Best-First Beam Search

Clara Meister[⋆] **Tim Vieira**[♯] **Ryan Cotterell**[∗,⋆]

[⋆]ETH Zürich [♯]Johns Hopkins University [∗]University of Cambridge
`clara.meister@inf.ethz.ch tim.vieira@gmail.com`
`ryan.cotterell@inf.ethz.ch`

Abstract

Decoding for many NLP tasks requires an effective heuristic algorithm for approximating exact search because the problem of searching the full output space is often intractable, or impractical in many settings. The default algorithm for this job is beam search—a pruned version of breadth-first search. Quite surprisingly, beam search often returns *better* results than exact inference due to *beneficial* search bias for NLP tasks. In this work, we show that the standard implementation of beam search can be made up to 10x faster in practice. Our method assumes that the scoring function is monotonic in the sequence length, which allows us to safely prune hypotheses that cannot be in the final set of hypotheses early on. We devise effective monotonic approximations to popular nonmonontic scoring functions, including length normalization and mutual information decoding. Lastly, we propose a memory-reduced variant of best-first beam search, which has a similar beneficial search bias in terms of downstream performance, but runs in a fraction of the time.

1 Introduction

Beam search is a common heuristic algorithm for decoding structured predictors (e.g., neural machine translation models and transition-based parsers). Because of the widespread adoption of recurrent neural networks and other non-Markov models, traditional dynamic programming solutions, such as the Viterbi algorithm (Viterbi, 1967), are prohibitively inefficient; this makes beam search a common component of many state-of-the-art NLP systems. Despite offering no formal guarantee of finding the highest-scoring hypothesis under the model, beam search yields impressive performance on a variety of tasks—

unexpectedly providing a beneficial search bias over *exact* search for many tasks (Stahlberg and Byrne, 2019).

Within NLP, most research on beam search has focused on altering the log-probability scoring function to return improved results, for example, higher BLEU scores (Wu et al., 2016; Murray and Chiang, 2018; Shu and Nakayama, 2018; Yang et al., 2018) or a more diverse set of outputs (Vijayakumar et al., 2016). However, little work has been done to speed up beam search itself. Filling this gap, this paper focuses on reformulating beam search in order to make it *faster*. We propose best-first beam search, a prioritized version of traditional beam search that is up to an order of magnitude faster in practice while still returning the same set of results. We additionally discuss an even faster heuristic version of our algorithm that further limits the number of candidate solutions, leading to a smaller memory footprint while still finding good solutions.

Concretely, we offer a novel interpretation of beam search as an agenda-based algorithm where traditional beam search is recovered by utilizing a length-based prioritization scheme. We prove that a specific best-first prioritization scheme, as in classic A∗ search (Hart et al., 1968), allows for the elimination of paths that will necessarily fall off the beam; for many scoring functions, including standard log-probability scoring, we can still guarantee the same k hypotheses as traditional beam search are returned. Indeed, our algorithm returns beam search's top hypothesis the first time it encounters a complete hypothesis, allowing the program to stop early. Further, we discuss the application of best-first beam search to several popular scoring functions in the literature (He et al., 2016; Li et al., 2016); this demonstrates that we have a general framework for adapting a variety of rescoring methods and alternate objectives to work with our algorithm.

Empirically, we compare best-first beam search to ordinary beam search on two NLP sequence-to-sequence tasks: neural machine translation (NMT) and abstractive summarization (AS). On NMT, we find that our algorithm achieves roughly a 30% speed-up over traditional beam search with increased gains for larger beams (e.g., $\approx$ 10x for a beam of 500). We find similar results hold for AS. Finally, we show that our memory-reduced version, which limits the number of active hypotheses, leads to additional speed-ups over best-first beam search across beam sizes while maintaining similar BLEU scores.

2 Sequence Transduction

A core operation in structured prediction models is the determination of the highest-scoring output for a given input under a learned scoring model.

$$\mathbf{y}^\star \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \operatorname{score}(\mathbf{x}, \mathbf{y}) \qquad (1)$$

where $\mathbf{x}$ is an input and $\mathcal{Y}(\mathbf{x})$ is a set of well-formed outputs for the input. An important example of (1) is maximum a posteriori (MAP),

$$\mathbf{y}^{\text{MAP}} \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} p(\mathbf{y} \mid \mathbf{x}) \qquad (2)$$

Our work focuses on sequence-to-sequence transduction: predicting an output sequence given an input sequence. One such task is machine translation, wherein a source-language sentence is mapped (''transduced'') to a target-language sentence. While our exposition focuses on sequence-to-sequence prediction, our algorithms are directly applicable to any sequential structured prediction model, such as transition-based parsers (Nivre et al., 2008) and sequence taggers (McCallum et al., 2000; Lafferty et al., 2001).

Notation. Let $\mathbf{x} = \langle x_1, \ldots, x_{N_{\mathbf{x}}} \rangle$ be an input sequence of length $N_{\mathbf{x}}$ and, likewise, let $\mathbf{y} = \langle y_1, \ldots, y_{N_{\mathbf{y}}} \rangle$ be an output sequence of length $N_{\mathbf{y}}$. Each y_t is an element of $\mathcal{V}$, the set of output tokens. Finally, let $\mathcal{Y}(\mathbf{x})$ be the set of all valid output sequences (i.e., complete hypotheses). For the task of language generation, which we focus on experimentally, this set is defined as

$$\mathcal{Y}(\mathbf{x}) \stackrel{\text{def}}{=} \{\text{BOS} \circ \mathbf{v} \circ \text{EOS} \mid \mathbf{v} \in \mathcal{V}^{<n_{max}}\} \qquad (3)$$

where $\circ$ is string concatenation and $\mathcal{V}^{<n_{max}(\mathbf{x})}$ is the set of all subsets of $\mathcal{V}^\star$ of size $< n_{max}(\mathbf{x})$. In

words, every valid sequence begins and ends with distinguished tokens (BOS and EOS, respectively).[1] Furthermore, each sequence has *at most* length $n_{max}(\mathbf{x})$—which is typically dependent on $\mathbf{x}$—a restriction we impose to ensure termination. Some applications may require a stronger coupling between $\mathcal{Y}(\mathbf{x})$ and $\mathbf{x}$ (e.g., $|\mathbf{x}| = |\mathbf{y}|$). We drop the dependence of $\mathcal{Y}$ and n_{max} on $\mathbf{x}$ when it is clear from context.

Scoring. We consider a general additively decomposable scoring model of the form

$$\operatorname{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{N_{\mathbf{y}}} \operatorname{score}(\mathbf{x}, \mathbf{y}_{<t} \circ y_t) \qquad (4)$$

This framework covers a variety of modeling methodologies including probabilistic transducers (both globally and locally normalized) and non-probabilistic models such as maximum-margin techniques (Taskar et al., 2004). Most importantly, (4) covers MAP decoding (2) of neural sequence-to-sequence models à la Sutskever et al. (2014):[2]

$$\operatorname{score}_{s2s}(\mathbf{x}, \mathbf{y}_{<t} \circ y_t) = \log p(y_t \mid \mathbf{y}_{<t}, \mathbf{x}) \qquad (5)$$

We note that (5) is the scoring function used for decoding many language generation models.

Beam search. The worst-case running time of exactly computing (1) is exponential in n_{max}; namely, $\mathcal{O}(|\mathcal{V}|^{n_{max}})$.[3] Beam search is a commonly used approximation to (1) in NMT and language generation tasks. It is used in many (if not most) state-of-the-art NLP systems (Wu et al., 2016; Serban et al., 2017; Edunov et al., 2018; Yang et al., 2019). Beam search may be understood as a pruned version of the classic path-search algorithm, breadth-first search (BFS), where the breadth is narrowed to the beam size k. Pseudocode is given in (1).

Although, beam search does not solve (1) exactly, it is a surprisingly useful approximation for NLP models. In many settings, beam

---

[1] BOS and EOS are typically members of $\mathcal{V}$. Often, EOS counts towards the n_{max} length limit while BOS does not. This is reflected in (3).

[2] To see why, apply $\exp$ (an order-preserving transformation): $\exp(\operatorname{score}_{s2s}(\mathbf{x}, \mathbf{y})) = \exp\left(\sum_{t=1}^{N_{\mathbf{y}}} \log p(y_t \mid \mathbf{y}_{<t}, \mathbf{x})\right) = \prod_{t=1}^{N_{\mathbf{y}}} p(y_t \mid \mathbf{y}_{<t}, \mathbf{x}) = p(\mathbf{y} \mid \mathbf{x})$.

[3] This can be improved if, for example, $\operatorname{score}(\cdot, \cdot)$ admits a low-order Markov factorization (Viterbi, 1967; Vieira et al., 2016). We do not discuss that setting in this paper because it limits the scoring model's expressive power.

Algorithm 1 Standard beam search[4]

Input: $\mathbf{x}$: source sentence
$\quad\quad k$: maximum beam size
$\quad\quad n_{max}$: maximum hypothesis length
$\quad\quad \text{score}(\cdot, \cdot)$: scoring function
1: $B_0 \leftarrow \{\langle 0, \text{BOS} \rangle\}$
2: **for** $t \in \{1, \ldots, n_{max}-1\}$:
3: $\quad B \leftarrow \varnothing$
4: $\quad$ **for** $\langle s, \mathbf{y} \rangle \in B_{t-1}$:
5: $\quad\quad$ **if** $\mathbf{y}.\text{last}() = \text{EOS}$:
6: $\quad\quad\quad B.\text{add}(\langle s, \mathbf{y} \rangle)$
7: $\quad\quad\quad$ **continue**
8: $\quad\quad$ **for** $y \in \mathcal{V}$:
9: $\quad\quad\quad s \leftarrow \text{score}(\mathbf{x}, \mathbf{y} \circ y)$
10: $\quad\quad\quad B.\text{add}(\langle s, \mathbf{y} \circ y \rangle)$
11: $\quad B_t \leftarrow B.\text{top}(k)$
12: **return** $B.\text{max}()$

search *outperforms* exact methods in terms of downstream evaluation (Koehn and Knowles, 2017; Stahlberg and Byrne, 2019). For the remainder of this paper, we will pivot our attention away from exact solutions to (1) to exact solutions to the *beam search* output.

Definition 2.1. *k-optimal hypothesis.* We say that a hypothesis is k-optimal if it is the top hypothesis returned by beam search with beam size k.

3 A* Beam Search

We develop a meta-algorithm that is parameterized by several choice points. Our general search algorithm for decoding (Alg. 2) takes an arbitrary prioritization function, stopping criterion, and search heuristic. With certain values of these attributes, we recover many common search algorithms: greedy search, beam search, best-first search (Dijkstra, 1959), and A* search (Hart et al., 1968). We propose an alternate prioritization function for beam search that allows for faster decoding while still returning the same k-optimal set of hypotheses.

---

[4]Often, the score function is additively decomposable in t, such as (5). Implementations can exploit this fact to make each score evaluation (line 9) $\mathcal{O}(1)$ rather than $\mathcal{O}(t)$. We did not make this implementation detail explicit in Alg. 1 or Alg. 2 for generality and simplicity.

Algorithm 2 General decoding scheme.[4,5] Highlighted sections are choice points in the algorithm for which values determine the search strategy. See § 3.1 for detailed explanation.

Input: $\mathbf{x}$: source sentence
$\quad\quad n_{max}$: maximum hypothesis length
$\quad\quad \text{score}(\cdot, \cdot)$: scoring function
$\quad\quad \oslash$: comparator ①
$\quad\quad \text{stop}(\cdot, \cdot)$: stopping criterion ②
$\quad\quad k$: maximum beam size ③
$\quad\quad h(\cdot, \cdot)$: heuristic function ④
1: $\mathcal{Q} \leftarrow \text{priority\_queue}(\oslash)$
2: $\mathcal{Q}.\text{push}(\langle 0, \text{BOS} \rangle)$
3: $\text{POPS} \leftarrow \text{counter}()$
4: **while not** $\text{stop}(\mathcal{Q})$ **and not** $\mathcal{Q}.\text{empty}()$:
5: $\quad \langle s_h, \mathbf{y} \rangle \leftarrow \mathcal{Q}.\text{pop}()$
6: $\quad$ **if** $\text{POPS}[|\mathbf{y}|] \geq k$ **or** $|\mathbf{y}| > n_{max}$:
7: $\quad\quad$ **continue**
8: $\quad \text{POPS}[|\mathbf{y}|] \leftarrow \text{POPS}[|\mathbf{y}|] + 1$
9: $\quad$ **if** $\mathbf{y}.\text{last}() = \text{EOS}$:
10: $\quad\quad \mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ \text{EOS} \rangle)$
11: $\quad$ **else**:
12: $\quad\quad$ **for** $y \in \mathcal{V}$:
13: $\quad\quad\quad s \leftarrow \text{score}(\mathbf{x}, \mathbf{y} \circ y)$
14: $\quad\quad\quad s_h \leftarrow s + h(\mathbf{x}, \mathbf{y} \circ y)$
15: $\quad\quad\quad \mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ y \rangle)$
16: **return** $\mathcal{Q}.\text{pop}()$ **if not** $\mathcal{Q}.\text{empty}()$ **else null**

3.1 Choice Points of 2

Here we review the components of our meta algorithm (the highlighted sections in Alg. 2) that can be varied to recover different search strategies:

① $\oslash : \mathbf{y} \times \mathbf{y} \rightarrow \{\text{True}, \text{False}\}$. A priority queue $\mathcal{Q}$ maintains the set of active hypotheses. Elements in this set are ordered according to a generic comparator $\oslash$. When its $\text{peek}()$ (or $\text{pop}()$) methods are called, the first element ordered by $\oslash$ is returned (or returned and removed).

② $\text{stop}(\cdot) : \text{Collection}\langle \mathbf{y} \rangle \rightarrow \{\text{True}, \text{False}\}$. The algorithm terminates according to configurable stopping criterion based on the current set of elements in $\mathcal{Q}$.

---

[5]If the last token of $\mathbf{y}'$ is the end symbol (e.g., EOS), then $\mathbf{y}'$ is not expanded any further. One can either regard $\mathbf{y}'$ as any other hypothesis albeit with $\mathbf{y}' \circ y_t = \mathbf{y}'$ or keep appending EOS (i.e., $\mathbf{y}' \circ y_t = \mathbf{y}' \circ \text{EOS}$) so that time step and length can be regarded as synonymous. We adopt the latter standard for comparability with subsequent algorithms.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00346/1923790/tacl_a_00346.pdf by guest on 20 May 2023

	Beam Search	Best-First Beam Search	A* Beam Search
①	$\langle s_h, \mathbf{y}\rangle \oslash \langle s_h', \mathbf{y}'\rangle \iff \|\mathbf{y}\| < \|\mathbf{y}\|'$ **or** $(\|\mathbf{y}\| = \|\mathbf{y}\|'$ **and** $s_h \geq s_h')$	$\langle s_h, \mathbf{y}\rangle \oslash \langle s_h', \mathbf{y}'\rangle \iff s_h > s_h'$ **or** $(s_h = s_h'$ **and** $\|\mathbf{y}\| < \|\mathbf{y}'\|)$	$\langle s_h, \mathbf{y}\rangle \oslash \langle s_h', \mathbf{y}'\rangle \iff s_h > s_h'$ **or** $(s_h = s_h'$ **and** $\|\mathbf{y}\| < \|\mathbf{y}'\|)$
②	$\text{stop}(\mathcal{Q}) \iff$ $\mathbf{y}.\text{last}() = \text{EOS} \quad \forall \mathbf{y} \in \mathcal{Q}$	$\text{stop}(\mathcal{Q}) \iff$ $\mathcal{Q}.\text{peek}().\text{last}() = \text{EOS}$	$\text{stop}(\mathcal{Q}) \iff$ $\mathcal{Q}.\text{peek}().\text{last}() = \text{EOS}$
③	$k = $ beam size	$k = $ beam size	$k = $ beam size
④	0	0	any admissible heuristic

	Breadth-First Search	Best-First Search	A* Search
①	$\langle s_h, \mathbf{y}\rangle \oslash \langle s_h', \mathbf{y}'\rangle \iff \|\mathbf{y}\| < \|\mathbf{y}\|'$ **or** $(\|\mathbf{y}\| = \|\mathbf{y}\|'$ **and** $s_h \geq s_h')$	$\langle s_h, \mathbf{y}\rangle \oslash \langle s_h', \mathbf{y}'\rangle \iff s_h > s_h'$ **or** $(s_h = s_h'$ **and** $\|\mathbf{y}\| < \|\mathbf{y}'\|)$	$\langle s_h, \mathbf{y}\rangle \oslash \langle s_h', \mathbf{y}'\rangle \iff s_h > s_h'$ **or** $(s_h = s_h'$ **and** $\|\mathbf{y}\| < \|\mathbf{y}'\|)$
②	$\text{stop}(\mathcal{Q}) \iff$ $\mathbf{y}.\text{last}() = \text{EOS} \quad \forall \mathbf{y} \in \mathcal{Q}$	$\text{stop}(\mathcal{Q}) \iff$ $\mathcal{Q}.\text{peek}().\text{last}() = \text{EOS}$	$\text{stop}(\mathcal{Q}) \iff$ $\mathcal{Q}.\text{peek}().\text{last}() = \text{EOS}$
③	$k = \infty$	$k = \infty$	$k = \infty$
④	0	0	any admissible heuristic

Table 1: Values at choice points for various search algorithms. Note that any admissible heuristic may be used for variants of A* search.

③ $k \in \mathbb{N}_{>0}$. Only k paths of a given length are considered. If the algorithm has already encountered k paths of a given length, subsequent paths of that length are not evaluated. If we take $k = \infty$, we recover unpruned search algorithms.

④ $h(\cdot, \cdot) : \mathbf{x} \times \mathbf{y} \to \mathbb{R}$. A heuristic function $h(\mathbf{x}, \mathbf{y})$ can be used during search to change the priority in which paths are evaluated. We note that with pruning, a heuristic may change the value of the k-optimal hypothesis (see § 4.1).

Recovering Beam Search. To recover beam search from Algorithm 2, we use the choice points from Table 1. Explicitly, the comparator prioritizes hypotheses from earlier time steps first, but breaks ties with the hypotheses' scores under the model. We note that while the standard algorithm for beam search does not prioritize by score within a time step, variations of the algorithm use this strategy so they can use early-stopping strategies (Klein et al., 2017; Huang et al., 2017). Beam search terminates once either *all* hypotheses end in EOS or the queue is empty (i.e., when the k beams have been extended n_{max} time steps but none end in EOS). In the second case, no complete hypothesis is found. Finally, choosing the heuristic $h(\mathbf{x}, \mathbf{y}) = 0$ makes the algorithm a case of standard best-first search.

Note that, while standard beam search returns a set, Alg 2 only returns the k-optimal hypothesis.

This behavior is sufficient for the majority of use cases for beam search. However, if the full set of k hypotheses is desired, the stopping criterion can be changed to evaluate true only when k hypotheses are complete. Under the other beam search settings, this would probably return the same set as beam search (see § 4.1).

Recovering A*. To recover the traditional A* search algorithm, we use the comparator that prioritizes hypotheses with a higher score first; ties are broken by hypothesis length. The algorithm terminates when the first item of $\mathcal{Q}$ contains an EOS. If we take $k = \infty$, best-first beam search recovers A*. Any admissible heuristic may be used for $h(\mathbf{x}, \mathbf{y})$.

Definition 3.1. *Admissible Heuristic.* A heuristic h is admissible if it never overestimates the future cost—or underestimates the future reward—of continuing down a path.

3.2 Best-First Beam Search

In its original form, A* search may traverse the entire $\mathcal{O}(|\mathcal{V}|^{n_{max}})$ graph, which as discussed earlier, is intractable for many decoding problems. While standard beam search addresses this problem by limiting the search space, it still has computational inefficiencies—namely, we *must* analyze k hypotheses of a given length (i.e., time step), regardless of how poor their scores may already be, before considering longer hypotheses.

However, prioritization by length is not strictly necessary for finding a k-optimal hypothesis. As is done in A*, we can use score as the prioritization scheme and still guarantee optimality–or k-optimality–of the paths returned by the algorithm.

We define A* beam search as the A* algorithm where breadth is limited to size k. Further, we define best-first beam search as the case of A* beam search when no heuristic is used (see Table 1 for algorithm settings). This formulation has two large advantages over standard beam search: (1) we gain the ability to remove paths from the queue that are guaranteed to fall off the beam and (2) we can terminate the algorithm the first time a complete hypothesis is encountered. We can therefore reduce the computation required for decoding while still returning the same set of results.

The mathematical property that makes this short-circuiting of computation possible is the monotonicity of the scoring function. Note that not all scoring functions are monotonic, but many important ones are, including log-probability (5). We discuss effective approximations for popular non-monotonic scoring functions in § 5.

Definition 3.2. *Monotonicity.* A scoring function score$(\cdot, \cdot)$ is monotonic in t if for all $\mathbf{x}$, $\mathbf{y}_{<t} = \langle y_1 \ldots y_{t-1} \rangle, y_t \in \mathcal{V}, 1 \leq t \leq n_{max}$

$$\text{score}(\mathbf{x}, \mathbf{y}_{<t}) \geq \text{score}(\mathbf{x}, \mathbf{y}_{<t} \circ y_t)$$

Clearly, (5) is a monotonic scoring function in t because score$_{s2s} \leq 0$, that is, the score of a partial hypothesis $\mathbf{y}_{<t}$ can only decrease if we extend it by another symbol y_t. This implies we can order our search according to score$(\mathbf{x}, \mathbf{y}_{<t})$ without fear of overlooking a hypothesis whose score would increase over time. Furthermore, once k hypotheses of a given length t have been evaluated, we no longer need to consider any hypothesis where $|\mathbf{y}| < t$ because such hypotheses would necessarily fall off the beam. We can therefore remove such hypotheses from the queue and avoid wasting computational power on their evaluation. We prove this formally in § 4.1.

Another implication of the monotonicity property of score is that we may terminate best-first beam search once a hypothesis containing EOS is encountered (i.e., the end state is found). If the full set of k complete hypotheses is desired, then we simply continue until k hypotheses have

reached EOS. We prove the k-optimality of these hypotheses under best-first beam search in § 4.1.

3.3 Implementation Details

Standard beam search forms a separate set of active hypotheses for each time step, that is, each B_t is its own set. Once B_t has been narrowed down to the top k, the previous $B_{<t}$ can be forgotten. However in best-first beam search, because hypotheses are not evaluated in order of time step, we may need to keep B_t from several time steps at any given point.

A naive implementation of best-first beam search is to keep a single priority queue with all the active hypotheses ordered by current score. However, each push to the queue would then require $\mathcal{O}(\log(n_{max}k|\mathcal{V}|))$ time. We can reduce this runtime by instead keeping a priority queue of beams, where the priority queue is ordered by the highest-scoring hypothesis from each beam. Further, each beam can be represented by a min-max queue (Atkinson et al., 1986); this allows us to limit the size of B_t to k: we can check in $\mathcal{O}(1)$ time if a hypothesis is in the top-k before adding it to B_t.

A potential inefficiency, which we avoid, comes from updating B_{t+1}, which we must do when evaluating a hypothesis from B_t. Because all beams are stored in a queue, there is no guarantee of the location in the queue of B_{t+1}. To avoid $\mathcal{O}(n_{max})$ lookup, we can keep a pointer to each beam, indexed by t making the lookup $\mathcal{O}(1)$. However, we acquire a $\mathcal{O}(\log n_{max})$ term to update the queue of beams as B_{t+1} may change priority.

Memory-Reduced Best-First Beam Search. A major drawback of the A* algorithm is its memory usage, which in the worst-case is $\mathcal{O}(b^d)$ for breadth width b and maximum depth d. In the A* formulation of beam search, where the breadth width is limited to the beam size, this amounts to worst-case $\mathcal{O}(k \cdot n_{max})$ memory usage, where standard beam search has $\mathcal{O}(k)$ memory usage. Whereas in many settings the multiplicative factor may be insignificant, for neural sequence models it can be prohibitive; this is due to the large amount of memory required to store each hypothesis (e.g., prior hidden states needed to compute subsequent scores for scoring functions parameterized by neural networks).

We propose a variant of best-first beam search that limits memory usage, that is, the queue capacity. Specifically, if we reach the chosen queue capacity, we remove the worst scoring active hypothesis from the earliest active time step. This can easily be done in $\mathcal{O}(1)$ time given our pointer to each beam.

4 Algorithm Analysis

4.1 Correctness

We show the equivalence of the top hypothesis[6] returned by beam search and best-first beam search when $\text{score}(\cdot, \cdot)$ is monotonically decreasing in t, length-based prioritization is used, and the beam size k is the same for both algorithms. Without loss of generality, we hold $\mathbf{x}$ constant in all the following proofs.

Note that we take the terms *pop* and *push* from queue terminology. Specifically, ''popping a hypothesis'' refers to making it past line 7 of Alg. 2, where a hypothesis $\mathbf{y}$ is expanded by $y_t \in \mathcal{V}$. In path search terminology, this would be equivalent to visiting a node and adding the edges from that node as potential paths to explore. Lastly, we refer to the priority queue used by beam search and best-first beam search as $\mathcal{Q}_{\text{BS}}$ and $\mathcal{Q}_{\text{A}^*}$, respectively.

Lemma 4.1. *Best-first beam search evaluates all hypotheses of a given length t in order of their score.*

Proof. We prove the lemma by induction. The lemma holds trivially for the base case of hypotheses of length 0 because the only hypothesis of length 0 is $\langle \text{BOS} \rangle$.

Now, by the inductive hypothesis, suppose Lemma 4.1 holds for all hypotheses of length $< t$. We will show it must also hold for hypotheses of length t. Consider two competing hypotheses: $\mathbf{y} = \mathbf{y}_{<t} \circ y_t$ and $\mathbf{y}' = \mathbf{y}'_{<t} \circ y'_t$. Note that $|\mathbf{y}_{<t}| = |\mathbf{y}'_{<t}| = t - 1$. Suppose $\text{score}(\mathbf{x}, \mathbf{y}') < \text{score}(\mathbf{x}, \mathbf{y})$.

Case 1: $\text{score}(\mathbf{x}, \mathbf{y}'_{<t}) < \text{score}(\mathbf{x}, \mathbf{y}_{<t})$. Then by induction, $\mathbf{y}_{<t}$ popped first and $\mathbf{y}$ is pushed to $\mathcal{Q}$ before $\mathbf{y}'$. Because $\text{score}(\mathbf{x}, \mathbf{y}') < \text{score}(\mathbf{x}, \mathbf{y})$, $\mathbf{y}$ will be popped before $\mathbf{y}'$.

Case 2: $\text{score}(\mathbf{x}, \mathbf{y}_{<t}) < \text{score}(\mathbf{x}, \mathbf{y}'_{<t})$. Then by induction, $\mathbf{y}'_{<t}$ is popped first and $\mathbf{y}'$ is added to $\mathcal{Q}$ before $\mathbf{y}$. But, because $\text{score}(\mathbf{x}, \mathbf{y}') < \text{score}(\mathbf{x}, \mathbf{y}) \leq \text{score}(\mathbf{x}, \mathbf{y}_{<t})$ by monotonicity, then $\mathbf{y}_{<t}$ will be popped before $\mathbf{y}'$. Consequently, $\mathbf{y}$ will be pushed to $\mathcal{Q}$ before $\mathbf{y}'$ is evaluated. By the rules of the priority queue $\mathbf{y}$ will be evaluated before $\mathbf{y}'$.

Case 3: $\text{score}(\mathbf{x}, \mathbf{y}') = \text{score}(\mathbf{x}, \mathbf{y})$. The lemma holds if either $\mathbf{y}$ or $\mathbf{y}'$ is popped first.

By the principle of induction, Lemmma 4.1 holds for all $t \in \mathbb{N}_{>0}$. $\square$

Lemma 4.2. *The first hypothesis that best-first beam search pops that ends in EOS is k-optimal.*

Proof. Let $\mathbf{y}$ be the first hypothesis popped by best-first beam search ending in EOS. By rules of the priority queue, no other active hypothesis has a higher score than $\mathbf{y}$. Additionally, by monotonicity of the scoring function, no other hypothesis can subsequently have score greater than $\mathbf{y}$. Therefore $\mathbf{y}$ must be k-optimal. $\square$

Lemma 4.3. *If best-first beam search pops a hypothesis, then beam search necessarily pops that same hypothesis.*

Proof. We prove the lemma by induction on hypothesis length. The base case holds trivially: For hypotheses of length 0, both best-first beam search and beam search must pop the $\langle \text{BOS} \rangle$ as it is the only item in the queue after initialization.

By the inductive hypothesis, suppose Lemma 4.3 holds for hypotheses of length $< t$. Suppose best-first beam search pops a hypothesis $\mathbf{y} = \mathbf{y}_{<t} \circ y_t$ of length t.

Case 1: Best-first beam search pops k hypotheses of length $t - 1$ before popping $\mathbf{y}$, which is of length t. The sets of hypotheses of length $t - 1$ that each algorithm pops are necessarily the same by the inductive hypothesis and the fact that they have the same cardinality. If best-first beam search pops $\mathbf{y}$, which is of length t, then it must be in the top-k highest-scoring hypotheses of length t in $\mathcal{Q}_{\text{A}^*}$ by the rules of the priority queue. Consequently, it must be in the top-k in $\mathcal{Q}_{\text{BS}}$.

Case 2: Best-first beam search has popped fewer than k hypotheses of length $t - 1$ before popping $\mathbf{y}$. Then, all remaining hypotheses of length $t - 1$ in $\mathcal{Q}_{\text{A}^*}$ must have $\text{score}(\mathbf{x}, \mathbf{y}'_{<t}) < \text{score}(\mathbf{x}, \mathbf{y})$ by the rules of the priority queue. By the monotonicity of the score function,

all extensions of those $\mathbf{y}'_{<t}$ will also have $\text{score}(\mathbf{x}, \mathbf{y}'_{<t} \circ y'_t) < \text{score}(\mathbf{x}, \mathbf{y})$. Because none of $\mathbf{y}'_{<t} \circ y'_t$ has greater score than $\mathbf{y}$, $\mathbf{y}$ must be in B_t. □

Corollary 4.3.1. *Best-first beam search will never pop more hypotheses than beam search.*

Theorem 4.4. *Once best-first beam search has popped k hypotheses of length t, hypotheses from time steps $< t$ do not need to be popped.*

Proof. This follows from Lemma 4.1. If k hypotheses of length t have been popped, then these must be the top-k hypotheses from time step t. Therefore no hypothesis from time step $< t$ that is still in $\mathcal{Q}_{\text{A}^*}$ would be in the top-k at time step t. □

Theorem 4.5. *Let $\mathcal{H}_{\text{BS}}$ and $\mathcal{H}_{\text{A}}$ be the set of k hypotheses returned by beam search and best-first beam search, respectively. $\mathcal{H}_{\text{BS}} = \mathcal{H}_{\text{A}}$.*

Proof. Because $|\mathcal{H}_{\text{BS}}| = |\mathcal{H}_{\text{A}}| = k$, we only need to show $\mathbf{y} \in \mathcal{H}_{\text{BS}} \implies \mathbf{y} \in \mathcal{H}_{\text{A}}$.

Suppose, by way of contradiction, there exists a hypothesis $\mathbf{y} \in \mathcal{H}_{\text{BS}}$ such that $\mathbf{y} \notin \mathcal{H}_{\text{A}}$. If $\mathbf{y} \notin \mathcal{H}_{\text{A}}$ then we must not pop the prefix $\mathbf{y}_{<t}$ (where $\mathbf{y} = \mathbf{y}_{<t} \circ \mathbf{y}_{t:|\mathbf{y}|}$) for some time step $t < |\mathbf{y}|$.

Case 1: At some time step $t + j$ ($j \geq 0$), we pop k partial hypotheses $\{\mathbf{y}^{(1)}_{\leq t+j}, \ldots, \mathbf{y}^{(k)}_{\leq t+j}\}$ where $\mathbf{y}_{\leq t+j} \notin \{\mathbf{y}^{(1)}_{\leq t+j}, \ldots, \mathbf{y}^{(k)}_{\leq t+j}\}$. By Lemma 4.1, it must be that $\text{score}(\mathbf{x}, \mathbf{y}^{(i)}_{\leq t+j}) > \text{score}(\mathbf{x}, \mathbf{y}_{\leq t+j})$ $\forall i \in 1, \ldots, k$. This implies that for beam search, $\mathbf{y}_{\leq t+j}$ would not be in the top-k paths at time step $t + j$ since by Lemma 4.3, paths $\{\mathbf{y}^{(1)}_{\leq t+j}, \ldots, \mathbf{y}^{(k)}_{\leq t+j}\}$ would also be evaluated by beam search. Therefore $\mathbf{y}$ cannot be in $\mathcal{H}_{\text{BS}}$, which is a contradiction.

Case 2: For no time step $t + j$ ($j \geq 0$) do we pop k paths. This can only happen if the algorithm stops early, namely, we have found k complete hypotheses $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}$. If this is the case, then by rules of the priority queue, each $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}$ must have score greater than $\text{score}(\mathbf{x}, \mathbf{y}_{<t})$. By monotonicity of the score function, $\text{score}(\mathbf{x}, \mathbf{y}^{(i)}) > \text{score}(\mathbf{x}, \mathbf{y})$. This implies $\mathbf{y}$ cannot be in $\mathcal{H}_{\text{BS}}$, which is a contradiction. □

Non-monotonic Scoring Functions. Non-monotonic scoring functions (Definition 3.2) break the assumptions of § 4.1, in which case best-first beam search is not guaranteed to return a k-optimal hypothesis. However, when the scoring function is boundable from above, we can alter the original stopping criterion (②in Alg. 2) such that k-optimality is again guaranteed.

Given our assumed restriction on the search space—namely, $|\mathbf{y}^* \in \mathcal{Y}(\mathbf{x})| \leq n_{max}(\mathbf{x})$—we can upper-bound the maximal score of any hypothesis under the scoring function in use. Formally, for any function score we have:

$$\text{stop}(\mathcal{Q}) \iff$$
$$\text{score}(\mathbf{x}, \hat{\mathbf{y}}) \geq \text{score}(\mathbf{x}, \mathbf{y}') + \mathcal{U}(\mathbf{x}, \mathbf{y}')$$
$$\forall \mathbf{y}' \in \mathcal{Q} \quad (6)$$

where $\hat{\mathbf{y}}$ is the best complete hypothesis found so far and $\mathcal{U}(\mathbf{x}, \mathbf{y}')$ is the score function-dependent upper bound on how much the score of $\mathbf{y}'$ can increase as $\mathbf{y}'$ is expanded further.[7] In this situation, best-first beam search only terminates once no other hypothesis in $\mathcal{Q}$ can have a score greater than the best finished hypothesis. We note that Huang et al. (2017) use a similar scheme for optimal stopping with bounded length normalization. We discuss examples of non-monotonic scoring functions in § 5.

A Note on Heuristics. Our analysis shows the equivalence of beam search and best-first beam search, that is, when $h(\mathbf{x}, \mathbf{y}) = 0$. The analysis does *not* hold for arbitrary admissible heuristics. A poor heuristic (e.g., one that grossly overestimates the future score of continuing down one path) may cause other items to be pruned from best-first beam search that otherwise would have remained on the beam in standard beam search.

4.2 Runtime

Theorem 4.6. *The runtime of best-first beam search is $\mathcal{O}(n_{max}k(|\mathcal{V}|\log(k) + \log(n_{max})))$*

Proof. We pop at most $n_{max} \cdot k$ items. Each pop requires us to push $|\mathcal{V}|$ items. Each push requires $\log(k)$ time when the priority queue is implemented with a min–max heap (Atkinson et al., 1986) and incrementally pruned so that it has no more than k items. After pushing those $|\mathcal{V}|$ items, we have to perform a percolation in the priority queue of priority queues, which requires $\log(n_{max})$ time. This yields $\mathcal{O}(n_{max}k(|\mathcal{V}|\log(k) + \log(n_{max})))$ time. □

Theorem 4.7. *The runtime of standard beam search is $\mathcal{O}(n_{max}k|\mathcal{V}|\log(k))$.*

---

[7]For monotonic scoring functions, we have $\mathcal{U}(\mathbf{x}, \mathbf{y}') = 0$.

Proof. The proof is the same as Theorem 4.6, but we can forgo the percolation step in the queue of queues because standard beam search proceeds in order of hypothesis length. This yields $\mathcal{O}(n_{max} k |\mathcal{V}| \log(k))$. $\qquad\square$

Although the theoretical bound of best-first beam search has an additional log factor compared with standard beam search, we find this to be negligible in practice. Rather, we find number of calls to score, the scoring function under our model (e.g., a neural network), is often the bottleneck operation when decoding neural networks (see § 6 for empirical evidence). In terms of this metric, the beam search algorithm makes $\mathcal{O}(k n_{max})$ calls to score, as score is called once for each active hypothesis in B and B may evolve for n_{max} rounds. The worst-case number of calls to score will be the same as for beam search, which follows from Lemma 4.3.

5 Scoring Functions

Even before the findings of Stahlberg and Byrne (2019), it was well known that the best-scoring hypothesis with respect to the traditional likelihood objective can be far from ideal in practice (Wu et al., 2016; Murray and Chiang, 2018; Yang et al., 2018). For language generation tasks specifically, the results returned by neural models using the standard scoring function are often short and default to high-frequency words (Vinyals and Le, 2015; Shen et al., 2016).

To alleviate such problems, methods that revise hypothesis scores to incorporate preferences for longer, less repetitive, or more diverse options have been introduced and are often used in practice. While most such techniques change the scoring function such that it is no longer monotonic, we can still guarantee the k-optimality of the returned hypothesis for (upper) bounded scoring functions using the methods discussed in § 4.1. In the remainder of this section, we present alternate scoring schemes adapted to work with best-first beam search. Additionally, we present several heuristics which, while breaking the k-optimality guarantee, provide another set of decoding strategies worth exploring.

Length Normalization. Length normalization is a widely used hypothesis scoring method that aims to counteract the propensity for shorter sequences to have higher scores under neural mod-

els; this is done by normalizing scores by hypothesis length (see Murray and Chiang [2018] for more detail).

For early stopping in beam search with length normalization, Huang et al. (2017) propose bounding the additive length reward as the minimum of a pre-determined optimal sequence length ratio r and the final sequence length $N_{\mathbf{y}}$:

$$\begin{aligned} \text{score}_{\text{LN}}(\mathbf{x}, \mathbf{y}) &= \text{score}(\mathbf{x}, \mathbf{y}) \\ &+ \beta \cdot \min\{r|\mathbf{x}|, N_{\mathbf{y}}\} \end{aligned} \tag{7}$$

where β is the scaling parameter for the reward. We note, however, that the same can be done with the maximum sequence length n_{max} such that the traditional length reward used by He et al. (2016) is recovered:

$$\begin{aligned} \text{score}_{\text{LN}}(\mathbf{x}, \mathbf{y}) &= \text{score}(\mathbf{x}, \mathbf{y}) + \beta \min\{n_{max}, N_{\mathbf{y}}\} \\ &= \text{score}(\mathbf{x}, \mathbf{y}) + \beta N_{\mathbf{y}} \end{aligned} \tag{8}$$

We formally propose two methods for length normalization. We use the scoring functions in (7) or (8) with either: (1) the following heuristic:

$$h(\mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{for } \mathbf{y}.\text{last}\,() = \text{EOS} \\ \beta \max\{b - |\mathbf{y}|, 0\} & \text{for } \mathbf{y}.\text{last}\,() \neq \text{EOS} \end{cases} \tag{9}$$

where b can be $r|\mathbf{x}|$ or n_{max};[8] or (2) stopping criterion as in (6) albeit with scoring function score_{LN} and upper-bound function:

$$\mathcal{U}(\mathbf{x}, \mathbf{y}) = \beta \max\{0, b - |\mathbf{y}|\} \tag{10}$$

Despite their similarities, these two methods are not guaranteed to return the same results. Whereas the second method will return the same k-optimal hypotheses as beam search, using a heuristic during pruned search means we can no longer guarantee the k-optimality of the results with respect to the scoring function as the heuristic may push hypotheses off of the beam. We present experimental results for both methods in § 6.

Mutual Information. Maximum mutual information decoding (Li et al., 2016) aims to alleviate the inherent preference of neural models for high-frequency tokens when using the log-probability

---

[8]We enforce $r|\mathbf{x}| < n_{max}$.

decoding objective. Rather than choosing the hypothesis $\mathbf{y}$ to maximize conditional probability with respect to the input $\mathbf{x}$, we instead choose $\mathbf{y}$ to maximize pointwise mutual information (PMI):

$$\text{PMI}(\mathbf{x};\mathbf{y}) = \log \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} \qquad (11)$$

Note that (11) is equivalent to $\log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})}$, which can be rewritten as $\log p(\mathbf{y} \mid \mathbf{x}) - \log p(\mathbf{y})$, making the objective additive and thus (11) can conform to (4).

From this last form, we can see how mutual information decoding penalizes high-frequency and generic outputs; the negative $p(\mathbf{y})$ term, as Li et al. (2016) point out, acts as an "anti-language model." One unfortunate side effect of this objective is that ungrammatical and nonsensical outputs, which have probabilities close to 0 under a language model like $p(\mathbf{y})$, end up with high scores because of the second term in the score function. To address this problem, and to upper-bound the scoring function, we propose lower-bounding the language model term by a hyperparameter $1 \geq \varepsilon > 0$. We additionally use the strength hyperparameter λ employed by Li et al. (2016):

$$\begin{aligned} \text{score}_{\text{PMI}}(\mathbf{x},\mathbf{y}) = {} & \log p(\mathbf{y} \mid \mathbf{x}) \\ & - \lambda \log \max\{p(\mathbf{y}),\varepsilon\} \end{aligned} \quad (12)$$

Similarly to our methods for length normalization, we can use the scoring function in (12) either with the heuristic:

$$h(\mathbf{x},\mathbf{y}) = \begin{cases} 0 & \text{for } \mathbf{y}.\text{last}\,() = \text{EOS} \\ -\lambda \log \varepsilon(n_{max}-|\mathbf{y}|) & \text{for } \mathbf{y}.\text{last}\,() \neq \text{EOS} \end{cases} \quad (13)$$

or with stopping criterion as in (6) albeit with $\text{score}_{\text{PMI}}$ and upper-bound function:

$$\mathcal{U}(\mathbf{x},\mathbf{y}) = -\lambda \log \varepsilon(n_{max} - |\mathbf{y}|) \qquad (14)$$

Because $-\lambda \log \varepsilon$ is the best possible score at any given time step, clearly we can bound the increase in $\text{score}_{\text{PMI}}$ by the above function. However, as with our length normalization strategy, we lose the k-optimality guarantee with the heuristic method for mutual information decoding. We present experimental results for both methods in § 6.

6 Experiments

We run our algorithm on several language-related tasks that typically use beam search for decoding: NMT and AS. Specifically, experiments are performed on IWSLT'14 De-En (Cettolo et al., 2012), WMT'17 De-En (Bojar et al., 2017), MTTT Fr-En (Duh, 2018), and CNN-DailyMail (Hermann et al., 2015) using both Transformers (Vaswani et al., 2017) and Convolutional sequence-to-sequence models (Gehring et al., 2017).

For reproducibility, we use the data pre-processing scripts provided by fairseq (Ott et al., 2019) and follow their methods for training sequence transduction models. Hyperparameters are set in accordance with previous works. Specifically, on IWSLT'14 and MTTT tasks, we follow the recommended Transformer settings for IWSLT'14 in fairseq,[9] which are based on Vaswani et al. (2017) and Gehring et al. (2017). Hyperparameters for models trained on the WMT task are set following version 3 of the Tensor2Tensor toolkit (Vaswani et al., 2018). We use byte-pair encoding (BPE; Sennrich et al. 2016) for all languages. Vocabulary sizes for WMT and IWSLT'14 are set from recommendations for the respective tasks in fairseq; for the MTTT tasks, vocabulary sizes are tuned on models trained with standard label-smoothing regularization. Similarly, the CNN/DailyMail dataset is pre-processed and uses BPE following the same steps as (Lewis et al., 2019); model hyperparameters are likewise copied. Details are available on fairseq's Web site.[10]

We use BLEU (Papineni et al., 2002) (evaluated using SacreBLEU [Post, 2018]) for MT metrics and ROUGE-L (Lin, 2004) for abstractive summarization metrics. We build our decoding framework in SGNMT.[11]

6.1 Running Time

In Table 2, we report values as the average number of calls to the scoring function per input; we do not use wall-clock time as this is heavily dependent on hardware. See Fig. 1 for empirical justification of the correlation between calls to the scoring function and runtime on the hardware our

---

[9]https://github.com/pytorch/fairseq/tree/master/examples/translation.
[10]https://github.com/pytorch/fairseq/blob/master/examples/bart/README.cnn.md.
[11]https://github.com/ucam-smt/sgnmt.

803

	IWSLT'14 De-En				MTTT Fr-En			CNN-DailyMail		
	$k=5$ **(35.6)**	$k=10$ **(35.4)**	$k=100$ **(34.7)**	$k=500$ **(7.9)**	$k=10$ **(33.0)**	$k=100$ **(9.9)**	$k=500$ **(1.2)**	$k=5$ **(31.5)**	$k=10$ **(30.9)**	$k=100$ **(29.1)**
BF beam search	93 **(24%)**	169 **(36%)**	1275 **(79%)**	1168 **(736%)**	184 **(16%)**	867 **(138%)**	885 **(836%)**	200 **(33%)**	305 **(43%)**	2960 **(92%)**
Beam search (ES)	107 **(7%)**	210 **(9%)**	2047 **(12%)**	7685 **(27%)**	196 **(9%)**	1310 **(58%)**	4182 **(98%)**	224 **(19%)**	357 **(22%)**	3942 **(59%)**
Beam search	115	229	2286	9770	214	2066	8281	266	435	5673

Table 2: Average number of calls (rounded to nearest whole digit) to score, the sequence transduction model, per generated sequence when using different decoding algorithms. Green percentages are performance improvements over standard beam search. Beam search (ES) refers to the OpenNMT early-stopping method (Klein et al., 2017). All methods provably return the same solution and thus, evaluation metrics (in dark blue) for a given beam size are identical.
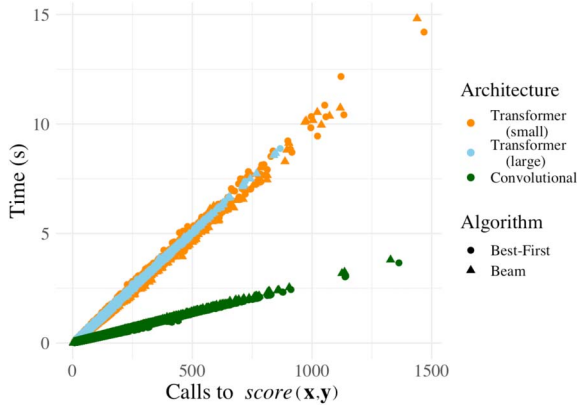

Figure 1: Number of calls to scoring function score vs. total sequence generation time. Each point is a decoded sequence. Colors represent different model architectures and shapes signify the decoding algorithm used (beam sizes 3 and 10 are included for each). There is no notable difference in the overhead (time-wise) of best-first beam search and beam search.

experiments were run on. For reference, in our experiments, the scoring function took on average $> 99\%$ of the total computation time, even with larger beam sizes, when overhead of the search algorithm is most significant.

We find that best-first (BF) beam search leads to significant speed-ups over both traditional beam search and beam search with early stopping, with a performance increase[12] of $\approx 8x$ for a beam size of 500. We likewise find that best-first beam search offers speed-ups over early stopping methods that are *not* guaranteed to return the same results as standard beam search (see Table 3).

6.2 Length Normalization

We experiment with both forms of length normalization presented in § 5 and provide results

[12]Performance increase is defined as $(\text{old} - \text{new})/\text{new}$.

k	method	search error	BLEU	# calls
	IWSLT'14 De-En			
10	shrinking	0%	35.4	229 **(0%)**
	early	0%	35.4	225 **(2%)**
	BF BS	–	35.4	**169 (36%)**
100	shrinking	31.7%	13.2	2278 **(0%)**
	early	31.7%	13.2	1738 **(31%)**
	BF BS	–	34.7	**1275 (79%)**
	WMT'17 De-En			
10	shrinking	0%	28.6	260 **(0%)**
	early	0%	28.6	252 **(3%)**
	BF BS	–	28.6	**230 (12%)**
100	shrinking	1.7%	26.4	2587 **(0%)**
	early	1.7%	26.4	2402 **(8%)**
	BF BS	–	26.9	**2046 (26%)**

Table 3: BLEU, search error, and average number of calls to score for different stopping criterion. ''shrinking'' refers to the shrinking beam method of Bahdanau et al. (2015) and ''early'' refers to the stopping criterion of Huang et al. (2017). Note that neither method is guaranteed to return the same result as standard beam search. Search error and performance increases are with respect to standard beam search.

in Table 4. We find that both methods, that is, changing the stopping criterion and using a heuristic during search, provide improvements over baseline BLEU scores albeit with different hyperparameter settings; increases are similar to improvements reported by Murray and Chiang (2018). Notably, using a heuristic causes a large percentage of search errors with respect to standard beam search using the same scoring function. However, the difference in results appears to be *beneficial* in terms of BLEU.

804

	k	β	b	# calls	search error	BLEU
IWSLT'14 De-En						
Heuristic	5	0.8	$\|\mathbf{x}\|$	115 **(0%)**	40.6%	33.9 **+0.3**
	10	1.2	$\|\mathbf{x}\|$	229 **(0%)**	54.7%	33.8 **+0.5**
Stopping Criterion	5	0.5	n_{max}	73 **(58%)**	–	33.7 **+0.1**
	10	0.5	n_{max}	130 **(76%)**	–	33.7 **+0.4**
MTTT Fr-En						
Heuristic	5	0.8	$.7\|\mathbf{x}\|$	100 **(8%)**	16.2%	33.5 **+0.2**
	10	1.0	$.7\|\mathbf{x}\|$	196 **(9%)**	25.2%	33.6 **+0.6**
Stopping Criterion	5	1.0	n_{max}	65 **(66%)**	–	34.1 **+0.8**
	10	1.2	n_{max}	88 **(143%)**	–	34.1 **+1.1**

Table 4: BLEU search error, and average number of calls to SCORE for output obtained with length normalization scoring function on the IWSLT'14 De-En and MTTT Fr-En test sets. Increase in BLEU is over baseline with no length normalization. Search error and performance increases are with respect to standard beam search decoding using the same scoring function.

6.3 Mutual Information

We train a language model on the IWSLT dataset and use it to calculate $p(\mathbf{y})$ from (12) as marginalizing over $\mathbf{y}$ is intractable (see Li et al. [2016] for further justification). We run experiments using both of the methods discussed in § 5 and present results in Table 5. We find that both methods provide results of equivalent BLEU score compared with the baseline output, namely, results obtained with the unbounded PMI objective and beam search. Again, despite the high search error rate demonstrated by the heuristic method, evaluation metrics are still comparable.

6.4 Memory Usage

We conduct a set of experiments where we limit total queue capacity to $k \cdot \gamma$ for $\gamma \in \{1, \ldots, n_{max}\}$, as described in § 3.3, and report the BLEU score of the resulting set of hypotheses.

As shown in Table 6, we find that restricting the queue capacity does not harm output quality and, additionally, leads to even greater runtime performance increase. For example, runtime for decoding of IWSLT'14 with a beam size of 10 can be improved by $> 3x$ while returning results with better evaluation metrics. We find that improvements are even more pronounced for larger beam sizes. Across beam widths and tasks, we find that search error (with respect to standard beam search) is quite low for $\gamma = 5$. Additionally, for smaller γ, the change in BLEU

	k	ε	β	# calls	search error	BLEU
Baseline	5	–	.05	115	–	33.2
	10	–	.05	229	–	33.0
Heuristic	5	.02	.05	129 **(0%)**	42.7%	33.2
	10	.02	.05	256 **(0%)**	42.7%	33.0
Stopping Criterion	5	3e–4	.05	114 **(1%)**	29.2%	33.2
	10	5e–5	.05	224 **(2%)**	26.6%	33.0

Table 5: BLEU scores with mutual information scoring function on IWSLT'14 De-En. Baseline is PMI decoding with *unbounded* $p(\mathbf{y})$, that is, $\varepsilon = 0$. Search error is with respect to beam search decoding of baseline with same β.

k	γ	search error	BLEU	# calls
IWSLT'14 De-En				
5	2	22.7%	35.7 **+0.1**	43.8 **(163%)**
	5	4.4%	35.8 **+0.2**	79.8 **(44%)**
	n_{max}	–	35.6	93.0 **(24%)**
10	2	22.6%	35.7 **+0.3**	48.4 **(374%)**
	5	4.5%	35.6 **+0.2**	126.9 **(81%)**
	n_{max}	–	35.4	169.0 **(36%)**
WMT'17 De-En				
5	2	29.0%	29.7 **+0.2**	77.5 **(75%)**
	5	1.2%	29.5 **+0.0**	115.8 **(12%)**
	n_{max}	–	29.5	118.8 **(10%)**
10	2	36.6%	29.5 **+0.2**	97.3 **(165%)**
	5	2.6%	29.3 **+0.0**	230.0 **(12%)**
	n_{max}	–	29.3	230.2 **(12%)**

Table 6: BLEU scores and the number of calls to SCORE on the IWSLT'14 De-En validation set and WMT'17 De-En test set with queue size restricted to $n_{max} \cdot k$. Note that $\gamma = n_{max}$ is the standard best-first beam search algorithm. Performance increases are over standard beam search. Search error is with respect to beam search with same beam width.

score demonstrates that search error in this context does not necessarily hurt the quality of results.

7 Related Work

Our work is most similar to that of Zhou and Hansen (2005), who propose beam stack search. However, they are focused on exact inference and still evaluate hypotheses in breadth-first order.

805

Additionally, their algorithm requires $\mathcal{O}(n_{max}k)$ memory; although best-first beam search has the same requirements, we introduce effective methods for reducing them, namely, memory-reduced best-first beam search.

Huang et al. (2017) propose and prove the optimality of an early-stopping criterion for beam search. The authors find in practice though that reduction in computation from their algorithm was generally not significant. We build on this work and introduce additional methods for avoiding unnecessary computation. Our method leads to better performance, as shown in Table 2.

Klein and Manning (2003) use A* for PCFG parsing; however, they use the un-pruned version for exact search, which is not applicable for NMT or AS as the memory requirements of the algorithm are far too large for these tasks. Subsequently, Pauls and Klein (2009) provide a method for pruning this search algorithm, albeit using a threshold rather than explicitly limiting the state space. Huang et al. (2012) also adapt A* for a k-best decoding algorithm. Although their methods differ notably from ours, they likewise use pruning techniques that allow for substantial speedups.

Stahlberg and Byrne (2019) create an exact inference algorithm for decoding and use it to analyze the output of neural NMT models. Whereas they likewise utilize the monotonicity of the scoring function to make their method tractable, they do not focus on speed or mimicking the results of standard beam search.

8 Conclusion

We propose best-first beam search, an algorithm that allows for faster decoding while still guaranteeing k-optimality. We provide results on several sequence-to-sequence transduction tasks that show the speed-ups that our algorithm provides over standard beam search for decoding neural models. We adapt several popular alternate scoring functions to best-first beam search and provide a framework that can be used to adapt other scoring methods such as coverage normalization (Wu et al., 2016) or diverse beam search (Vijayakumar et al., 2016). We also provide a memory-reduced version of our algorithm, which returns competitive results in a fraction of the time needed for standard beam search.

References

M. D. Atkinson, J. R. Sack, N. Santoro, and T. Strothotte, 1986. Min-max heaps and generalized priority queues. *Communications of ACM*: 29(10). **DOI:** https://doi.org/10.1145/6617.6621

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation. In *Proceedings of the Conference on Machine Translation, Volume 2: Shared Task Papers*, Copenhagen, Denmark. **DOI:** https://doi.org/10.18653/v1/W17-4717

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit³: Web inventory of transcribed and translated talks. In *Proceedings of the Conference of the European Association for Machine Translation*.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1). **DOI:** https://doi.org/10.1007/BF01386390

Kevin Duh. 2018. The multitarget TED talks task. http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D18-1045

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers). Vancouver, Canada. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P17-1012, **PMID:** 28964987, **PMCID:** PMC6754825

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). **DOI:** https://doi.org/10.1109/TSSC.1968.300136

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with SMT features. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend, *Advances in Neural Information Processing Systems*.

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*, Copenhagen, Denmark. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D17-1227, **PMID:** 28564569

Zhiheng Huang, Yi Chang, Bo Long, Jean-Francois Crespo, Anlei Dong, Sathiya Keerthi, and Su-Lin Wu. 2012. Iterative Viterbi A* algorithm for k-best sequential decoding. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. **DOI:** https://doi.org/10.3115/1073445.1073461

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017.

OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, Vancouver, Canada. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P17-4012

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, Vancouver. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/W17-3204

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, ICML '01, San Francisco, CA, USA.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *arXiv:1910.13461*. **DOI:** https://doi.org/10.18653/v1/2020.acl-main.703

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*, San Diego, California. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain. Association for Computational Linguistics.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning*, ICML 00. San Francisco, CA, USA.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation.

In *Proceedings of the Third Conference on Machine Translation: Research Papers*, Belgium, Brussels. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/W18-6322

Joakim Nivre, Igor M. Boguslavsky, and Leonid L. Iomdin. 2008. Parsing the SynTagRus treebank of Russian. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK. Coling 2008 Organizing Committee. **DOI:** https://doi.org/10.3115/1599081.1599162

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT: Demonstrations*. **DOI:** https://doi.org/10.18653/v1/N19-4009

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*. **DOI:** https://doi.org/10.3115/1073083.1073135

Adam Pauls and Dan Klein. 2009. Hierarchical search for parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/1620754.1620835

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Conference on Machine Translation: Research Papers*. **DOI:** https://doi.org/10.18653/v1/W18-6319

Rico Sennrich, Barry Haddow, and Alexandra Birch.2016. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. **DOI:** https://doi.org/10.18653/v1/P16-1162

Iulian Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany. Association for Computational Linguistics. **DOI:** https://doi.org/10.18v653/v1/P16-1159, **PMID:** 27069146

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)*, Melbourne, Australia. 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 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China. **DOI:** https://doi.org/10.18653/v1/D19-1331

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

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *Advances in Neural Information Processing Systems*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017.

Attention is all you need. In *Advances in Neural Information Processing Systems*.

Tim Vieira, Ryan Cotterell, and Jason Eisner. 2016. Speed-accuracy tradeoffs in tagging with variable-order CRFs and structured sparsity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. **DOI:** `https://doi.org/10.18653/v1/D16-1206`

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

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. **DOI:** `https://doi.org/10.1109/TIT.1967.1054010`

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2).

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D18-1342`

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*.

Rong Zhou and Eric A. Hansen. 2005. Beam-stack search: Integrating backtracking with beam search. In *Proceedings of the International Conference on International Conference on Automated Planning and Scheduling*, ICAPS05.