Massive-scale Decoding for Text Generation using Lattices

Anonymous ACL submission

Abstract

Conditional neural text generation models generate high-quality outputs, but often concentrate around a mode when what we really want is a diverse set of options. We present a search algorithm to construct lattices encoding a massive number of generation options. First, we restructure decoding as a *best-first search*, which explores the space differently than beam search and improves efficiency by avoiding pruning paths. Second, we revisit the idea of *hypothesis recombination*: we can identify pairs of similar generation candidates during search and merge them as an approximation. On both summarization and MT, we show that our algorithm encodes thousands of diverse options that remain grammatical and high-quality into one lattice. This algorithm provides a foundation for building downstream generation applications on top of massive-scale diverse outputs.[1]

1 Introduction

Although pre-trained text generation models (Lewis et al., 2020; Raffel et al., 2020) have achieved impressive results across a range of tasks, these models do not always deliver what system developers want. Machine generated text may be non-factual (Kryscinski et al., 2020; Maynez et al., 2020; Goyal and Durrett, 2021) or toxic (Gehman et al., 2020). We might patch these problems by applying discriminators over the output (Holtzman et al., 2018; Yang and Klein, 2021) to enforce these properties post-hoc; we could, for instance, apply a secondary model as a reranker over a small collection of outputs. However, if the generator returns a homogeneous set of candidates, we may fail to find *any* usable generation output. What if generation models could return *massive* numbers of candidates rather than a few outputs with optimal score? With a large set of candidates, our secondary model

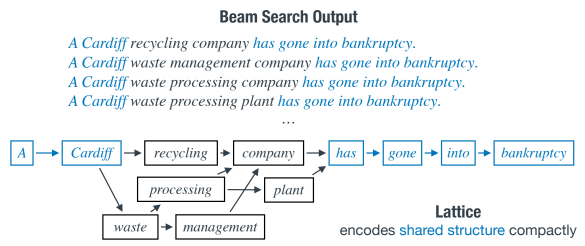

Beam Search Output
A Cardiff recycling company has gone into bankruptcy.
A Cardiff waste management company has gone into bankruptcy.
A Cardiff waste processing company has gone into bankruptcy.
A Cardiff waste processing plant has gone into bankruptcy.
...
A
Cardiff
recycling
company
has
gone
into
bankruptcy
processing
plant
waste
management
Lattice
encodes shared structure compactly

Figure 1: A lattice of outputs yielded by path recombination is a more efficient way to represent and explore related generation outputs compared to beam search.

could more easily find an acceptable one without having to take more extreme steps like re-training the initial generation model. Output diversity has separately been established as a useful goal for for applications such as dialogue and story generation (Li et al., 2016; Fan et al., 2019).

Standard approaches including beam search (BS) and sampling methods fall short of our goal. Beam search uses significant computational resources to explore similar hypotheses, and much of the computation in the search process is invested into paths that could be acceptable generation outputs, but are ultimately pruned. Sampling approaches like nucleus sampling (Holtzman et al., 2020), although achieving better diversity than beam search, often re-discover seen hypotheses and can be harder to control for quality. A central problem with both methods is that they do not handle very similar hypotheses efficiently.

In this paper, we present a decoding framework with two key components. First, we argue that a modified **best-first search** (BFS) is the right way to explore the search space. We augment standard best-first search with a depth-first path completion strategy: we eagerly expand each node until we reach an EOS token, thereby guaranteeing that each node is part of some completed path returned to the user. This generation strategy avoids exploring large numbers of states which end up being pruned.

[1] Code and demo will be available on GitHub.

BFS is also more flexible than static beam search and can prioritize exploration in more uncertain parts of the generation.

Second, our algorithm returns a massive number of generation options encoded in a lattice, with different hypotheses **recombined** in an approximate fashion. Beam search preserves similar outputs such as "*A Cardiff recycling company has gone into*" and "*A Cardiff waste management company has gone into*" as different states. However, these prefixes actually have very similar distributions of following words under the model; if we identify states like this, we can recombine them (Figure 2) and treat them as the same from the perspective of future continuations. In Figure 1, we show an illustration of the lattice structure this recombination can form for document summarization. We broaden a recombination method used previously in beam search for machine translation (Och et al., 2001; Zhang et al., 2018), enabling us to compactly encode large number of generation candidates and achieve dense lattices.

We show results for both document summarization and machine translation in three language pairs. For each setting, we show that our lattice encodes a large number of high-quality candidates, including good matches with annotated reference generations. We further show that a variant of our method can still achieve strong results with a lower number of nodes expanded than the baselines, suggesting that this can be a path towards saving computational resources. We believe that computing thousands of high-quality generation candidates within a single compact data structure can provide a powerful starting point for various downstream purposes: diversity, factuality, customizability, and more.

2 Problem & Setup

We define our algorithm in the context of conditional text generation (Sutskever et al., 2014; Bahdanau et al., 2014). Conditional text generation is formulated as sequence transformation from a source input $\mathbf{x}$ to target output $\mathbf{y} = (y_1, \dots, y_n)$ via a neural text generation model parameterized by θ. Each y_i is a symbol in a vocabulary $\mathcal{V}$. The probability of a decoded sequence is $p(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{t=1}^{n} p(y_t \mid y_{<t}, \mathbf{x}; \theta)$. Decoding text from a model can be framed as a search problem, where the search objective is to find the output sequence that maximizes the conditional probability under the model: $\arg\max_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}} \mid \mathbf{x}; \theta)$. Because

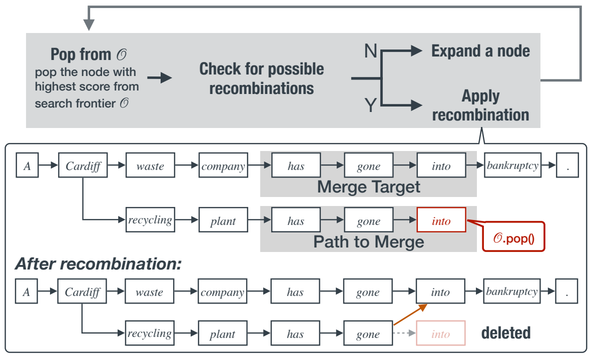


Figure 2: Our search algorithm. At each step, the algorithm pops a node from search frontier $\mathcal{O}$, checks for possible recombinations with existing nodes, and merges the nodes if a match is found. "*waste company*" and "*recycling plant*" are interchangeable paraphrases which do not affect the continuation from the model's perspective.

$p(\hat{y}_t \mid \hat{\mathbf{y}}_{<t}, \mathbf{x}; \theta)$ depends on the entire generated sequence so far, this decoding problem is intractable to solve exactly.

While typically the goal of decoding is to find the hypothesis with the highest possible model score, we instead focus on finding a large set of "good enough" hypotheses. That is, finding a set $\mathcal{Y}$:

$$\arg\max_{\mathcal{Y}} \; |\mathcal{Y}| \quad \text{s.t.} \;\; p(\mathbf{y} \mid \mathbf{x}; \theta) > \epsilon \text{ for all } \mathbf{y} \in \mathcal{Y} \qquad (1)$$

for some threshold ϵ. ϵ emerges naturally by adjusting search hyperparameters to control the number of returned hypotheses. Our goal in this paper is to design an algorithm that can efficiently find $\mathcal{Y}$.

Notation We encode predicted generation candidates $\hat{\mathbf{y}}$ in a lattice. A **lattice** $L = (N, E)$ is a directed graph where each **node** represent a word token and paths defined by directed edges encode candidates. A **path** π in L from a unique start-of-sequence node n_{SOS} to any node n represents a (partially) decoded string, consisting of the words in that path. All completed paths start with n_{SOS} and end at (potentially different) end-of-sequence nodes n_{EOS}. The search graph L is constructed iteratively through a search procedure. We maintain the closed graph $\mathcal{C}$ with explored nodes and edges as well as a search frontier $\mathcal{O}$, a set consisting of successors to nodes currently in the graph. For each node, there are $|\mathcal{V}|$ possible successors.

We define the **search budget** as the number of nodes expanded from the search frontier. Our experiments will seek to compare different methods using the same search budget. We will define this more precisely in Sec. 6.

Algorithm 1 Best-first search with depth-first completion and path recombination

Input: Generation model θ with vocabulary $\mathcal{V}$, search budget b, $\mathcal{O}$ and $\mathcal{C}$ denote open set (max priority queue) and closed set, $isRecomb$ and $doRecomb$ are functions checking and running path recombination.
Output: All completed paths P

```
 1: $\mathcal{O} \leftarrow \{(\infty, n_{\text{SOS}})\}, \mathcal{C} \leftarrow \emptyset, expanded \leftarrow 0$.
 2: while $expanded < b$ do
 3:     $\hat{h} \leftarrow \mathcal{O}.\texttt{pop()}$
 4:     if $isRecomb(\hat{h}, \mathcal{C})$ then
 5:         $doRecomb(\hat{h}, \mathcal{C})$
 6:         continue
 7:     end if
 8:     if $\hat{h} \neq$ EOS then
 9:         $v_{greedy} = \arg\max_{v \in \mathcal{V}} p(v \mid \hat{h}, \mathbf{x}; \theta)$
10:         for $v \in \mathcal{V}$ do
11:             score $\leftarrow s(v, \hat{h})$
12:             if $v = v_{greedy}$ then
13:                 score $\leftarrow \infty$        // depth-first completion
14:             end if
15:             $\mathcal{O} \leftarrow \mathcal{O} \cup$ (score, $n_v$)
16:         end for
17:         $expanded \leftarrow expanded + 1$
18:     end if
19:     $\mathcal{C} \leftarrow \mathcal{C} \cup \hat{h}$
20: end while
```

Inadequacies of Beam Search Beam search has several properties that make it inadequate for our goal, including that it optimizes for the wrong objective, lacks diversity, and aggressively prunes hypotheses that could still be useful. We show experiments on these aspects in Appendix A.

3 Modified Best-first Search

As established in the previous section, beam search prunes many paths that would potentially yield high-quality summaries and wastes computational resources expanding nodes that aren't included in a final search graph. We tackle this issue by changing from beam search to *best-first search* (BFS) (Hart et al., 1968; Pearl, 1984). BFS prioritizes searching over nodes according to a scoring function, giving us more flexibility in how we explore the space. Our chief modification of the base algorithm is a heuristic we call depth-first completion.

Depth-first Path Completion Neural text generation is a search problem with large branching factor ($\mathcal{V}$) and deep search depth (sequence length). As a result, applying BFS with the scoring function being the model score of a state often leads to a broad search that rarely returns a valid path. One solution to this problem is to incorporate a heuristic based on length. Model score is monotonically decreasing as a sequence grows in length, so prior

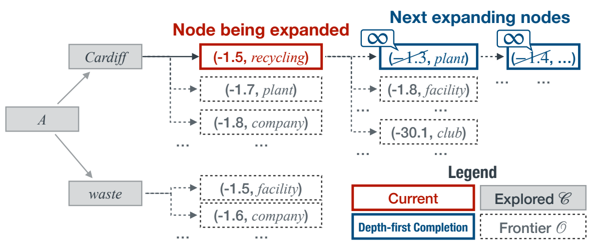


Figure 3: Depth-first completion. The red node is the current node being expanded. We depth-first expand a sequence of nodes (in blue) to get a completed path.

work (Wu et al., 2016; Zhang et al., 2018; Meister et al., 2020b) has used a length reward term to alleviate this issue.[2] We found that, even with a length heuristic, BFS will still have "dangling" nodes that are not part of any path to an EOS (goal) token, and it might return few or no valid hypotheses.

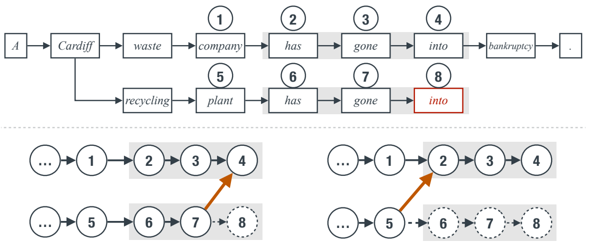
Recognizing our objective from Equation 1, we can take a simple step to ensure that every node ends up on some completed path: *eagerly do a greedy search from each node until we reach n_{eos} or exceed a maximum length*. In Algorithm 1, we implement this by modifying the priority of the highest scored token with ∞ (line 12), so it will be explored depth-first immediately after the current time step. In Figure 3, we show an illustrative example of depth-first completion.

Search Algorithm We describe BFS with depth-first completion in Algorithm 1. The algorithm is a modified best-first search algorithm applied to text generation. $s(\cdot)$ is a function to evaluate the value of a path. Typically it is defined as $s(\mathbf{y}) = \sum \log p(y_t \mid y_{<t})$. b is the budget for total model calls to neural text generation model. Note that $isRecomb$ and $doRecomb$ do not invoke the neural generation model, so they do not count towards the computation budget we defined here. In practice, we only consider top 5 expansions rather than the whole vocabulary $\mathcal{V}$ for line 10.

4 Path Recombination

Path recombination, also known as hypothesis recombination, was originally proposed and used in phrase-based machine translation (Och et al., 2001; Koehn et al., 2003; Zhang et al., 2018). The idea of path recombination is to *combine similar paths if what the model predicts for them in the future*

[2]This can be considered a heuristic like in (weighted) A* search, but it is not necessarily admissible or consistent. Interpreting it this way does make our approach best-first search with modified scores, hence why we describe it this way.



(a) RCB (b) ZIP

Figure 4: Illustration of two path recombination strategies, RCB and ZIP. Orange lines are the merging edges (MRG) built by recombination. Dotted lines and circles are removed after recombination. The key difference of RCB and ZIP is how much the recombination propagates, 1 step or n steps.

is the same, reflecting a similar dynamic programming principle as the Viterbi algorithm. We focus on finding hypotheses which approximately exhibit this property, and show that merging them can yield high-quality outputs. Figure 2 shows an example of recombination. The two hypotheses being merged here roughly convey the same intent, and it turns out that the shared suffix "*has gone into*" is a strong indicator that the model will treat them similarly in the rest of the generation.

Prerequisites of Recombination Theoretically, two search states should only be recombined if they yield the exact same distribution over future generation decisions (see strong equivalence in Appendix B). However, this is intractable even to check approximately; we define a weaker criterion:

Definition 4.1 (Weak equivalence)**.** *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be two prefix strings starting with* n_{sos}. $\mathbf{a}$ *and* $\mathbf{b}$ *are weakly equivalent if greedy completions of these two strings are the same:* $\arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{a}) = \arg\max_{\mathbf{y}'} P(\mathbf{y}' \mid \mathbf{b})$.

This criterion can be checked empirically, but it is still not practical to do so during search itself.

To approximate equivalence, we define a similarity function $\texttt{merge}(\mathbf{h}, \hat{\mathbf{h}})$ to determine if an expanded node $\hat{\mathbf{h}}$ should be merged with an existing expanded node $\mathbf{h}$. A similar recombination idea was explored in Zhang et al. (2018). Following their work, we explore a family of rule-based heuristics for merging. There are two rules: (1) two strings share a common n-gram suffix, (2) the length difference of two strings is less than α. Assume that the canonical paths for $\mathbf{h}$ and $\hat{\mathbf{h}}$ are lengths l and $\hat{l}$, then $\texttt{merge}(\mathbf{h}, \hat{\mathbf{h}}) = \mathbb{1}[\pi(\mathbf{h})_{l-n+1,\dots,l} = \pi(\hat{\mathbf{h}})_{\hat{l}-n+1,\dots,\hat{l}} \wedge |l - \hat{l}| < \alpha]$ where α and n are hyper-parameters.[3] For a large enough value of n, note that the shared suffixes encourage hypotheses like this in Figure 4 that share large parts of the structure already.

Prior Work: BSZBEAM Zhang et al. (2018) use their merging criterion in the context of beam search for neural machine translation. If the merging criteria hold, $\hat{\mathbf{h}}$ will be recombined with $\mathbf{h}$. However, $\hat{\mathbf{h}}$ will not be considered as a future merging candidate. We call this merging strategy ZBEAM. We implement this model together with its merging criteria and denote it as BSZBEAM. This strategy is tailored to beam search and, as we discuss later, explores a more limited set of merges than one might want to consider.

Canonical Paths After recombination, a single node may represent multiple different possible sentence prefixes. If an edge is created due to the extension of search graph via model's prediction, we call it a GEN edge. Otherwise, the edge is created due to path recombination, and we call it a MRG edge. We define the notion of a canonical path, which represents the single path used to score candidate expansions.

Definition 4.2 (Canonical Path)**.** *Let n be a node. The canonical path to n is defined as the unique path from n_{sos} to n consisting only of* GEN *edges.*

Theorem 4.1. *For any node n in the graph except n_{sos}, there exists exactly one canonical path.*

We present the proof in Appendix. C. We define the path of a node n, $\pi(n)$, as returning the sequence of words corresponding to the canonical path of that node. Expanding n computes $P(y \mid \pi(n))$ under the neural model.

5 Recombination Mechanism

We illustrate the two major recombination techniques we introduce, RCB and ZIP, in Figure 4.

RCB: Generalization of ZBEAM ZBEAM has a major limitation: a limited set of merging candidates. The potential merge candidates in ZBEAM are only nodes in the current beam hypotheses and their previous steps, so the method cannot merge

[3]In Zhang et al. (2018), there is one extra constraint requiring $P(\hat{\mathbf{h}} \mid \mathbf{x}) < P(\mathbf{h} \mid \mathbf{x})$, which requires that the path getting recombined has lower model score than the existing path. However, we found that model score is not always a good indicator for merging, as suggested in Fig. 7, partially because it is challenging to calibrate scores across different sequence lengths, so we disregard this constraint.

with nodes from earlier timesteps. For example, "*A waste plant has gone into*" cannot be merged with the hypothesis with ending in node 4 shown in Figure 4. The proposed generalization, RCB, addresses this limitation. We index all of the nodes in the lattice across all timesteps by their n-grams using a hash table, making it $O(1)$ time to look up an n-gram pattern and retrieve potential merge candidates if they exist.

ZIP: Recombining More If we take a closer look at RCB in Figure 4, we see that even in the merged structure, nodes 3 and 7 and nodes 2 and 6 are preserved as separate. They do not pass the recombination criterion themselves, but these nodes are part of the suffix matched strings, still correspond to the same words, and have the same directly generated next word. There is reason to believe that these might be equivalent as well. Hence, we explore a variant called ZIP that propagates the merge backwards through the lattice. This change relaxes the merging criterion and up to n pairs of nodes are combined when a merge is identified, leading to a more compact lattice. We describe some of the details in Appendix D.

6 Evaluation

To evaluate the proposed methods, we conduct experiments on abstractive text summarization and machine translation. Our evaluation focuses on two questions: (1) how **large and diverse** are our lattices; (2) are the candidates encoded in the lattices **high quality and grammatical**?

6.1 Datasets & Base Models

We obtain all the models and certain baseline decoding methods from the Transformers library (Wolf et al., 2020). Since our methods are inference techniques with rule based heuristics, we do not retrain any models. For **summarization**, we use XSum (Narayan et al., 2018), a popular English news summarization dataset. We sample 100 examples from the validation set. The base model we use is `BART-large-XSum` (Lewis et al., 2020). For **machine translation**, we study our models on the English-French (en-fr) pairs from WMT 2014 (Bojar et al., 2014) and Chinese-to-English (zh-en) pair from WMT 2019 (Barrault et al., 2019). We use `mBART` (Liu et al., 2020), a state-of-the-art neural machine translation model. We set the max decoding length to be twice the input length, so it varies per example.

6.2 Search Budget

To fairly compare the resource usage of all methods, we define the search budget as the number of calls to the neural model, equivalent to the number of nodes expanded.[4] With beam size k and maximum length T, beam search methods are given a theoretical budget of kT. We could simply allow best-first search and sampling methods to expand this number of nodes. However, since hypotheses may terminate before they reach EOS, empirically there is a gap between effective length (the average generated hypothesis length) and max length for both beam search and sampling. To balance the computation used across the different methods, we apply a correction factor so that the different methods are expanding the same number of nodes in aggregate. We increase the beam size k by 50% for translation, from 8 to 12, and 25% for summarization, from 16 to 20, for our baseline methods: k to BS, DBS, NCLS, TEMP, and BSZBEAM. This correction was empirically determined to balance the number of nodes expanded between our method and the baselines. We emphasize that this correction improves the baseline performance relative to our methods.

6.3 Search Algorithms

We implemented GREEDY, BS, DBS, NCLS, and TEMP as baseline methods. NCLS$_{0.9}$ represents nucleus sampling method with $p = 0.9$. We refer to Appendix E for detailed descriptions. We also experiment with basic **BFS** without path recombination, but including our depth-first path completion technique to ensure that finished hypotheses are produced. BSZBEAM is our implementation of Zhang et al. (2018). We integrate RCB with nucleus sampling and best-first search as NCLSRCB and BFSRCB. We also test BFS with the ZIP strategy. 🍃BFSZIP is a resource-efficient version of BFSZIP where only 25% of the search budget is used, exploring what this method can achieve with a lower budget given its more aggressive merges.

6.4 Evaluation Metrics

We describe our metrics to evaluate both quality and diversity. Several of our methods build on ROUGE and BLEU (Papineni et al., 2002; Post,

[4]We incur negligible overhead from rule-based matching in the merging step, as well as the computational costs of computing diversity term in DBS and modifying sampling distributions in sampling methods.

	Diversity					OR	SP	GRM
	↑	↑	↑	↓	↑	↑	≥	↓
Model	|path|	N1	N2	SBL	ED	R2	R2	ERR
GREEDY	1	22	23	100	0	17.3	17.3	0.5%
BS	20	42	61	87	31	26.3	17.7	0.3%
DBS	19	59	91	79	53	25.5	15.9	0.5%
NCLS$_{0.8}$	20	124	237	57	72	30.2	14.5	0.5%
NCLS$_{0.9}$	20	143	273	50	76	28.1	13.3	0.8%
TEMP$_{1.5}$	20	170	319	51	82	26.6	11.6	1.4%
BFS	30	88	167	68	60	30.1	15.6	0.4%
	+ Path Recombination							
BSZBEAM	4,701	66	118	75	51	33.0	16.0	0.7%
NCLS$_{0.8}$RCB	52	167	308	53	79	28.8	13.0	1.0%
NCLS$_{0.9}$RCB	36	207	363	50	87	25.9	11.0	1.7%
BFSRCB	7,758	111	239	65	64	35.8	15.2	0.8%
BFSZIP	95,744	124	274	53	77	36.8	13.2	1.4%
∅BFSZIP	297	58	92	80	49	29.2	15.2	0.8%

Table 1: Results decoding text summaries on XSum. Diversity metrics are rounded to integers to save space. We use ↑, ↓ and ≥ to denote the desired trend, the higher the better, the lower the better, or good if it passes some threshold. Among the methods with path recombination excluding ∅BFSZIP, we highlight the best, second and third best, and the worst one.

2018) for evaluating the generated text compared to reference summaries or translations.

Diversity-oriented Metrics We evaluate the diversity of generated texts with the following metrics. (1) |path| is the average number of unique paths in the produced lattice.[5] (2) Number of unique n-grams encoded in the lattice; this captures a different type of diversity than the number of paths, since there could be many paths reusing the same words. N1 and N2 are average number of novel unigrams and bigrams in the graph. (3) SBL is the average self-BLEU among m samples (Zhu et al., 2018). The samples are drawn from a uniform random walk from n_{SOS}. The range of SBL is $[0, 100]$. (4) ED is the average edit-distance among m samples. We set $m = 5$ in our experiment.

Quality: Grammaticality We adopt GECToR a neural grammatical error correction model (Omelianchuk et al., 2020) to automatically assess the grammaticality of generated texts. We report GRMERR(%), the average number of grammar errors per token, for all English-output experiments.

Quality: Oracle Reference Match Given the reference, we find the path with highest ROUGE or BLEU over all found paths. Oracle ROUGE is defined as $\text{OR}(\mathcal{Y}, \mathbf{y}^*) = \max_{\mathbf{y}\in\mathcal{Y}}(\text{R2}(\mathbf{y}, \mathbf{y}^*))$. This metric captures both quality and diversity: the algorithm needs to find something close to the reference, but a diverse lattice will have a higher chance of exhibiting a good candidate all else being equal.

Quality: Average Reference Match Although our method focuses on deriving diverse text summaries or translations, we aim to guarantee that the generated text is highly relevant to the generation target and is of high quality in general. We sample 1,000 paths from the lattice with replacement and evaluate the average ROUGE or BLEU compared to the reference. We denote this metric as SP.

7 Results

Text Summarization We present the experimental results on the dev set of XSum in Table 1. Full results are kept in Table 4 for reference. Among non-recombination methods, BS and DBS are the least diverse methods. Sampling based methods including TEMP are generally more diverse, but the oracle ROUGE is lower than that of BFS. Given the sacrificed text quality (lower sample ROUGE and more grammar errors) of sampling based methods, we argue that **modified best-first search is a strong decoding strategy even without path recombination**. The bottom half shows all methods with path recombination techniques. **Recombination significantly improves the diversity of generated outputs**, with a much higher number of paths. The self-BLEU of the recombination variants are lower than their non-recombination counterparts.

In terms of search quality, the proposed BFSRCB and BFSZIP methods obtain significantly higher oracle ROUGE compared to all other methods. We show these results later in Figure 9: our approach can find much better oracle solutions, even compared with beam search method with quadruple the amount of computation resources. The design of the oracle ROUGE metric is also motivated by a real use case: if you want a specific summary (e.g., a summary covering a specific entity or topic), does it exist in the search graph? Higher oracle ROUGE indicates a closer match, meaning a strategy using some kind of reranking model could help find the user's preferred outputs.

Comparison: RCB & ZIP The ZIP method yields even more diverse output at the cost of text quality. There are a few reasons for this: 1) recom-

[5]Due to the exponentially growing number of paths in some of our models, we cap the number of paths from n_{SOS} to each node to $C = 10^4$.

Model	zh-en ↑ |path|	Diversity ↑ N1	↑ N2	↓ sBL	↑ ED	↑ OR BL	≥ SP BL	↓ GRM ERR	fr-en ↑ |path|	Diversity ↑ N1	↑ N2	↓ sBL	↑ ED	↑ OR BL	≥ SP BL	↓ GRM ERR
GREEDY	1	35	40	100	0	24.7	24.7	0.5%	1	28	31	100	0	40.0	40.0	0.9%
BS	12	45	63	95	20	32.2	25.0	0.2%	12	37	50	93	13	52.6	38.1	1.0%
DBS	11	55	84	89	59	29.7	20.5	0.5%	11	45	67	88	37	46.4	30.5	1.1%
NCLS$_{0.8}$	12	94	188	72	82	31.5	17.5	0.7%	11	62	107	80	46	51.0	31.2	1.0%
NCLS$_{0.9}$	12	110	226	67	94	30.4	15.8	0.9%	12	75	134	77	57	48.3	27.4	1.2%
TEMP$_{1.5}$	12	140	280	62	105	27.0	12.7	1.3%	12	102	184	69	71	43.7	21.6	1.6%
BFS	18	60	104	86	54	32.7	20.7	0.5%	27	59	102	84	37	53.2	33.7	1.1%
+ Path Recombination																
BSZBEAM	18,336	64	117	77	65	40.1	19.1	0.8%	16,729	59	107	77	43	61.2	28.2	1.3%
NCLS$_{0.8}$RCB	81	138	263	67	91	26.8	13.9	1.1%	344	140	246	64	67	48.2	26.6	1.2%
NCLS$_{0.9}$RCB	38	188	343	58	114	23.9	10.6	1.7%	123	205	352	55	92	41.1	20.2	2.1%
BFSRCB	17,535	81	171	76	72	42.1	19.4	0.9%	47,577	85	193	68	52	64.6	25.3	1.6%
BFSZIP	59,020	94	205	66	88	42.4	15.5	1.4%	146,163	111	259	56	63	56.8	16.9	2.5%
🍃BFSZIP	511	50	75	89	38	33.0	21.2	0.7%	4,531	50	81	82	35	59.5	29.4	1.4%

Table 2: Results on WMT14 Fr-En and WMT19 Zh-En. Columns are the same as for summarization, although BLEU is used instead of ROUGE. Trends are roughly similar, with BFSRCB providing high diversity at good quality and 🍃BFSZIP offering a strong tradeoff between computational resources and diversity.

bination of more nodes makes the lattice denser, increasing the number of paths but also potential errors; 2) elimination of unexplored children from merged branch reduces the waste of exploration which means ZIP can explore more hypotheses than RCB. With the same amount of computational resources, ZIP explores a larger search space while RCB explores a smaller collection more reliably. 🍃ZIP exploits the efficiency of ZIP to achieve high diversity, and by searching through fewer states, it manages to achieve higher quality as well.

Machine Translation We show the result on machine translation in Table 2 and 6. Results on translation tasks show the consistent gains of diversity from path recombination models. In Table 2, we show two translation tasks where the target language is English. BFSRCB works better than BFSZIP because it disables some aggressive and bad merges which explores bad hypotheses. Compared to summarization, we found the search space in MT to be more constrained, so there was less room for aggressive merging and exploration to improve over RCB. Our lower-resource method, 🍃BFSZIP approach, actually performs quite well on most metrics with only 25% of search budget. It has better diversity performance than any non-recombination methods, and comes with quality better than most of the recombination methods. The usage of BFS and path recombination methods like BFSRCB and BFSZIP is promising for being able to find a better cost-diversity tradeoff in MT.

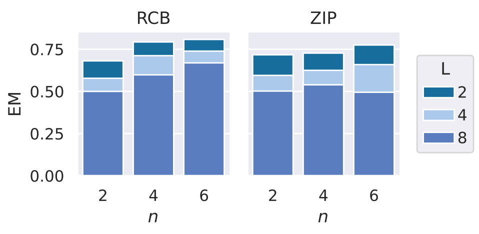


Figure 5: Empirical verification of merging criteria. We experiment with $n = \{2, 4, 6\}$ for n-gram suffix matching. We sample 1,000 recombinations from BFSRCB and BFSZIP respectively, and run greedy inference based on merged hypotheses. We use Exact Match (EM) to measure how often two merged hypotheses give the same greedy future generations considering the next L tokens after the merge.

Validating the Merging Criterion Our merging criterion is fundamentally an approximation of the equivalence criteria described in Section 4. Our question is: **what fraction of nodes merged by our merging criterion satisfy the weak equivalence assumption?** We conduct an experiment to verify this on XSum. We compute the greedy completion for L timesteps and check whether the continuation of the base candidates would be the same. In Figure 5, we show the fraction of merged pairs for which the generations match exactly under three values of the recombination criterion. For BFSRCB, when using $n = 4$ the greedy continuation over 4 timesteps is the same 71.2% of the time. For BFSZIP it is the same 62.5% of the time. Following the weak equivalence criterion is a strong

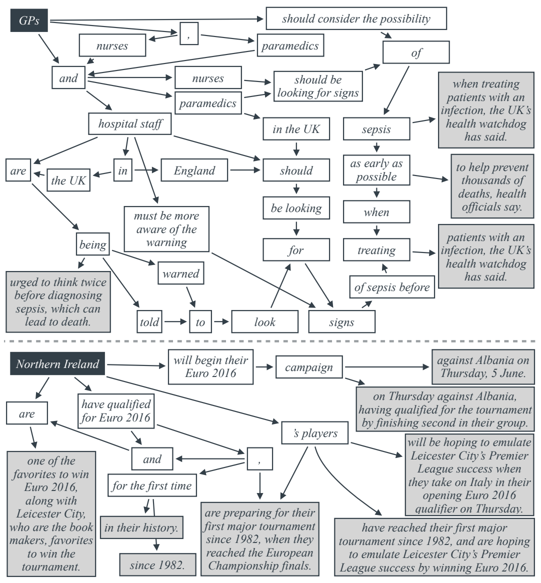

GPs
should consider the possibility
,
nurses
paramedics
of
and
nurses
should be looking for signs
paramedics
when treating patients with an infection, the UK's health watchdog has said.
hospital staff
in the UK
sepsis
are
in
England
should
as early as possible
to help prevent thousands of deaths, health officials say.
the UK
be looking
when
must be more aware of the warning
being
for
treating
patients with an infection, the UK's health watchdog has said.
warned
of sepsis before
urged to think twice before diagnosing sepsis, which can lead to death.
told
to
look
signs
Northern Ireland
will begin their Euro 2016
campaign
against Albania on Thursday, 5 June.
on Thursday against Albania, having qualified for the tournament by finishing second in their group.
are
have qualified for Euro 2016
's players
will be hoping to emulate Leicester City's Premier League success when they take on Italy in their opening Euro 2016 qualifier on Thursday.
and
,
one of the favorites to win Euro 2016, along with Leicester City, who are the book makers, favorites to win the tournament.
for the first time
in their history.
are preparing for their first major tournament since 1982, when they reached the European Championship finals.
have reached their first major tournament since 1982, and are hoping to emulate Leicester City's Premier League success by winning Euro 2016.
since 1982.

Figure 6: Two examples on XSum by BFSZIP. The start of sentence is denoted in dark color, and all the endings are in gray. We combine tokens to phrases when possible for visualization purpose. More examples are presented in Appendix. I.

indication that these hypotheses can admit many of the same continuations. RCB is more reliable than ZIP, but both methods show moderate adherence to the equivalence criterion.

Error Analysis & Visualization In Figure 6, we present two examples on XSum by BFSZIP. The upper example has more word level recombination and paraphrasing while the bottom one has more ways of ending and more diverse content coverage. We show more examples on both summarization and translation in Appendix. I.

We manually examine the output and found a few common types of errors introduced by our algorithm. (1) Factual errors at high entropy nodes. Our approach assumes that high-scoring candidates under the model are good quality, but this assumption is violated in certain cases, like when the model attempts to hallucinate information. For example, given the prefix "*The company, founded in*" will cause the model to guess answers like "*1989*" or "*1999*". Encoding all of these in the lattice is incorrect. However, we did not see significant factual errors introduced by merging specifically. (2) Aggressive bad merges. In the upper example in Figure 6, the cluster of "*GPs*", "*nurses*", "*paramedics*" is an example case. The lattice encodes paths like "*GPs, nurses and nurses should ...*". This could be fixed by heuristics or rules in future work.

8 Related Work

The techniques used in this work partially reflect an outgrowth of a few lines of literature: understanding the behavior of text generation models (Xu et al., 2020; Xu and Durrett, 2021; Zhong et al., 2021), investigations into beam search (Stahlberg and Byrne, 2019; Meister et al., 2020a), and studies of diversity in generation.

In terms of search strategies, best-first beam search (Meister et al., 2020b) is a method integrating best-first search with beam search. Some other variants of search have also been studied in previous work (Meister et al., 2021b,a). Beam search has been critically examined in some recent work (Huang et al., 2017; Stahlberg and Byrne, 2019), but largely of focused on specific challenges in MT.

As for diverse generation, neural text degeneration has been discussed in Radford et al. (2019); Holtzman et al. (2020); Welleck et al. (2020), which led to an interest in diverse generation models. Diverse text generation has been studied in previous work (Yu et al., 2017), including in dialogue (Li et al., 2016), story generation (Fan et al., 2019), and particularly paraphrasing (Iyyer et al., 2018; Goyal and Durrett, 2020). Our method can also diversify content coverage (Gehrmann et al., 2018) and word choice (Cao and Wang, 2021).

9 Discussion & Conclusion

We presented an algorithm for decoding in text generation with two main components: best-first search to more efficiently structure exploration of the space and hypothesis recombination to encode summaries in a lattice structure. We showed that across summarization and machine translation, these lattices successfully encode large numbers of high-quality generation options.

There are a few limitations of our method. First, we currently benchmark these techniques using number of nodes expanded, not wall clock time. There are strategies for parallelizing the BFS expansion (Shu and Nakayama, 2018), but it remains to be seen how this parallelism compares to the parallelism achieved by beam search. Regardless, the dramatically larger number of hypotheses we return outweighs efficiency differences for now. Second, we focus on auto-regressive methods in this paper. However, we believe our framework could also be applied and adopted to non auto-regressive generation models (Song et al., 2021).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Shuyang Cao and Lu Wang. 2021. Inference time style control for summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5942–5953, Online. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Florence, Italy. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

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

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

Tanya Goyal and Greg Durrett. 2021. Annotating and modeling fine-grained factuality in summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1449–1462, Online. Association for Computational Linguistics.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139, Copenhagen, Denmark. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, 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.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

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

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

Clara Meister, Afra Amini, Tim Vieira, and Ryan Cotterell. 2021a. Conditional Poisson stochastic beams. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 664–681, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Clara Meister, Martina Forster, and Ryan Cotterell. 2021b. Determinantal beam search. 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 6551–6562, Online. Association for Computational Linguistics.

Clara Meister, Tim Vieira, and Ryan Cotterell. 2020b. Best-first beam search. *Transactions of the Association for Computational Linguistics*, 8:795–809.

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.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Judea Pearl. 1984. Heuristics: intelligent search strategies for computer problem solving.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. In *arXiv eprint 1910.10683*.

Raphael Shu and Hideki Nakayama. 2018. Improving beam search by removing monotonic constraint for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 339–344, Melbourne, Australia. 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.

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 (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China. 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.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

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

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jiacheng Xu, Shrey Desai, and Greg Durrett. 2020. Understanding neural abstractive summarization models via uncertainty. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6275–6281, Online. Association for Computational Linguistics.

Jiacheng Xu and Greg Durrett. 2021. Dissecting generation modes for abstractive summarization models via ablation and attribution. 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 6925–6940, Online. 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.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Zhisong Zhang, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Hai Zhao. 2018. Exploring recombination for efficient decoding of neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4785–4790, Brussels, Belgium. Association for Computational Linguistics.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2856–2878, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

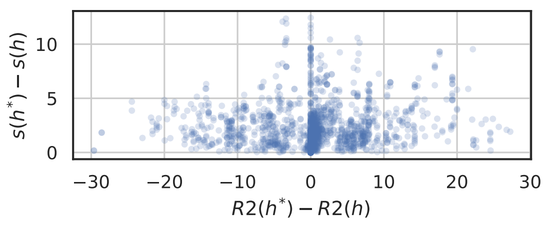


Figure 7: Correlation of ROUGE-2 and model score in beam search. For each example, we compare the hypothesis with the highest model score, h^*, with all other hypotheses. x and y-axis show the gaps of R2 and model score. The Pearson's ρ is 0.092 which suggests very low correlation between R2 and model score.

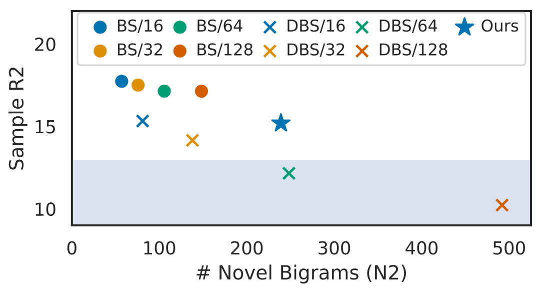


Figure 8: Results of BS/DBS on XSum with larger beam size k, compared to a proposed model introduced later (blue star) with equivalent beam size $k = 16$. We consider sample ROUGE-2 lower than 13 as low relevance/quality generations. Diversity of BS does not scale well with k and DBS generations become low quality.

k	16		8	
$\mathcal{D}$	XSum	zh-en	fr-en	en-fr
BS	71.3%	63.3%	54.0%	59.2%
DBS	71.2%	56.1%	50.4%	55.7%

Table 3: Pruning ratio of BS and DBS on different tasks and datasets with beam size k. We report the average percentage of explored nodes getting pruned and not appearing in a finished hypothesis.

A Inadequacies of Beam Search

As we have alluded to, beam search is inadequate for our goal for several reasons.

Better Model Score $\nRightarrow$ Better Hypothesis The most critical issue is that beam search is designed to efficiently approximate $\arg\max \hat{\mathbf{y}} = p(\hat{\mathbf{y}} \mid \mathbf{x}; \theta)$, but the optimal model score is neither our goal nor a guarantee of a good hypothesis. In Figure 7, we compare the correlation of model score and ROUGE under beam search for text summarization. The Pearson correlation between these two variables is very weak. Beyond ROUGE score, the example in Fig. 1 shows that the main differences between these summaries may be minor differences in surface realization that have little effect on our qualitative judgments of summary quality. **Finding the best model score does not substantially improve the quality over a near-optimal model score.** Allocating resources to eke out slight improvements over the greedy hypothesis, as beam search does, is a poor use of resources for most applications.

Lack of Diversity in (Diverse) Beam Search Are the model outputs from BS and DBS diverse? We use Self-BLEU (SBL) (Zhu et al., 2018) to measure the BLEU score for randomly sampled pairs from each algorithm's output. The lower the self-BLEU, the more dissimilar the pairs are. On decoding summaries on XSum, the SBL for BS/DBS are 87/79 while a nucleus sampling method can achieve 57/50 depending on configuration. Although DBS slightly improves the diversity compared to the original variant, **the overlap of outputs from beam search based method is still very high, and the diversity remains a challenge.**

Poor Scaling Behavior In spite of these shortcomings, perhaps beam search could still be viable with larger beam sizes if more computational resources are available. We experiment with beam sizes of $2^{\{4,5,6,7\}}$ and see how diversity scales with beam size. In Figure 8, we found that the exponential increase of beam size does not scale with the increase of number of novel bigram in beam search. In DBS, the diversity does ramp up, but the quality of the generated text is getting very bad very soon. **For BS and DBS, increasing beam size is not an effective solution for better diversity.** We also show that increasing beam size does not scale well in terms of finding better hypotheses, which is shown in Appendix. G.

Poor Efficiency from Pruning One final issue with beam search is that **most of its computation is not even useful in producing finished hypotheses**; that is, the set $\mathcal{Y}$ of answers produced does not contain most of the nodes expanded in the typical course of operation. We conduct an empirical pruning study on a summarization dataset and three translation datasets and show the results in

Table 3. For all studied cases, beam search and diverse beam search prunes over half of the expanded nodes. Many pruned hypotheses are not truly ungrammatical or low quality, but are merely slightly less likely than other nodes. How we can preserve more of the explored lattice and do so efficiently is addressed in the next section by our use of best-first search.

B Strong Equivalence of Path recombination

In the strictest form, recombining two hypotheses assumes the following equivalence between them:

Definition B.1 (Strong equivalence)**.** *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be two prefix strings starting with* n_{sos}*.* $\mathbf{a}$ *and* $\mathbf{b}$ *are strongly equivalent if* $P(\mathbf{y} \mid \mathbf{a}) = P(\mathbf{y} \mid \mathbf{b})$ *holds for all* $\mathbf{y}$*.*

Merging such states in the search tree is valid with no loss of information, as any expanded node will receive the same score under both prefixes. However, this assumption is not realistic since seq2seq models condition on the entire sequence so far, and any tiny perturbation changes the predicted distribution. To relax the assumption, we then propose the weak alternative.

C Proof of Theorem 4.1

Proof by induction. Base case: we begin with just n_{SOS} in the lattice, which has exactly one canonical path consisting of itself.

Inductive case: assume every node in the lattice has exactly one canonical path. We have to consider two cases when expanding a node in the lattice:

(1) Expanding the node as normal. In this case, the node is on the search frontier due to its parent node n' being expanded, which establishes a GEN edge from n' to n. Since n' has exactly one canonical path, n then has exactly one canonical path consisting of the canonical path to n' extended to n.

(2) Applying recombination. This operation only adds MRG edges and deletes nodes, neither of which have any impact on the canonical paths.

D Implementation Details: ZIP

We summarize the key differences of ZBEAM, RCB and ZIP in Table 5. In ZIP, nodes that are already expanded might be removed from the lattice due to recombination. For example, in Figure 4, node 6 and 7 are removed in this fashion. In general, we handle this by re-mapping the eliminated node to its surviving counterpart. Any reference to node 7 is routed to node 3, or whatever node 3 is mapped to. This procedure is defined and implemented as a union-find data structure.

Deduplication of Unexplored Successors After the ZIP procedure, we also remove the unexplored successors of the merged nodes, like node 6, 7, and 8 in Fig. 4. We show a more detailed example in Figure 10. In ZIP, we will merge node 3 and node 6. If we take a closer look at the successors of these two nodes, the distributions could be similar and in fact are expected to be if the equivalence criteria hold. We remove the unexplored direct successors of the merged node as part of the merging process, and the surviving node (node 3) captures these with similar probabilities regardless.

E Baselines

GREEDY is the deterministic greedy decoding method that always selects the highest probability token as prediction. The equivalent beam size for this approach is 1 since we only run one pass.

BS & DBS stand for beam search and its variant diverse beam search (Vijayakumar et al., 2016). In our configuration, we use Hamming distance as the diversity function and set the diversity strength to 1.5, following Vijayakumar et al. (2016).

NCLS is the nucleus sampling method proposed in Holtzman et al. (2020), which encourages quality by truncating the distribution over the vocabulary with a parameter p before sampling. We experiment it with $p = 0.9$ and $p = 0.8$.

TEMP changes the temperature of softmax function to reshape the prediction distribution (Ficler and Goldberg, 2017). We set the temperature parameter $\tau = 1.5$ so the prediction picks more low-scored tokens than $\tau = 1$.

F Implementation Details: Beam Search

In our beam search implementation, the size of the search frontier $\mathcal{O}$ is up to the beam size k. When a path is completed, we move it from the search frontier $\mathcal{O}$ to a completed set $\mathcal{F}$ to free up the beam for exploring unfinished hypotheses. Naturally, finished hypotheses $\mathcal{F}$ in the end can be of variable length. After reaching the max generation step T, we sort all hypotheses in $\mathcal{F}$ according to the model

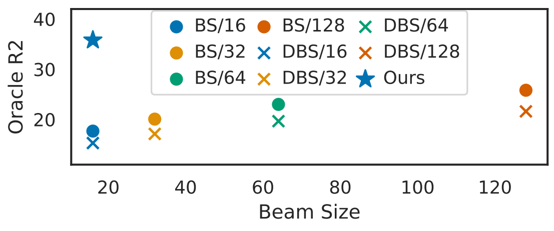


Figure 9: Oracle R2 of BS/DBS with larger beam size k. Blue star represents BFSRCB with equivalent $k = 16$.

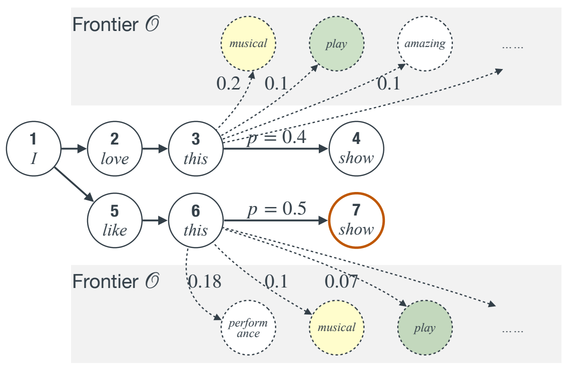


Figure 10: An illustration of removing unexplored hypotheses from search frontier in ZIP.

score. Following common practice in libraries such as Transformers (Wolf et al., 2020), we return a number of completed hypotheses equal to the beam size.

G Budget Scaling Behavior: Optimality

As a search algorithm, how do BS and DBS with larger beam size perform at finding solutions close to the reference? We compare the oracle R2 of BS/DBS with larger beam size in Figure 9. The oracle R2 increases slowly as k doubles, but our model BFSRCB with $k = 16$ achieves 35.8, much higher than all BS/DBS cases.

H Results of WMT14 English to French

Table 6 shows an additional experiment on English-French. We do not evaluate on grammaticality due to the GECToR model being specialized to English. The results show broadly similar trends as those in Figure 2, discussed in the main text.

I Examples

We show three examples with visualization in Figure 11,12 and 13. We use PyVis as the visualization tool.[6] More examples are available at anonymized.

J Computational Considerations

Resources Used All experiments were conducted on a server with 32GB RAM and Intel Xeon E5-2630 CPU, using a single NVIDIA GTX1080. The total computational budget in GPU hours is within 50 hours for experiments in text summarization and machine translation.

Memory and Runtime Although the final lattices returned encode large numbers of paths, they do not take large amounts of memory. Because the number of nodes in a lattice is no larger than the number of node expansion operations during beam search, it is always less than the search budget and can be stored compactly.

Moreover, the wall clock time of our BFS-Recomb strategy is manageable, on the order of between 1 and 10 seconds for summarization. As mentioned in the Conclusion, additional parallelism can be combined with our BFS search to further improve the time and make it comparable to beam search. However, even this version of the algorithm can be "embarrassingly" parallelized across examples to improve efficiency.

Descriptive Statistics We randomly sample 100 data instances from the validation set for each dataset, and they are used by all methods. When sampling is needed, we take 1,000 samples for each data instance, so all the metrics are reported on 100,000 translations/summaries for one dataset.

K Risks

By generating additional outputs from a generation model, we may cause a system to produce outputs that are biased, factually inaccurate, or contain hallucinations. However, we do not believe these risks are substantially increased from the original model. Moreover, because we present many options, we believe our approach more appropriately reflects the model's uncertainty over its output.

[6] https://github.com/WestHealth/pyvis

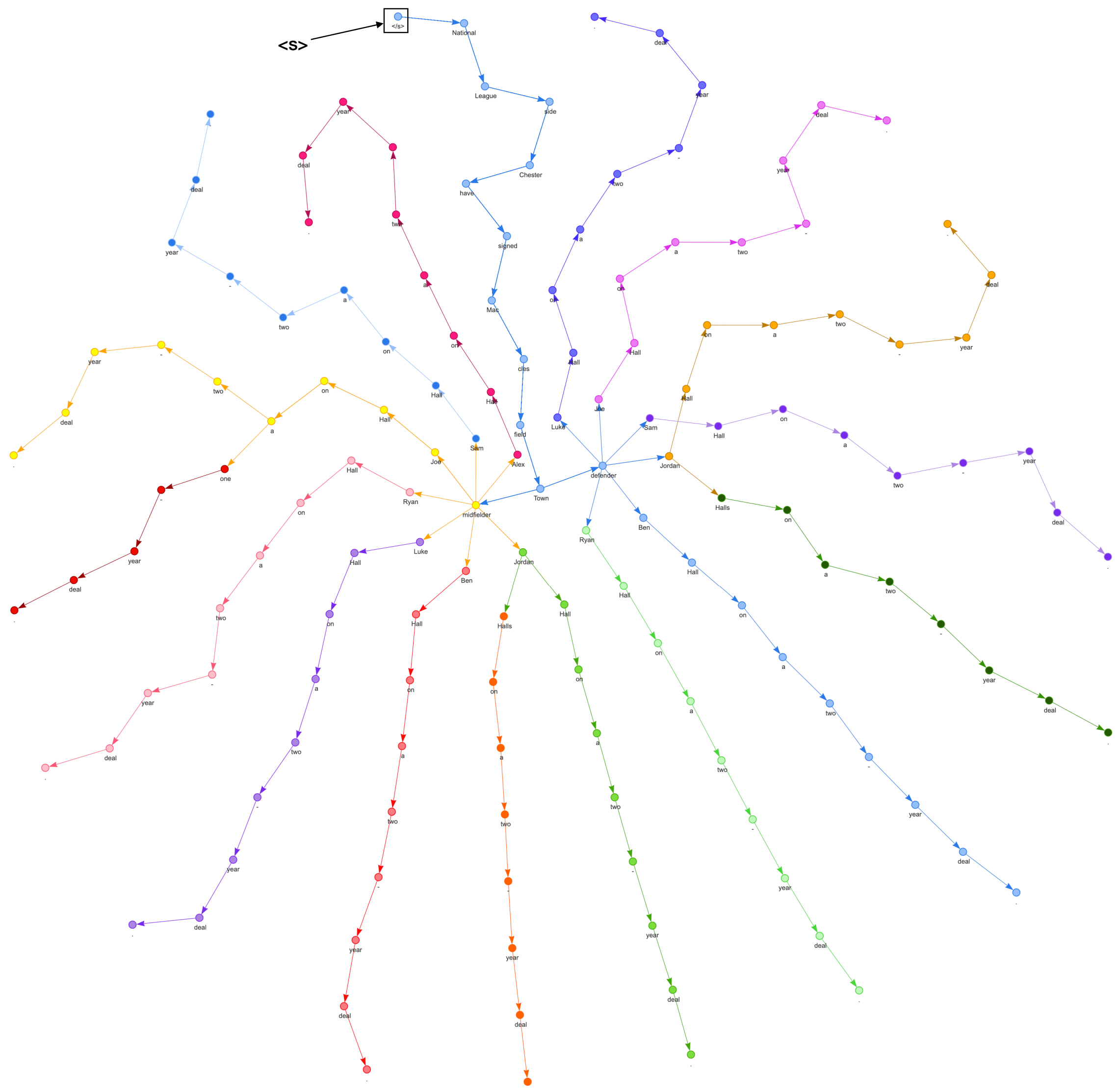


Figure 11: Visualization of one example output for beam search on XSum. n_{SOS} is labeled. Each color represents one unique ending.

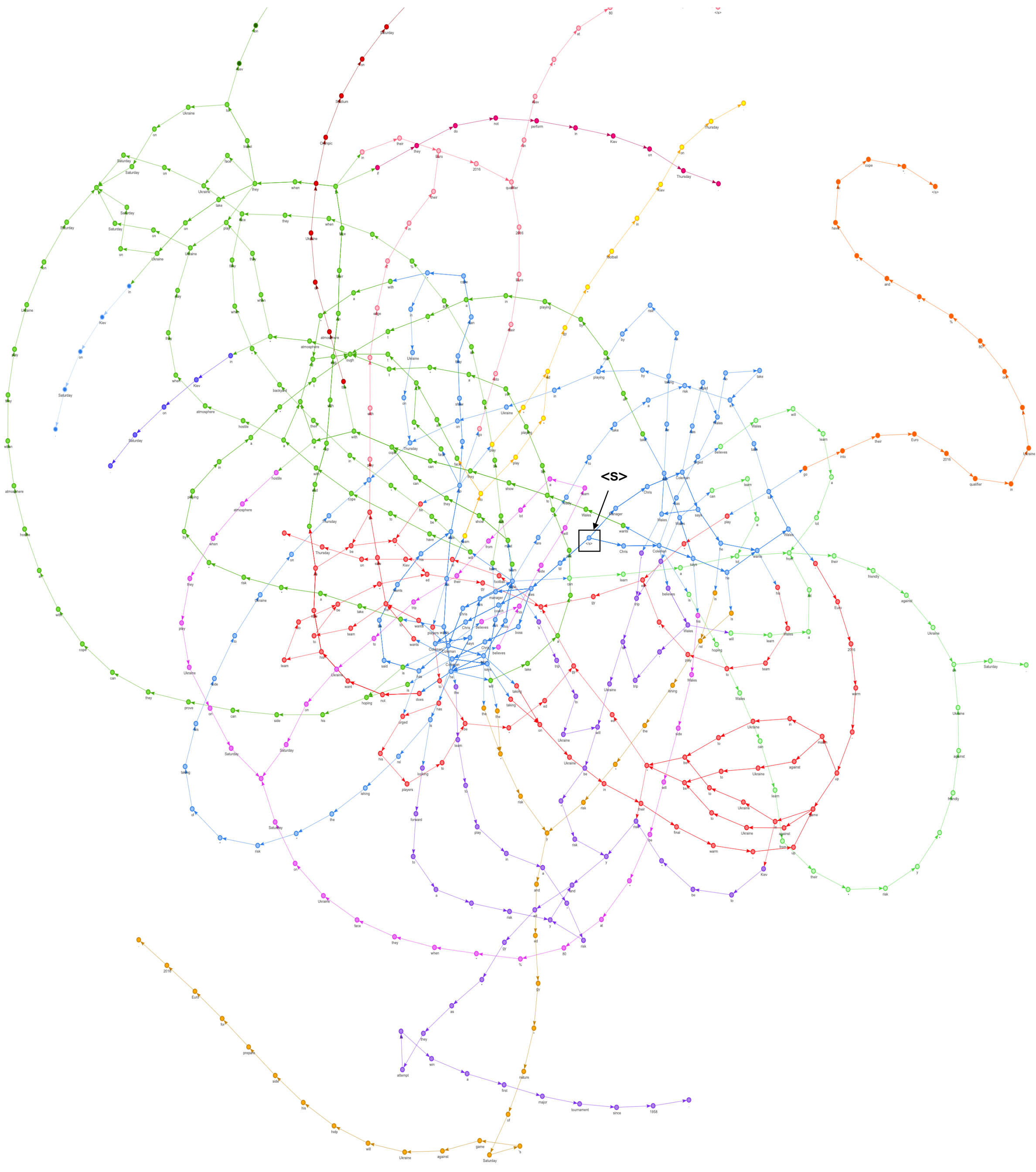


Figure 12: Visualization of one example output for BFSRCB on XSum.

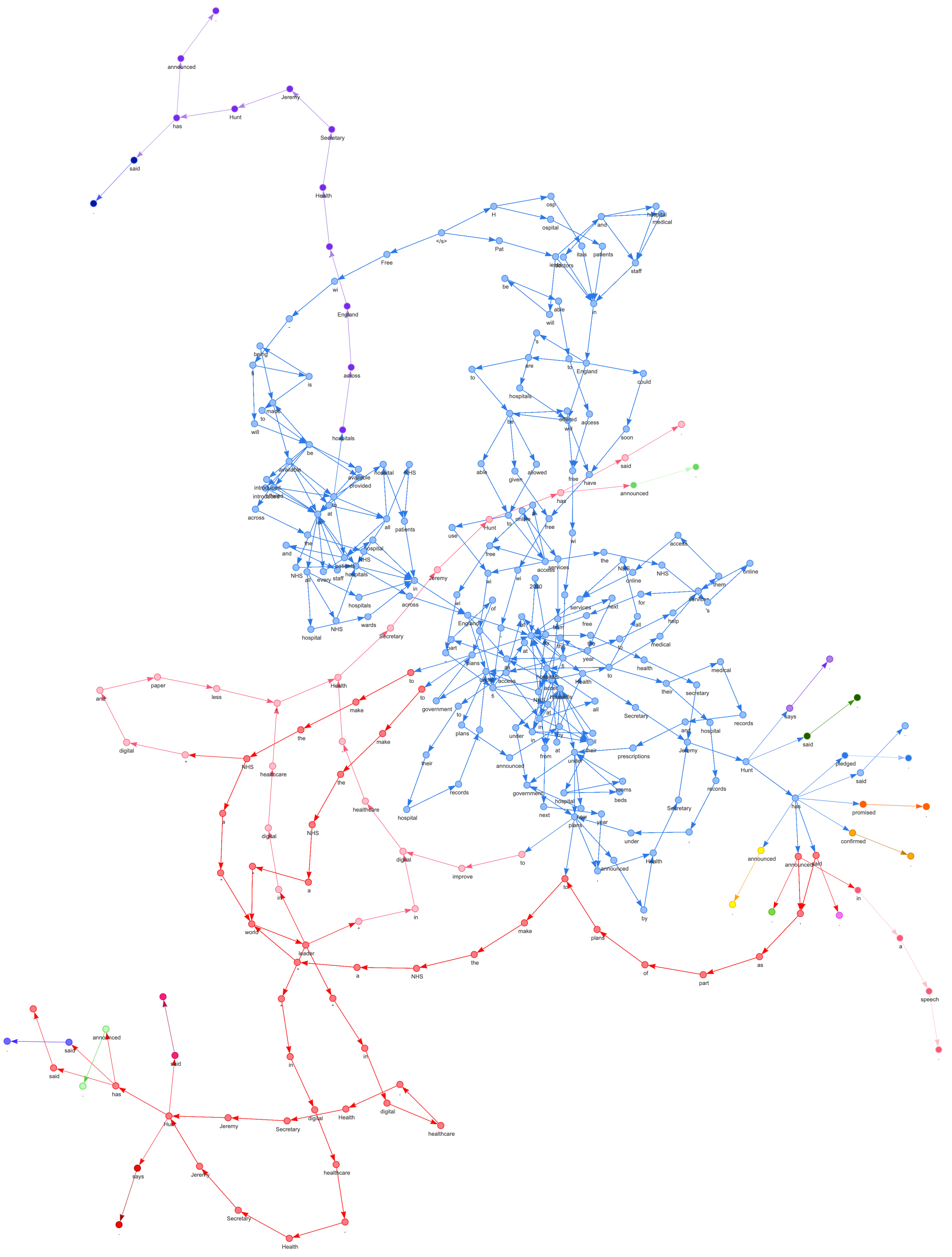


Figure 13: Visualization of one example output for BFSZIP on XSum.

	Diversity					Oracle			Sample			Grm
	↑	↑	↑	↓	↑	↑	↑	↑	≥	≥	≥	↓
Model	|path|	N1	N2	sBl	ED	R1	R2	RL	R1	R2	RL	Err
Greedy	1	22	23	100	0	41.4	17.3	33.5	41.4	17.3	33.5	0.5%
BS	20	42	61	87	31	47.6	26.3	40.3	41.5	17.7	33.6	0.3%
DBS	19	59	91	79	53	47.0	25.5	39.1	38.5	15.9	30.3	0.5%
Ncls$_{0.8}$	20	124	237	57	72	50.4	30.2	44.2	37.4	14.5	29.5	0.5%
Ncls$_{0.9}$	20	143	273	50	76	48.0	28.1	42.2	36.1	13.3	28.5	0.8%
Temp$_{1.5}$	20	170	319	51	82	45.0	26.6	38.5	34.1	11.6	26.3	1.4%
Bfs	30	88	167	68	60	50.8	30.1	44.0	39.0	15.6	30.8	0.4%
					+ Path Recombination							
BSZBeam	4,701	66	118	75	51	52.2	33.0	45.7	40.0	16.0	32.3	0.7%
Ncls$_{0.8}$Rcb	52	167	308	53	79	49.0	28.8	41.8	35.0	13.0	27.8	1.0%
Ncls$_{0.9}$Rcb	36	207	363	50	87	44.6	25.9	38.7	32.1	11.0	25.1	1.7%
BfsRcb	7,758	111	239	65	64	55.2	35.8	49.3	38.5	15.2	30.8	0.8%
BfsZip	95,744	124	274	53	77	55.6	36.8	48.8	36.8	13.2	28.7	1.4%
🍃BfsZip	297	58	92	80	49	49.6	29.2	42.8	38.8	15.2	31.0	0.8%

Table 4: Full results for all methods decoding text summaries on XSum.

Method	Algos	Cand	Len	Dedup
BSZBeam	BS	last step	1	N
Rcb	any	all	1	N
Zip	any	all	n	Y

Table 5: Key differences in path recombination methods. BSZBeam is the recombination method used in Zhang et al. (2018). Algos shows which search or decoding methods this method is used with. Cand is where the merge candidates come from in the lattice. Len reflects how many nodes are recombined per operation. Dedup denotes whether duplicates on the merged branch will be removed from heap.

	Diversity					Or	Sp
	↑	↑	↑	↓	↑	↑	≥
Model	|path|	N1	N2	sBl	ED	Bl	Bl
Greedy	1	32	35	100	0	28.5	28.5
BS	12	42	57	93	13	37.8	27.5
DBS	10	51	73	89	38	33.1	22.7
Ncls$_{0.8}$	12	95	171	72	56	35.4	20.4
Ncls$_{0.9}$	12	116	214	66	73	33.4	17.6
Temp$_{1.5}$	12	150	274	61	89	28.4	13.1
Bfs	17	62	98	85	35	38.8	25.0
			+ Path Recombination				
BSZBeam	17,508	67	117	78	40	46.4	21.2
Ncls$_{0.8}$Rcb	59	151	261	67	78	29.3	16.3
Ncls$_{0.9}$Rcb	32	190	317	53	101	26.9	12.6
BfsRcb	18,663	90	180	74	42	46.6	20.8
BfsZip	49,507	104	213	65	53	45.9	16.7
🍃BfsZip	386	49	70	88	25	39.5	25.7

Table 6: Results on machine translation WMT14 English to French. BfsRcb and BfsZip are strong in both diversity and quality.