREER Award IIS-2145280, a grant from Open Philanthropy, a gift from Salesforce, Inc., a gift from Amazon, and a gift from Adobe. Thanks to Darcey Riley, André Martins, Ben Peters, Ricardo Rei, António Farinhas, Perez Ogayo, and José Souza for discussion and feedback during the completion of this paper. Thanks as well to the anonymous reviewers for their helpful feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loic Barrault, Ondrej Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussa, Christian Federmann, Mark Fishel, Alexander Fraser, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Tom Kocmi, Andre Martins, Makoto Morishita, and Christof Monz, editors. 2021. *Proceedings of the Sixth Conference on Machine Translation*. Association for Computational Linguistics, Online.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Sumanta Bhattacharyya, Amirmohammad Rooshenas, Subhajit Naskar, Simeng Sun, Mohit Iyyer, and Andrew McCallum. 2021. Energy-based reranking: Improving neural machine translation using energy-based models. 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 4528–4537, Online. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 858–866, Los Angeles, California. Association for Computational Linguistics.

Patrick Fernandes, António Farinhas, Ricardo Rei, José De Souza, Perez Ogayo, Graham Neubig, and Andre Martins. 2022. Quality-aware decoding for neural machine translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412, Seattle, United States. Association for Computational Linguistics.

Markus Freitag, David Grangier, Qijun Tan, and Bowen Liang. 2022. High quality rather than high model probability: Minimum Bayes risk decoding with neural metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. 2022. RankGen: Improving Text Generation with Large Ranking Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2021. Lattice-BERT: Leveraging multi-granularity representations in Chinese pretrained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1716–1731, Online. Association for Computational Linguistics.

Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislar, Lespiau Jean-Baptiste, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. 2021. Machine translation decoding beyond beam search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8410–8434, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

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

Ke Li, Daniel Povey, and Sanjeev Khudanpur. 2021. A parallelizable lattice rescoring strategy with neural language models. *CoRR*, abs/2103.05081.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization.

Zhifei Li and Sanjeev Khudanpur. 2012. Forest reranking for machine translation with the perceptron algorithm.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pretraining for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2022. Typical decoding for natural language generation. *CoRR*, abs/2202.00666.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Prabhat Pandey, Sergio Duarte Torres, Ali Orkan Bayer, Ankur Gandhe, and Volker Leutnant. 2021. Lattention: Lattice-attention in asr rescoring. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7877–7881.

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, Dublin, Ireland. Association for Computational Linguistics.

Ricardo Rei, Ana C Farinha, Chrysoula Zerva, Daan van Stigt, Craig Stewart, Pedro Ramos, Taisiya Glushkova, André F. T. Martins, and Alon Lavie. 2021. Are references really needed? unbabel-IST 2021 submission for the metrics shared task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 1030–1040, Online. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Kaiqiang Song, Bingqing Wang, Zhe Feng, and Fei Liu. 2021. A new approach to overgenerating and scoring abstractive summaries. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1392–1404, Online. Association for Computational Linguistics.

Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel. 2019. Self-attentional models for lattice inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1185–1197, Florence, Italy. Association for Computational Linguistics.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Qingyun Wang, Semih Yavuz, Xi Victoria Lin, Heng Ji, and Nazneen Rajani. 2021. Stage-wise fine-tuning for graph-to-text generation. In *Proceedings of the ACL-IJCNLP 2021 Student Research Workshop*, pages 16–22, Online. Association for Computational Linguistics.

Jiacheng Xu, Siddhartha Jonnalagadda, and Greg Durrett. 2022. Massive-scale decoding for text generation using lattices. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4659–4676, Seattle, United States. Association for Computational Linguistics.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. 2019a. Lattice transformer for speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6475–6484, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

A TFR Model Details

A.1 Model Architecture

While the over-arching idea of TFRs stays the same, our TFR models all differ slightly in the setup of the final hidden layers. For the NOUN-TFR model, we actually don't encode the input, as it isn't necessary for the task, and thus, for a given candidate, we only run the feedforward network at the end on individual token hidden states, as opposed to the concatenated vector with the product and difference between it and the pooled input state.

For the MT-TFR and the TAB-TFR model, we follow the output format described in Section 5.2. The only difference is that for MT-TFR, we use the same encoder model for both the input and the output, and for TAB-TFR we follow an encoder-decoder architecture, where the input is acquired by the encoder, and the output by the decoder, both which are fine-tuned separately during training. We do this as the structure of data is more divergent in table-to-text, and thus reasoned that separate encodings of input and output would lead to greater success.

Model	Raw Val Size	Val Size
MT EN-DE	294,497	500
MT FR-EN	156,121	500
MT EN-RU	265,807	500
XSum	11,334	500
WebNLG	1863	500

Table 4: Information on evaluation sets used for our results sections. We randomly sample 500 examples to use for our evaluation from different base task datasets to use for our exploration. MT data comes from the WMT-2019 NewsTest Corpus, XSum comes from the XSum test set, and WebNLG from the WebNLG Test Set

A.2 Training

For training, as a rough heuristic of how well a model would perform as a re-ranker, we optimized for correlation metrics between TFR model scores and downstream metric scores (Pearson, Spearman, and Kendall's Tau correlations). For our MT-TFR validation set, our model reached .879 Pearson, .854 Spearman, and .690 Kendall correlation with COMET score. For our NOUN-TFR model, our model reached .971 Perason, .940 Spearman, and .800 Kendall correlations with gold NLTK noun count. Lastly, for our TAB-TFR model, our model reached .646 Pearson, .632 Spearman, and .470 Kendall correlation. Interestingly, though the correlation metrics weren't good for the TAB-TFR model, the downstream re-ranking still worked well, outperforming model score with similar margins to MT-TFR. We trained each model for an average of roughly 12 hours of training for the TAB-TFR and MT-TFR models, and roughly 3 hours of training for the NOUN-TFR model.

Note the size of the train / validation set for MT-TFR were 958,122/74,522, for TAB-TFR were 260,668/28,964, and for NOUN-TFR were 90,000/10,001.

A.3 Format Validation

In order to validate that token-factoring doesn't inherently lead to degradations of a model, we test a token-factored version of COMET-QE (2020), a referenceless re-ranking metric. On a validation set, we measure its correlations on randomly sampled WMT Direct Assessments scores to be .479 Spearman, .431 Pearson, and .333 Kendall's Tau correlations. We then modified the model to be token-factored (re-ordered pooling and feed-forward pass), and fine-tuned the model on Direct Assessments data. We found that we were able to reproduce these correlations with .477 Spearman, .431 Pearson, and .333 Kendall's Tau correlations. More-over we measured similar re-ranking performance on a set of beam search 50 candidate sets, thus validating that token-factoring re-rankers doesn't inherently lead to better or worse re-ranking.

B Degradation Analysis

By design, on any token-level task, or task that can be decomposed into a token-level task (we examine reranking), EEL can perform perfectly on lattices without recombination, while offering greater efficiency. This is because the Transformer lattice encoding scheme with causal models in these circumstances is mathematically equivalent to evaluating each path separately: the use of masking and modified positional encodings enables the computation to exactly replicate the standard setting. While we use beam search lattices to demonstrate this, it can apply to a variety of sequence-modeling tasks with Transformers.

Note the graphs that we look at have 163 (B-12), 551 (B-50), and 431 (LATT) nodes on average, with lattice decoding lattices generally having 15+ recombination points, for an average sequence length of around 40 tokens. The lattice-decoding lattices we examine encode between 1000-4000 candidates per lattice .

Sources of remaining degradation On lattices with recombination, we still experience some degradation. We can pinpoint remaining degradation as observed in results to 2 sources. Firstly, as masks are randomly generated, there's a chance (increasingly small with each new mask) chance that the true top-1 candidate isn't covered. By the nature of our single-context masks, and how they cover every node in a lattice, its often the case that even if the true top-1 path isn't traced by a mask connection, our approach is likely to extract something similar. Additionally, due to recombination, the absolute position ids we encode tokens with will occasionally mismatch at certain points, leading to slightly different scores than if EEL were not used.

C Robustness

TFR-n-samp / EEL We control for randomness in several of our experiments as follows. For the TFR-N-SAMP rows, we sample 10,000 times from each input lattice, averaging those values to get our final results; we note that across robust runs the numbers don't change up to the 3 decimal precision that we report. Likewise, for the EEL numbers, we randomly generate 32 masks, and then sample results 1000 times respectively from those masks. It's further worth noting that past 16 random masks, the 1-best result often repeats from earlier versions (hence the diminishing returns in our ablation study), so we reason this to be a reasonable cut-off. We follow a similar procedure for the downstream table.

Timing While the absolute magnitudes of our timing results are subject to variance as a result of varying compute setting and load, we run our timing results several times at different points in time, and note that the relative ranking and overall patterns remain similar across runs. We run these experiments as 10 runs over the entire sets of 500 which we report results on. We apply batching evenly across our approaches and baselines, and we believe that the comparisons are representative of actual performance deltas, as further validated by our C/N results.

D Responsible NLP Checklist

D.1 Artifacts

We use several scientific artifacts in this work.

Data We use the WMT datasets (2017-2021, licensed under CC-BY-SA-4.0), the WebNLG Dataset (2017, licensed under CC BY-NC-SA 4.0), and the XSUM dataset (MIT License).

Code We use open-source code from the COMET repository (APACHE License 2.0), PARENT score (Dhingra et al., 2019), and lattice generation code from the original lattice decoding paper (Xu et al., 2022).

Generation Models For generations models we use bart-large-xsum, facebook/mbart-large-50-many-to-one-mmt, facebook/mbart-large-50-one-to-many-mmt, and bart-large fine-tuned on web-nlg data from (Ribeiro et al., 2021).

D.2 Model Hyperparameters / Infrastructure

We run all of our experiments on a single PNY NVIDIA Quadro RTX 8000 GPU.

As we found it to work well across models, for training our TFR models, we use the following common hyperparameters: **encoder learning rate**: 1e-5, **learning rate**: 3.1e-5, **layerwise decay**: 0.9, **dropout:** 0.1, **number of encoder frozen epochs**: 0.3, and a **batch size** of 8.

We run our NOUN-TFR model for 40000 train steps, our MT-TFR model for 140000 train steps, and our TAB-TFR model for 190000 train steps. This came out to roughly 28 GPU hours total to train the models that we used. Note that beyond manual adjustments, we don't use any sort of hyperparameter grid search.

We further report another approximately 8 GPU hours for generating and reranking the lattices we used for our evaluation.

D.3 Libraries Used

We use version 3.7 of the NLTK python library (Bird et al., 2009) to extract part of speech tags to use for our NOUN-TFR setting.

E Generation Output

Label	COMET	Text
Source	-	Depuis longtemps, plusieurs éléments mettent en péril l'économie américaine :
Reference	-	A number of worrying factors about the US economy have been around for a long time:
E-TFR #1	.683	For a long time, several factors have threatened the US economy:
E-TFR #2	.683	For a long time, several factors have threatened the US economy:
E-TFR #3	.669	For a long time, several factors have threatened the American economy:
model score rerank #1	.420	The U.S. economy has long been threatened by several factors:
model score rerank #2	.420	The U.S. economy has long been threatened by several factors:
model score rerank #3	.683	For a long time, several factors have threatened the US economy:
oracle over lattice	.749	For a long time, a number of factors have been threatening the US economy:

Table 5: Example 1, French to English, Reranked on LATT

Label	COMET	Text
Source	-	Une enquête d'opinion menée de longue date à travers l'Europe permet de relier les deux.
Reference	-	A pan-European opinion survey, which has been carried out for many years, allows us to relate the two.
E-TFR #1	.533	A long-standing opinion poll across Europe makes it possible to link the two.
E-TFR #2	.549	A long-term opinion poll across Europe makes it possible to link the two.
E-TFR #3	.374	A long-standing public opinion survey across Europe links the two.
model score rerank #1	.207	A long-standing opinion poll across Europe links the two.
model score rerank #2	.533	A long-standing opinion poll across Europe makes it possible to link the two.
model score rerank #3	.549	A long-term opinion poll across Europe makes it possible to link the two.
oracle over lattice	.733	A long-term opinion poll conducted across Europe makes it possible to link these two.

Table 6: Example 2, French to English, Reranked on LATT

Label	PARENT-P	Text
Source	-	<H> 250 Delaware Avenue <R> architectural Style <T> Postmodern architecture
Reference	-	250 Delaware Avenue has the Postmodern style of architecture.
E-TFR #1	.883	The architecture style of 250 Delaware Avenue is Postmodern.
E-TFR #2	.519	250 Delaware Avenue is in the Postmodern architectural style.
E-TFR #3	.589	250 Delaware Avenue is located in Postmodern architecture style.
model score rerank #1	.519	250 Delaware Avenue is in the Postmodern architectural style.
model score rerank #2	.883	The architecture style of 250 Delaware Avenue is Postmodern.
model score rerank #3	.389	250 Delaware Avenue is in the postmodern architectural style.
oracle over lattice	1.0	The architectural style of 250 Delaware Avenue is Postmodern.

Table 7: Example 3, Table to Text, Reranked on LATT

Label	PARENT-P	Text
Source	-	<H> Bakso <R> ingredient <T> Celery <H> Celery <R> family <T> Apiaceae
Reference	-	Celery is a member of the Apiaceae family and is an ingredient of Bakso.
E-TFR #1	.857	Celery is a member of the Apiaceae family and is an ingredient in Bakso.
E-TFR #2	.668	Celery is part of the Apiaceae family and is an ingredient in Bakso.
E-TFR #3	.828	Celery is part of the Apiaceae family and is an ingredient of Bakso.
model score rerank #1	.447	Celery is part of the Apiaceae family and is one of the ingredients in Bakso.
model score rerank #2	.857	Celery is a member of the Apiaceae family and is an ingredient in Bakso.
model score rerank #3	.468	Celery is part of the Apiaceae family and is one of the ingredients of Bakso.
oracle over lattice	1.0	Celery is a member of the Apiaceae family and is an ingredient of Bakso.

Table 8: Example 4, Table to Text, Reranked on LATT

Label	Unique Nouns	Text
Source	-	The Death of Poor Joe, which dates back to March 1901, was discovered by British Film Institute...
Reference	-	The oldest surviving film featuring a Charles Dickens character has been discovered, in the year of the 200th anniversary of the author's birth.
TFR #1	11	The earliest known film of Charles Dickens' A Christmas Carol is to be shown in the UK as part of a celebration of the author's bicentenary next year.
TFR #2	11	The earliest known film of Charles Dickens' A Christmas Carol is to be screened in March as part of a celebration of the author's bicentenary next year.
TFR #3	11	The earliest known film of Charles Dickens' A Christmas Carol is to be screened in London as part of a celebration of the author's bicentenary next year.
oracle over lattice	11	The earliest known film of Charles Dickens' A Christmas Carol is to be screened in March as part of a bicentenary celebration of the author's work.

Table 9: Example 5, Summarization, Unique Nouns, Reranked on LATT

Label	Unique Nouns	Text
Source	-	Regulator Ofcom ruled the performance, by Alexandr Magala of Moldova, was "in line with audience expectations"...
Reference	-	ITV show Britain's Got Talent will not be investigated by the broadcasting watchdog over a sword-swallowing act that drew 33 complaints.
TFR #1	13	A daredevil sword act on Britain's Got Talent drew no complaints, despite the stunt leaving one contestant fearing for his life, will not be investigated, TV watchdog Ofcom has said
TFR #2	13	A daredevil sword act on Britain's Got Talent drew no complaints, despite the stunt leaving one contestant fearing for his life, will not be investigated, TV watchdog Ofcom has said.
TFR #3	11	A stunt in which a man slid down a pole with a sword lodged in his mouth on Britain's Got Talent will not be investigated, TV watchdog Ofcom has said
oracle over lattice	13	A daredevil sword act on Britain's Got Talent will not be investigated over a stunt in which a man fell down a pole with a sword stuck in his mouth, the media watchdog has said.

Table 10: Example 6, Summarization, Unique Nouns, Reranked on LATT

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Right after Section 10

☒ A2. Did you discuss any potential risks of your work?
This work is improving the quality of text generation systems. We believe that the risks of these methods are essentially the same as the risks of broader text generation systems, which have been documented at length in other publications.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Yes, they are the first 2 sections.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Yes, we use scientific artifacts, and discuss them in the appendix, specifically in Appendix Section E

☑ B1. Did you cite the creators of artifacts you used?
Yes, we cite them throughout the paper, as well as in Appendix Section E

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Yes, we document these in Appendix Section E

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Yes, we only use data and code specifically prepared for research contexts, and thus stay in line with intended use of the artifacts we use, as we only use them in a research context.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No, these are standard datasets that do not contain PII to our knowledge.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Yes, we document languages clearly for all sets we use them in (All tables in the paper).

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Yes, we document this in Appendix A.2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ **Did you run computational experiments?**

Yes, we report details in Appendix E, in addition to descriptions in Section 5.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We report these in Appendix E.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Yes, we discuss this in Appendix E.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Yes, we discuss our mechanism for computing results robustly in detail in Appendix B.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Yes, this is included in Appendix E.

D ☒ **Did you use human annotators (e.g., crowdworkers) or research with human participants?**

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.