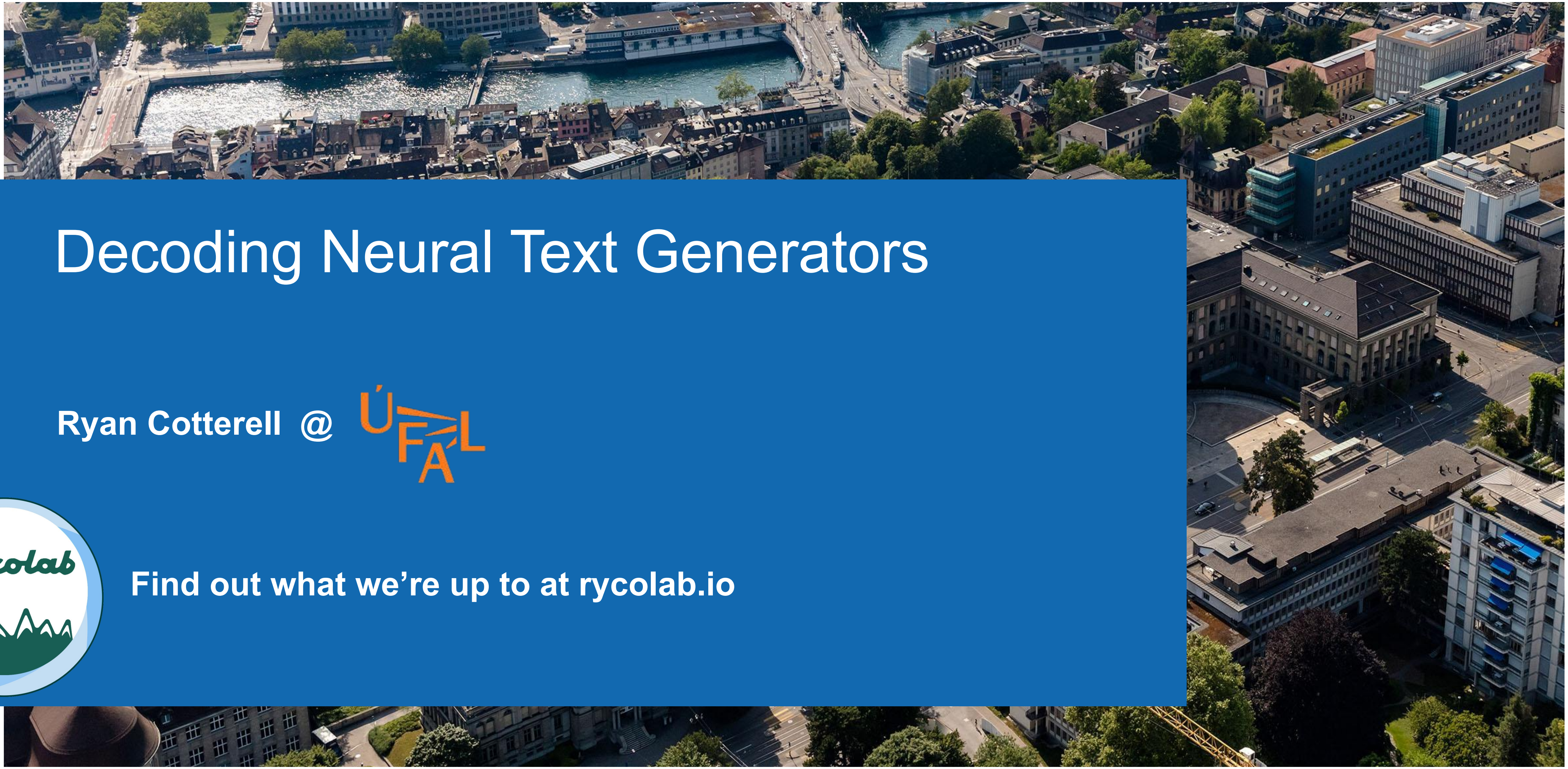
ETH zürich



Decoding Neural Text Generators

Ryan Cotterell @ ÚFAL

Rycolab

Find out what we're up to at rycolab.io

Insight Number One:
Can we Prioritize Beam Search?

Thanks to my Amazing Collaborators

Clara is a second-year PhD Student at ETH Zürich advised by Ryan. She did her Bachelor's and Master's in statistics at Stanford before moving over to the dark side (NLP)

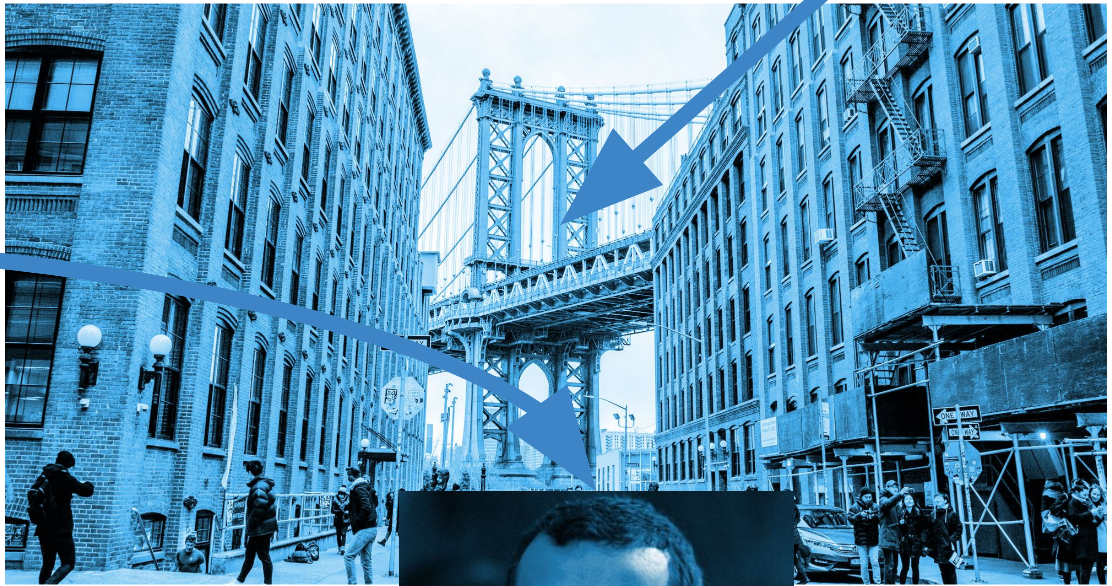
Tim is a final-year PhD Candidate at Johns Hopkins advised by Jason Eisner. He likes hand-stands *inter alia*.

The Origin Story

On a Brisk Before-Times Day in February, 2020

ETH Zurich

Brooklyn



Work Published at TACL on a Dreary After-Times Day

@ TACL 2020 (December)

Zurich

Zurich

Brklyn

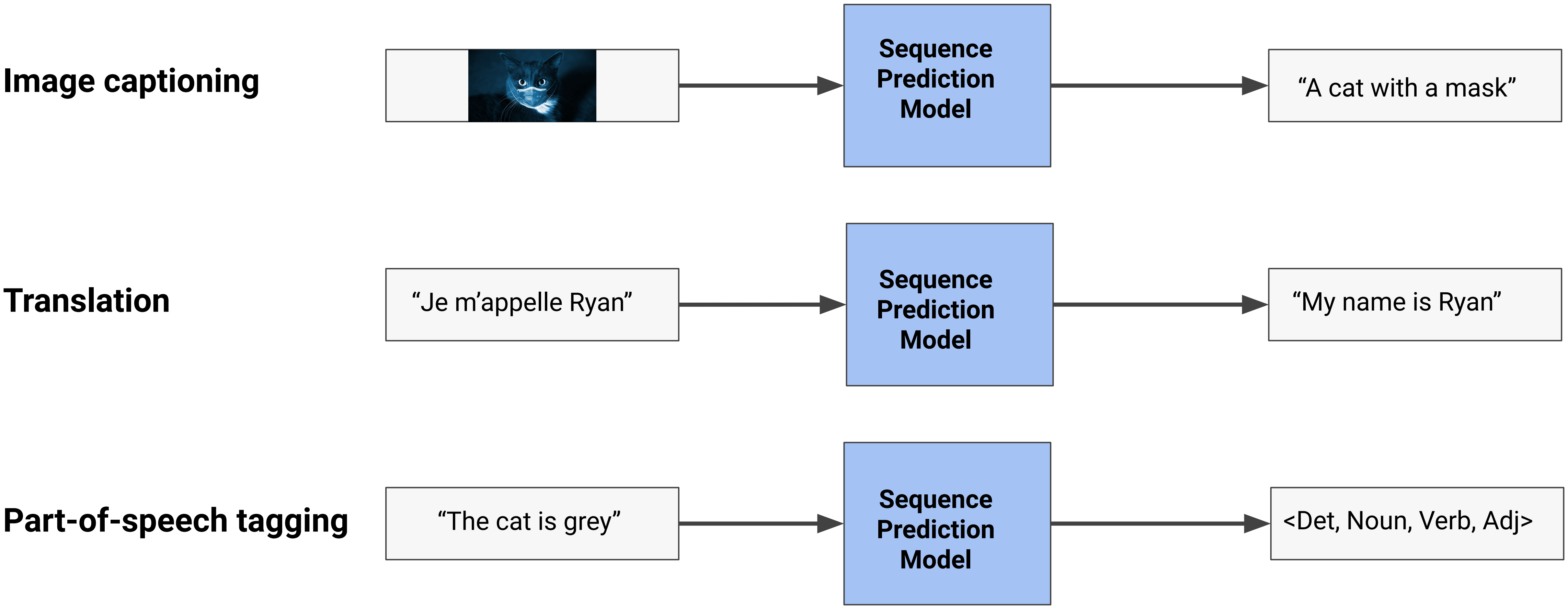
Best-First Beam Search

Clara Meister[⋆] **Tim Vieira**[ℓ] **Ryan Cotterell**[∗,⋆]
[⋆]ETH Zürich [ℓ]Johns Hopkins University [∗]University of Cambridge
clara.meister@inf.ethz.ch tim.vieira@gmail.com
ryan.cotterell@inf.ethz.ch

Decoding Neural Language Generators

Sequence Prediction in Natural Language Processing

Many core NLP tasks involve predicting sequences!



Sequence Prediction in Natural Language Processing

The **Sequence Prediction Model** can be thought of as a blackbox scoring function:

$$\mathbf{score}(\mathbf{x}, \mathbf{y})$$

input **output sequence**

When we have a probabilistic model: $\mathbf{score}(\mathbf{x}, \mathbf{y}) = \log p_{\theta}(\mathbf{y} \mid \mathbf{x})$

model parameters

Decoding Sequence Models

Given an input $\mathbf{x}$, we can generate output sequences $\mathbf{y}$ according to the scoring function

$$\mathbf{score}(\mathbf{x}, \mathbf{y})$$

where we typically want to generate the prediction such that:

$$\mathbf{y}^\star = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \ \mathbf{score}(\mathbf{x}, \mathbf{y})$$

Aside: when $\mathbf{score}(\mathbf{x}, \mathbf{y}) = \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$, this is just maximum-a-posteriori (MAP) inference!

Decoding Sequence Models

Given an input $\mathbf{x}$, we can generate output sequences $\mathbf{y}$ according to the scoring function

$$\mathbf{score}(\mathbf{x}, \mathbf{y})$$

where we typically want to generate the prediction such that:

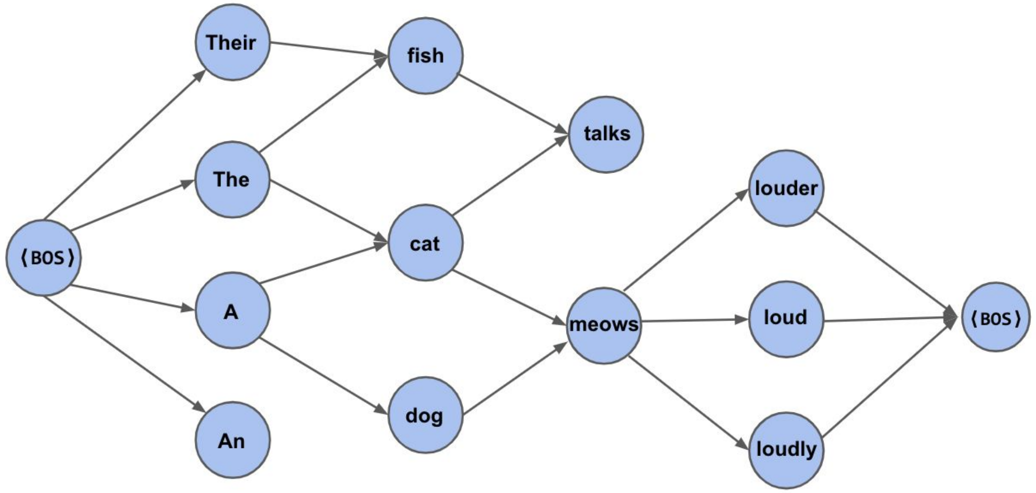
But how do we actually generate y???

$$\mathbf{y}^{\star} = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \ \mathbf{score}(\mathbf{x}, \mathbf{y})$$

Aside: when $\mathbf{score}(\mathbf{x}, \mathbf{y}) = \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$, this is just maximum-a-posteriori (MAP) inference!

Decoding Sequence Models

Let's frame decoding as a search problem:

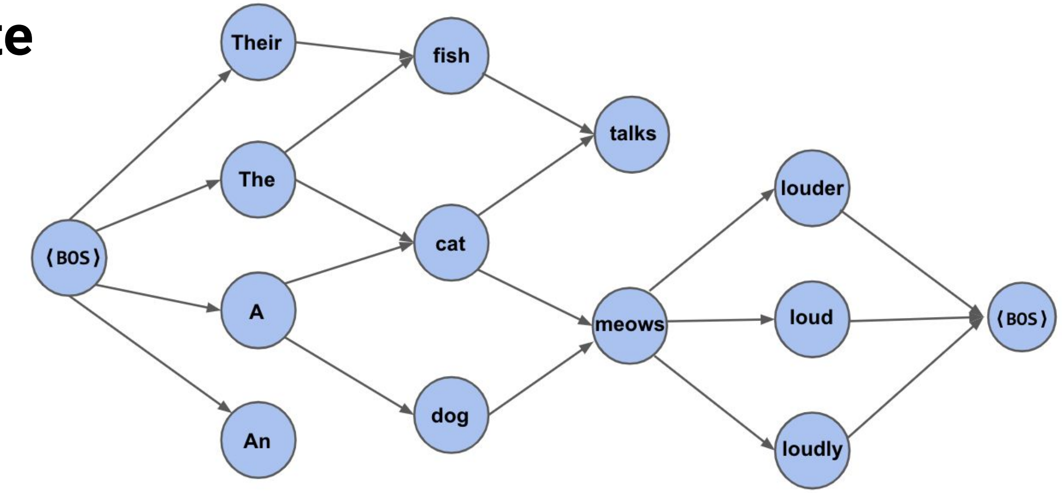


$$\mathbf{y}^{\star} = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \ \mathbf{score}(\mathbf{x}, \mathbf{y})$$

Decoding Sequence Models

Let’s frame decoding as a search problem:

<BOS> is our starting state



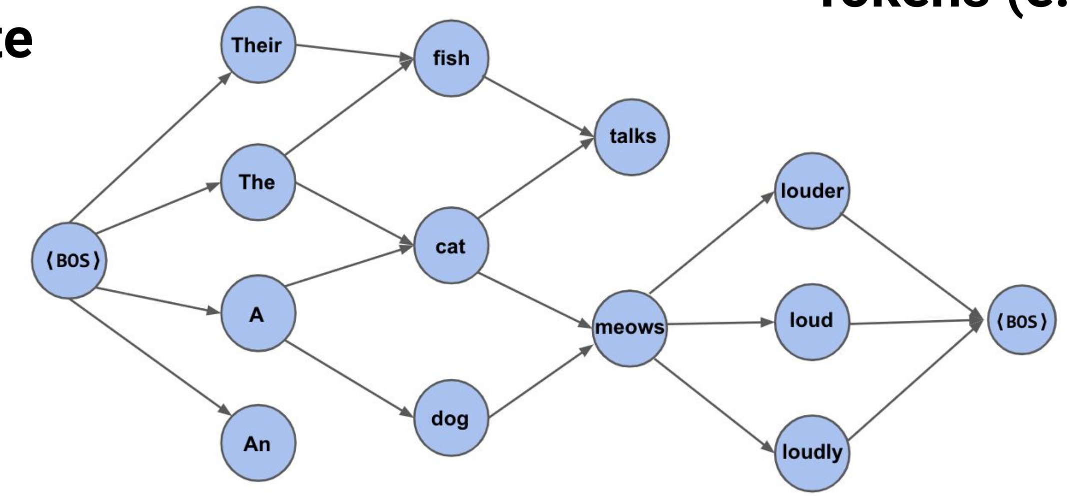
$$\mathbf{y}^\star = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \; \mathbf{score}(\mathbf{x}, \mathbf{y})$$

Decoding Sequence Models

Let’s frame decoding as a search problem:

<BOS> is our starting state

Tokens (e.g., words) are nodes



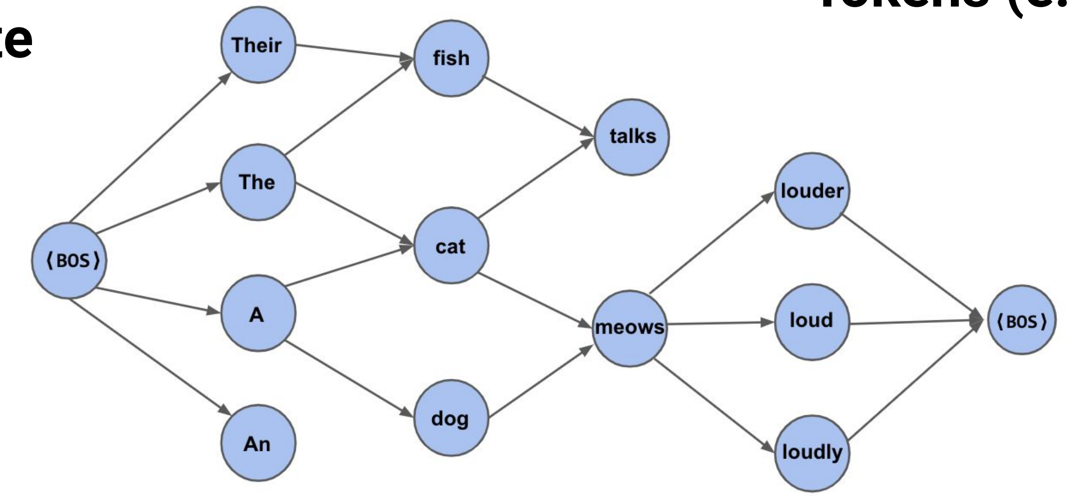
$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}{\operatorname{argmax}}\ \mathbf{score}(\mathbf{x},\mathbf{y})$$

Decoding Sequence Models

Let’s frame decoding as a search problem:

<BOS> is our starting state

Tokens (e.g., words) are nodes



Looking for “best” path to <EOS>

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{score}(\mathbf{x}, \mathbf{y})$$

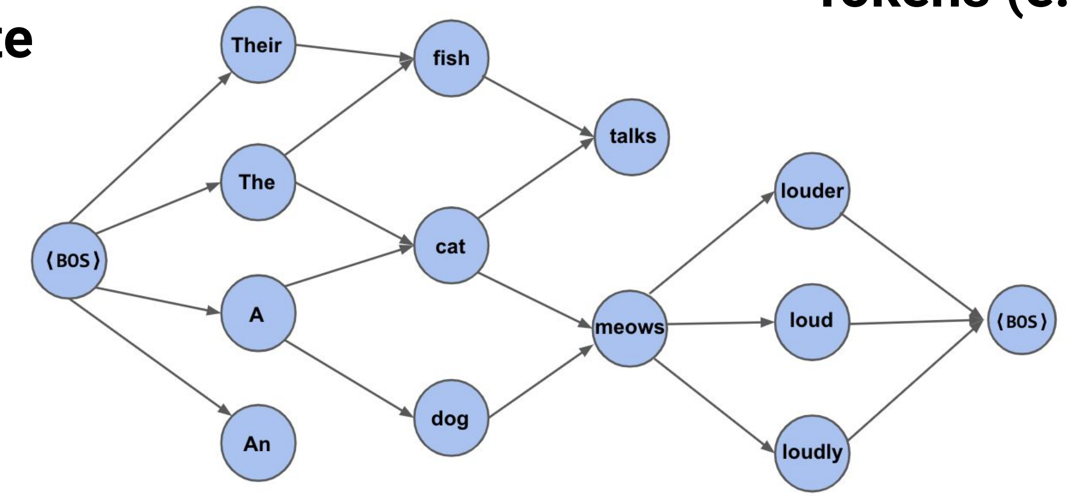
Decoding Sequence Models

Let’s frame decoding as a search problem:

Tokens (e.g., words) are nodes

<BOS> is our starting state

Looking for “best” path to <EOS>



$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y} \in \boxed{\mathcal{Y}(\mathbf{x})}} \mathbf{score}(\mathbf{x}, \mathbf{y})$$

But wait,
what’s this?

Decoding Sequence Models

- We only consider the task-specific set of “well-formed” predictions when decoding. We call this set our output space $\mathcal{Y}(\mathbf{x})$, which is often dependent on $\mathbf{x}$.

Decoding Sequence Models

- We only consider the task-specific set of “well-formed” predictions when decoding. We call this set our output space $\mathcal{Y}(\mathbf{x})$, which is often dependent on $\mathbf{x}$.

A closer look at $\mathcal{Y}(\mathbf{x})$:

Decoding Sequence Models

- We only consider the task-specific set of “well-formed” predictions when decoding. We call this set our output space $\mathcal{Y}(\mathbf{x})$, which is often dependent on $\mathbf{x}$.

A closer look at $\mathcal{Y}(\mathbf{x})$:

- **In machine translation**: It is the set of all word sequences of max length n
- **In image captioning**: It is also the set of all word sequences of max length n
- **In tagging**: It is the set of tag sequences that have the same length as $\mathbf{x}$

Decoding Sequence Models

- We only consider the task-specific set of "well-formed" predictions when decoding. We call this set our output space $\mathcal{Y}(\mathbf{x})$, which is often dependent on $\mathbf{x}$.

A closer look at $\mathcal{Y}(\mathbf{x})$:

- **In machine translation**: It is the set of all word sequences of max length n
- **In image captioning**: It is also the set of all word sequences of max length n
- **In parsing**: It is the set of all well formed parse trees for input $\mathbf{x}$

Exponentially large spaces!!!

Decoding Sequence Models

Good thing we have years of research on dynamic programming and combinatorial algorithms at our disposal...

Dijkstra

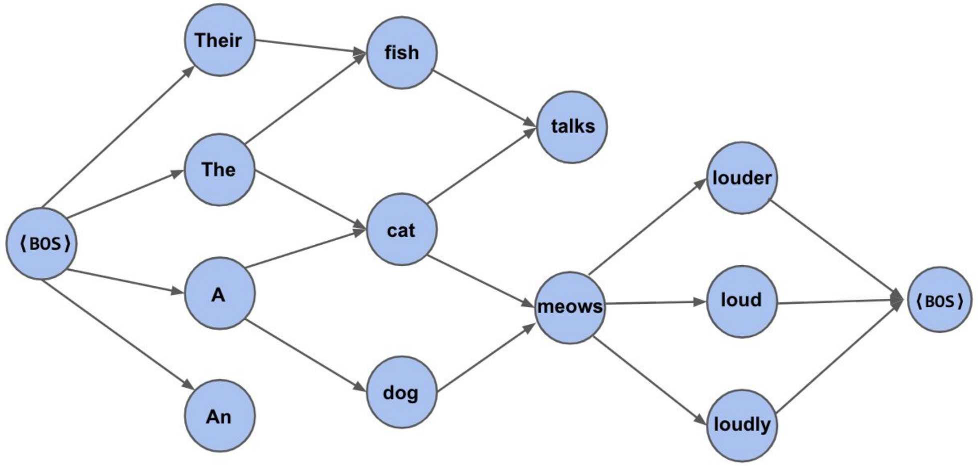
Bellman

Ford

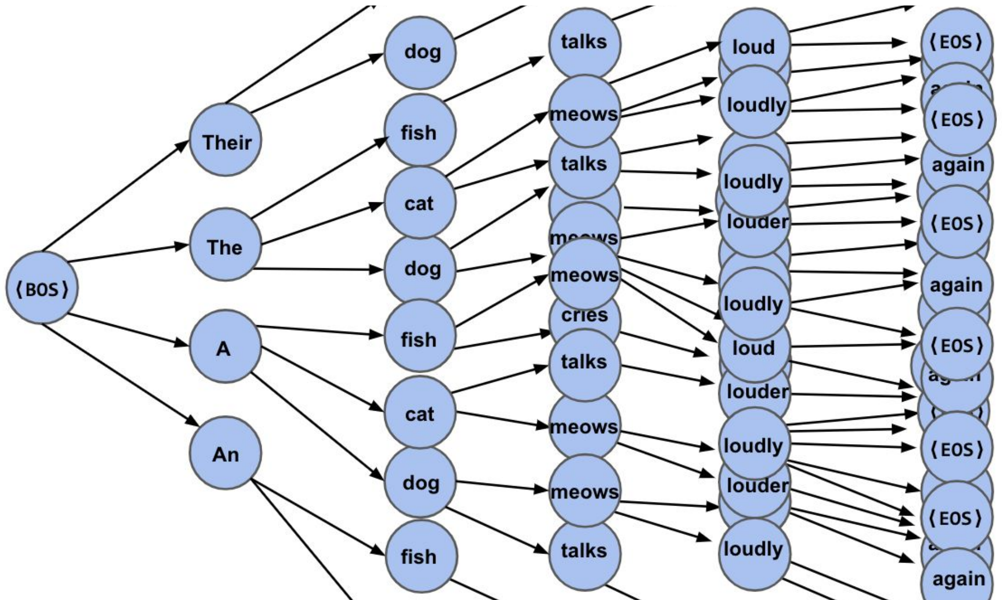
Viterbi

Decoding Sequence Models

But what if our score function doesn't nicely decompose like this:



what if, instead, it decomposes like this:



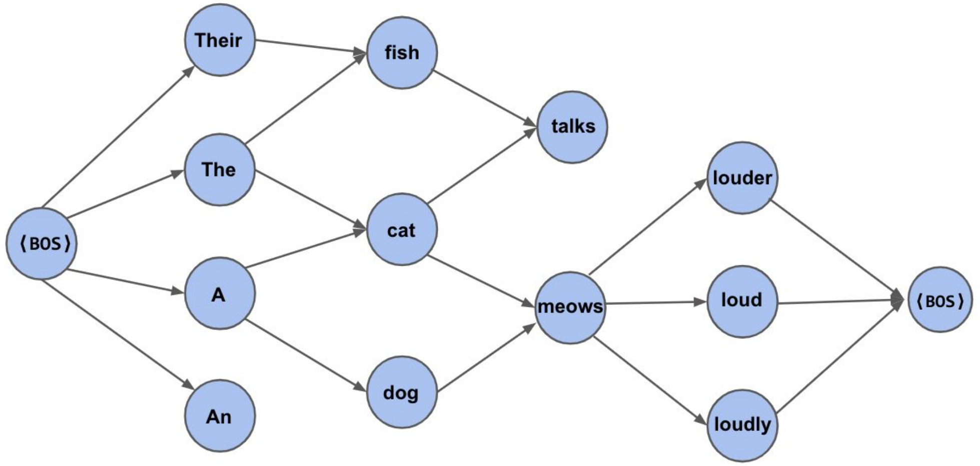
Decoding Sequence Models

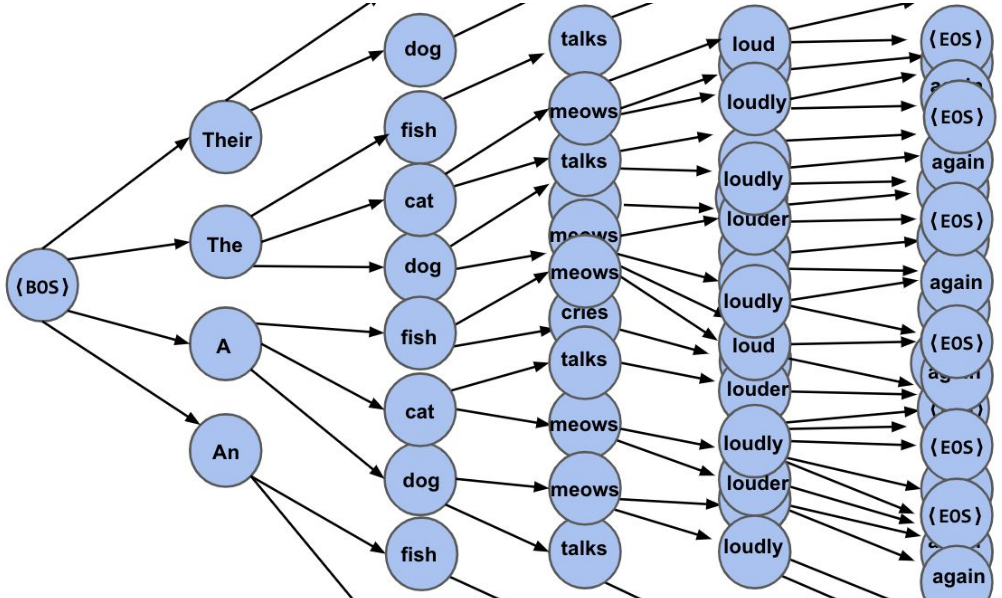
But what if our score function doesn't nicely decompose like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \langle y_{t-1}, y_t \rangle)$$

what if, instead, it decomposes like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$



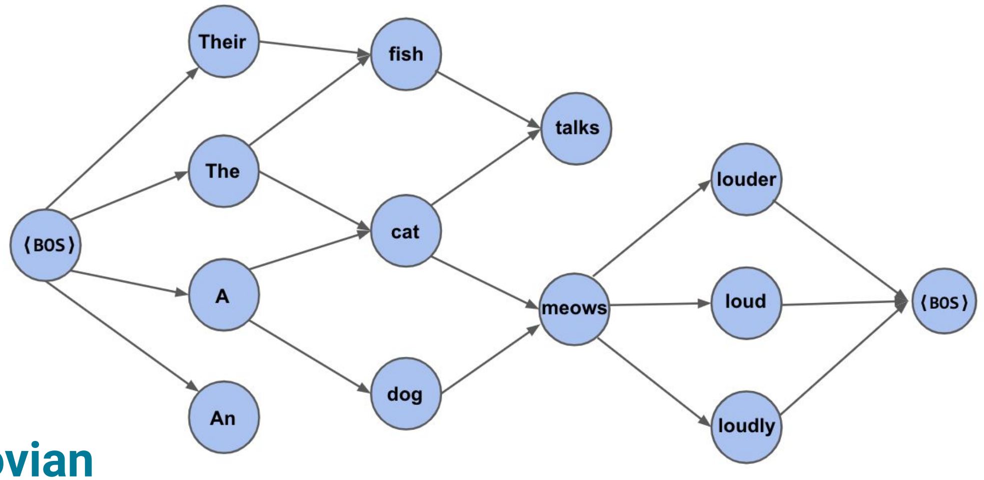


Decoding Sequence Models

But what if our score function doesn't nicely decompose like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \boxed{\langle y_{t-1}, y_t \rangle})$$

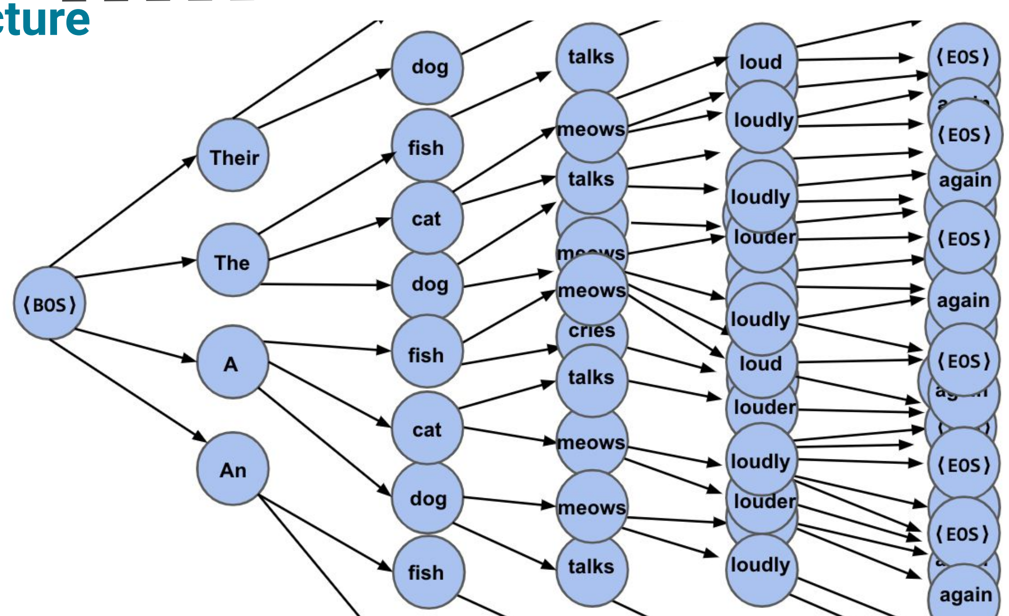
Markovian structure



what if, instead, it decomposes like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \boxed{\mathbf{y}_{<t}})$$

Non-Markovian structure



Decoding Sequence Models

But what if our score function doesn't nicely decompose like this:

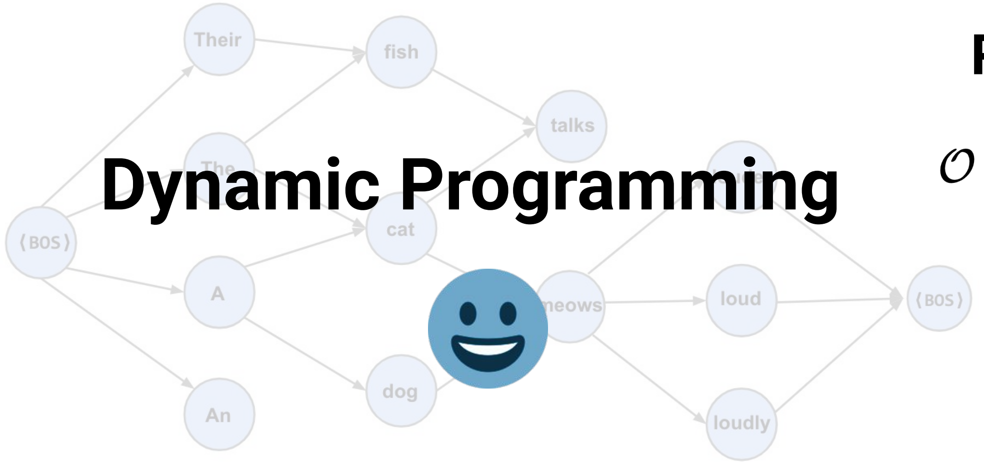
$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \langle y_{t-1}, y_t \rangle)$$

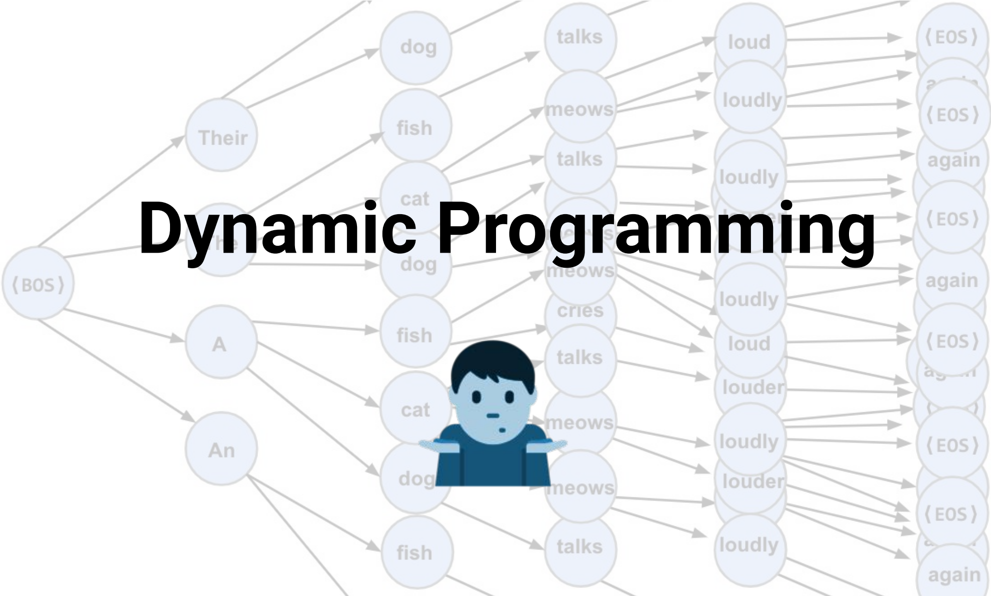
what if, instead, it decomposes like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

Runtime

$\mathcal{O}\left(|\mathcal{V}|^2 \cdot n_{\max}\right)$





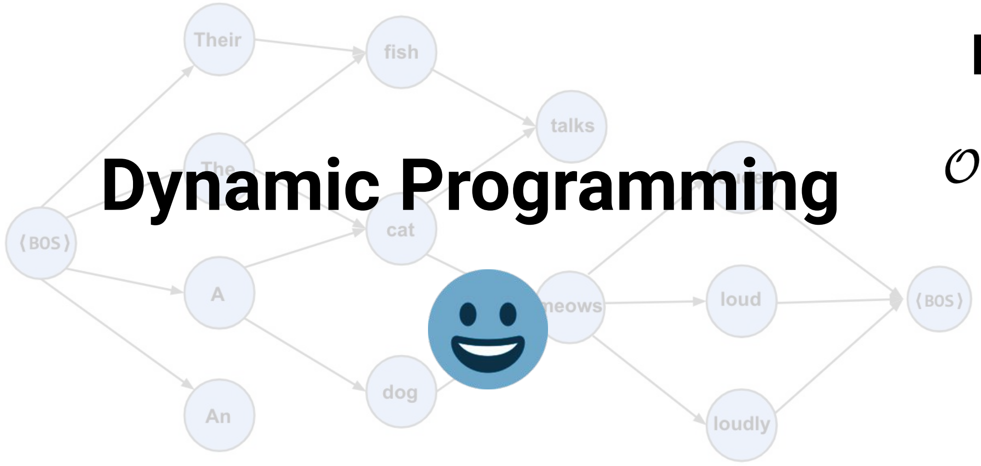
Decoding Sequence Models

But what if our score function doesn't nicely decompose like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \langle y_{t-1}, y_t \rangle)$$

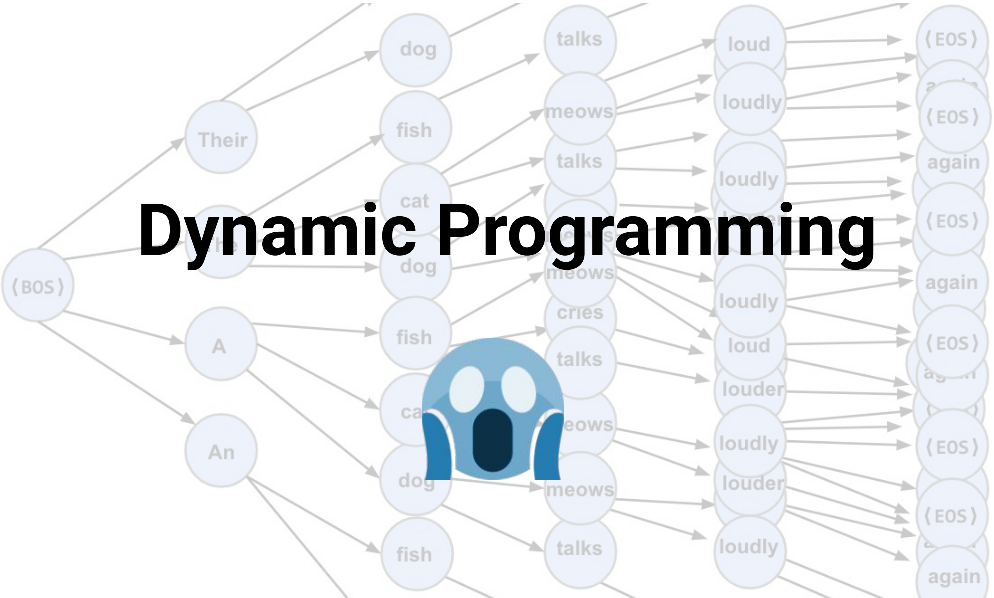
what if, instead, it decomposes like this:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$



Runtime

$\mathcal{O}\left(|\mathcal{V}|^2 \cdot n_{\max}\right)$

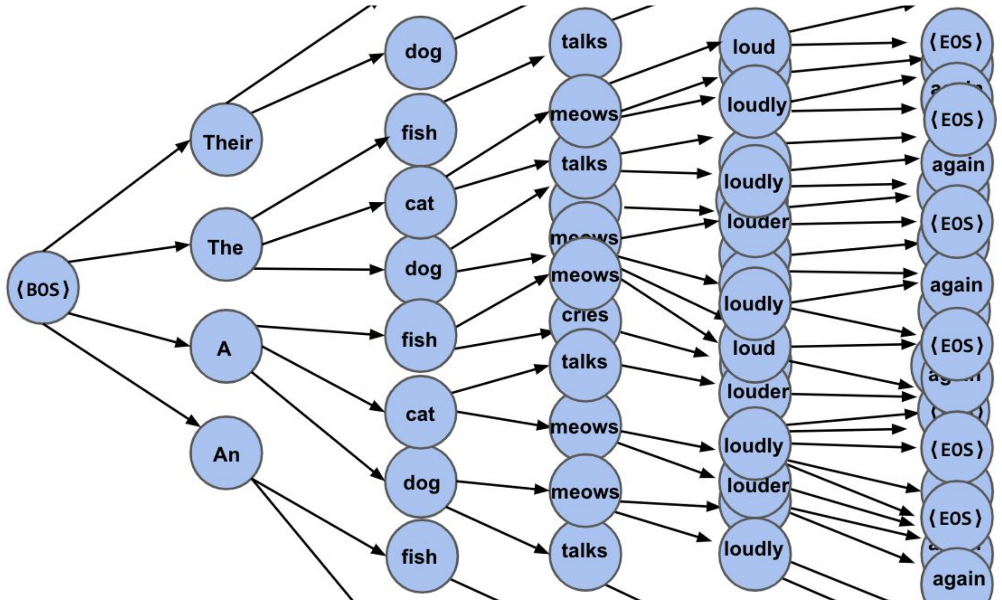


$\mathcal{O}\left(|\mathcal{V}|^{n_{\max}}\right)$

Decoding Sequence Models

- Many of the neural probabilistic models used today for sequence generation tasks exhibit this (lack of) structure

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$



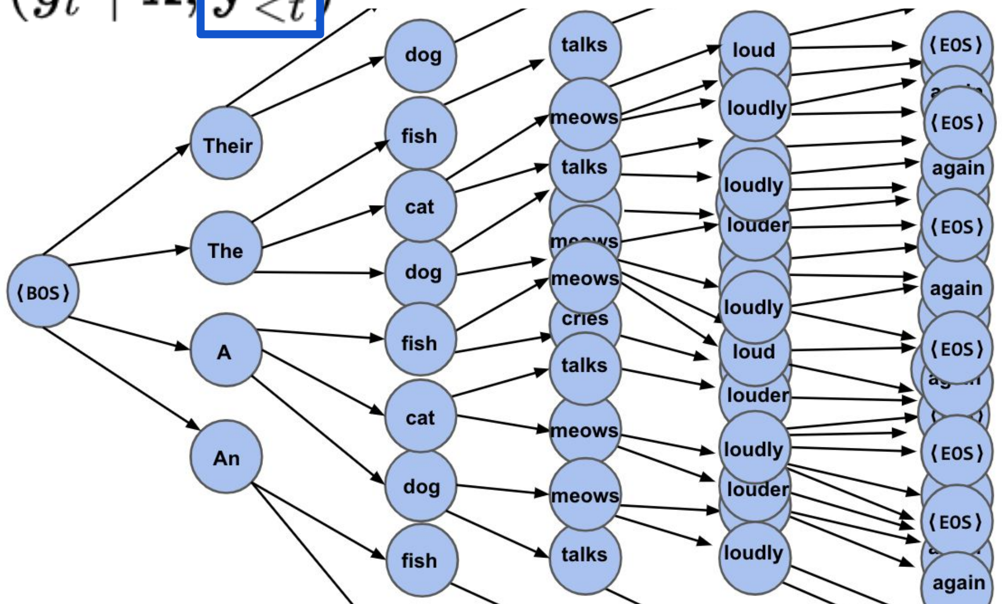
Decoding Sequence Models

- Many of the neural probabilistic models used today for sequence generation tasks exhibit this (lack of) structure
- E.g., most neural language models including machine translation systems

$$\mathrm{score}(\mathbf{x}, \mathbf{y}) = \log p_\theta(\mathbf{y} \mid \mathbf{x}) = \sum_{t=1}^{|\mathbf{y}|} \log p_\theta(y_t \mid \mathbf{x}, \boxed{\mathbf{y}_{<t}})$$

$$\mathrm{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathrm{score}(\mathbf{x}, \boxed{\mathbf{y}_{<t}})$$

Non-Markovian structure



Decoding Sequence Models

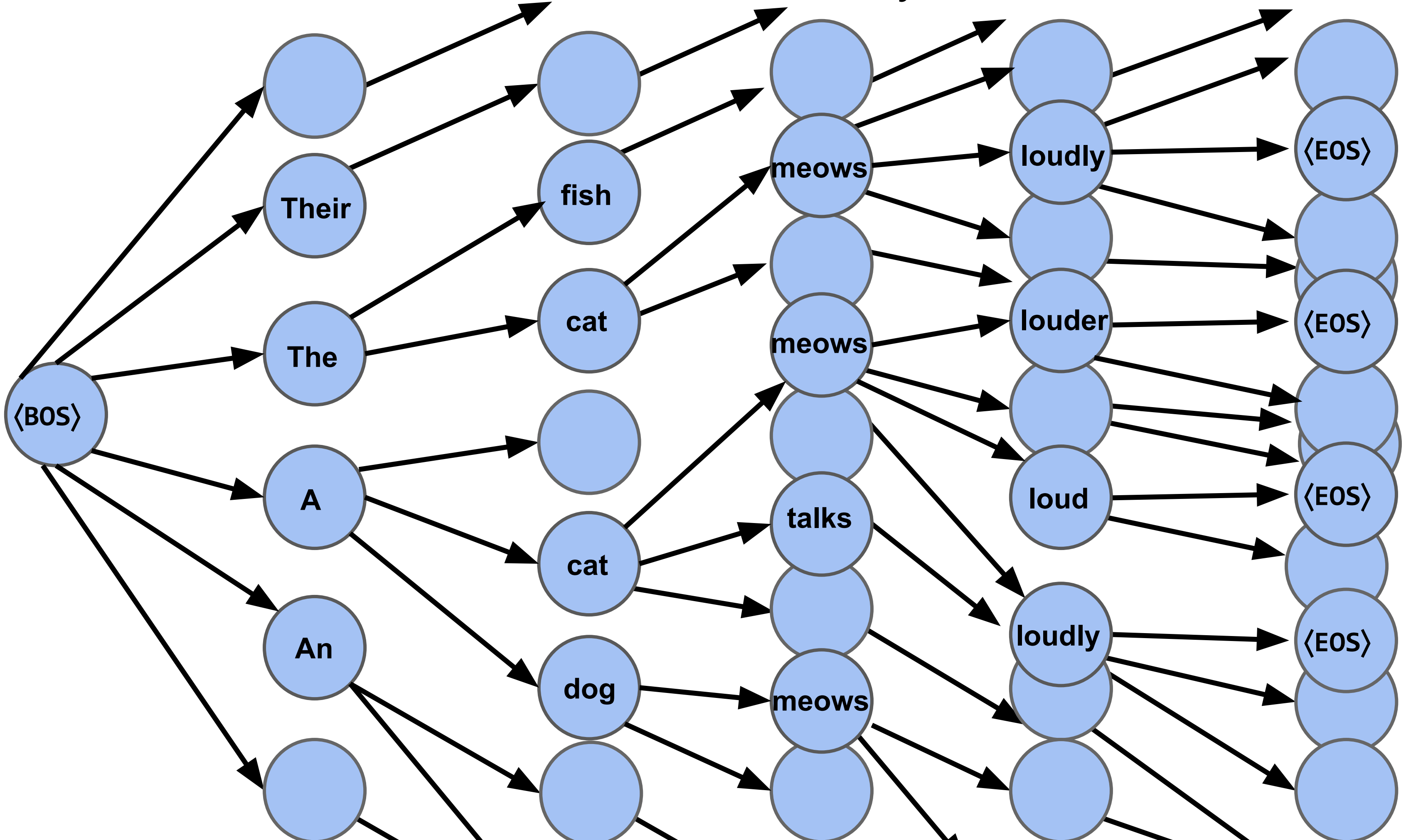
- Exact decoding for tasks like machine translation require us to explore all $|\mathcal{Y}(\mathbf{x})| = |\mathcal{V}|^{n_{\max}}$ paths independently to find the optimal solution
- For large $\mathcal{V}$ (think, a reasonable vocab of size 30,000) and a maximum sequence length of 20 words, that's more paths than the number of particles in the universe

Our traditional dynamic programming algorithms may never even terminate!

Beam Search

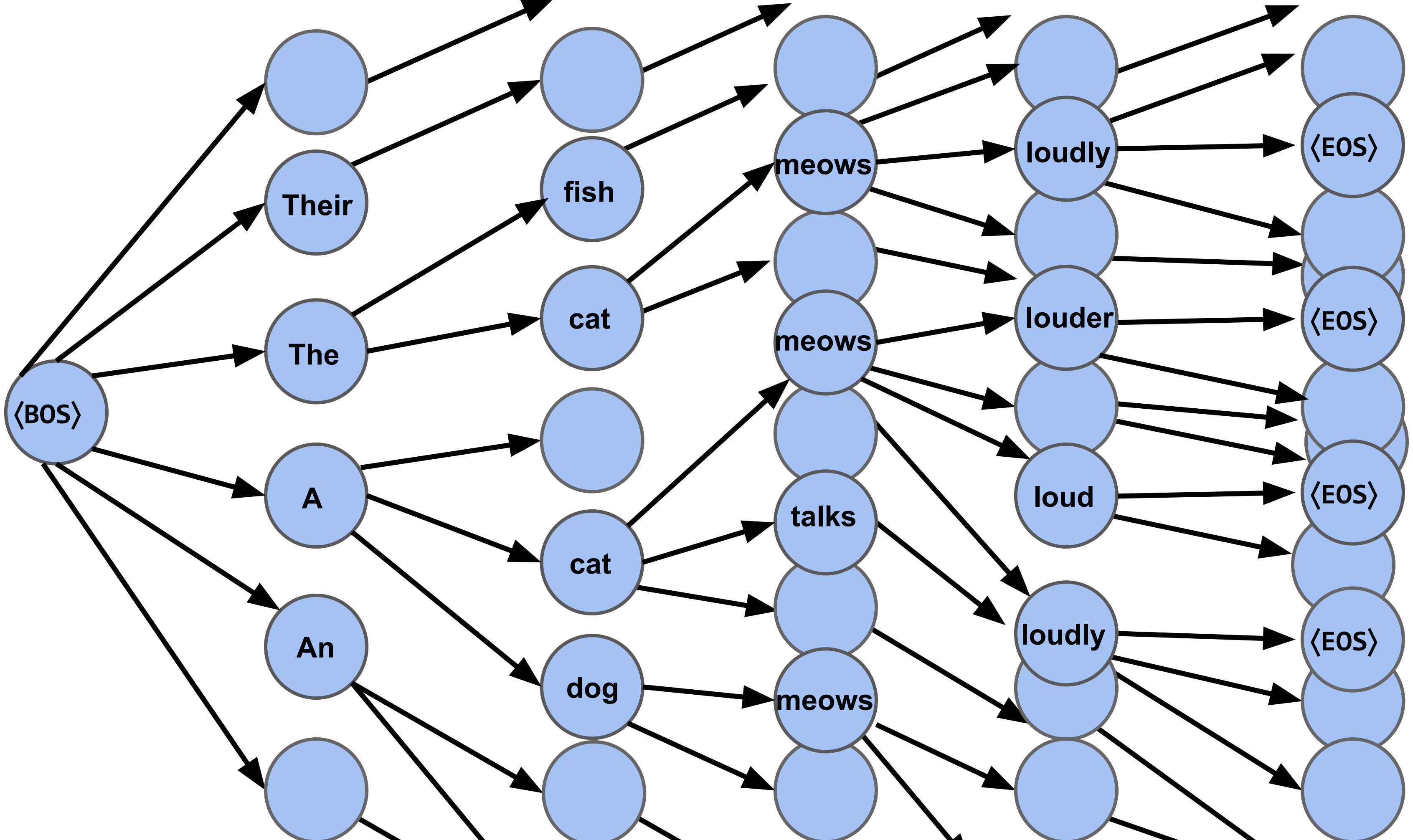
Beam Search

As we saw, with breadth-first search we may have a combinatorial explosion...



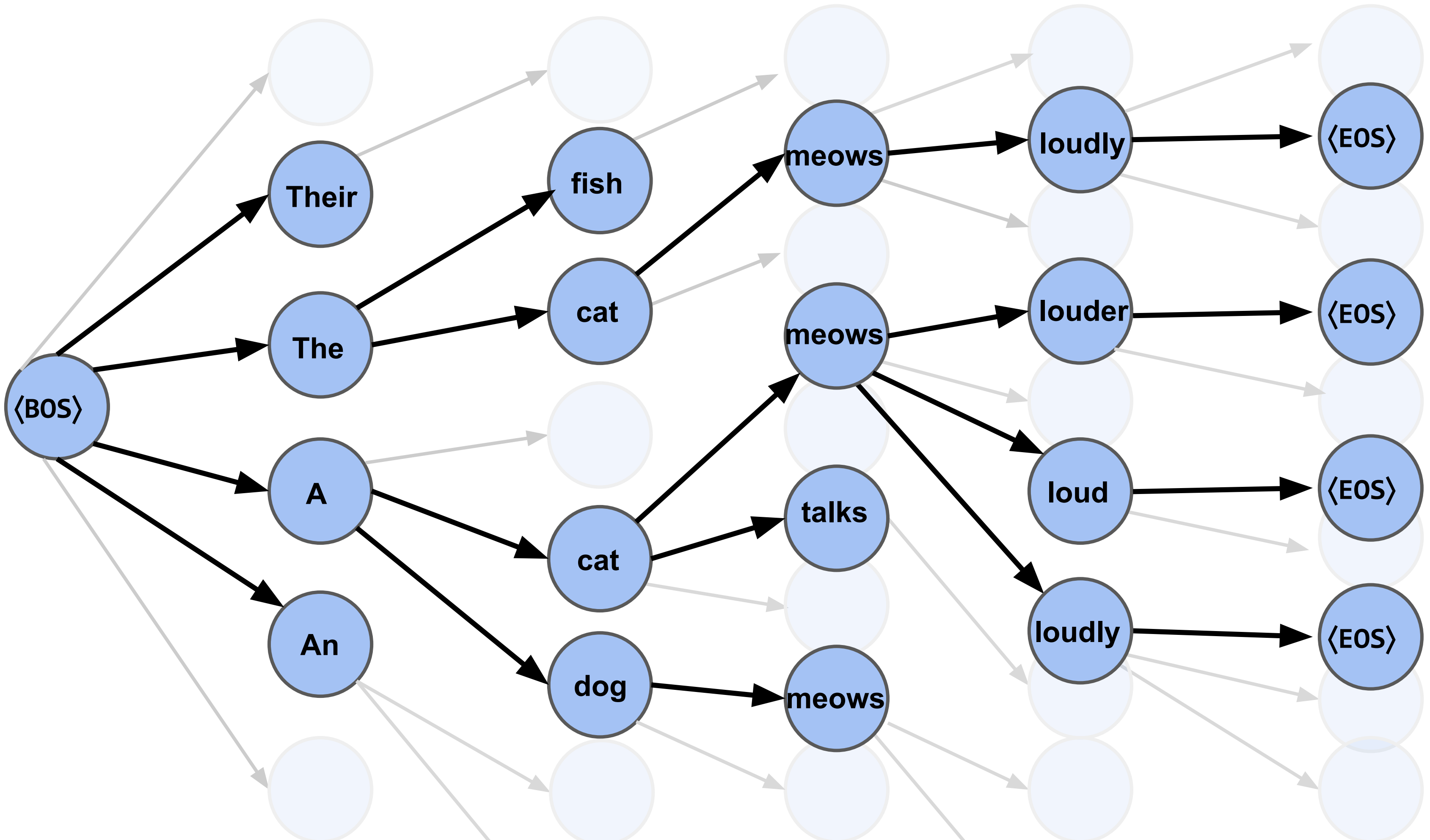
Beam Search

As we saw, with breadth-first search we may have a combinatorial explosion...



What if we limit the number of paths we explore?

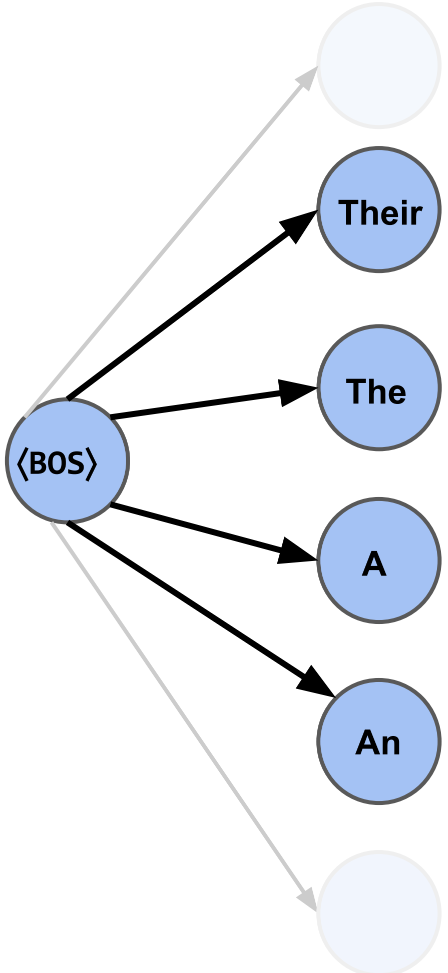
Beam Search



What if we limit the number of paths we explore?

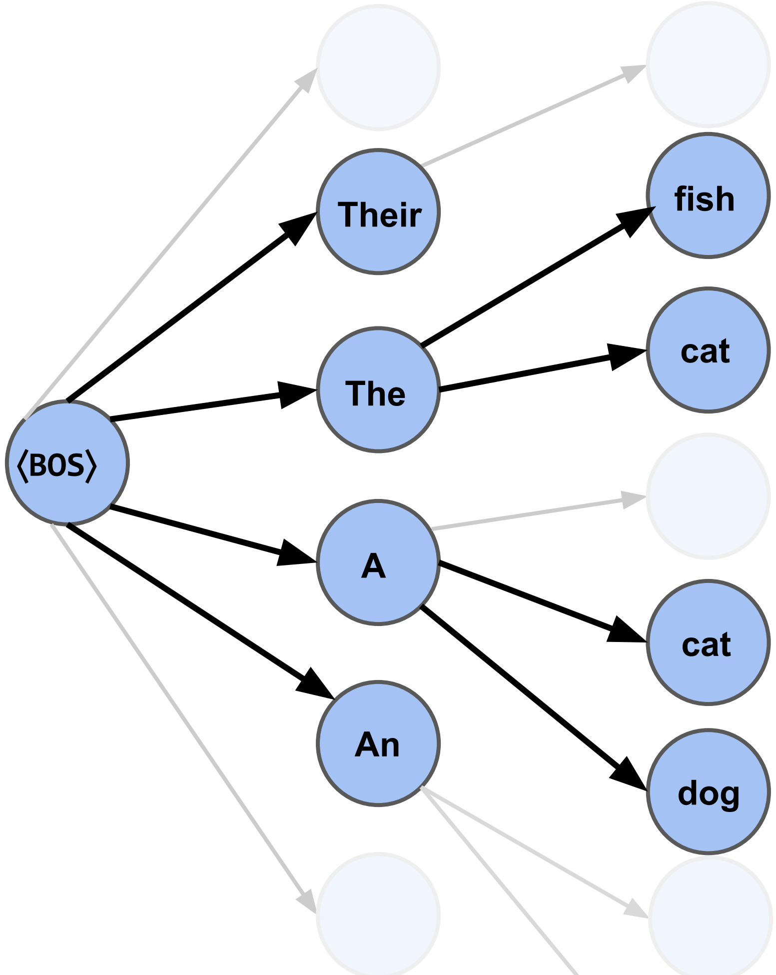
Beam Search In Action

In short: ***pruned*** breadth-first search where the breadth is limited to size k



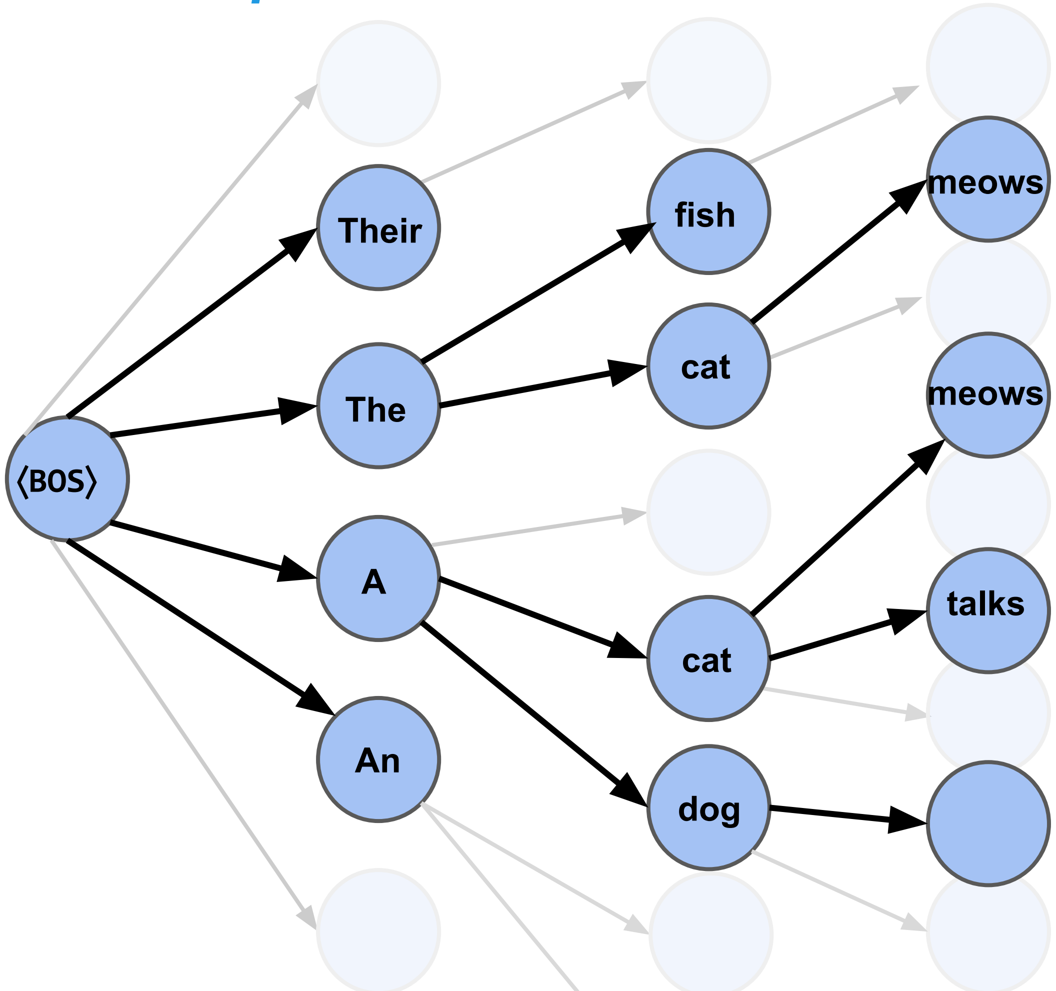
Beam Search In Action

In short: ***pruned*** breadth-first search where the breadth is limited to size k



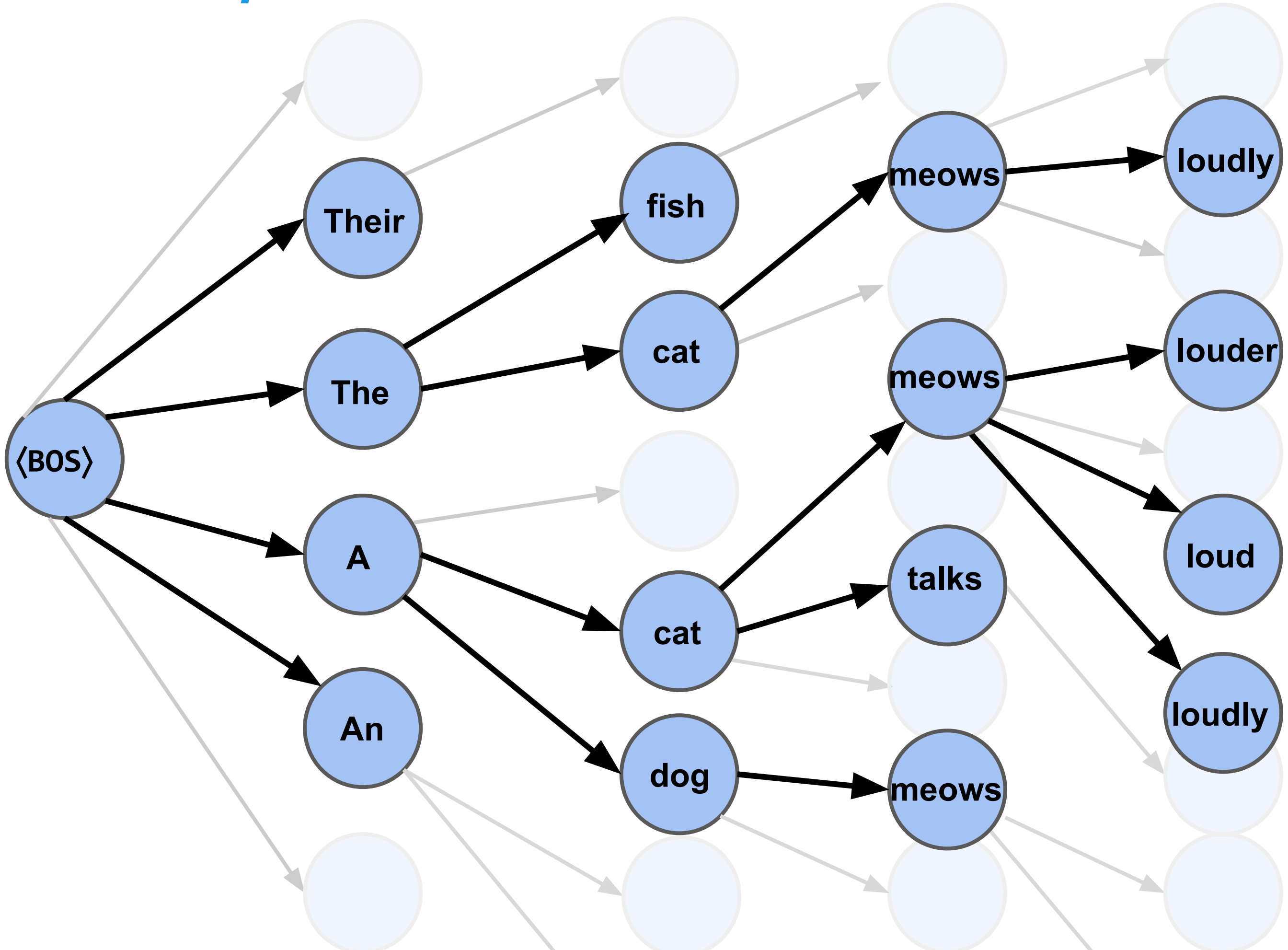
Beam Search In Action

In short: ***pruned*** breadth-first search where the breadth is limited to size k



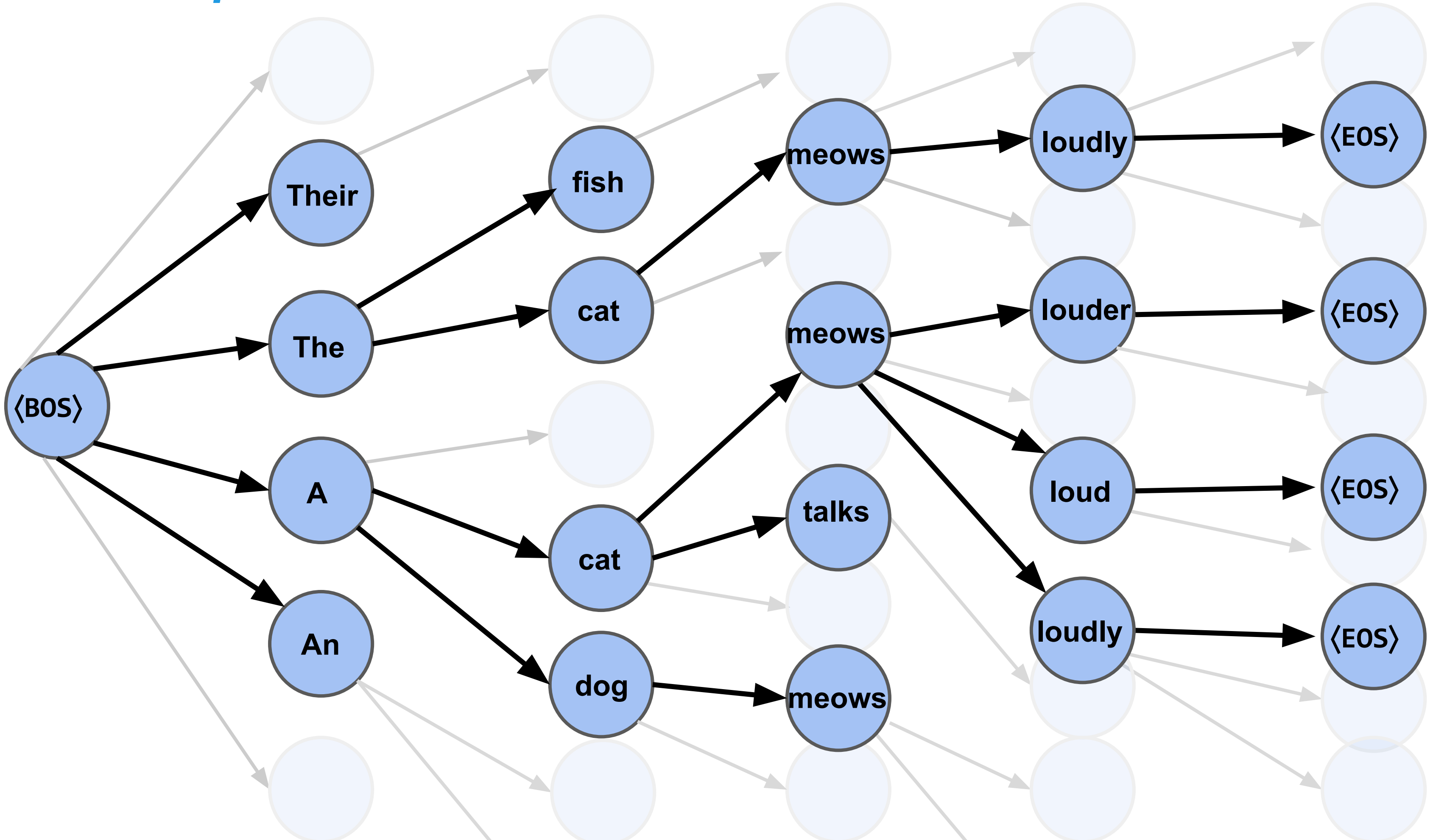
Beam Search In Action

In short: ***pruned*** breadth-first search where the breadth is limited to size k



Beam Search: A Summary

In short: ***pruned*** breadth-first search where the breadth is limited to size k



- **maximum of k paths kept at each time step**
- **greedy algorithm: no guarantee we'll find the optimal solution!**
- **despite lack of formal guarantees, does well in practice!**

How Fast is Beam Search? A Tutorial Rant

- Beam search is criminally incorrectly implemented!
- If we wish to decode a sequence of length *n* with a vocabulary *V* and with a beam size of *k*, we must do the following:
  - Select the top *k* of *n* lists (one for every time step *t*)
  - Each list has $k|V|$ items

Runtime

- $O(n\ |V|\ k^2)$ because you bubble-sorted

- $O(n\ |V|\ k \log k)$ because you merge-sorted

- **Conjecture**: $O(n\ |V|\ k)$ because you used median-of-medians

How Fast is Beam Search? A Tutorial Rant

- Beam search is criminally incorrectly implemented!

Don’t be that person!

e of length n with a vocabulary V and
do the following:

e for every time step t)

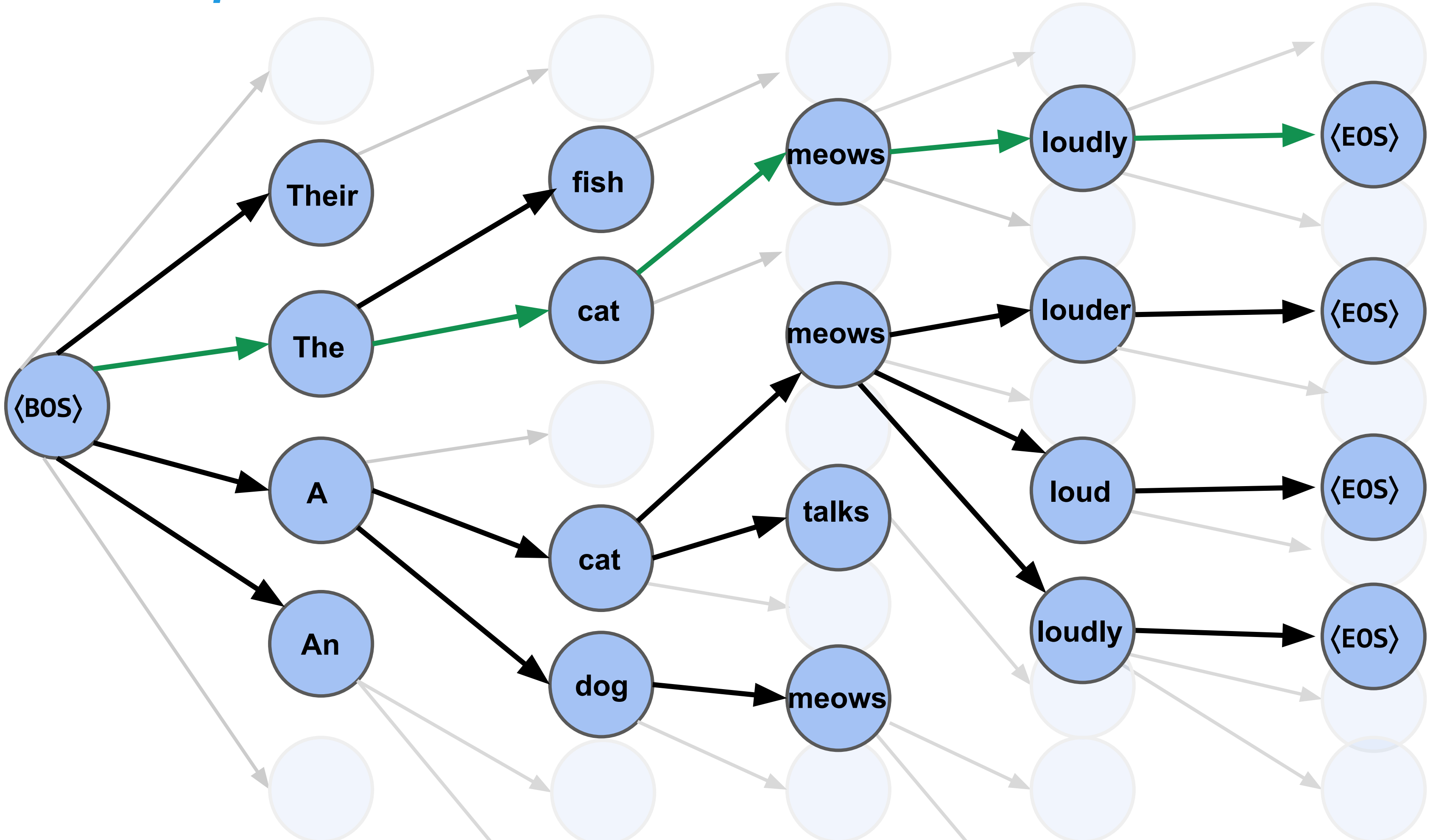
  - Each list has $k|V|$ items

Runtime

- $O(n\ |V|\ k^2)$ because you bubble-sorted
- $O(n\ |V|\ k \log k)$ because you merge-sorted
- $O(n\ |V|\ k)$ because you used median-of-medians

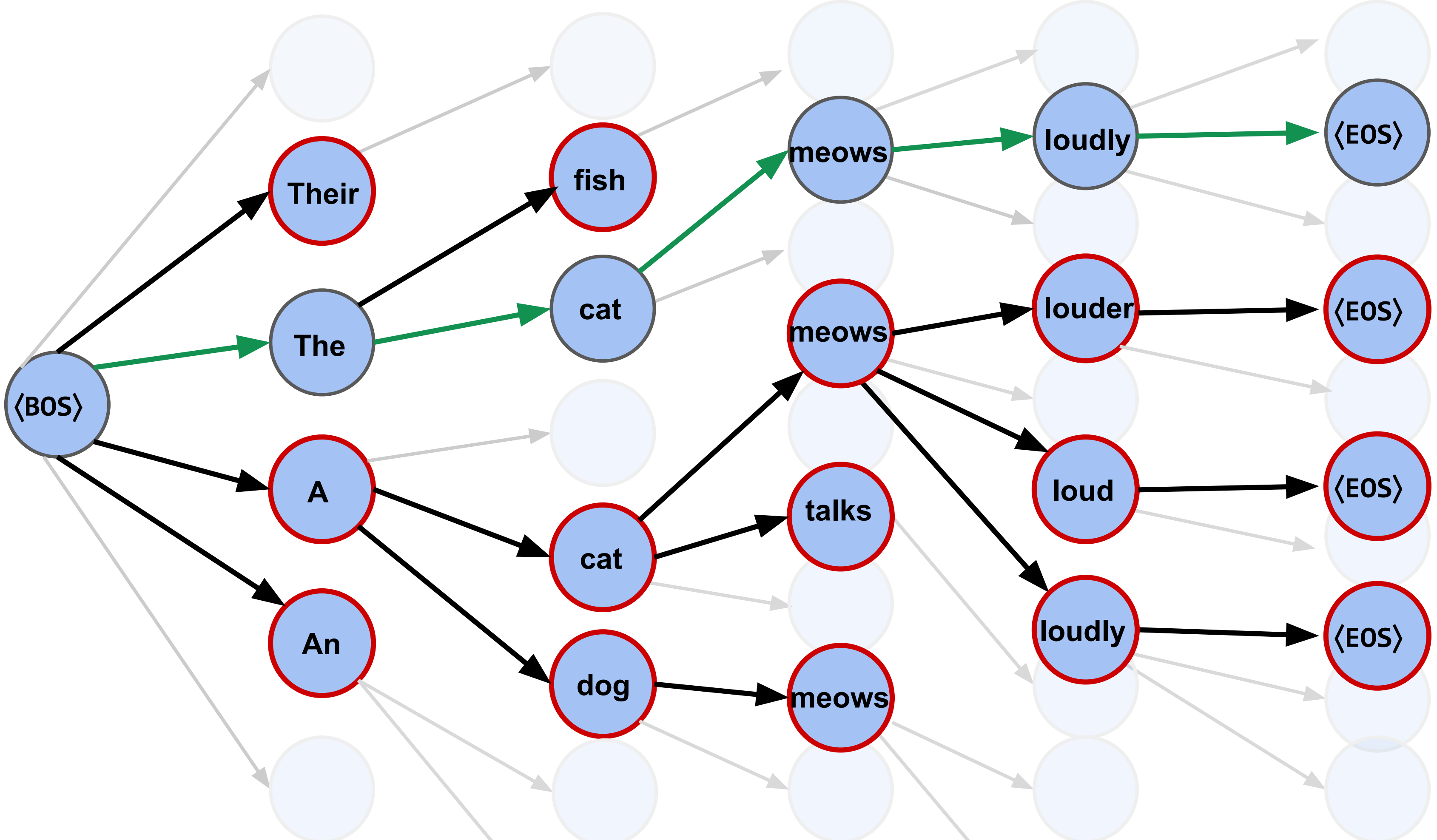
Beam Search

In short: ***pruned*** breadth-first search where the breadth is limited to size k



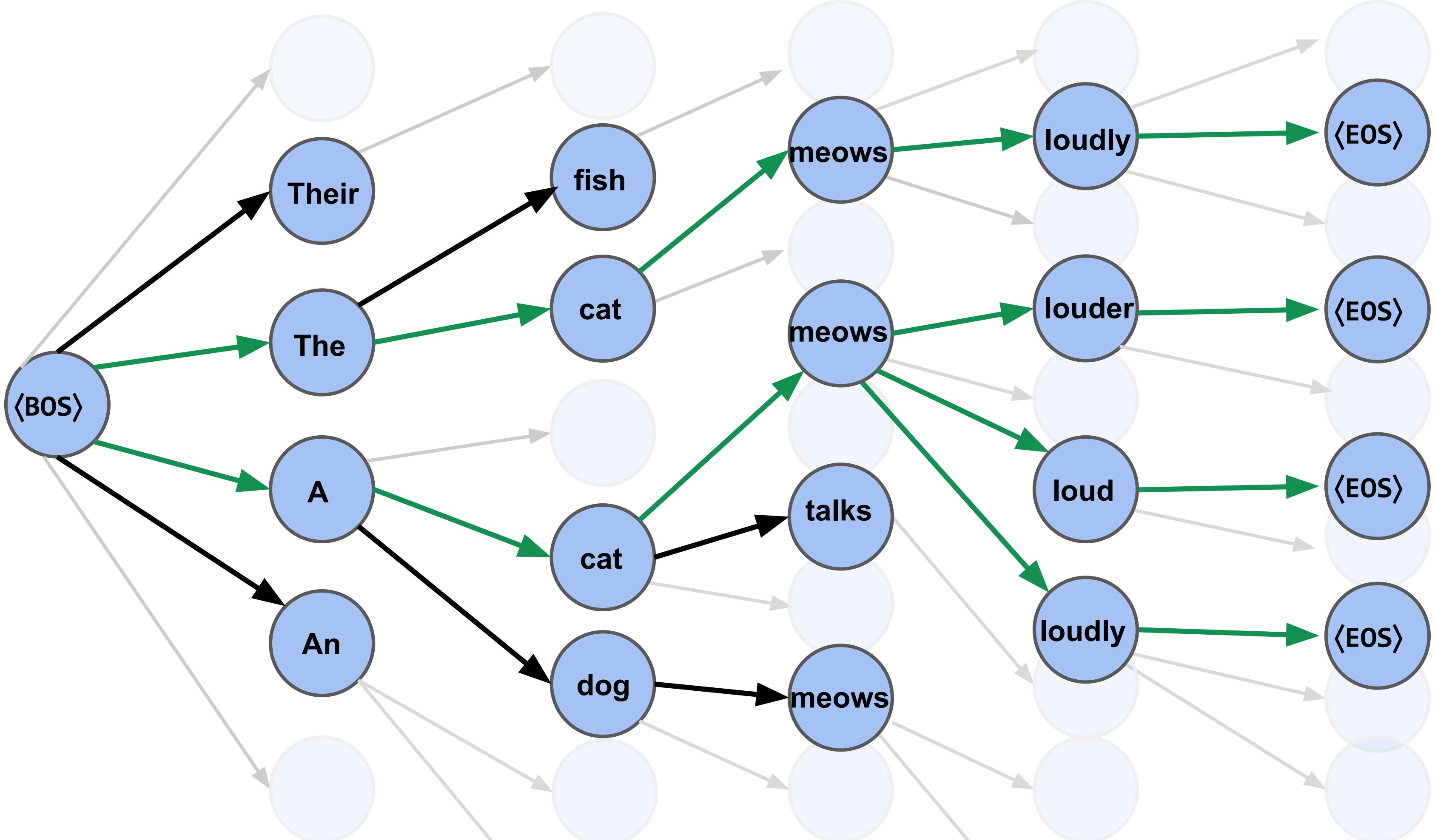
Beam Search Does Unnecessary Work!

If we only care about one path, why do we explore so many dead ends?



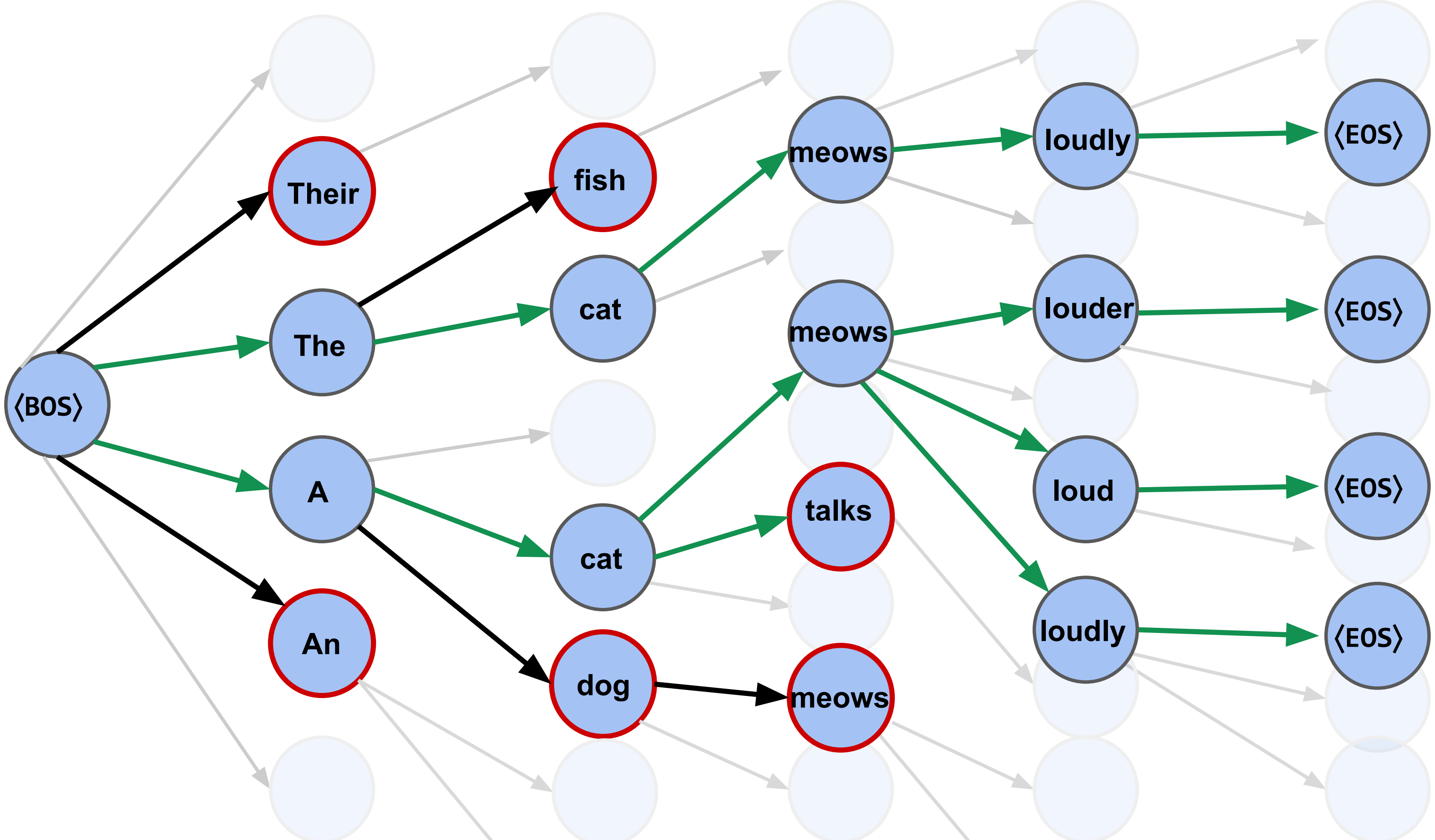
Beam Search Does Unnecessary Work!

What if we cared about more than one path?



Beam Search Does Unnecessary Work!

We still explore a large number of dead ends!



Best-first Beam Search

Key Insight Behind Best-First Beam Search

- Beam search decodes by selecting *k* hypotheses at every time step *t*
  - Indeed, we have *k* partial hypotheses for every *t* ***before we even consider*** those partial hypotheses for time step *t*+1
- What if we considered hypotheses out of order? Preference is given to those with a higher score under the model
  - Easily implemented with priority queue like Dijkstra's algorithm
- This algorithm returns the ***same set*** of hypotheses as beam search but in a different order

Best-first Beam Search

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

Best-first Beam Search

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{\boxed{t}=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

score monotonically
decreases in t

Best-first Beam Search

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

E.g., in a probabilistic model, we have:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$$

Best-first Beam Search

$$\mathbf{score}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x},\mathbf{y}_{<t})$$

E.g., in a probabilistic model, we have:

$$\mathbf{score}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x},\mathbf{y}_{<t})$$

$$\mathbf{score}(\mathbf{x},\mathbf{y}) = (-1)$$

Best-first Beam Search

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

E.g., in a probabilistic model, we have:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$$

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = (-1) + (-0.5)$$

Best-first Beam Search

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

E.g., in a probabilistic model, we have:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$$

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = (-1) + (-0.5) + (-2)$$

Best-first Beam Search

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} \mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$$

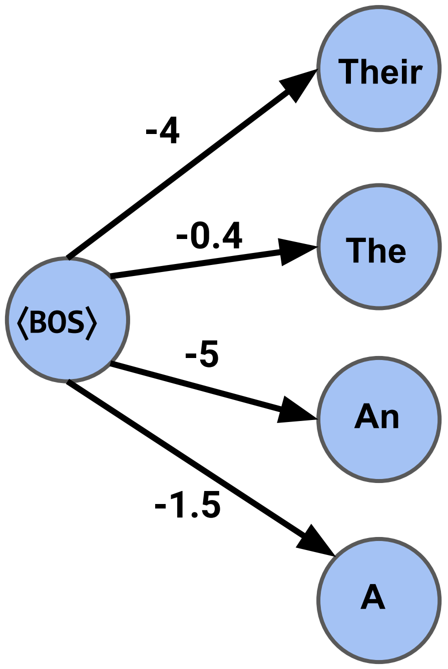
E.g., in a probabilistic model, we have:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$$

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = (-1) + (-0.5) + (-2) + (-3) + (-0.1) + \ldots$$

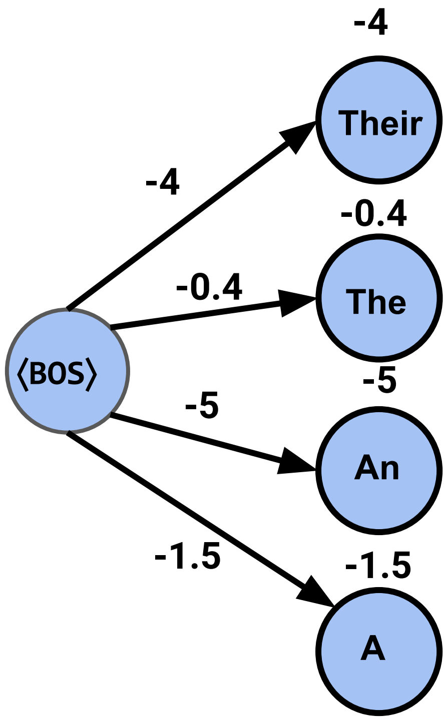
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



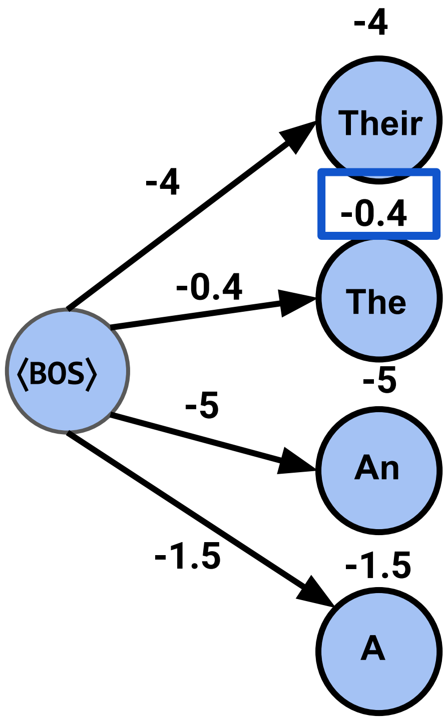
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



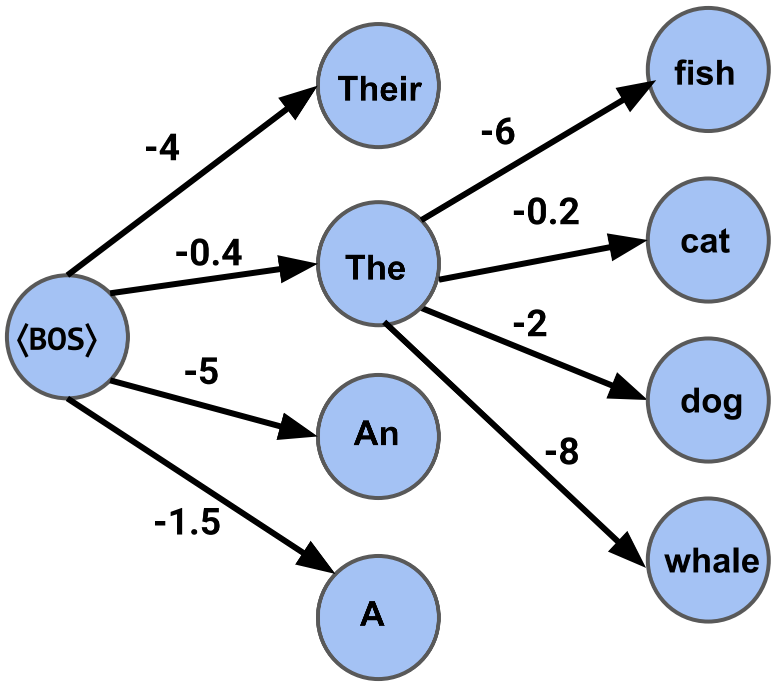
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



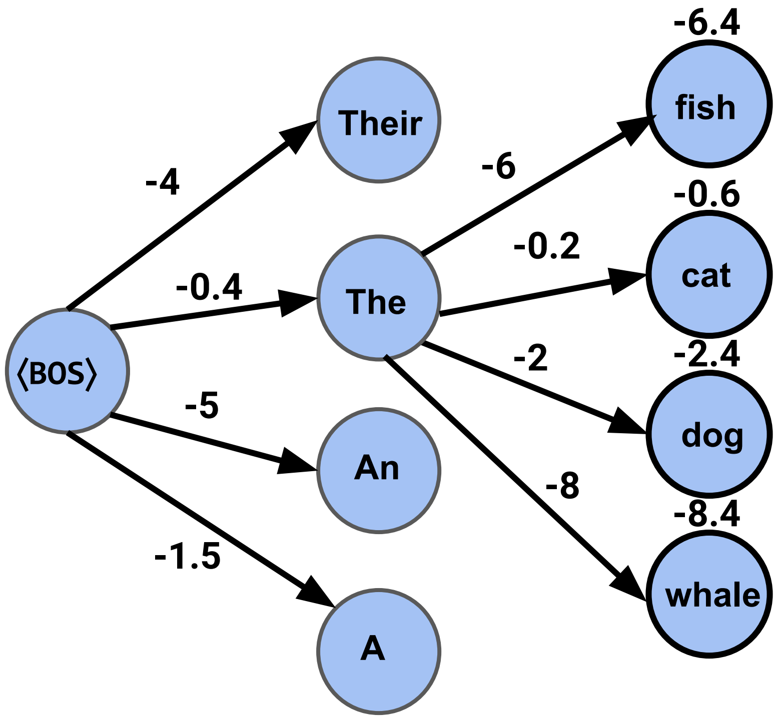
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



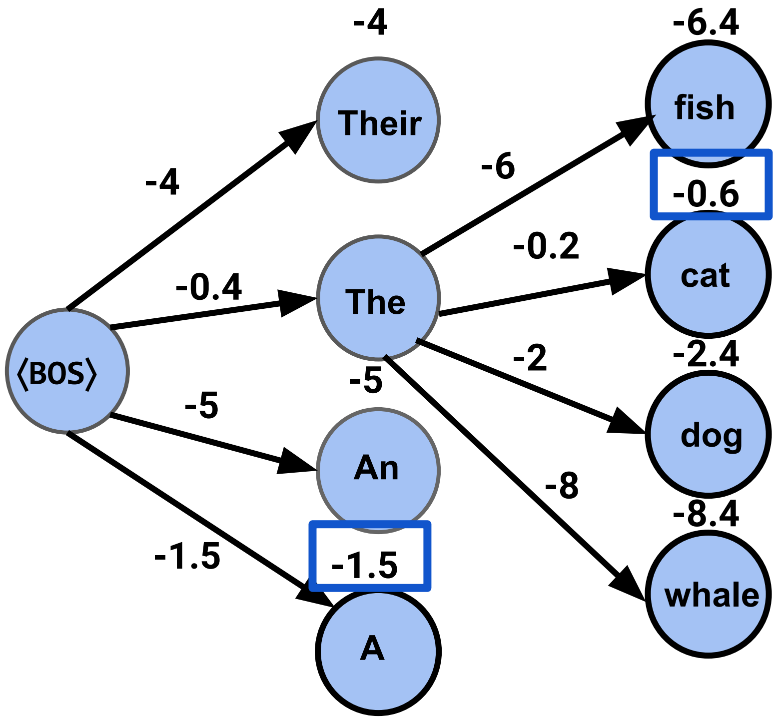
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



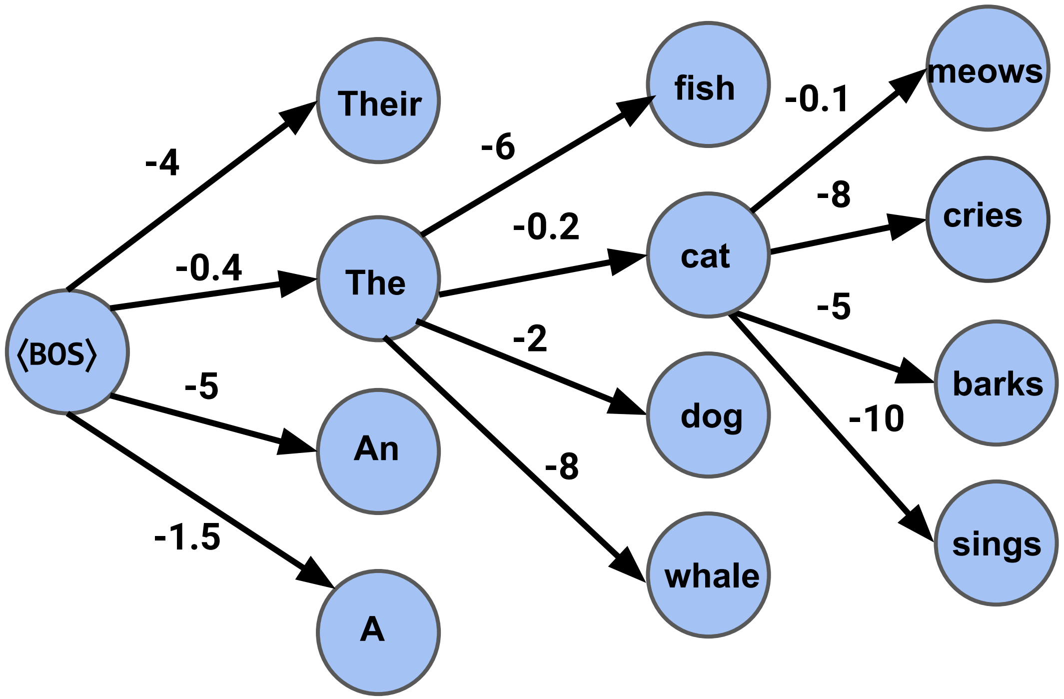
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



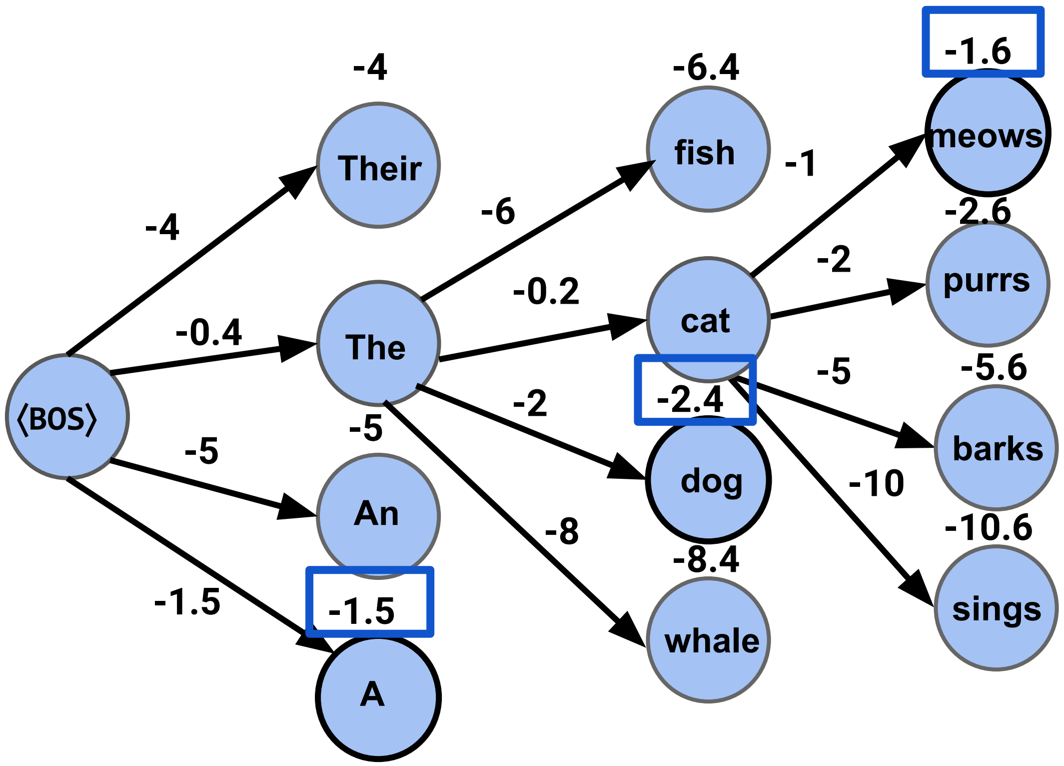
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



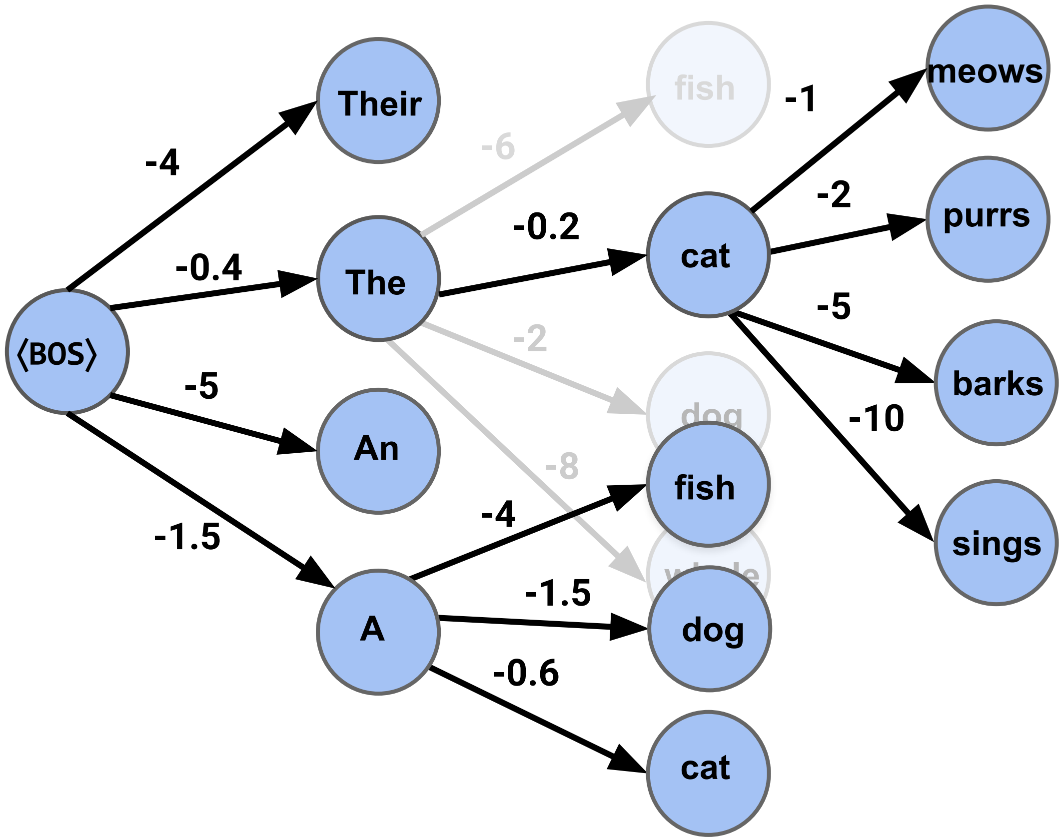
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



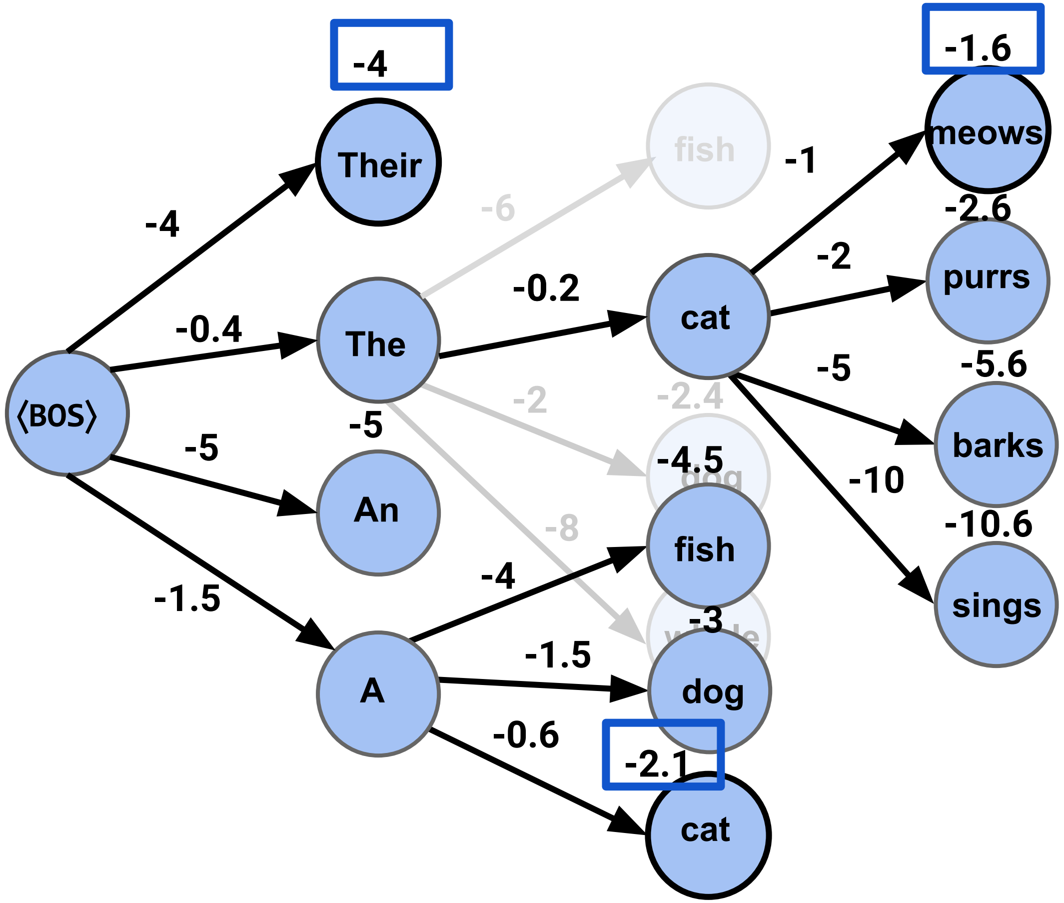
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



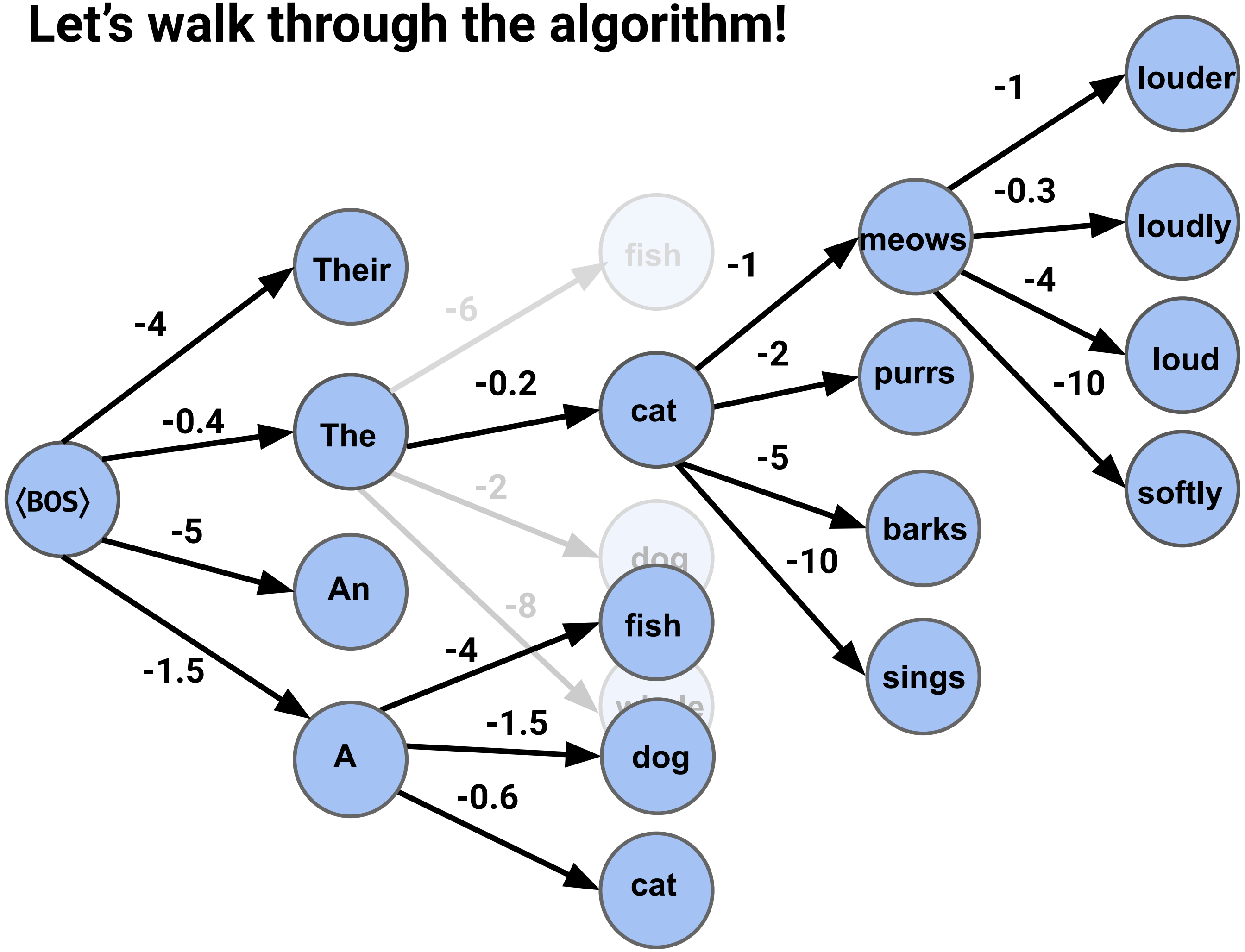
Best-first Beam Search

Let's walk through the algorithm with a beam size of 4!



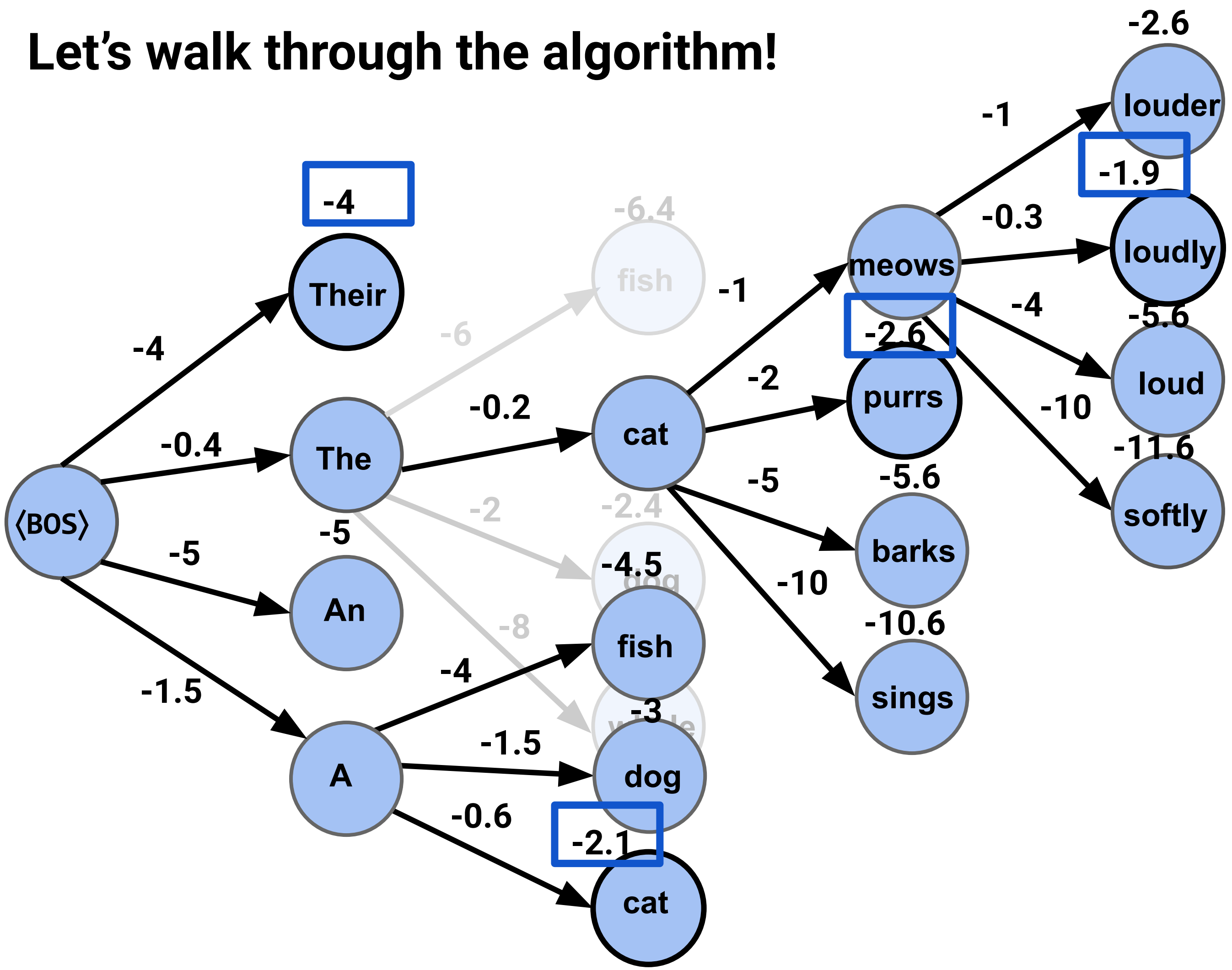
Best-first Beam Search

Let's walk through the algorithm!



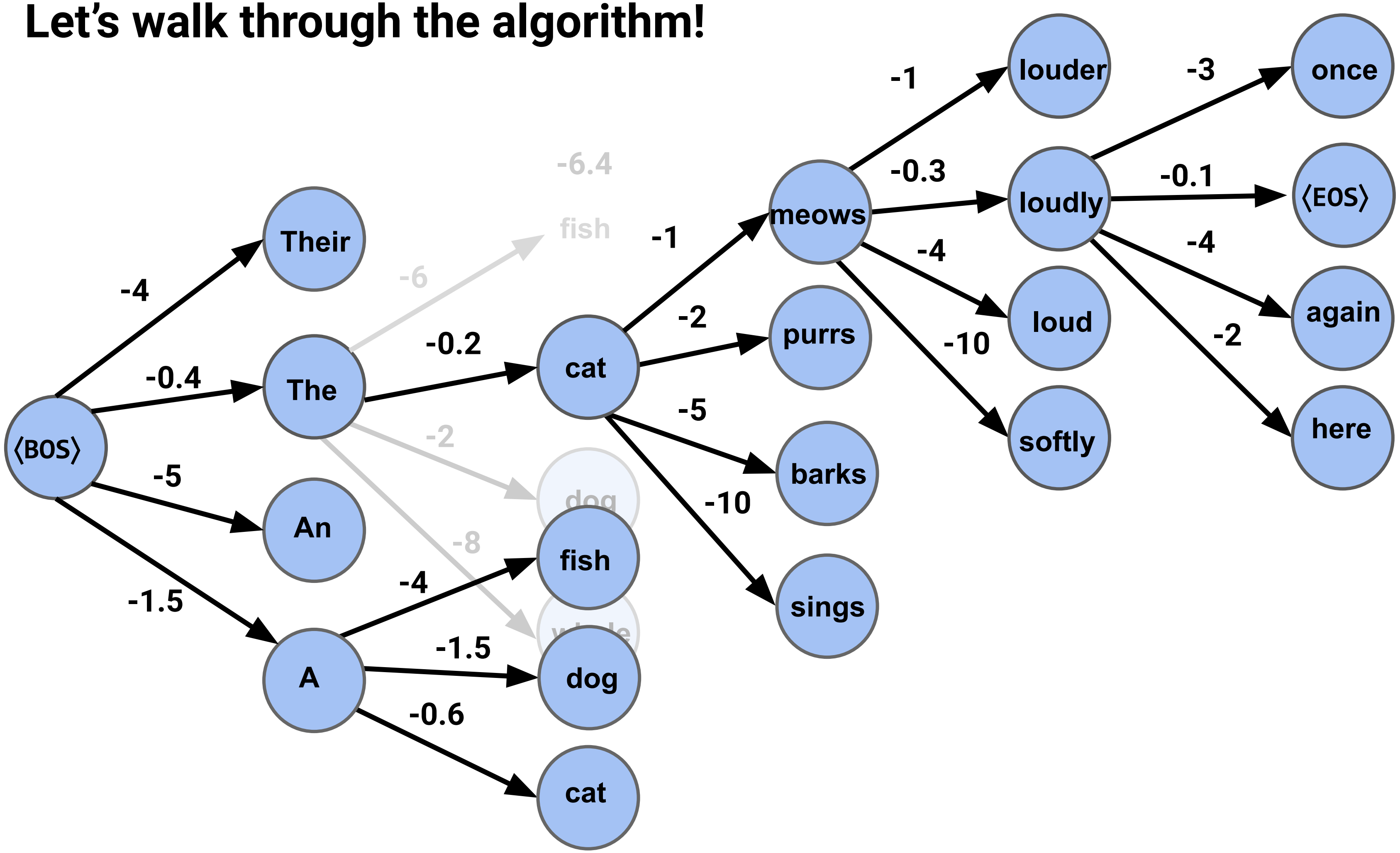
Best-first Beam Search

Let's walk through the algorithm!



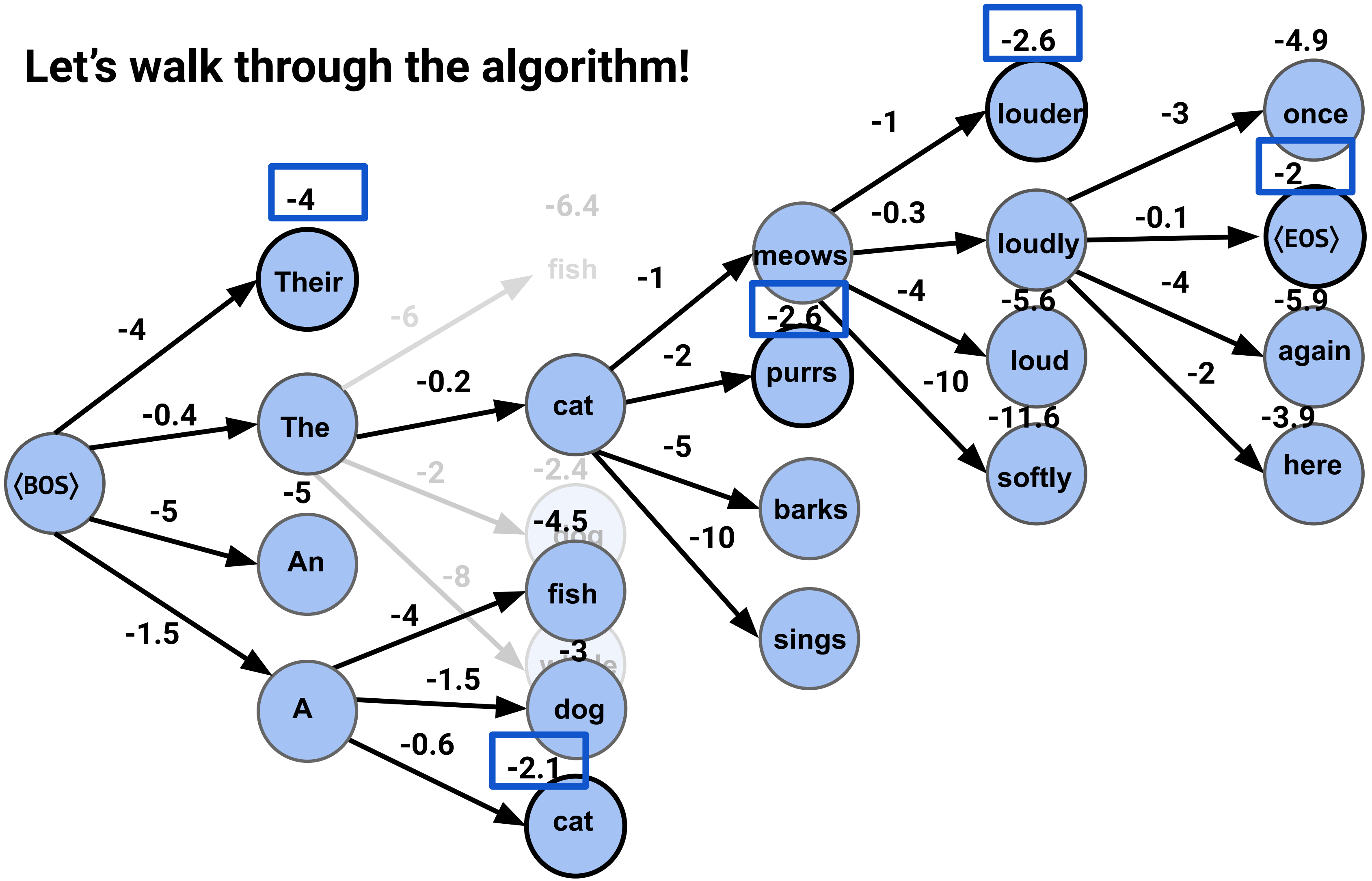
Best-first Beam Search

Let's walk through the algorithm!



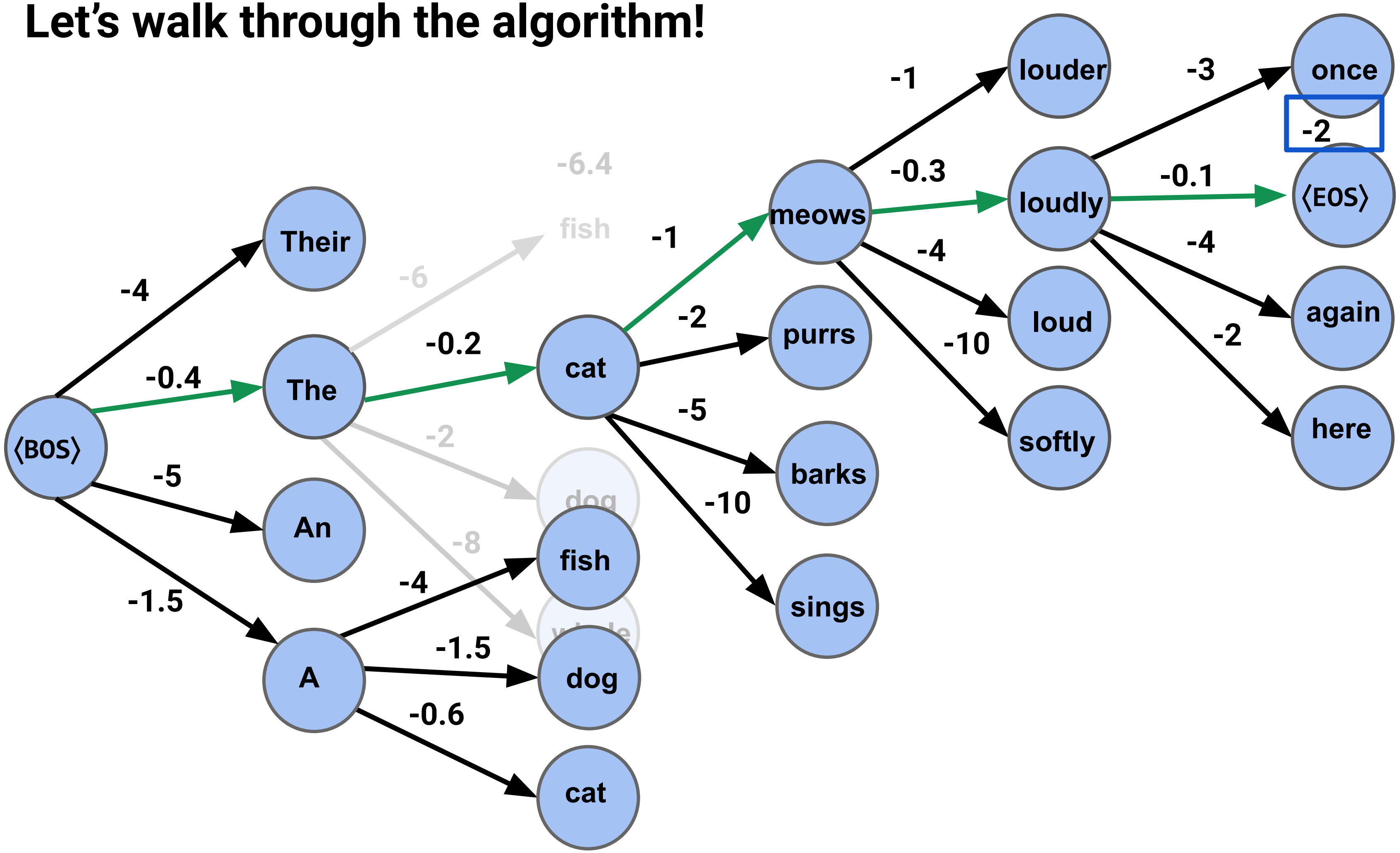
Best-first Beam Search

Let's walk through the algorithm!



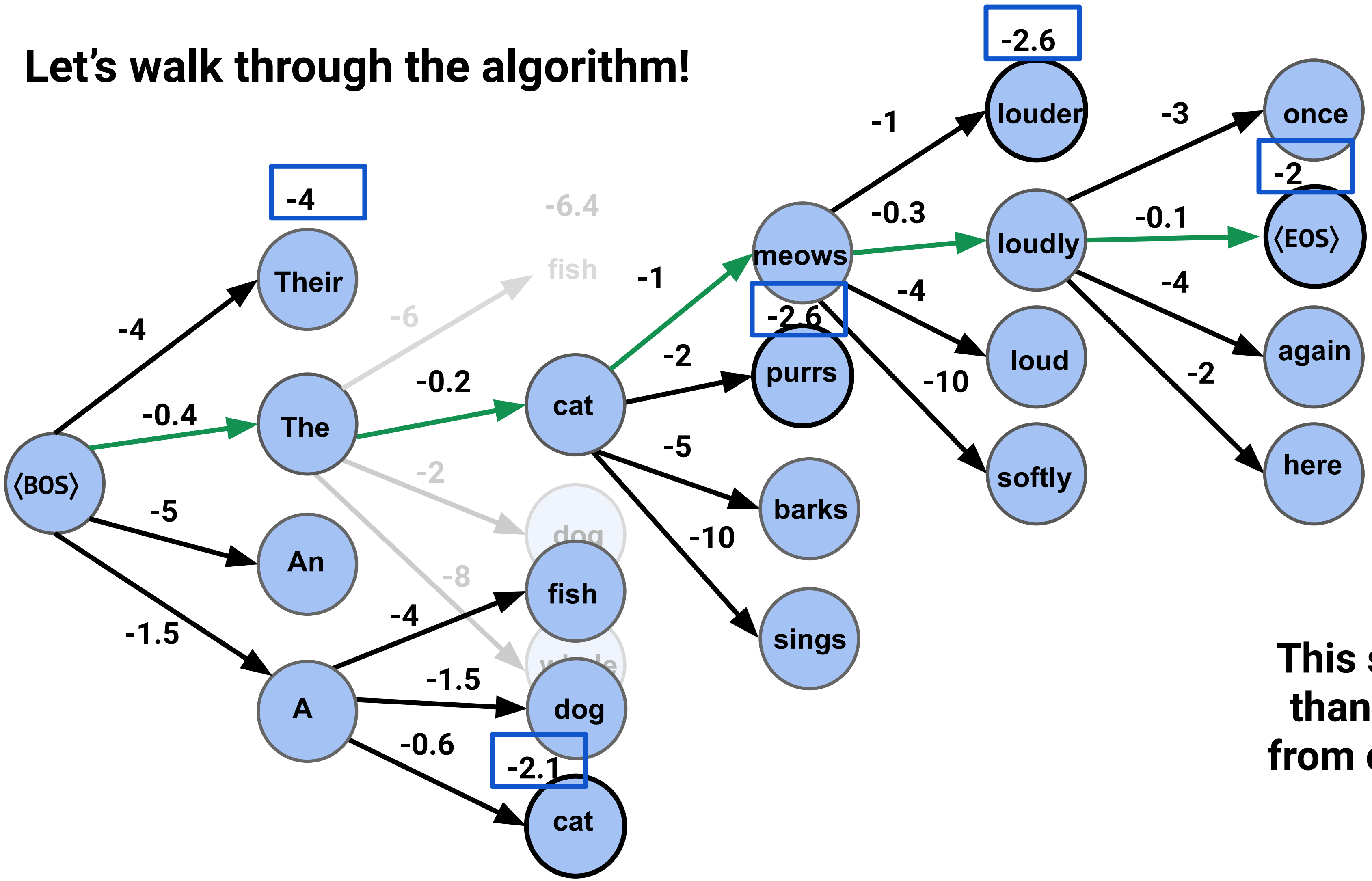
Best-first Beam Search

Let's walk through the algorithm!



Best-first Beam Search

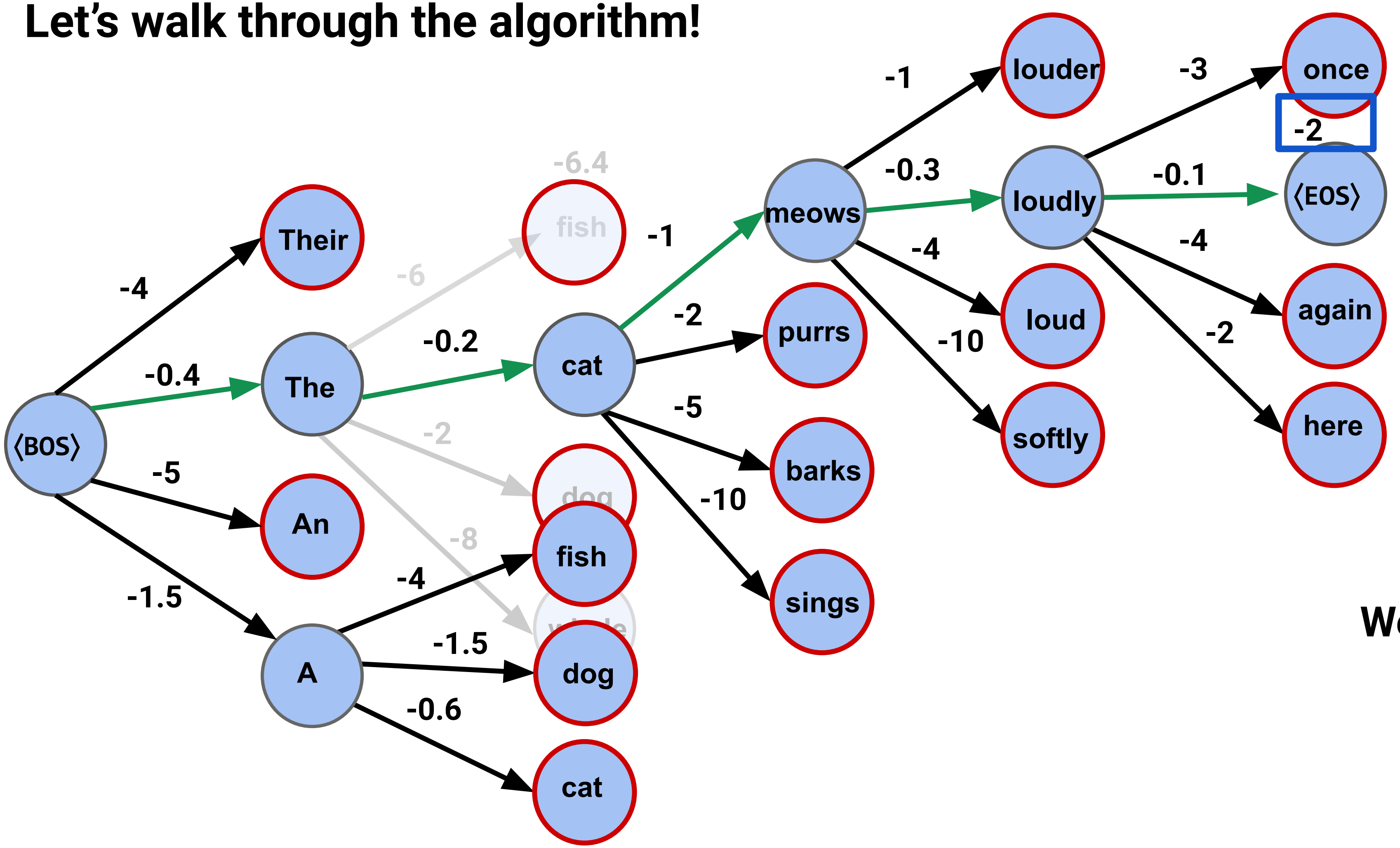
Let's walk through the algorithm!



This solution must be better than any of those resulting from children of other nodes!

Best-first Beam Search

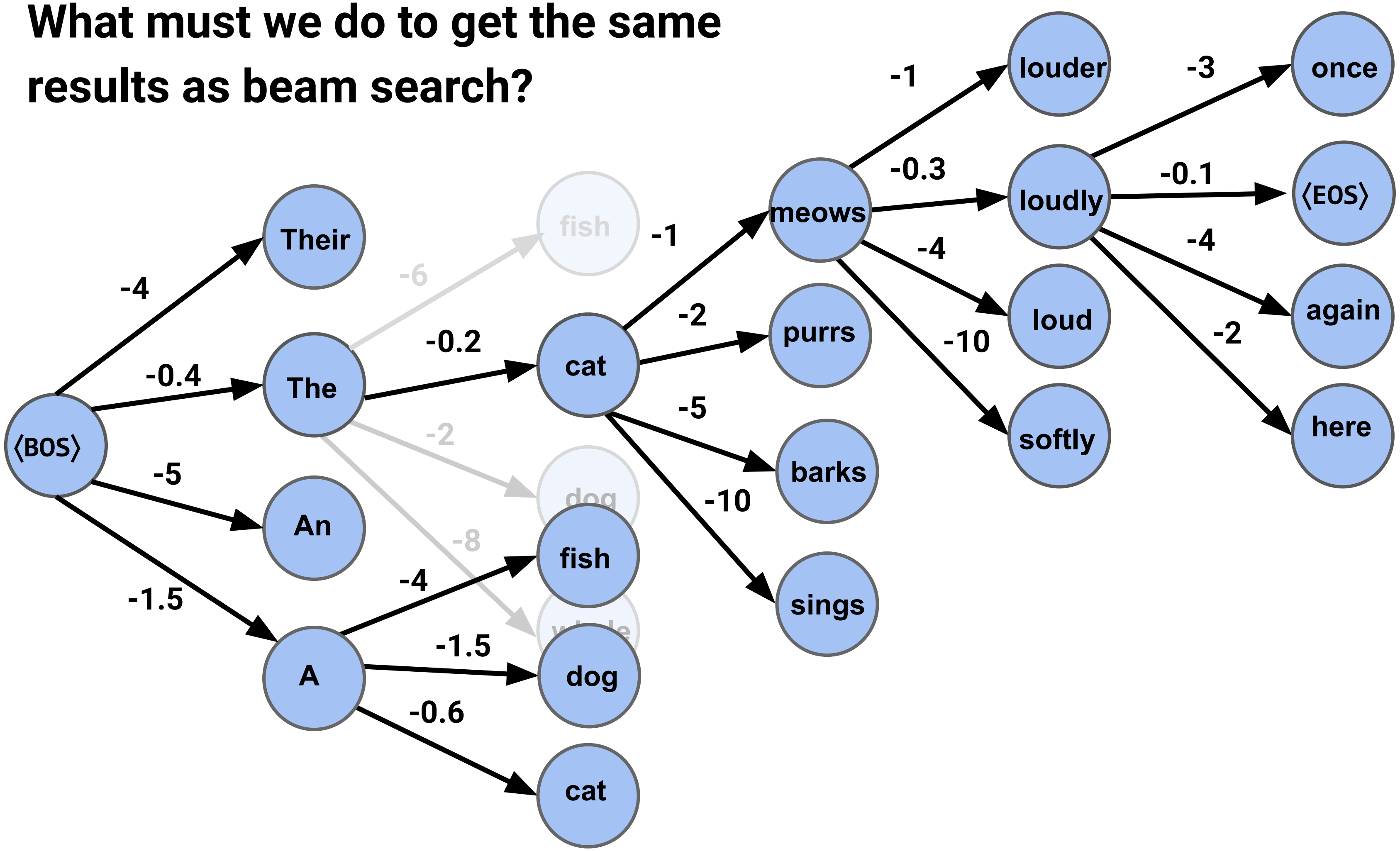
Let's walk through the algorithm!



We just saved a lot of computations…

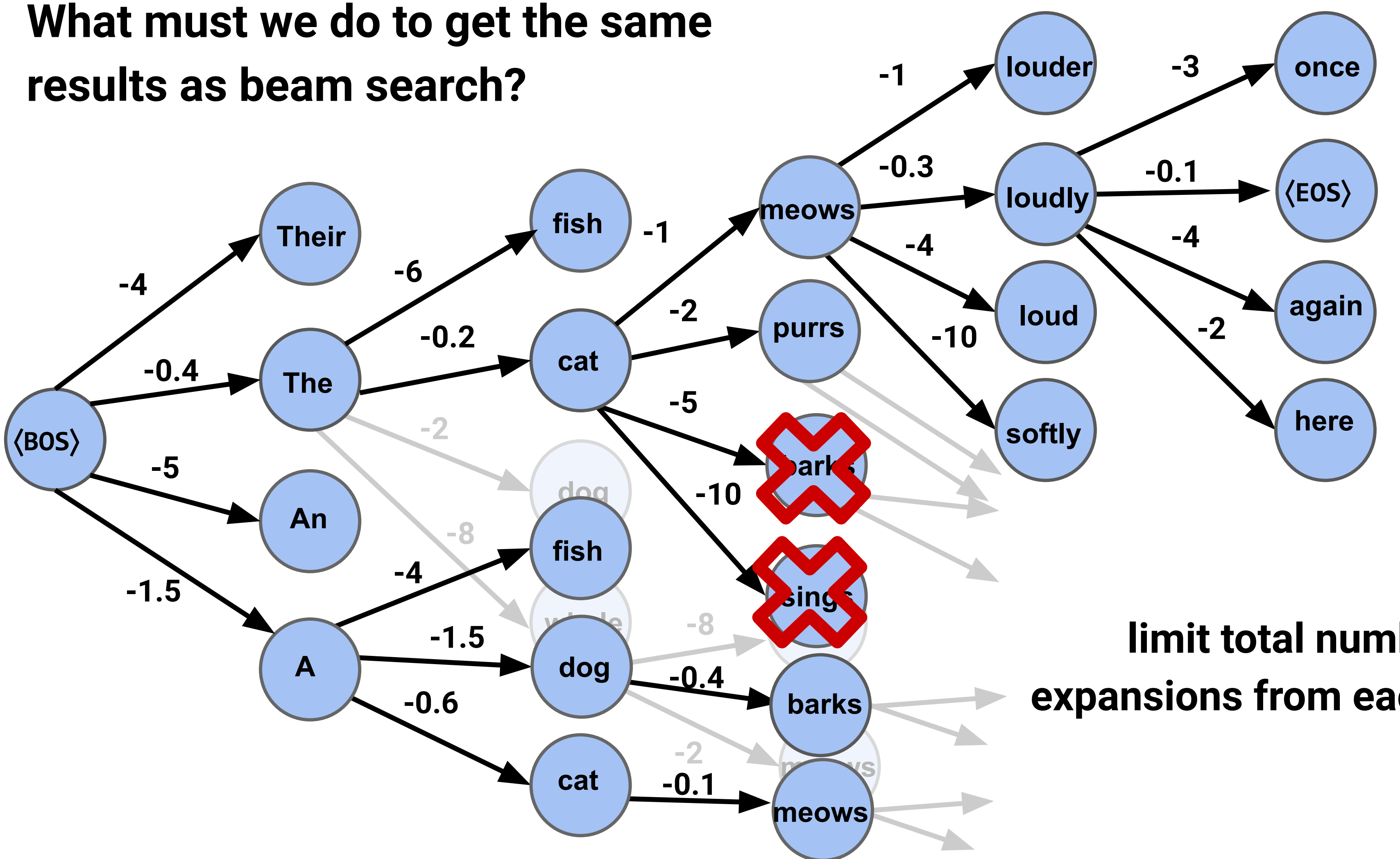
Best-first Beam Search

What must we do to get the same results as beam search?



Best-first Beam Search

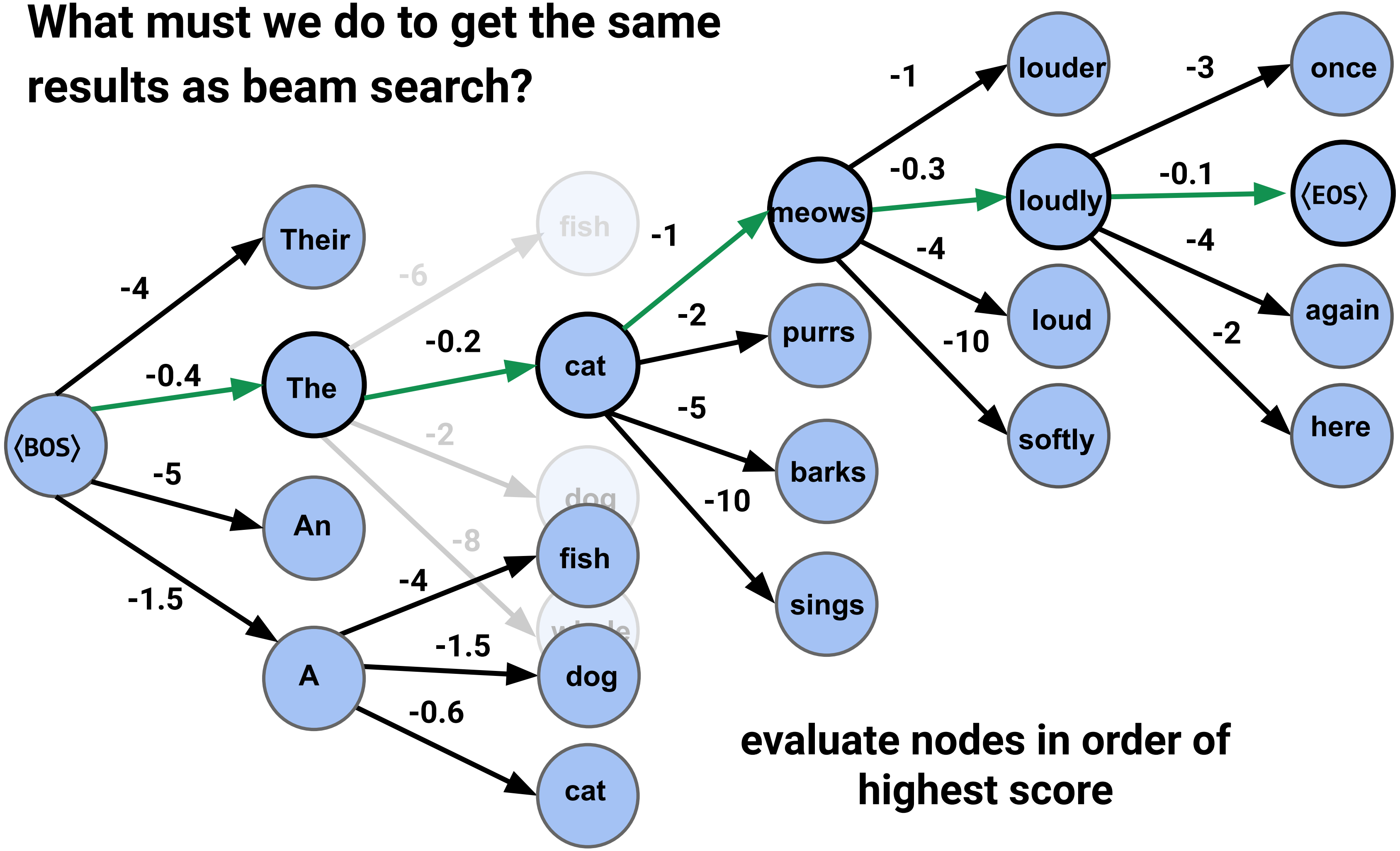
What must we do to get the same results as beam search?



limit total number of node expansions from each time step to k

Best-first Beam Search

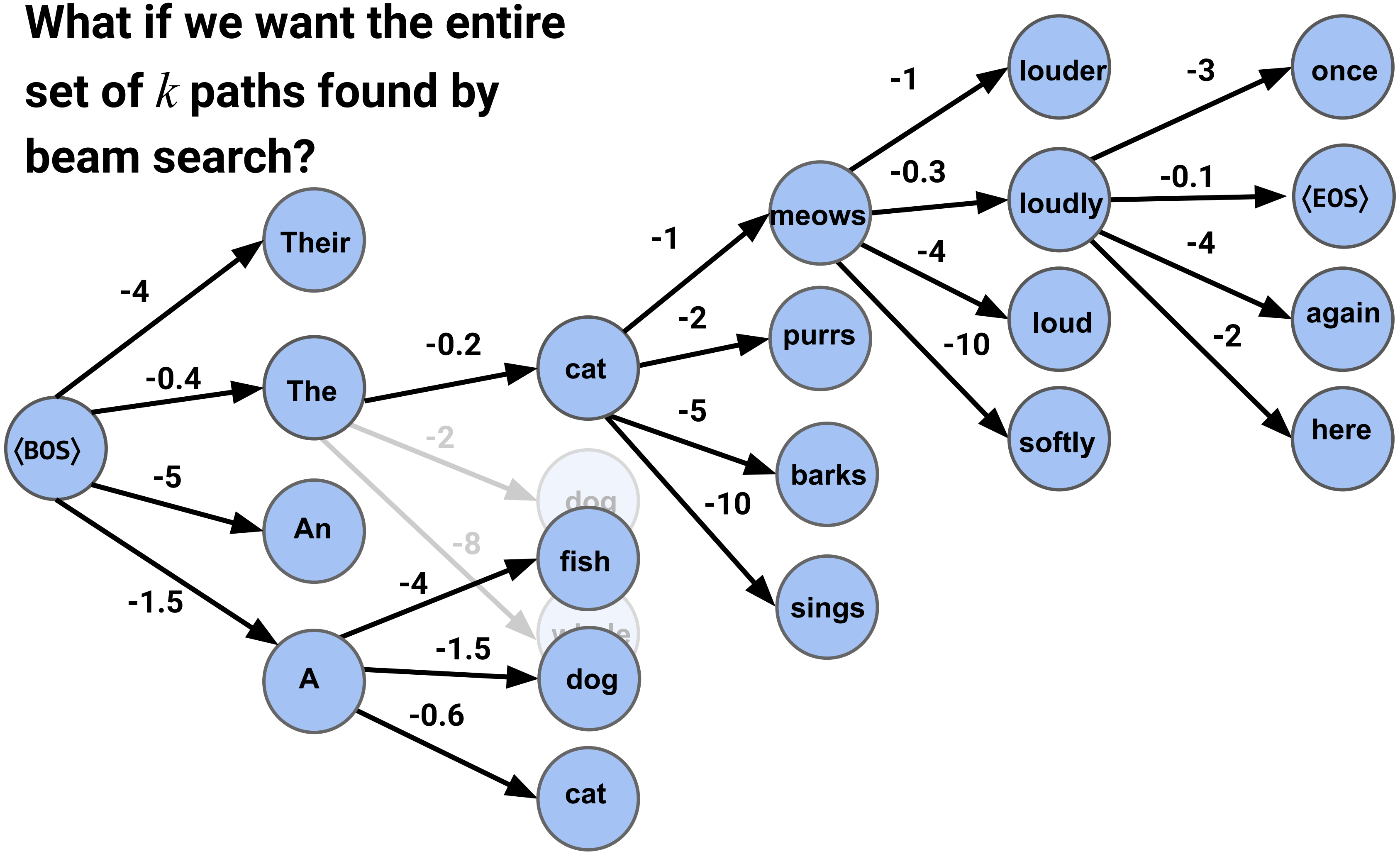
What must we do to get the same results as beam search?



evaluate nodes in order of highest score

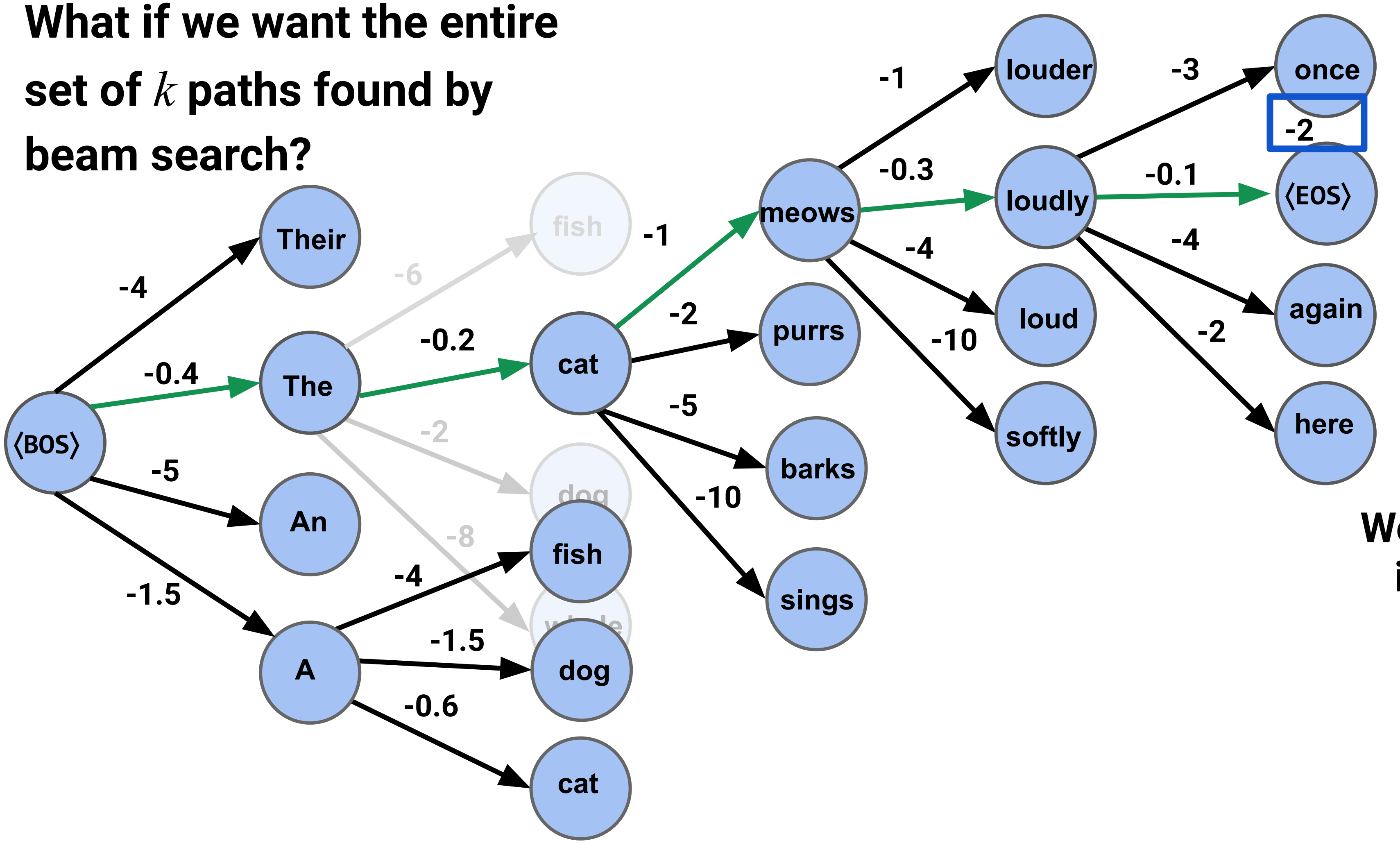
Best-first Beam Search

What if we want the entire set of k paths found by beam search?



Best-first Beam Search

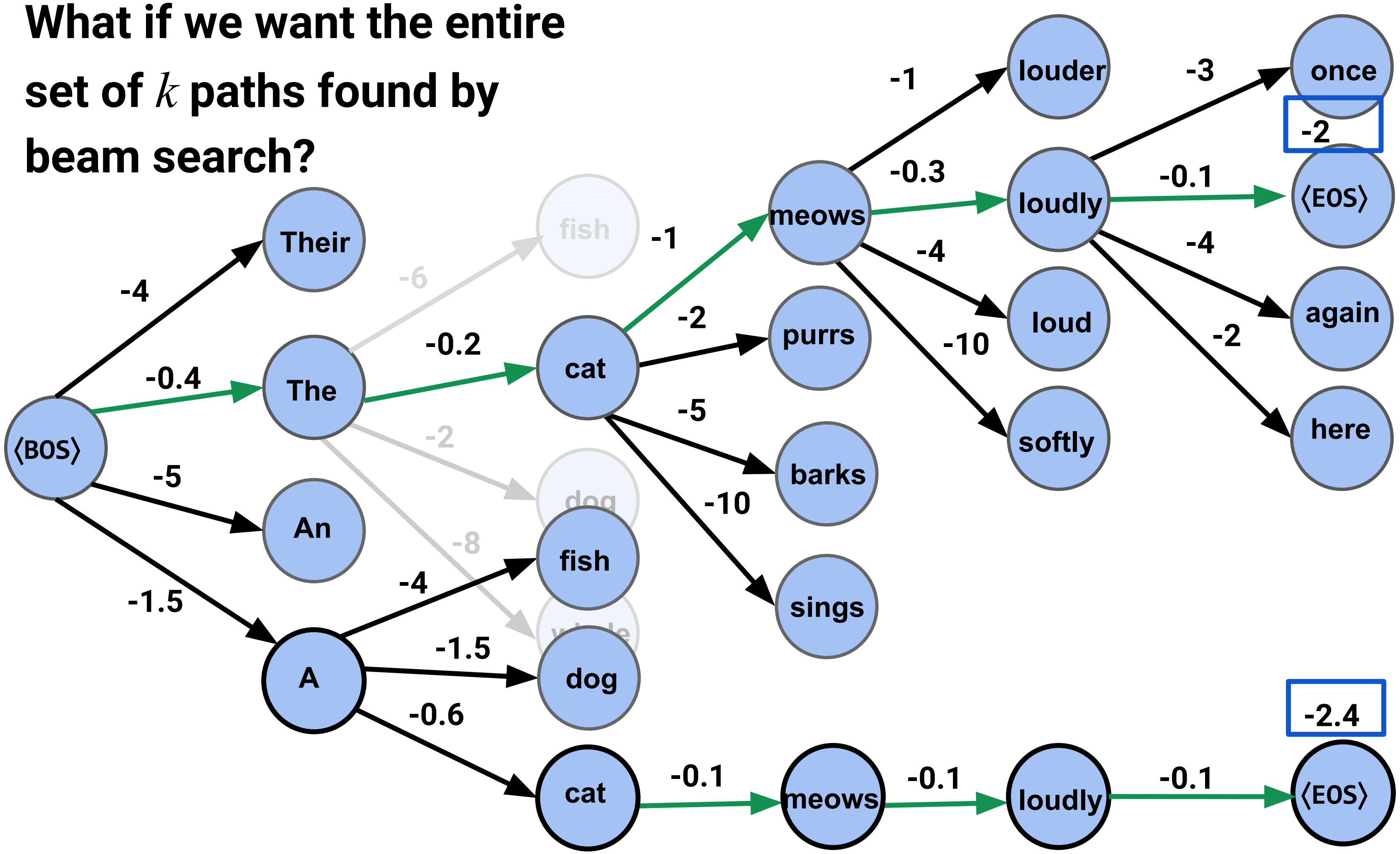
What if we want the entire set of k paths found by beam search?



We keep expanding paths in order of best score!

Best-first Beam Search

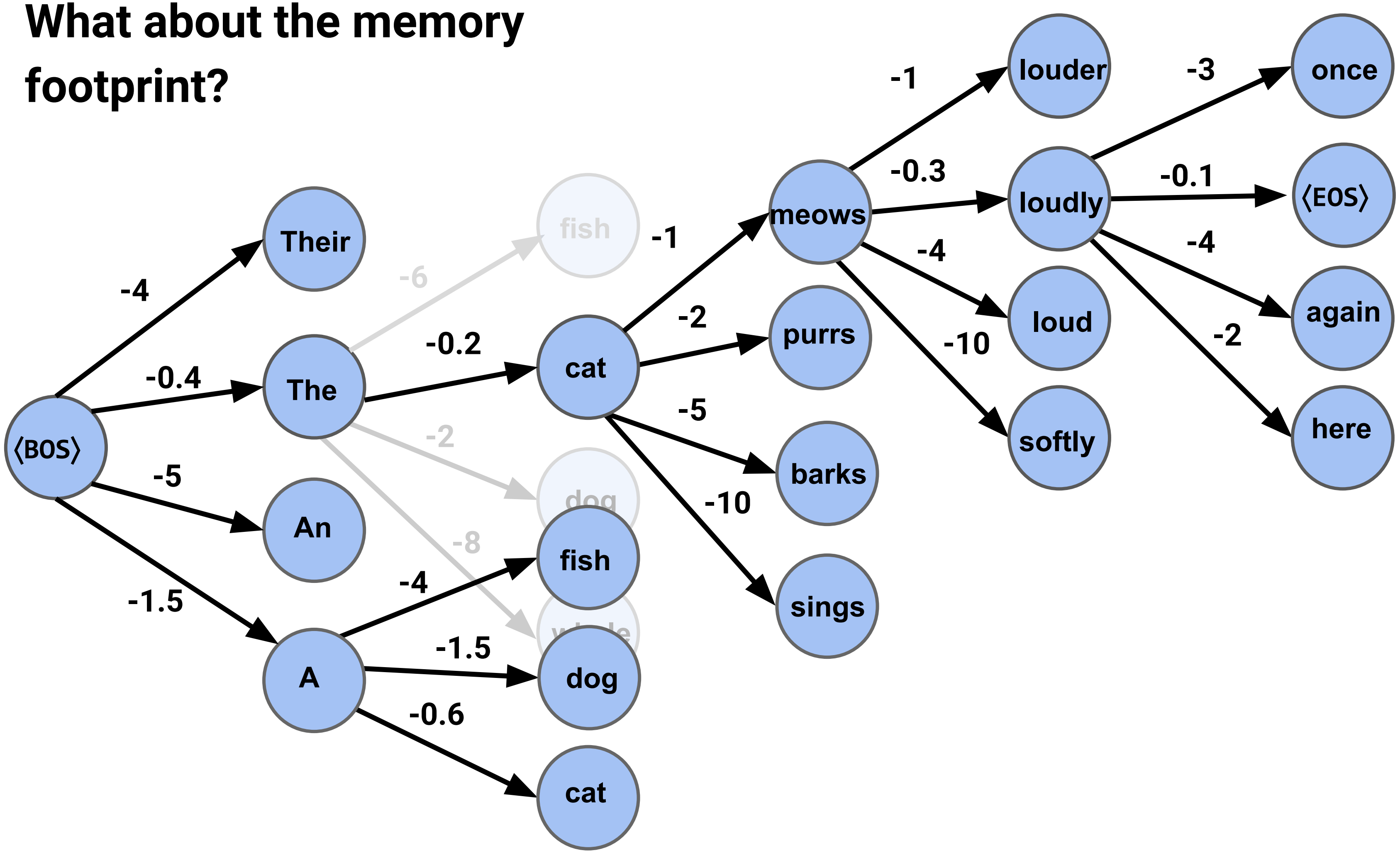
What if we want the entire set of k paths found by beam search?



if we keep expanding paths in order of best score, the next complete solution will be the second best found by beam search!

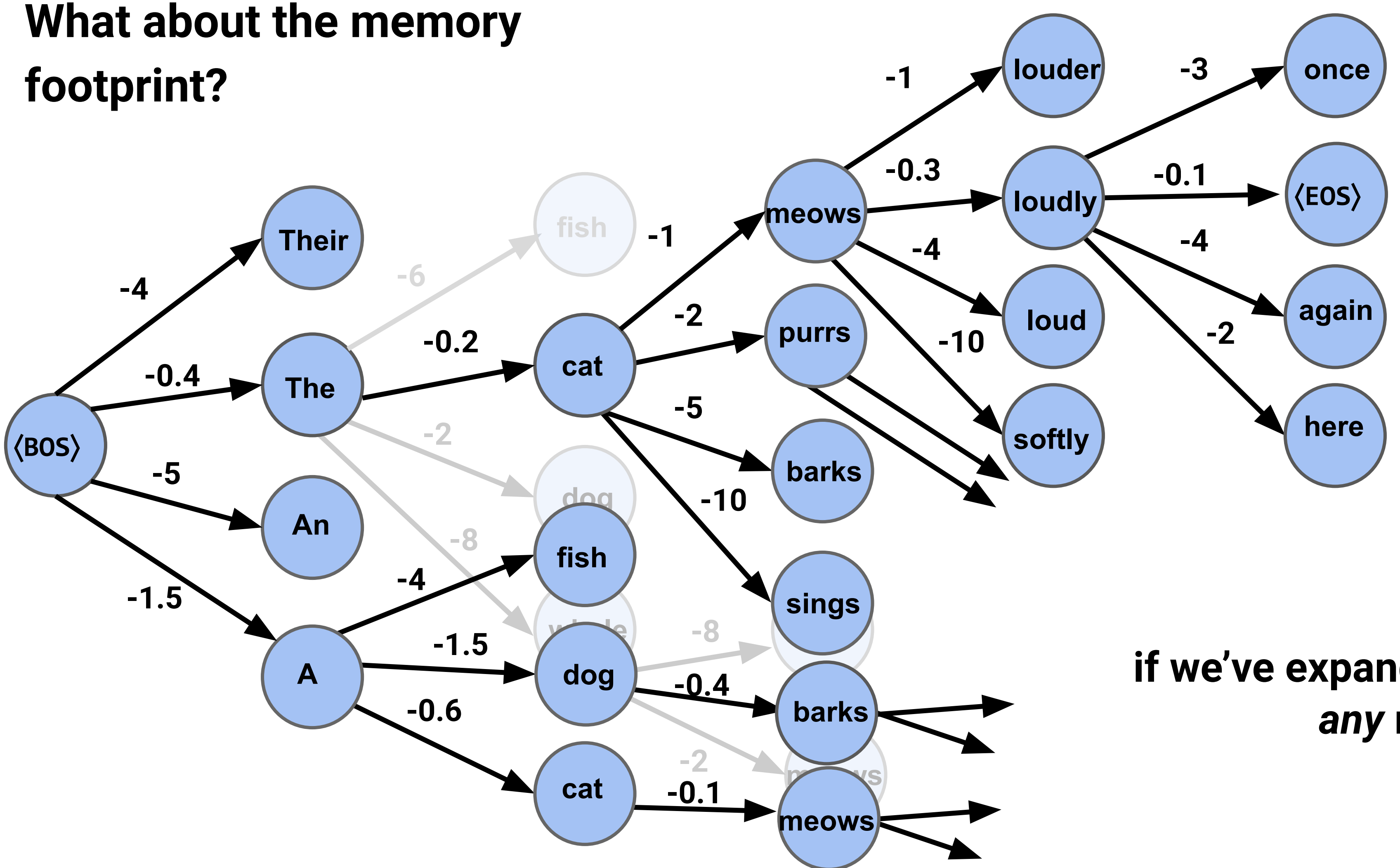
Best-first Beam Search

What about the memory footprint?



Best-first Beam Search

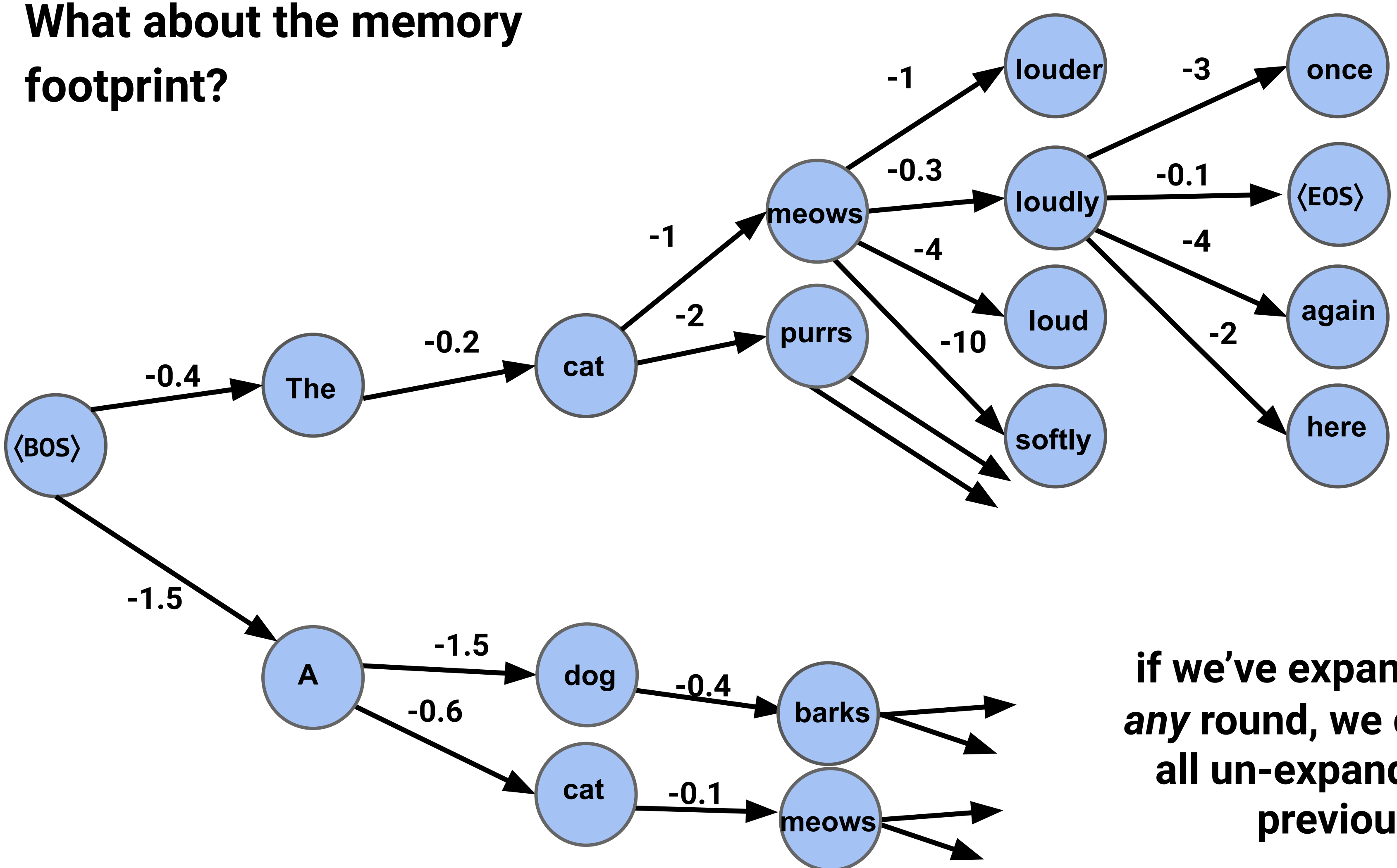
What about the memory footprint?



if we've expanded k nodes for *any* round

Best-first Beam Search

What about the memory footprint?

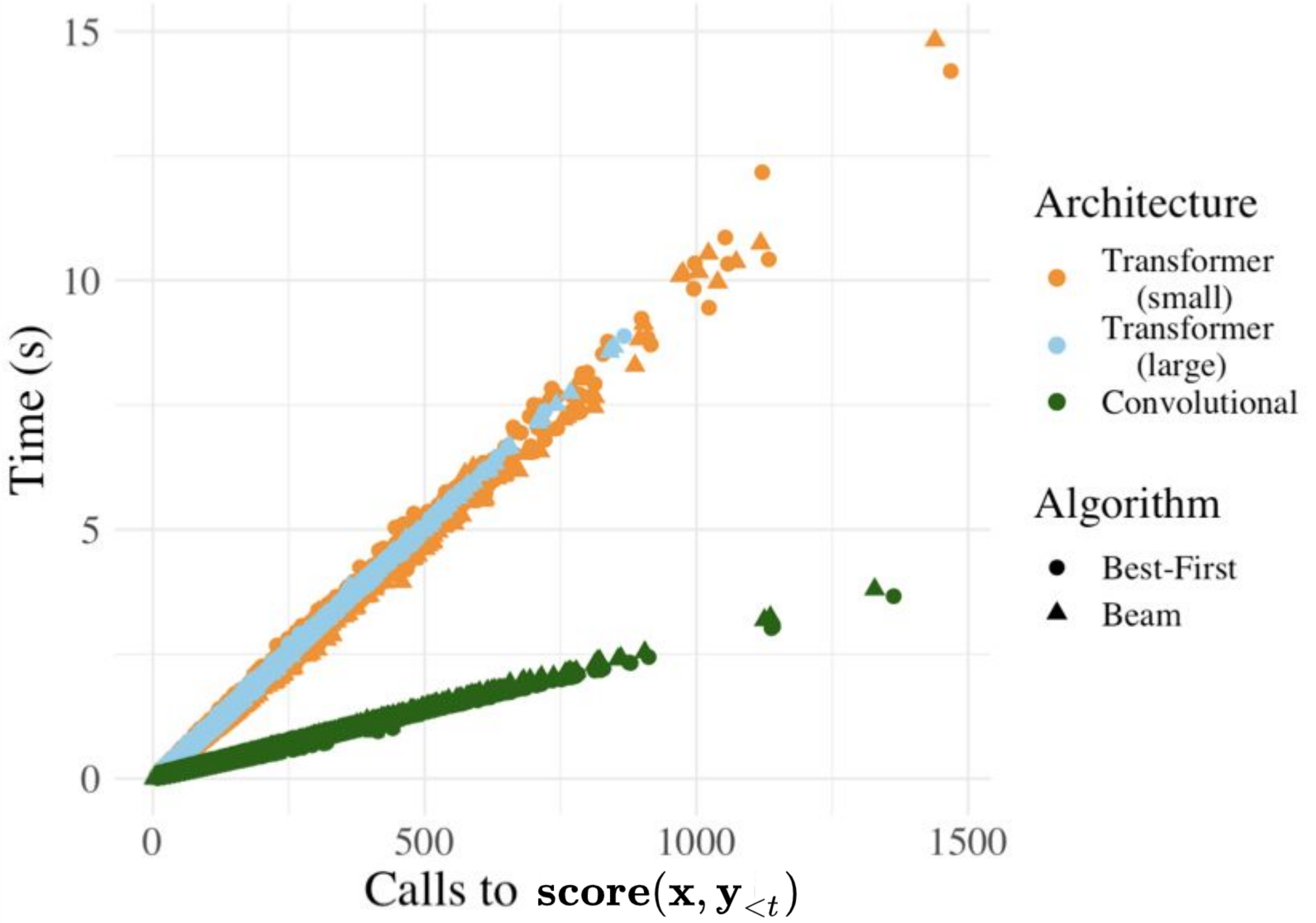


if we've expanded k nodes for *any* round, we can forget about all un-expanded paths from previous rounds!

Best-first Beam Search

Does this actually lead to speed-ups in practice?

We can measure speed-ups in terms of calls to our scoring function, which is directly related to the amount of time it takes to decode a sequence



Best-first Beam Search

Does this actually lead to speed-ups in practice? YES!!!

Calls to $\mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$

	k = 5	k = 10	k = 100	k = 500
Beam Search	115	229	2286	9770
Beam Search (ES)	107	210	2047	7685
BF Beam Search	93	169	1275	1168

Best-first Beam Search

Does this actually lead to speed-ups in practice? YES!!!

(· %) = percentage decrease in comparison to baseline

Calls to $\mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})$

	k = 5	*k* = 10	*k* = 100	*k* = 500
Beam Search	115	229	2286	9770
Beam Search (ES)	107 **(7%)**	210 **(8%)**	2047 **(10%)**	7685 **(21%)**
BF Beam Search	93 **(19%)**	169 **(26%)**	1275 **(44%)**	1168 **(88%)**

Best-first Beam Search

The monotonicity constraint on our score function is hindering…

length normalization:

$$\textbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\theta}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) + \lambda |\mathbf{y}|$$

mutual information decoding:

$$\textbf{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \log p_{\theta}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) - \log p_{\theta'}(\mathbf{y})$$

Can we get around it? Yes, if we can bound the non-monotonicity of the regularizer!

Insight Number Two

But, how good is beam search at search?

Beam Search

- How often *does* beam search find the global optimum in language generation tasks?

But, how good is beam search at search?

Beam Search

- How often *does* beam search find the global optimum in language generation tasks?

  Answer: Not often.*

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

*At least not for language generation tasks for which this question has been studied

But, how good is beam search at search?

Beam Search

- Yet how come it does so well?

  Answer: ????

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

But, how good is beam search at search?

Beam Search

- And how come it does **so much** better than exact search?

  Answer: ???????????????

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

But, how good is beam search at search?

Beam Search

- The solution to MAP inference is clearly not desirable text...

$$\mathbf{y}^\star = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \ \log p_\theta(\mathbf{y} \mid \mathbf{x})$$

Search	**BLEU**	**#Search Errors**	**#Empty**
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

But, how good is beam search at search?

Beam Search

- But the solution provided by beam search is...

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \ ?$$

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

Beam Search is Iterative Subset Optimization

Our (clunky) algorithm for beam search

$$Y_0 = \{\textsc{bos}\}$$

$$Y_t = \underset{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}}{\operatorname{argmax}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x})$$

$$\mathcal{B}_t = \left\{ \mathbf{y}_{t\text{-}1} \circ y \mid y \in \bar{\mathcal{V}} \textbf{ and } \mathbf{y}_{t\text{-}1} \in Y_{t\text{-}1} \right\}$$

Return $Y_{n_{\max}}$

What does beam search optimize?

Our (clunky) algorithm for beam search

$$Y_0 = \{\textsc{bos}\}$$

$$Y_t = \underset{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}}{\operatorname{argmax}} \log p_{\theta}(Y' \mid \mathbf{x})$$

$$\mathcal{B}_t = \left\{ \mathbf{y}_{t\text{-}1} \circ y \mid y \in \bar{\mathcal{V}} \text{ and } \mathbf{y}_{t\text{-}1} \in Y_{t\text{-}1} \right\}$$

Return $Y_{n_{\max}}$

$$\Rightarrow \quad Y^{\star} = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

Can we write this as a (sleek) optimization problem?

Insight Number Two:
What does beam search optimize?

What does beam search optimize?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \boxed{\mathbf{score}(\mathbf{x}, \mathbf{y})}$$

What does beam search optimize?

Easy Case: *k* = 1 (greedy search)

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \ \mathbf{score}(\mathbf{x},\mathbf{y})$$

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \left(\boxed{\log p_\theta(\mathbf{y} \mid \mathbf{x})} - \lambda \cdot \mathcal{R}(\mathbf{y})\right)$$

original objective

What does beam search optimize?

Easy Case: *k* = 1 (greedy search)

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \ \mathbf{score}(\mathbf{x},\mathbf{y})$$

"regularizer"

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \left(\boxed{\log p_\theta(\mathbf{y} \mid \mathbf{x})} - \boxed{\lambda \cdot \mathcal{R}(\mathbf{y})} \right)$$

original objective

What does beam search optimize?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \ \mathbf{score}(\mathbf{x}, \mathbf{y})$$

$$\Big\downarrow$$

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \ \Big(\log p_\theta(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \Big)$$

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \Big(\max_{y' \in \mathcal{V}} \log p_{\boldsymbol{\theta}}(y' \mid \mathbf{x}, \mathbf{y}_{<t}) - \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) \Big)^2$$

What does beam search optimize?

Easy Case: *k* = 1 (greedy search)

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \ \mathbf{score}(\mathbf{x}, \mathbf{y})$$

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \left(\log p_\theta(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \Big(\max_{y' \in \mathcal{V}} \underbrace{\log p_{\boldsymbol{\theta}}(y' \mid \mathbf{x}, \mathbf{y}_{<t}) - \log p_{\boldsymbol{\theta}}}_{\text{The distance between the log-prob of our chosen word and the max log-prob word at step t}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) \Big)^2$$

What does beam search optimize?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \ \mathbf{score}(\mathbf{x},\mathbf{y})$$

$$\Downarrow$$

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \left(\log p_\theta(\mathbf{y} \mid \mathbf{x}) - \boxed{\lambda} \cdot \mathcal{R}(\mathbf{y}) \right)$$

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(\max_{y' \in \mathcal{V}} \log p_{\boldsymbol{\theta}}(y' \mid \mathbf{x}, \mathbf{y}_{<t}) - \log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) \right)^2$$

The optimum of our regularized decoding problem as λ →∞ is the same as the solution found by greedy search!

What does beam search optimize?

General Case: $k > 1$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y| = k}}{\operatorname{argmax}} \quad ?$$

$$\Downarrow$$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y| = k}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(Y) \right)$$

Now we're dealing with sets!

What does beam search optimize?

General Case: *k* > 1

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

$$p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) := \prod_{\mathbf{y} \in Y} p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(Y) \right)$$

What does beam search optimize?

General Case: *k* > 1

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y| = k}}{\operatorname{argmax}} \quad ?$$

$$\Downarrow$$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y| = k}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(Y) \right)$$

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Big(\max_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'| = k}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x}) - \log p_{\boldsymbol{\theta}}(Y_t \mid \mathbf{x}) \Big)^2$$

What does beam search optimize?

General Case: ***k*** > 1

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y},\\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y},\\ |Y|=k}}{\operatorname{argmax}} \Big(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \boxed{\lambda \cdot} \mathcal{R}(Y) \Big)$$

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Big(\max_{\substack{Y' \subseteq \mathcal{B}_t,\\ |Y'|=k}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x}) - \log p_{\boldsymbol{\theta}}(Y_t \mid \mathbf{x}) \Big)^2$$

The optimum of our regularized decoding problem as λ →∞ is the same as the solution found by beam search!

A Cognitive Motivation for Beam Search?

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Big(\max_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x}) - \log p_{\boldsymbol{\theta}}(Y_t \mid \mathbf{x}) \Big)^2$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Big(\max_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x}) - \log p_{\boldsymbol{\theta}}(Y_t \mid \mathbf{x}) \Big)^2$$

surprisal:

$$u_0(\mathrm{BOS}) = 0$$
$$u_t(y) = -\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}, \mathbf{y}_{<t}), \ \textbf{for } t \geq 1$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Big(\max_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x}) - \log p_{\boldsymbol{\theta}}(Y_t \mid \mathbf{x}) \Big)^2$$

surprisal:

$$u_0(\text{BOS}) = 0$$
$$u_t(y) = -\log p_{\boldsymbol{\theta}}(y \,|\, \mathbf{x}, \mathbf{y}_{<t}), \;\; \textbf{for } t \geq 1$$

$$\Downarrow$$

$$u_0(\langle\text{BOS}\rangle) = 0$$
$$u_t(Y_t) = -\log p_{\boldsymbol{\theta}}(Y_t \,|\, \mathbf{x}), \;\; \textbf{for } t \geq 1$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)^2$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Bigg(\underbrace{u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y')}_{\text{(squared) distance from lowest surprisal choice}} \Bigg)^2$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \underbrace{\left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)^2}_{\text{(squared) distance from lowest surprisal choice}}$$

Beam search enforces low surprisal choices at each time step

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \underbrace{\left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)^2}_{\text{(squared) distance from lowest surprisal choice}}$$

Beam search enforces **low surprisal choices** at each time step

Beam search as a cognitively motivated search heuristic

Great! Why does that matter?

The **uniform information density hypothesis** (Levy, 2005; Levy and Jaeger, 2007; Jaeger 2010):

“Within the bounds defined by grammar, speakers prefer utterances that distribute information uniformly across the signal (information density). Where speakers have a choice between several variants to encode their message, they prefer the variant with more uniform information density (ceteris paribus)”

Beam search as a cognitively motivated search heuristic

Great! Why does that matter?

The **uniform information density hypothesis** (Levy, 2005; Levy and Jaeger, 2007; Jaeger 2010):

"Within the bounds defined by grammar, speakers prefer utterances that distribute information uniformly across the signal (information density). Where speakers have a choice between several variants to encode their message, they prefer the variant with more uniform information density (ceteris paribus)"

TL;DR: Humans prefer sentences that evenly distribute information across the sentence. We don't like moments of high surprisal; they're hard to process!

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is [$_{\text{NP}}$ the family$_{\text{i}}$ [$_{\text{RC}}$ (that) you cook for $\_{\text{i}}$]]?

This sentence is also grammatically correct (and relays the same message) without the word "that."

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is [$_{NP}$ the family$_{i}$ [$_{RC}$ (~~that~~) you cook for $\_{i}$]]?

This sentence is also grammatically correct (and relays the same message) without the word “that.”

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is $[_{NP}$ the family$_i$ $[_{RC}$ (~~that~~) you cook for $\_\_{}_i]]$?

This sentence is also grammatically correct (and relays the same message) without the word "that."

But it just sounds better with it….

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is [$_{NP}$ the family$_i$ [$_{RC}$ ~~(that)~~ you cook for $\_\_{}_i$]]?

Information-theoretic explanation:

- Without “that,” the word “you” conveys two pieces of information: the onset of a relative clause and part of its internal contents.

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is $[_{NP}$ the family$_i$ $[_{RC}$ (~~that~~) you cook for $\_{i}]]$?

Information-theoretic explanation:

- Without “that,” the word “you” conveys two pieces of information: the onset of a relative clause and part of its internal contents.
- Including the relativizer spreads information across two words, thereby **avoiding an instance of high surprisal** and **distributing information across the sentence** more uniformly.

Beam search as a cognitively motivated search heuristic

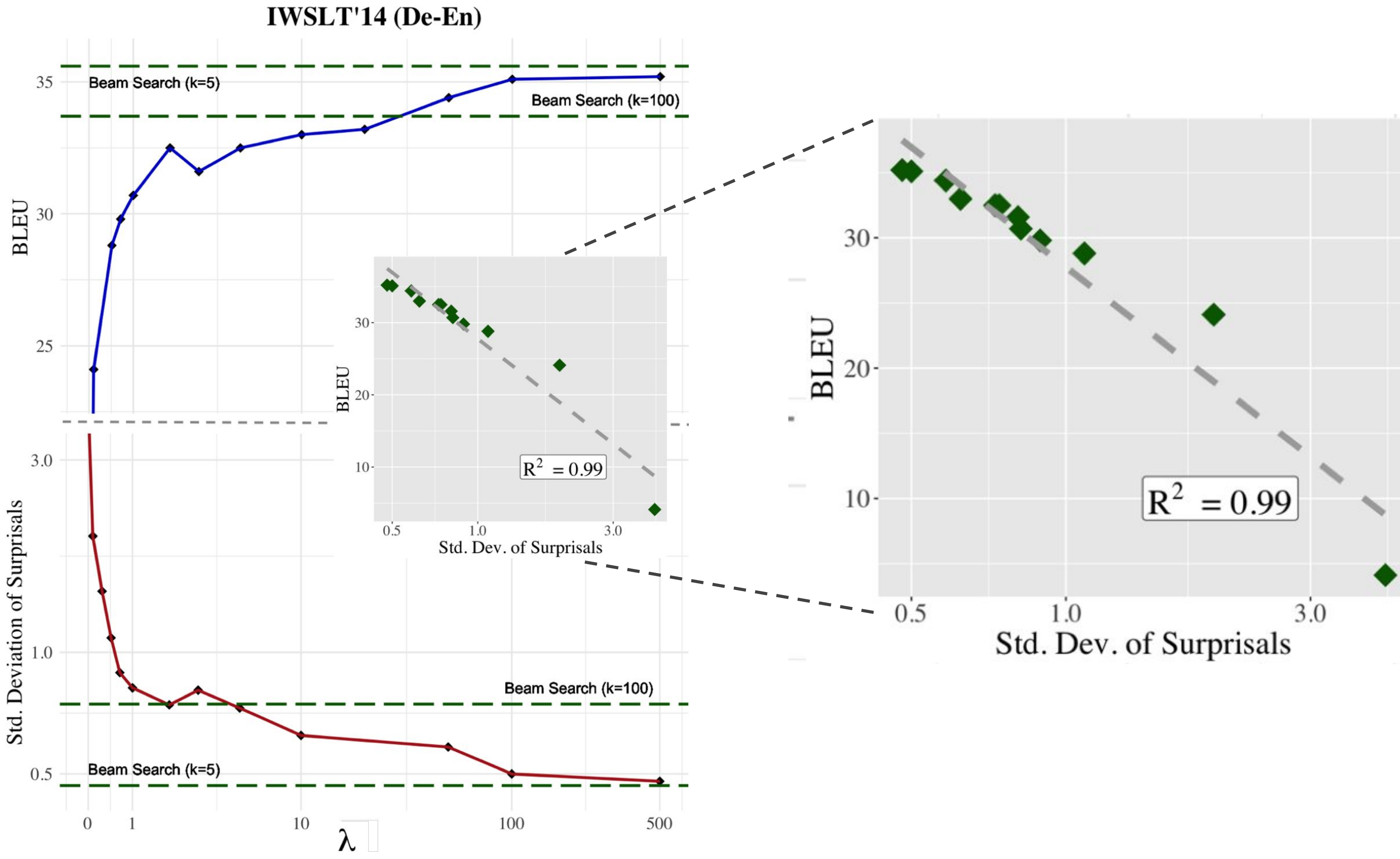
Nice hypothesis

but where's the proof?

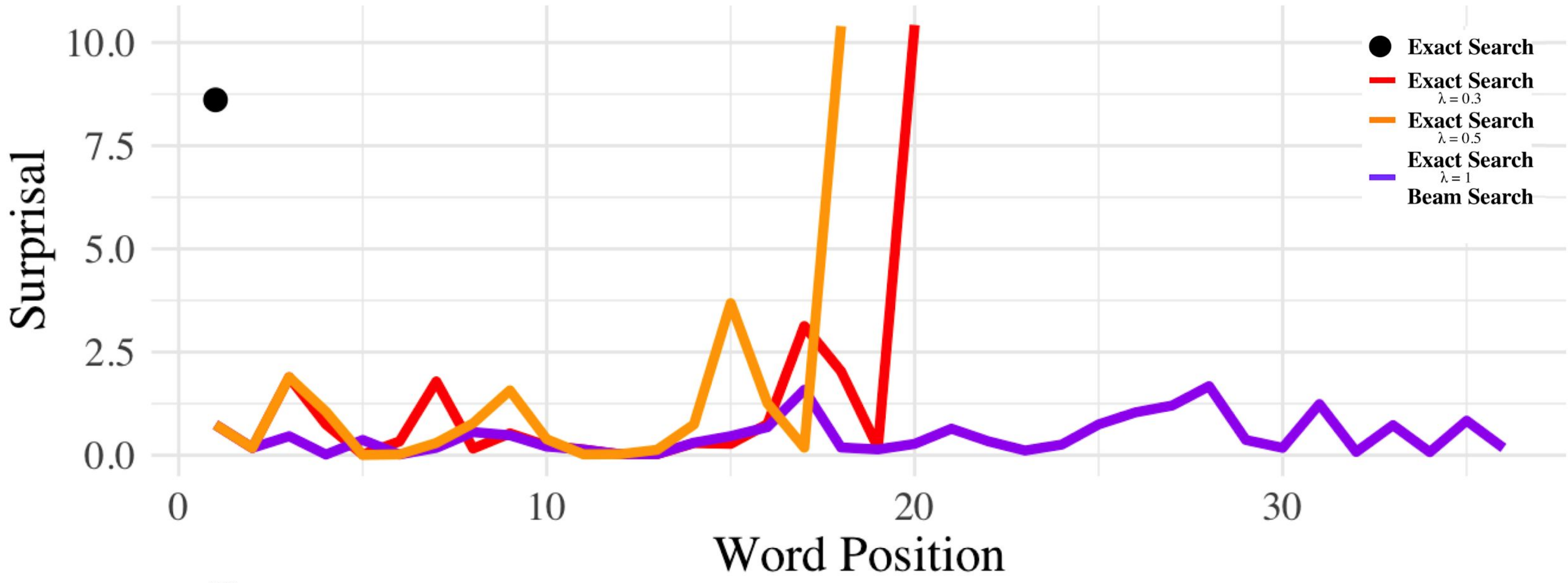
Beam search as a cognitively motivated search heuristic

Nice hypothesis

but where's the proof?



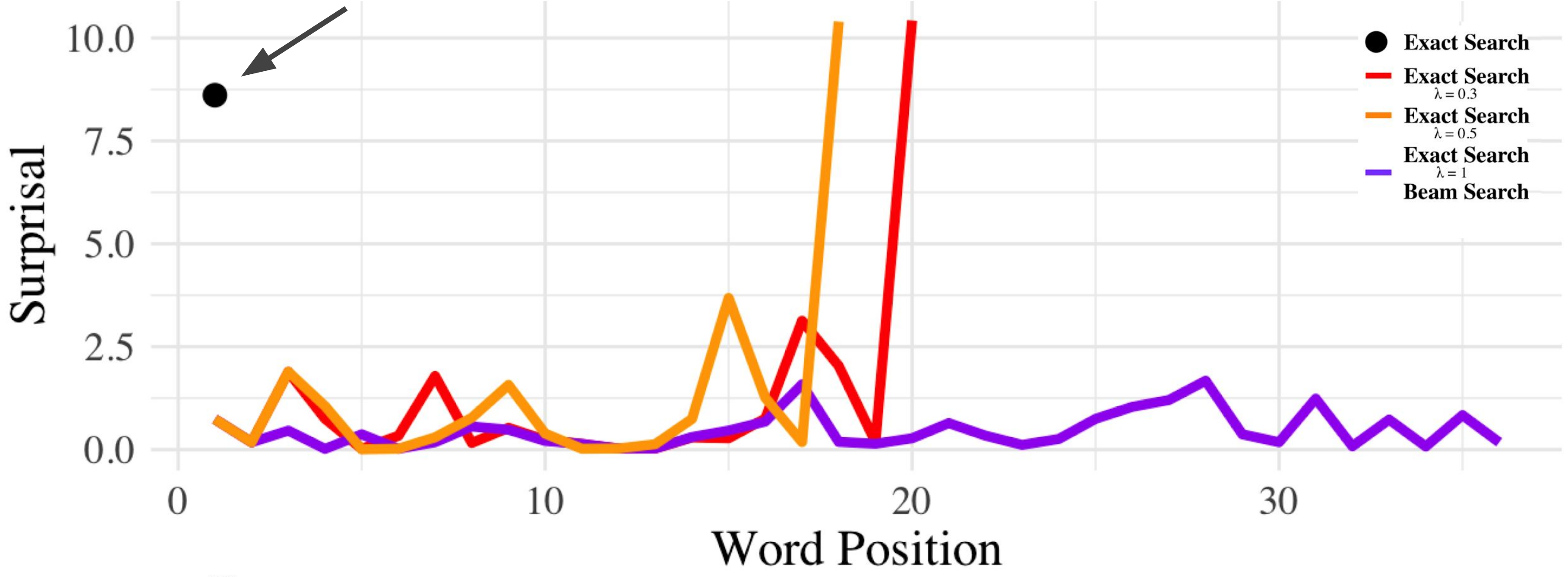
Beam search as a cognitively motivated search heuristic



Recall: our standard decoding objective explicitly minimizes the sum of surprisals, i.e., maximizes log-probability.

Beam search as a cognitively motivated search heuristic

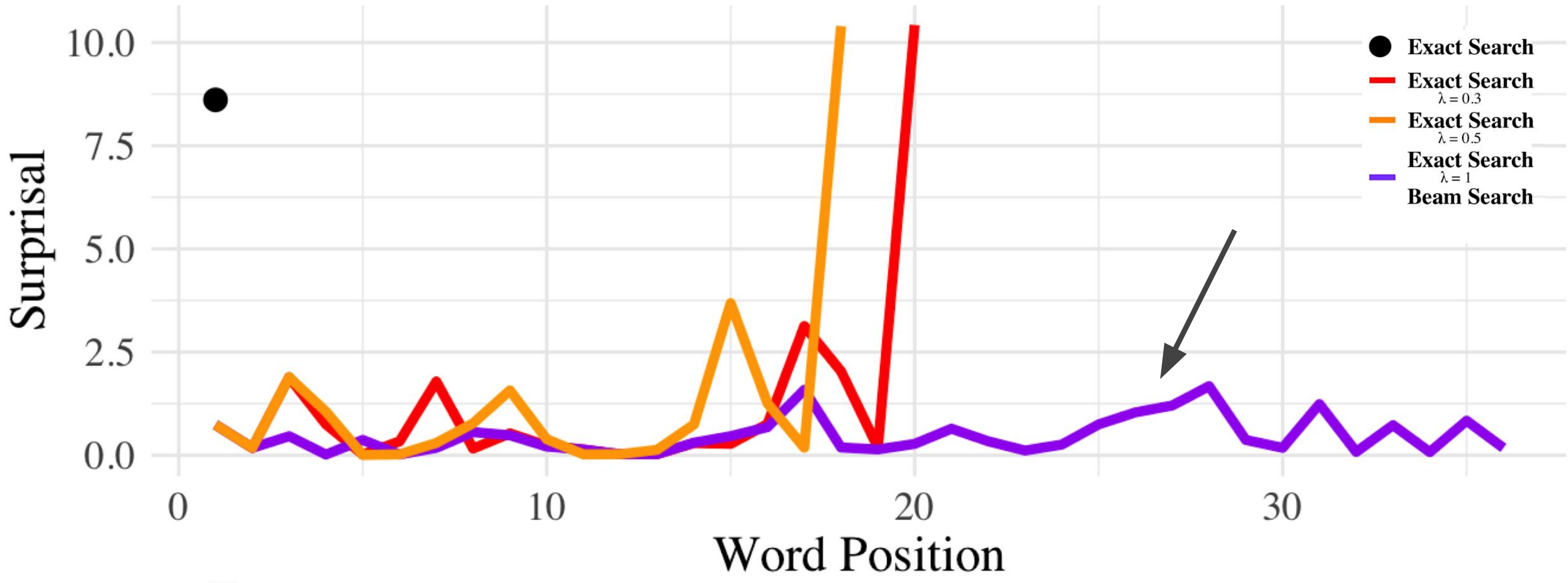
Exact search finds the highest probability sequence, regardless of local decisions



Recall: our standard decoding objective explicitly minimizes the sum of surprisals, i.e., maximizes log-probability.

Beam search as a cognitively motivated search heuristic

Beam search enforces UID!



Recall: our standard decoding objective explicitly minimizes the sum of surprisals, i.e., maximizes log-probability. Therefore, the only way the distribution of a solution can become distinctly non-uniform is when there are several high-surprisal decisions

Uniformation Information Density Regularization

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- Recall our regularized decoding objective:

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \left(\log p_\theta(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- Recall our regularized decoding objective:

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}{\operatorname{argmax}} \left(\log p_\theta(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

- In practice, it’s not practical do set optimization...

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- We started with a "greedy" regularizer that mimics beam search (k=1)

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - \min_{y' \in \mathcal{V}} u_t(y') \right)^2$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages high variance in surprisals**?

$$\mathcal{R}_{\text{var}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} \Big(u_t(y_t) - \mu \Big)^2$$

define:

$$\mu = {}^{1}/_{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages high variance in surprisals *locally***?

$$\mathcal{R}_{\mathrm{local}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} \Big(u_t(y_t) - u_{t-1}(y_{t-1}) \Big)^2$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages instances of high surprisal**?

$$\mathcal{R}_{\max}(\mathbf{y}) = \max_{t=1}^{|\mathbf{y}|} u_t(y_t)$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages consistently high surprisal**?

$$\mathcal{R}_{\text{square}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)^2$$

Uniform information density (UID) regularization

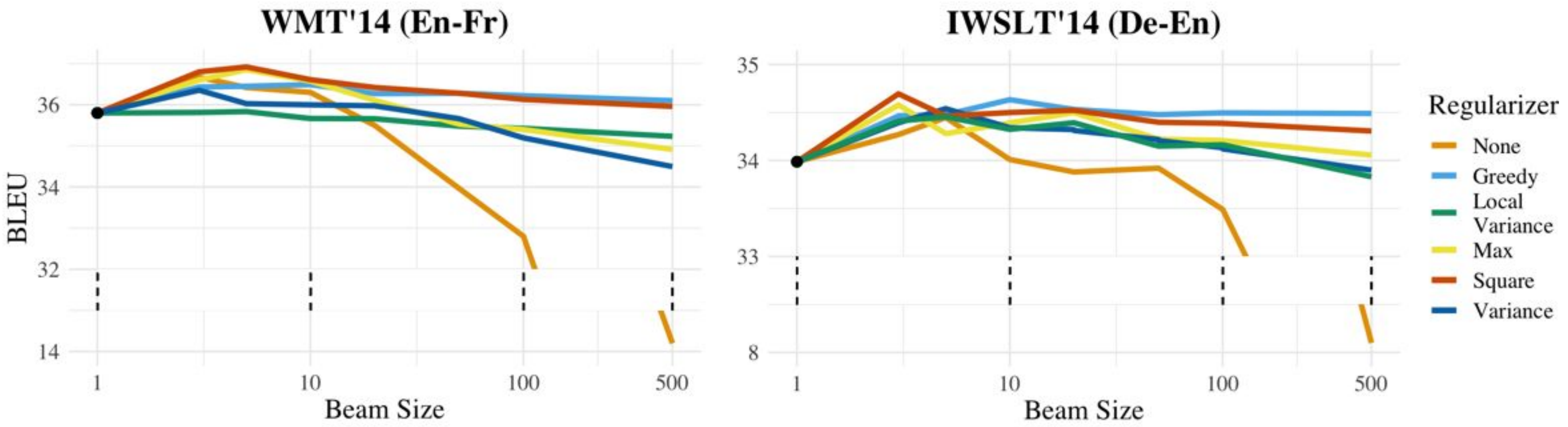
Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that discourages consistently high surprisal?

Let's see 'em in action!

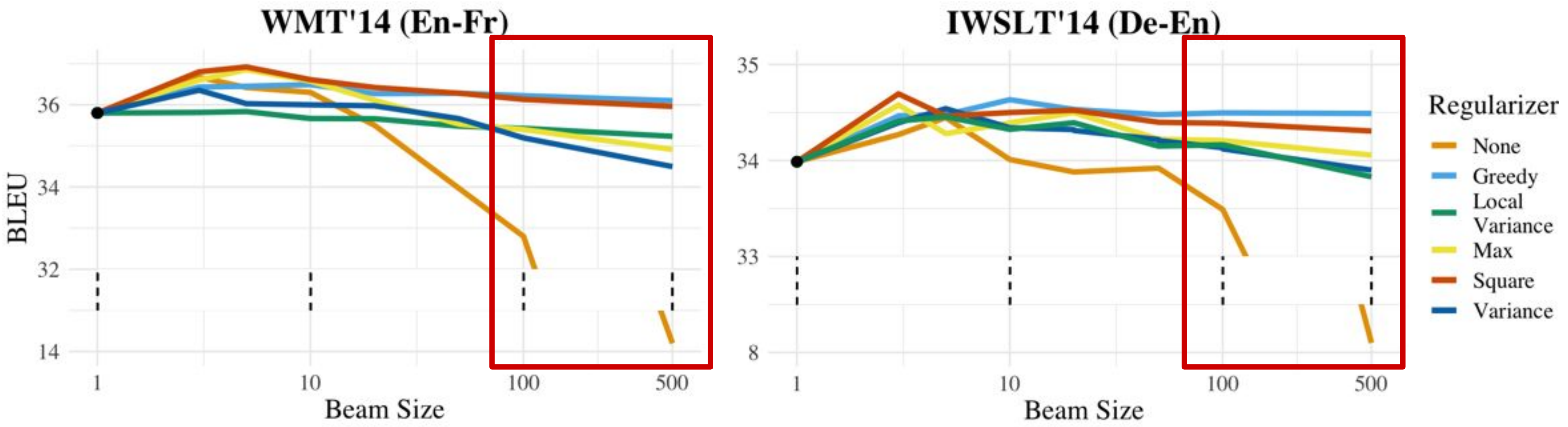
$$\mathcal{R}_{\text{square}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)^2$$

Uniform information density (UID) regularization



Experiments on NMT systems; decoding of models with various objectives for different beam sizes

Uniform information density (UID) regularization



Experiments on NMT systems; decoding of models
with various objectives for different beam sizes

Uniform information density (UID) regularization

	$k=5$	$k=10$	$k=100$	$k=500$
No Regularization	36.42	36.30	32.83	14.66
Squared Regularizer	**36.92**	36.42	36.13	35.96
Greedy Regularizer	36.45	36.49	36.22	36.15
Combined Regularizers	36.69	**36.65**	**36.48**	**36.35**
Length Normalization	36.02	35.94	35.80	35.11

Table 1: BLEU scores on first 1000 samples of Newstest2014 for predictions generated with various decoding strategies. Best scores per beam size are bolded.

Uniform information density (UID) regularization

	$k=5$	$k=10$	$k=100$	$k=500$
No Regularization	36.42	36.30	32.83	14.66
Squared Regularizer	**36.92**	36.42	36.13	35.96
Greedy Regularizer	36.45	36.49	36.22	36.15
Combined Regularizers	36.69	**36.65**	**36.48**	**36.35**
Length Normalization	36.02	35.94	35.80	35.11

Table 1: BLEU scores on first 1000 samples of Newstest2014 for predictions generated with various decoding strategies. Best scores per beam size are bolded.

Luckily, not a huge difference! One regularizer seems good enough

A Hot Take on Sampling from Probabilistic Text Generators

Ryan Cotterell @ ACL 2022

22ND – 27TH MAY | 60TH MEETING | DUBLIN

Rycolab

ETH zürich

UNIVERSITY OF CAMBRIDGE

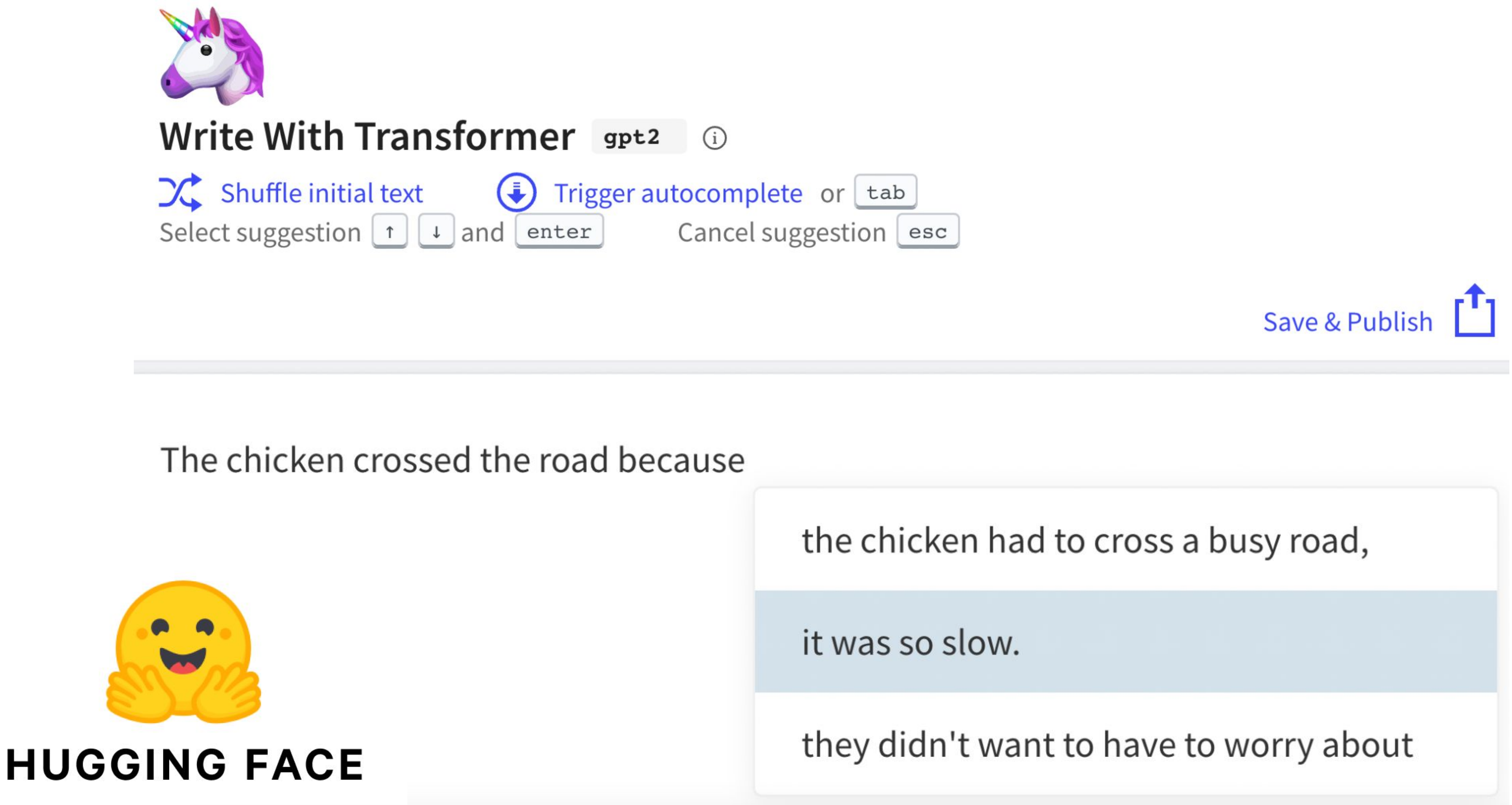
Thanks to my amazing collaborators!

Clara Meister
ETH Zürich
(2nd-year PhD student)

Gian Wiher
ETH Zürich
(MSc student)

Tiago Pimentel
Cambridge
(3rd-year PhD student)

Probabilistic Text Generation



Probabilistic Text Generation

What is a good algorithm to generate this text?

Write With Transformer `gpt2`

Shuffle initial text

Trigger autocomplete or `tab`

Select suggestion `↑` `↓` and `enter`

Cancel suggestion `esc`

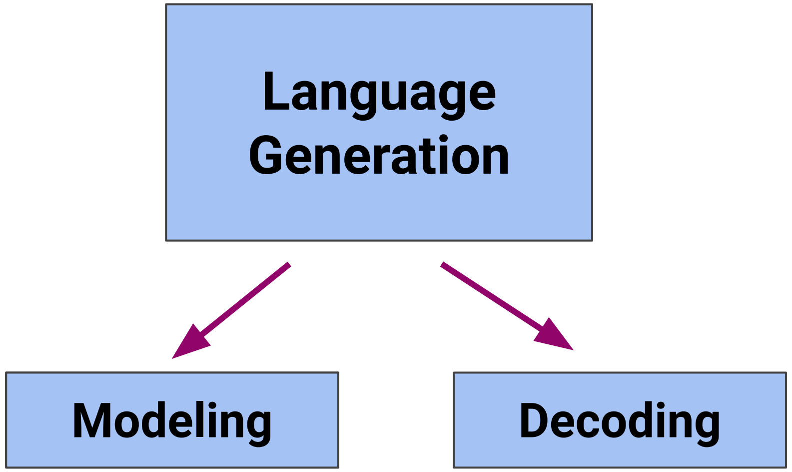
Save & Publish

The chicken crossed the road because

- the chicken had to cross a busy road,
- it was so slow.
- they didn't want to have to worry about

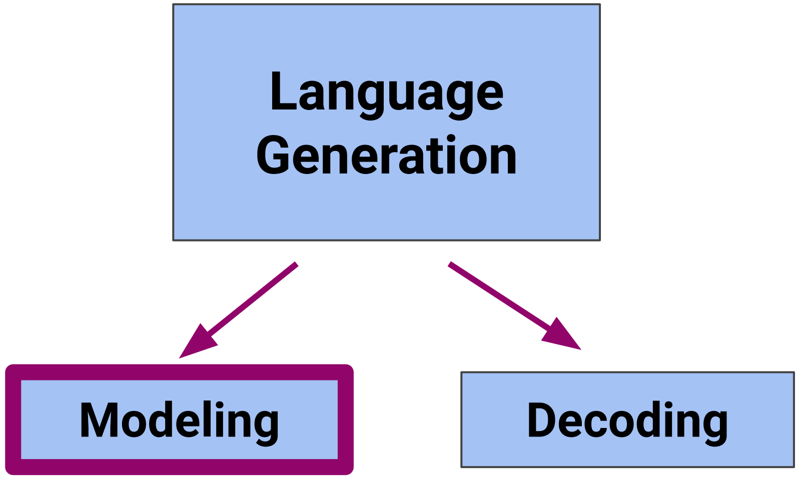
HUGGING FACE

Probabilistic Text Generation



We can view probabilistic natural language generation as a two part problem

Probabilistic Text Generation: Modeling



Modeling: Which probability distribution should we generate text from?

- Left-to-right "causal" language model?
- Cloze language model?
- Globally versus locally normalized models?

Probabilistic Text Generation: Modeling

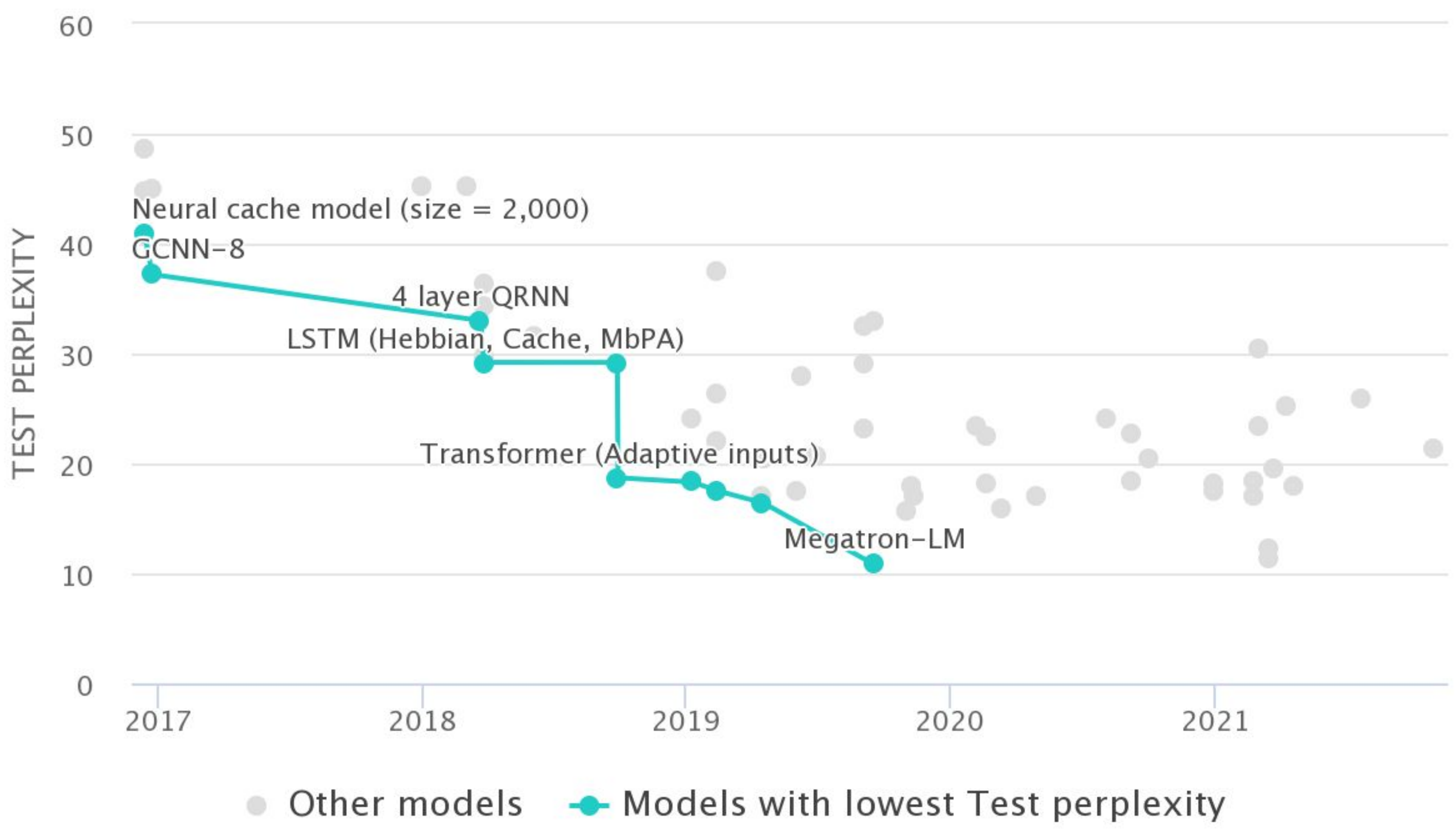
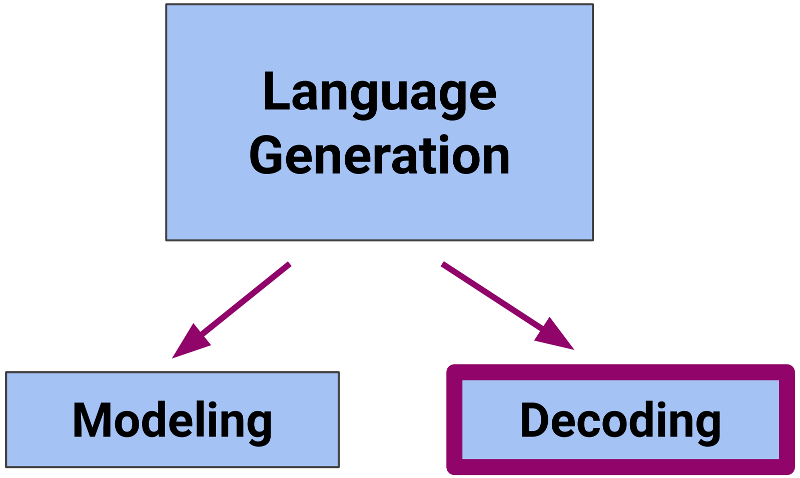


Figure: paperswithcode.com/sota/

Probabilistic Text Generation: Decoding



Decoding: Which ***decoding strategy*** should we use to generate the text?

- Ancestral sampling?
- Beam search?
- Dynamic programming (might be slow!)?

This Talk Focuses on Decoding!

(It’s surprising how much the decoding strategy matters!)

The Mechanics of Decoding

Let's focus on a traditional left-to-right language model that decomposes as follows

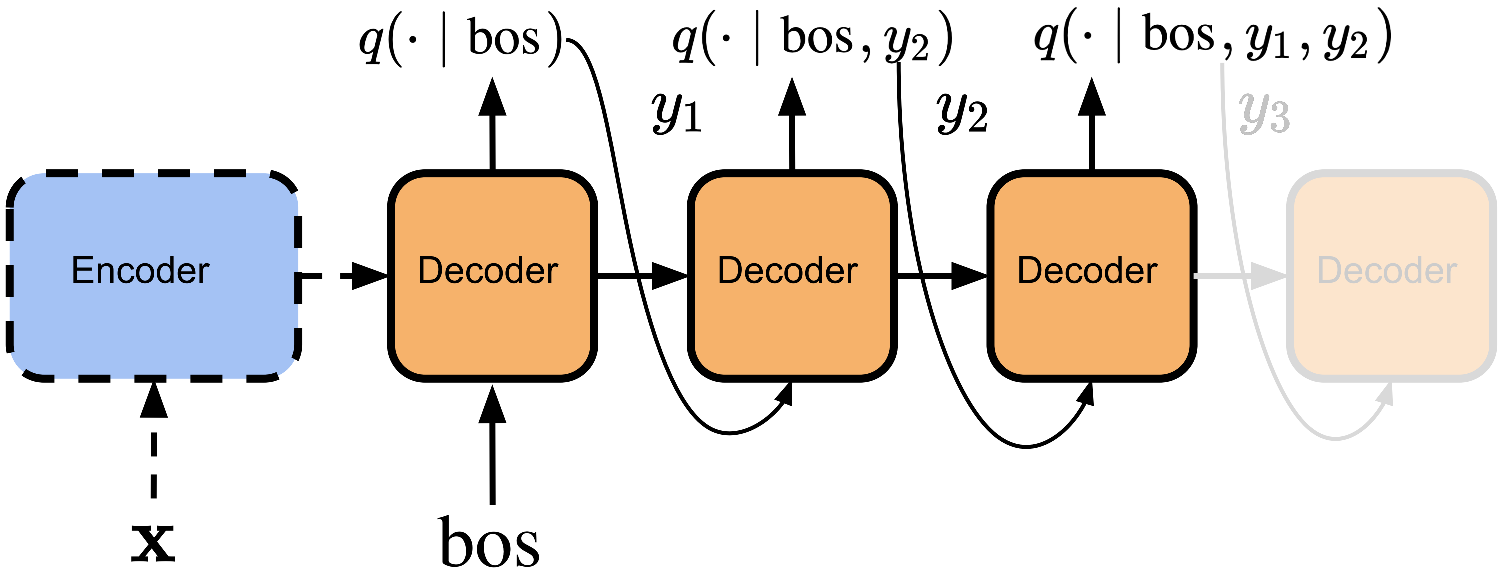
$$q(\mathbf{y}) = \prod_{t=1}^{T} q(y_t \mid y_1, \ldots, y_{t-1})$$

The Mechanics of Decoding

Let's focus on a traditional left-to-right language model that decomposes as follows

$$q(\mathbf{y}) = \prod_{t=1}^{T} q(y_t \mid y_1, \ldots, y_{t-1})$$

We then generate y_1 according to $q(\cdot \mid \mathrm{bos})$, y_2 according to $q(\cdot \mid \mathrm{bos}, y_1)$

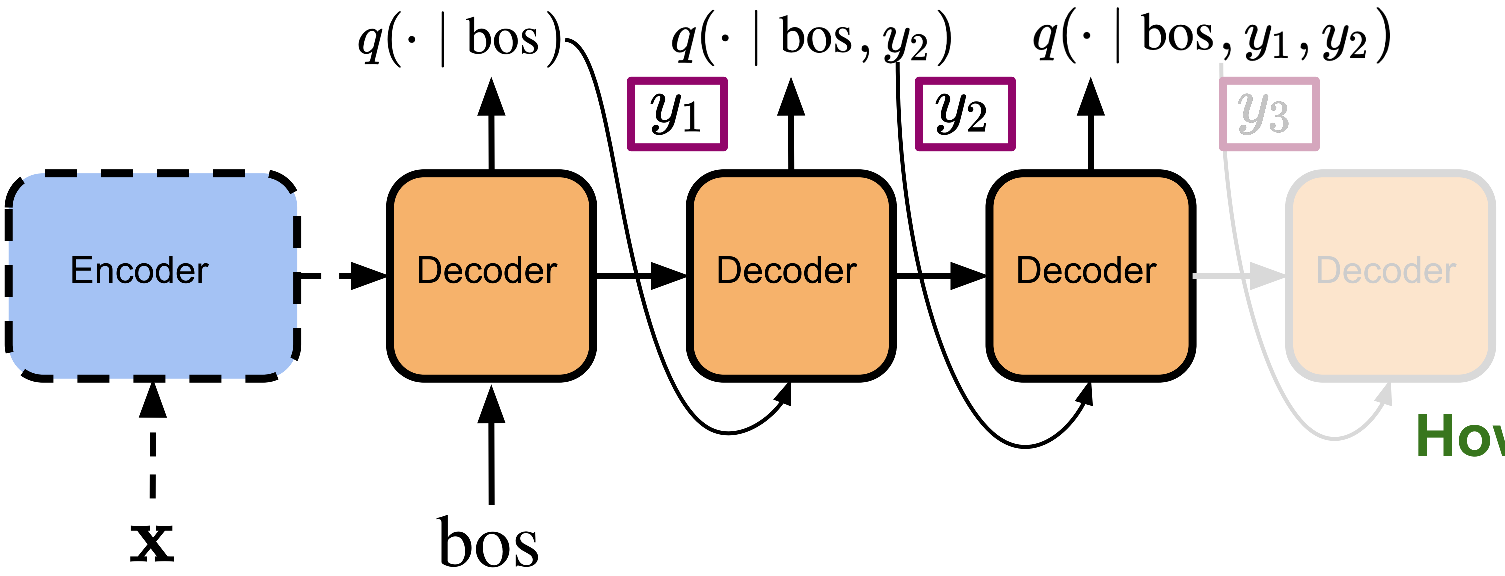


The Mechanics of Decoding

Let's focus on a traditional left-to-right language model that decomposes as follows

$$q(\mathbf{y}) = \prod_{t=1}^{T} q(y_t \mid y_1, \ldots, y_{t-1})$$

We then generate y_1 according to $q(\cdot \mid \text{bos})$, y_2 according to $q(\cdot \mid \text{bos}, y_1)$



How do we choose *y* at each step?

Decoding Strategy Example: Greedy Search

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$

0.001	jumped
….	….
….	….
0.15	$\boxed{\text{dog}}$
0.1	head

Greedy search says choose select the argmax at each time step:

$$y_t = \underset{y \in \overline{\mathcal{V}}}{\operatorname{argmax}} \; q(y \mid \mathbf{y}_{<t})$$

Decoding Strategy Example: Greedy Search

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$

0.001	jumped
....	....
....	....
0.15	dog
0.1	head

What can go wrong?

- *Often leads to dull or generic text*
- *Prone to repetitive loops!*

Greedy search says choose select the argmax at each time step:

$$y_t = \operatorname*{argmax}_{y \in \overline{\mathcal{V}}} \; q(y \mid \mathbf{y}_{<t})$$

Decoding Strategy Example: Ancestral Sampling

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$

0.001	jumped
….	….
….	….
0.15	$\boxed{\text{dog}}$
0.1	head

Ancestral sampling says sample according to *q* at each time step:

$$y_t \sim q(\cdot \mid \mathbf{y}_{<t})$$

Decoding Strategy Example: Ancestral Sampling

$$q(\cdot \mid \mathrm{bos}, \mathrm{My}) \rightarrow$$

0.001	jumped
....	....
....	....
0.15	dog
0.1	head

What can go wrong?

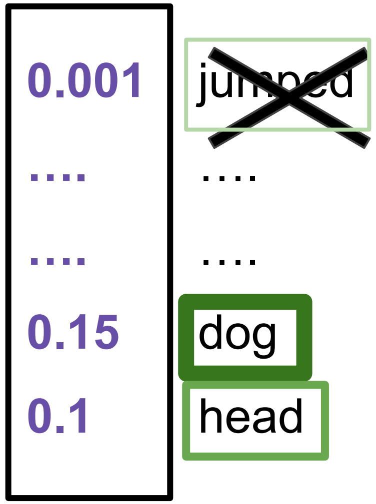
- *We often sample from the tail of the distribution, which leads to text that is not relevant or nonsensical*

Ancestral sampling says sample according to *q* at each time step:

$$y_t \sim q(\cdot \mid \mathbf{y}_{<t})$$

Decoding Strategy Example: Top-*k* Sampling

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$



Top-*k* sampling says sample one of the top *k* of *q* at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \textbf{if } y_t \in \text{top}_k(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \textbf{otherwise} \end{cases}$$

Decoding Strategy Example: Top-*k* Sampling

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$

0.001	~~jumped~~
….	….
….	….
0.15	$\boxed{\text{dog}}$
0.1	$\boxed{\text{head}}$

What can go wrong?

- *The generated text is higher quality, but we still occasionally observe degenerate behavior, e.g., repetitive loops*

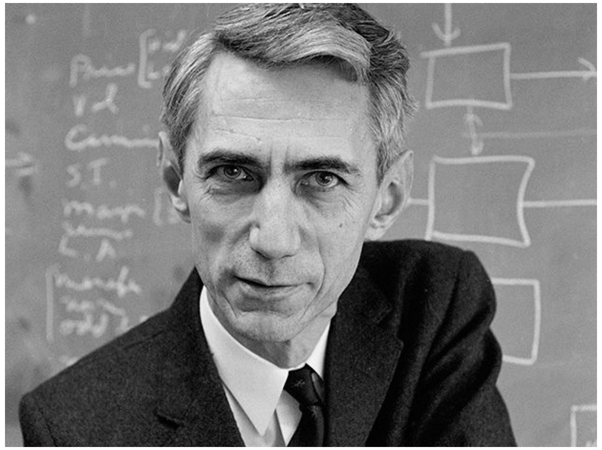
Top-*k* sampling says sample one of the top *k* of *q* at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \textbf{if } y_t \in \text{top}_k(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \textbf{otherwise} \end{cases}$$

An Information-Theoretic View of Language

An Information-Theoretic View of Language

Natural language is the primary means for human communication



Information theory is the mathematical study of communication

1. Can information theory help us determine when automatically generated text is human-like?

2. Can we use information-theoretic concepts to generate more human-like text?

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests ***two principles*** that guide what makes a good sentence:

- **Principle 1**: Information should be transmitted efficiently
- **Principle 2:** Sentences should be chosen to avoid miscommunication

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests ***two principles*** that guide what makes a good sentence:

- **Principle 1**: Keep the sentence short and information dense
- **Principle 2:** Avoid moments of high information, which are hard to process

Rephrased more colloquially

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests ***two principles*** that guide what makes a good sentence:

- **Principle 1**: Keep the sentence short and information dense
- **Principle 2:** Avoid moments of high information, which are hard to process

These two principles trade off!

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests ***two principles*** that guide what makes a good sentence:

- **Principle 1**: Keep the sentence short and information dense
- **Principle 2:** Avoid moments of high information, which are hard to process

These two principles trade off!

Solution: A natural solution to the above trade-off is for an algorithm to choose sentences that are around the ***average information content***. Intuitively, such sentences should be informative enough, but also avoid stretches of high information.

What's Special About Average Information?

- The average information content of a distribution goes by the **entropy**

- In the case of probabilistic language generators of the form

$$q(\mathbf{y}) = \prod_{t=1}^{T} q(y_t \mid y_1, \ldots, y_{t-1})$$

  it is most natural to talk about time-step dependent entropy

- In symbols, entropy (average information) at time step *t* is denoted as

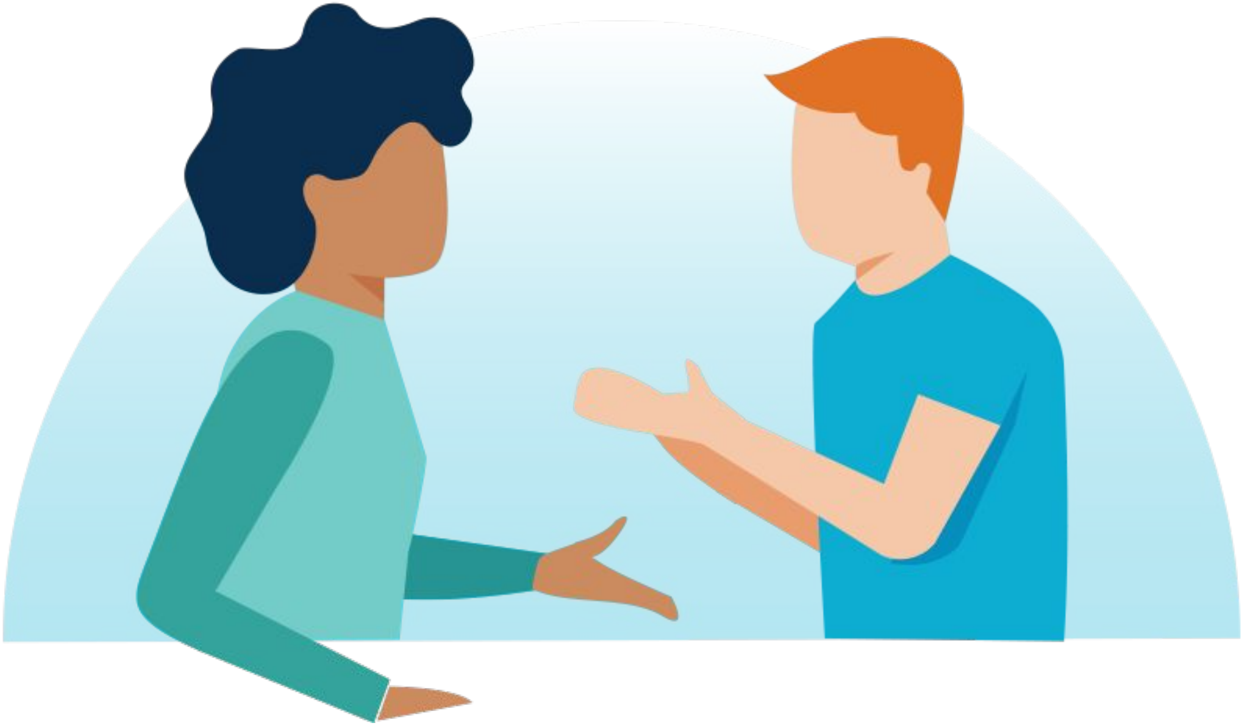
$$\mathrm{H}\left(q(\cdot \mid \mathbf{y}_{<t})\right) = \sum_{y \in \overline{\mathcal{V}}} q(y \mid \mathbf{y}_{<t}) \left[-\log q(y \mid \mathbf{y}_{<t})\right]$$

conditional entropy **information content**

The Expected Information Hypothesis (Meister et al. 2022a)

PS5-2: Generation, Tuesday 15:15-16:15 (Forum)

The Expected Information Hypothesis in a Picture



The Expected Information Hypothesis in a Picture

The Expected Information Hypothesis in a Picture

The Expected Information Hypothesis

Expected Information Hypothesis. Every word in a generated sentence should have an information content close to the conditional entropy of the distribution over words given prior context. That is, there exists an ε such that

$$\left|\mathrm{H}(q(\cdot \mid \mathbf{y}_{<t})) + \log q(y_t)\right| < \varepsilon$$

for every token y_t in sentence **y** with high probability.

A Useful Definition: Local Typicality

- So what *set* should speakers' utterances fall in? Let's define one!
- We define the ***locally typical set*** of the distribution *q* as follows

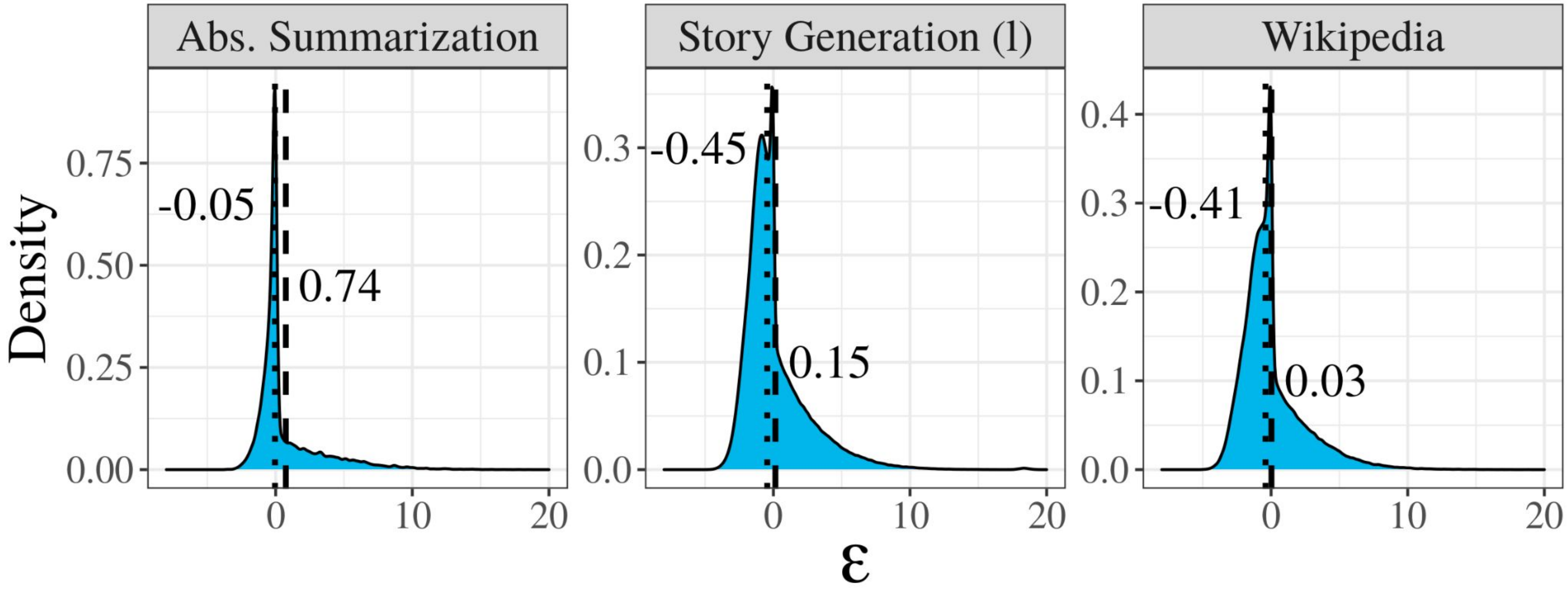
$$\mathcal{T}_\varepsilon(q) = \left\{ y : \Big| \underbrace{\mathrm{H}(q)}_{\text{entropy}} - \underbrace{-\log q(y)}_{\text{information content}} \Big| < \varepsilon \right\}$$

  This set is defined as those *y* whose information content has distance of less than ε from the entropy of the distribution *q*

- We have a free parameter ε that we get to choose!

- **Caveat**: This is not the standard definition of typicality that you will find in information theory, e.g. Cover and Thomas (2006). It is related, though.

Empirical Evidence for the Hypothesis



The per-token distribution of the deviation (ε) of information content from conditional entropy on human text. The true probabilities and entropies are approximated using probabilistic models trained on the data for each task. Labels and lines indicate the mean and median deviations

Still curious about expected information hypothesis?

- See Meister et al. (2022) at ***this conference***!
- We have many, many experiments that support the hypothesis

On the probability–quality paradox in language generation

Clara Meister Gian Wiher Tiago Pimentel Ryan Cotterell
ETH Zürich University of Cambridge
clara.meister@inf.ethz.ch gian.wiher@inf.ethz.ch
tp472@cam.ac.uk ryan.cotterell@inf.ethz.ch

Abstract

When generating natural language from neural probabilistic models, high probability does not always coincide with high quality: It appears to have an inflection point,[2] i.e., quality and probability are positively correlated only until a certain threshold, after which the correlation becomes negative. While the existence of such a trend has received informal explanations (see, e.g.,

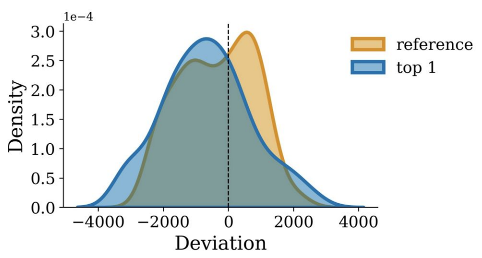


Figure 2: The distribution of the difference in total information content for (1) test-set references and (2) top-ranked model-generated strings from the (conditional) entropy of the model from which they were generated.

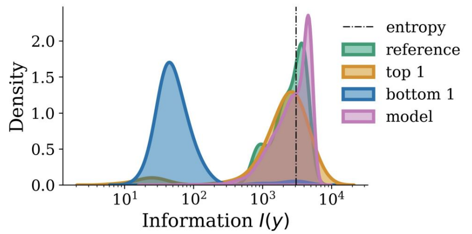
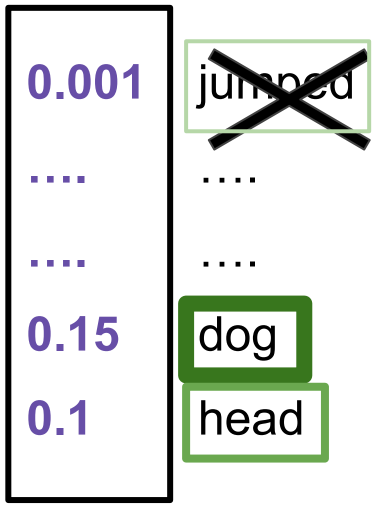


Figure 1: The distribution over information $\mathrm{I}(\mathbf{y})$ values of: MODEL, the model, as estimated using samples from q; REFERENCE, the reference strings; TOP 1 and BOTTOM 1, model-generated strings ranked first and last (respectively) among all decoding strategies by human annotators. The latter 3 are all w.r.t. a held-out test set. Same graph is reproduced for individual decoding strategies in App. B.

Typical sampling: From hypothesis to algorithm (Meister et al. 2022b)

A New Decoding Strategy: Typical Sampling

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$

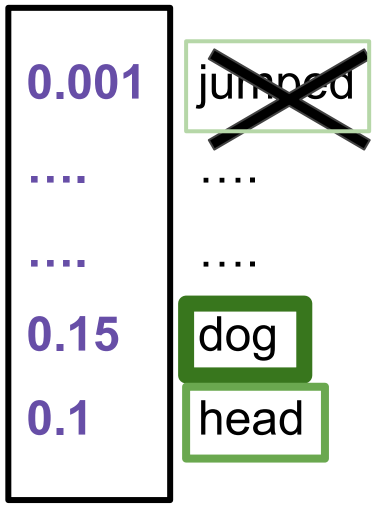


Typical sampling says sample from the locally typical set at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \textbf{if } y_t \in \mathcal{T}_\varepsilon(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \textbf{otherwise} \end{cases}$$

A New Decoding Strategy: Typical Sampling

$$q(\cdot \mid \text{bos}, \text{My}) \rightarrow$$



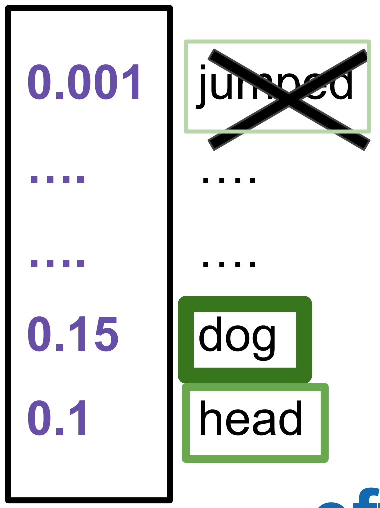
Locally Typical Set

Typical sampling says sample from the locally typical set at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \textbf{if } y_t \in \mathcal{T}_\varepsilon(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \textbf{otherwise} \end{cases}$$

A New Decoding Strategy: Typical Sampling

$$q(\cdot \mid \mathrm{bos}, \mathrm{My}) \rightarrow$$



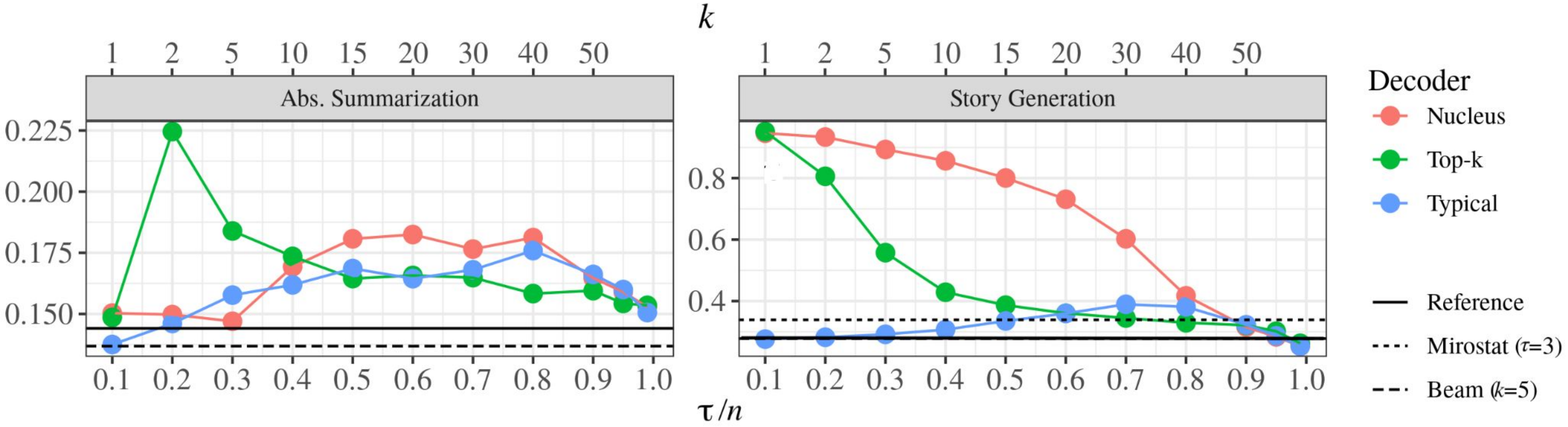
Can be calculated efficiently in O(|V|) time!

Typical sampling says sample from the locally typical set at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \textbf{if } y_t \in \mathcal{T}_\varepsilon(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \textbf{otherwise} \end{cases}$$

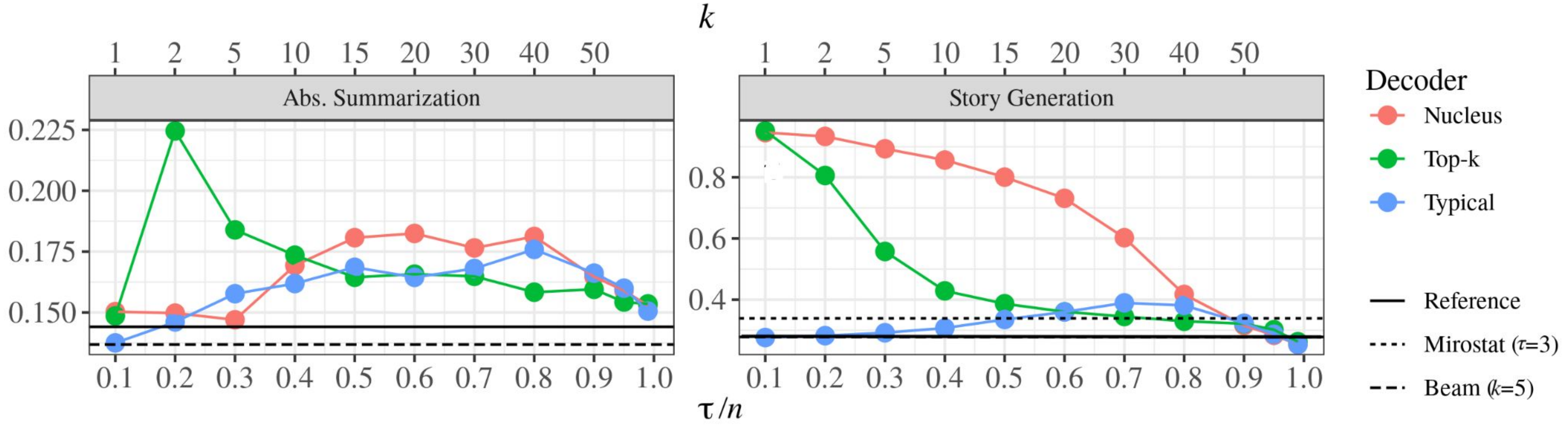
How well does it compare? nUmBeRs wEnt uP

	Abstractive Summarization					Story Generation				
	PPL	REP	Zipf	MAUVE	Human	PPL	REP	Zipf	MAUVE	Human
Reference	11.51	0.13	0.76	-	4.07	20.06	0.28	1.09	-	4.23
Nucleus	3.28	0.16	0.93	0.93	3.97	9.65	0.32	**1.25**	0.95	4.00
Top-k	3.12	0.16	0.93	0.96	3.97	7.77	0.34	1.42	**0.97**	4.08
Typical	**4.12**	**0.15**	**0.92**	**0.98**	**4.03**	**17.32**	**0.28**	1.26	0.92	**4.10**



How well does it compare? nUmBeRs wEnt uP

	Abstractive Summarization					Story Generation				
	PPL	REP	Zipf	MAUVE	Human	PPL	REP	Zipf	MAUVE	Human
Reference	11.51	0.13	0.76	-	4.07	20.06	0.28	1.09	-	4.23
Nucleus	3.28	0.16	0.93	0.93	3.97	9.65	0.32	**1.25**	0.95	4.00
Top-k	3.12	0.16	0.93	0.96	3.97	7.77	0.34	1.42	**0.97**	4.08
Typical	**4.12**	**0.15**	**0.92**	**0.98**	**4.03**	**17.32**	**0.28**	1.26	0.92	**4.10**



How well does it compare? Qualitative results

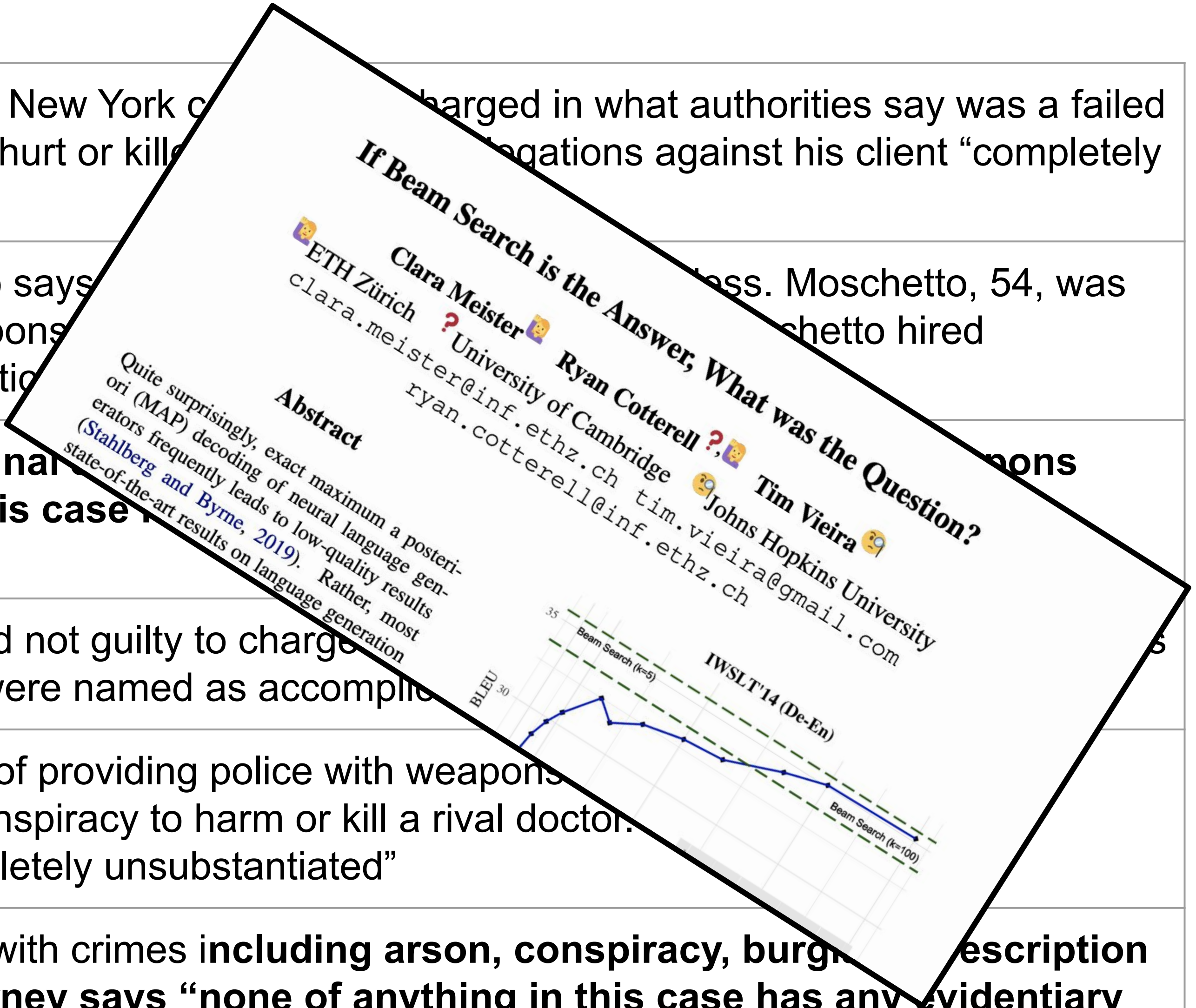
Prompt	(CNN) The attorney for a suburban New York cardiologist charged in what authorities say was a failed scheme to have another physician hurt or killed is calling the allegations against his client "completely unsubstantiated…
Reference	A lawyer for Dr. Anthony Moschetto says the charges against him are baseless. Moschetto, 54, was arrested for selling drugs and weapons, prosecutors say. Authorities allege Moschetto hired accomplices to burn down the practice of former associate.
Beam Search	Dr. Anthony Moschetto **faces criminal solicitation, conspiracy, burglary, arson and weapons charges. "None of anything in this case has any evidentiary value," his attorney says.**
Nucleus Sampling	Dr. Anthony Moschetto, 54, pleaded not guilty to charges Wednesday. Two men -- identified as James Chmela and James Kalamaras -- were named as accomplices.
Top-*k* Sampling	Dr. Anthony Moschetto is accused of providing police with weapons and prescription drugs. Authorities say he was part of a conspiracy to harm or kill a rival doctor. His attorney calls the allegations against his client "completely unsubstantiated"
Typical Sampling	Dr. Anthony Moschetto is charged with crimes i**ncluding arson, conspiracy, burglary, prescription sale, weapons charges. His attorney says "none of anything in this case has any evidentiary value"**

How well does it compare? Qualitative results

Prompt	(CNN) The attorney for a suburban New York cardiologist charged in what authorities say was a failed scheme to have another physician hurt or killed is calling the allegations against his client “completely unsubstantiated…
Reference	A lawyer for Dr. Anthony Moschetto says the charges against him are baseless. Moschetto, 54, was arrested for selling drugs and weapons, prosecutors say. Authorities allege Moschetto hired accomplices to burn down the practice of former associate.
Beam Search	Dr. Anthony Moschetto **faces criminal solicitation, conspiracy, burglary, arson and weapons charges. “None of anything in this case has any evidentiary value,” his attorney says.**
Nucleus Sampling	Dr. Anthony Moschetto, 54, pleaded not guilty to charges Wednesday. Two men -- identified as James Chmela and James Kalamaras -- were named as accomplices.
Top-*k* Sampling	Dr. Anthony Moschetto is accused of providing police with weapons and prescription drugs. Authorities say he was part of a conspiracy to harm or kill a rival doctor. His attorney calls the allegations against his client “completely unsubstantiated”
Typical Sampling	Dr. Anthony Moschetto is charged with crimes i**ncluding arson, conspiracy, burglary, prescription sale, weapons charges. His attorney says “none of anything in this case has any evidentiary value”**

How well does it compare? Qualitative results

Prompt	(CNN) The attorney for a suburban New York c … harged in what authorities say was a failed scheme to have another physician hurt or kill … egations against his client "completely unsubstantiated…
Reference	A lawyer for Dr. Anthony Moschetto says … ess. Moschetto, 54, was arrested for selling drugs and weapons … chetto hired accomplices to burn down the practic…
Beam Search	Dr. Anthony Moschetto **faces criminal** … **pons charges. "None of anything in this case** …
Nucleus Sampling	Dr. Anthony Moschetto, 54, pleaded not guilty to charg… … s Chmela and James Kalamaras -- were named as accompli…
Top-*k* Sampling	Dr. Anthony Moschetto is accused of providing police with weapons … Authorities say he was part of a conspiracy to harm or kill a rival doctor. … allegations against his client "completely unsubstantiated"
Typical Sampling	Dr. Anthony Moschetto is charged with crimes i**ncluding arson, conspiracy, burg** … **escription sale, weapons charges. His attorney says "none of anything in this case has any evidentiary value"**



If Beam Search is the Answer, What was the Question?

Clara Meister Ryan Cotterell Tim Vieira
ETH Zürich University of Cambridge Johns Hopkins University
clara.meister@inf.ethz.ch tim.vieira@gmail.com
ryan.cotterell@inf.ethz.ch

Abstract

Quite surprisingly, exact maximum a posteriori (MAP) decoding of neural language generators frequently leads to low-quality results (Stahlberg and Byrne, 2019). Rather, most state-of-the-art results on language generation

Try out typical sampling!

Typical sampling is implemented in the Hugging Face transformers library!

HUGGING FACE

```python
class TypicalLogitsWarper(LogitsWarper):
    def __init__(self, mass: float = 0.9, filter_value: float = -float("Inf"), min_tokens_to_keep: int = 1):
        mass = float(mass)
        if not (mass > 0 and mass < 1):
            raise ValueError(f"`typical_p` has to be a float > 0 and < 1, but is {mass}")

        self.filter_value = filter_value
        self.mass = mass
        self.min_tokens_to_keep = min_tokens_to_keep

    def __call__(self, input_ids: torch.LongTensor, scores: torch.FloatTensor) -> torch.FloatTensor:

        # calculate entropy
        normalized = torch.nn.functional.log_softmax(scores, dim=-1)
        p = torch.exp(normalized)
        ent = -(normalized * p).nansum(-1, keepdim=True)

        # shift and sort
        shifted_scores = torch.abs((-normalized) - ent)
        sorted_scores, sorted_indices = torch.sort(shifted_scores, descending=False)
        sorted_logits = scores.gather(-1, sorted_indices)
        cumulative_probs = sorted_logits.softmax(dim=-1).cumsum(dim=-1)

        # Remove tokens with cumulative mass above the threshold
        last_ind = (cumulative_probs < self.mass).sum(dim=1)
        last_ind[last_ind < 0] = 0
        sorted_indices_to_remove = sorted_scores > sorted_scores.gather(1, last_ind.view(-1, 1))
        if self.min_tokens_to_keep > 1:
            # Keep at least min_tokens_to_keep (set to min_tokens_to_keep-1 because we add the first one below)
            sorted_indices_to_remove[..., : self.min_tokens_to_keep] = 0
        indices_to_remove = sorted_indices_to_remove.scatter(1, sorted_indices, sorted_indices_to_remove)

        scores = scores.masked_fill(indices_to_remove, self.filter_value)
        return scores
```

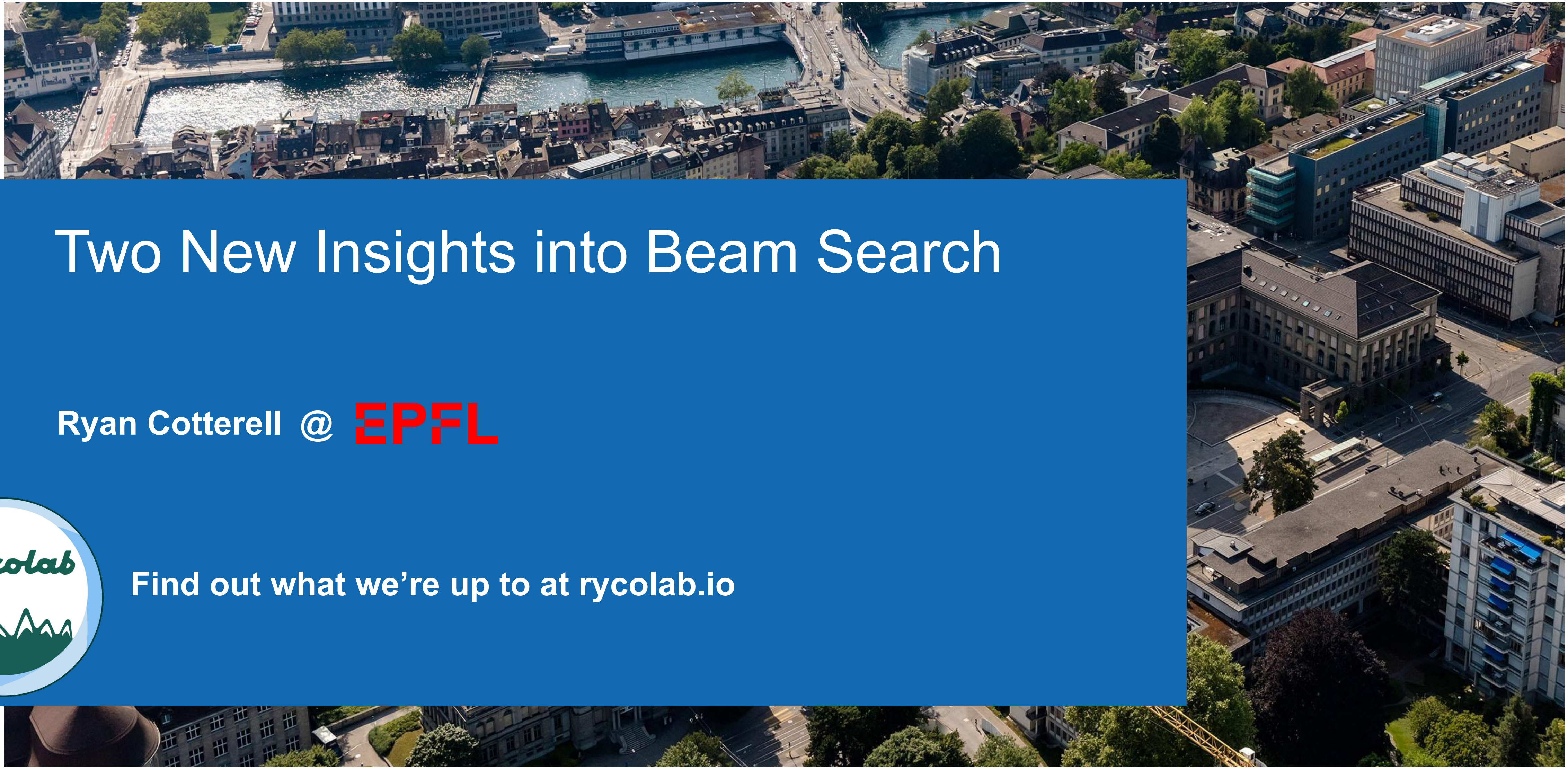
Fin

Fin

Find out more at https://rycolab.io/#publications

Link to Paper

ETH zürich



Two New Insights into Beam Search

Ryan Cotterell @ EPFL

Find out what we’re up to at rycolab.io

ETH zürich

Two New Insights into Beam Search

Ryan Cotterell @ THE UNIVERSITY OF MELBOURNE

Rycolab

Find out what we're up to at rycolab.io

Conclusion

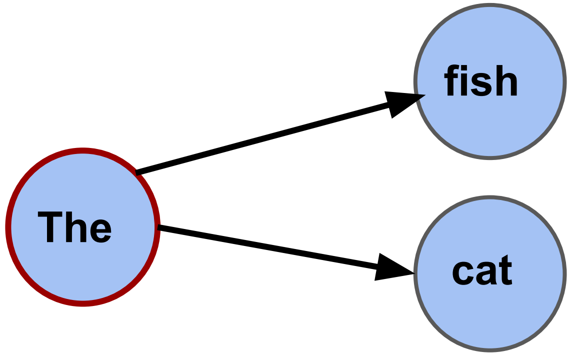
To conclude...

- We propose best-first beam search, an algorithm that allows for faster decoding while still guaranteeing the same set of results as standard beam search.
- We provide results on several sequence-to-sequence transduction tasks that show the speed-ups our algorithm provides over standard beam search for decoding neural models.
- We provide a general algorithm parameterized by a few choices that let us recover many standard search algorithms.
- We adapt popular alternate scoring functions to best- first beam search and find they offer competitive performance

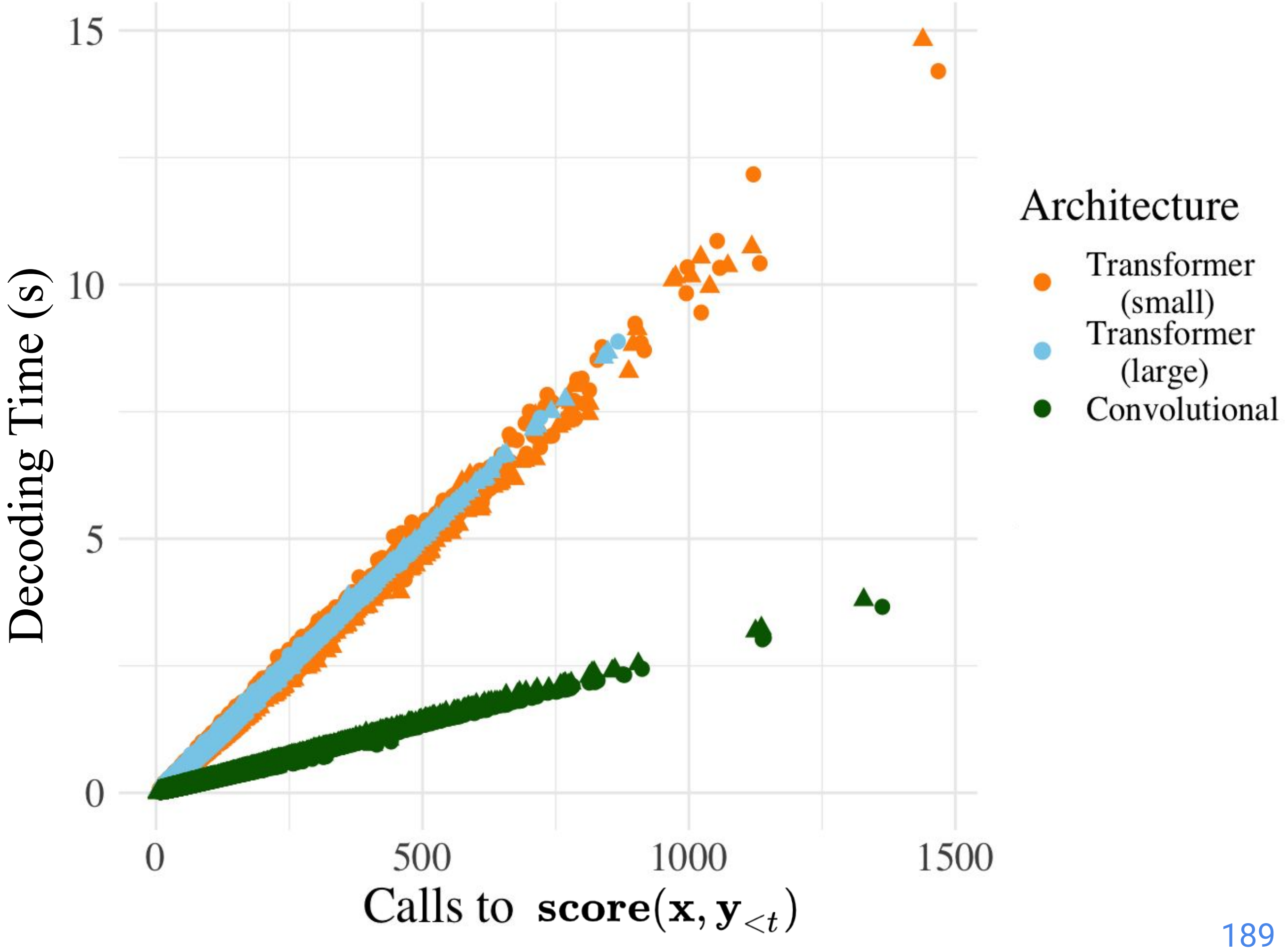
Fin

One Algorithm to Rule Them All

Decoding Sequence Models



$$\boxed{\mathbf{score}(\mathbf{x}, \mathbf{y}_{<t})}$$



One Algorithm to Rule Them All

General algorithm:

$\mathcal{Q} \leftarrow \text{priority\_queue}(\circleddash)$
$\mathcal{Q}.\text{push}(\langle 0, \text{BOS} \rangle)$
$\text{POPS} \leftarrow \text{counter}()$
while not $\text{stop}(\mathcal{Q})$ **and not** $\mathcal{Q}.\text{empty}()$:
  $\langle s_h, \mathbf{y} \rangle \leftarrow \mathcal{Q}.\text{pop}()$
  if $\text{POPS}[|\mathbf{y}|] \geqslant k$ **or** $|\mathbf{y}| > n_{max}$:
    continue
  $\text{POPS}[|\mathbf{y}|] \leftarrow \text{POPS}[|\mathbf{y}|] + 1$
  if $\mathbf{y}.\text{last}() = \text{EOS}$:
    $\mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ \text{EOS} \rangle)$
  else:
    for $y \in \mathcal{V}$:
      $s \leftarrow \text{score}(\mathbf{x}, \mathbf{y} \circ y)$
      $s_h \leftarrow s + h(\mathbf{x}, \mathbf{y} \circ y)$
      $\mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ y \rangle)$
return $\mathcal{Q}.\text{pop}()$ **if not** $\mathcal{Q}.\text{empty}()$ **else null**

Choice points:

$\circleddash$: comparator (1)
$\text{stop}(\cdot,\cdot)$: stopping criterion (2)
k: maximum beam size (3)
$h(\cdot,\cdot)$: heuristic function (4)

One Algorithm to Rule Them All

General algorithm:

```
Q ← priority_queue(⊜)
Q.push(⟨0, BOS⟩)
POPS ← counter()
while not stop(Q) and not Q.empty() :
    ⟨s_h, y⟩ ← Q.pop()
    if POPS[|y|] ⩾ k or |y| > n_max :
        continue
    POPS[|y|] ← POPS[|y|] + 1
    if y.last() = EOS :
        Q.push(⟨s_h, y ∘ EOS⟩)
    else:
        for y ∈ V :
            s ← score(x, y ∘ y)
            s_h ← s + h(x, y ∘ y)
            Q.push(⟨s_h, y ∘ y⟩)
return Q.pop() if not Q.empty() else null
```

Choice points:

⊜: comparator 1
stop(·, ·): stopping criterion 2
k: maximum beam size 3
$h(\cdot, \cdot)$: heuristic function 4

One Algorithm to Rule Them All

General algorithm:

$Q \leftarrow \text{priority\_queue}(\circleddash)$
$Q.\text{push}(\langle 0, \textsc{bos} \rangle)$
$\textsc{pops} \leftarrow \text{counter}()$
while not $\boxed{\text{stop}(Q)}$ **and not** $Q.\text{empty}()$
  $\langle s_h, \mathbf{y} \rangle \leftarrow Q.\text{pop}()$
  if $\textsc{pops}[|\mathbf{y}|] \geqslant k$ **or** $|\mathbf{y}| > n_{max}$:
    continue
  $\textsc{pops}[|\mathbf{y}|] \leftarrow \textsc{pops}[|\mathbf{y}|] + 1$
  if $\mathbf{y}.\text{last}() = \textsc{eos}$:
    $Q.\text{push}(\langle s_h, \mathbf{y} \circ \textsc{eos} \rangle)$
  else:
    for $y \in \mathcal{V}$:
      $s \leftarrow \text{score}(\mathbf{x}, \mathbf{y} \circ y)$
      $s_h \leftarrow s + h(\mathbf{x}, \mathbf{y} \circ y)$
      $Q.\text{push}(\langle s_h, \mathbf{y} \circ y \rangle)$
return $Q.\text{pop}()$ **if not** $Q.\text{empty}()$ **else null**

Choice points:

$\circleddash$: comparator 1
$\boxed{\text{stop}(\cdot,\cdot)\text{: stopping criterion } 2}$
k: maximum beam size 3
$h(\cdot,\cdot)$: heuristic function 4

One Algorithm to Rule Them All

General algorithm:

$\mathcal{Q} \leftarrow \text{priority\_queue}(\circleddash)$
$\mathcal{Q}.\text{push}(\langle 0, \text{BOS} \rangle)$
$\text{POPS} \leftarrow \text{counter}()$
while not $\text{stop}(\mathcal{Q})$ **and not** $\mathcal{Q}.\text{empty}()$:
  $\langle s_h, \mathbf{y} \rangle \leftarrow \mathcal{Q}.\text{pop}()$
  if $\text{POPS}[|\mathbf{y}|] \geqslant k$ **or** $|\mathbf{y}| > n_{max}$:
    continue
  $\text{POPS}[|\mathbf{y}|] \leftarrow \text{POPS}[|\mathbf{y}|] + 1$
  if $\mathbf{y}.\text{last}() = \text{EOS}$:
    $\mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ \text{EOS} \rangle)$
  else:
    for $y \in \mathcal{V}$:
      $s \leftarrow \text{score}(\mathbf{x}, \mathbf{y} \circ y)$
      $s_h \leftarrow s + h(\mathbf{x}, \mathbf{y} \circ y)$
      $\mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ y \rangle)$
return $\mathcal{Q}.\text{pop}()$ **if not** $\mathcal{Q}.\text{empty}()$ **else null**

Choice points:

$\circleddash$: comparator 1
$\text{stop}(\cdot, \cdot)$: stopping criterion 2
k: maximum beam size 3
$h(\cdot, \cdot)$: heuristic function 4

One Algorithm to Rule Them All

General algorithm:

$\mathcal{Q} \leftarrow \text{priority\_queue}(\circledcirc)$
$\mathcal{Q}.\text{push}(\langle 0, \text{BOS} \rangle)$
$\text{POPS} \leftarrow \text{counter}()$
while not $\text{stop}(\mathcal{Q})$ **and not** $\mathcal{Q}.\text{empty}()$:
    $\langle s_h, \mathbf{y} \rangle \leftarrow \mathcal{Q}.\text{pop}()$
    if $\text{POPS}[|\mathbf{y}|] \geqslant k$ **or** $|\mathbf{y}| > n_{max}$:
        continue
    $\text{POPS}[|\mathbf{y}|] \leftarrow \text{POPS}[|\mathbf{y}|] + 1$
    if $\mathbf{y}.\text{last}() = \text{EOS}$:
        $\mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ \text{EOS} \rangle)$
    else:
        for $y \in \mathcal{V}$:
            $s \leftarrow \text{score}(\mathbf{x}, \mathbf{y} \circ y)$
            $\boxed{s_h \leftarrow s + h(\mathbf{x}, \mathbf{y} \circ y)}$
            $\mathcal{Q}.\text{push}(\langle s_h, \mathbf{y} \circ y \rangle)$
return $\mathcal{Q}.\text{pop}()$ **if not** $\mathcal{Q}.\text{empty}()$ **else null**

Choice points:

$\circledcirc$: comparator 1
$\text{stop}(\cdot, \cdot)$: stopping criterion 2
k: maximum beam size 3
$\boxed{h(\cdot, \cdot)\text{: heuristic function 4}}$

One Algorithm to Rule Them All

Choice points:

$\circledcirc$: comparator ①
$\text{stop}(\cdot,\cdot)$: stopping criterion ②
k: maximum beam size ③
$h(\cdot,\cdot)$: heuristic function ④

	Beam Search	**Best-First Beam Search**	**A* Beam Search**
①	$\langle s_h, \mathbf{y}\rangle \circledcirc \langle s'_h, \mathbf{y}'\rangle \iff \|\mathbf{y}\| < \|\mathbf{y}\|'$ **or** $(\|\mathbf{y}\| = \|\mathbf{y}\|'$ **and** $s_h \geqslant s'_h)$	$\langle s_h, \mathbf{y}\rangle \circledcirc \langle s'_h, \mathbf{y}'\rangle \iff s_h > s'_h$ **or** $(s_h = s'_h$ **and** $\|\mathbf{y}\| < \|\mathbf{y}'\|)$	$\langle s_h, \mathbf{y}\rangle \circledcirc \langle s'_h, \mathbf{y}'\rangle \iff s_h > s'_h$ **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	$\langle s_h, \mathbf{y}\rangle \circledcirc \langle s'_h, \mathbf{y}'\rangle \iff \|\mathbf{y}\| < \|\mathbf{y}\|'$ **or** $(\|\mathbf{y}\| = \|\mathbf{y}\|'$ **and** $s_h \geqslant s'_h)$	$\langle s_h, \mathbf{y}\rangle \circledcirc \langle s'_h, \mathbf{y}'\rangle \iff s_h > s'_h$ **or** $(s_h = s'_h$ **and** $\|\mathbf{y}\| < \|\mathbf{y}'\|)$	$\langle s_h, \mathbf{y}\rangle \circledcirc \langle s'_h, \mathbf{y}'\rangle \iff s_h > s'_h$ **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

One Algorithm to Rule Them All

Choice points:

- $\oslash$: comparator 1
- $\boxed{\mathrm{stop}(\cdot,\cdot)\text{: stopping criterion } 2}$
- k: maximum beam size 3
- $\boxed{h(\cdot,\cdot)\text{: heuristic function } 4}$

length normalization:

$$\mathbf{score}(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{|\mathbf{y}|} p_\theta(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) + \lambda|\mathbf{y}|$$

One Algorithm to Rule Them All

Choice points:

$\oslash$: comparator 1
$\boxed{\text{stop}(\cdot,\cdot)\text{: stopping criterion } 2}$
k: maximum beam size 3
$\boxed{h(\cdot,\cdot)\text{: heuristic function } 4}$

We can guarantee the same results as beam search for slightly modified versions of traditional non-monotonic scoring functions!

For example:

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

length normalization!

$$\mathcal{U}(\mathbf{x}, \mathbf{y}) = \lambda \max\{0, n_{\max} - |\mathbf{y}|\}$$

One Algorithm to Rule Them All

Choice points:

- $\oslash$: comparator 1
- $\boxed{\text{stop}(\cdot,\cdot)\text{: stopping criterion 2}}$
- k: maximum beam size 3
- $\boxed{h(\cdot,\cdot)\text{: heuristic function 4}}$

We can guarantee the same results as beam search for slightly modified versions of non-monotonic scoring functions!

For example: length normalization

One Algorithm to Rule Them All

Choice points:

- $\oslash$: comparator (1)
- $\boxed{\text{stop}(\cdot,\cdot)\text{: stopping criterion (2)}}$
- k: maximum beam size (3)
- $\boxed{h(\cdot,\cdot)\text{: heuristic function (4)}}$

We can guarantee the same results as beam search for slightly modified versions of non-monotonic scoring functions!

For example: length normalization

IWSLT'14 De-En

k	λ	BLEU
5	0.5	33.7 (+0.1)
10	0.5	33.7 (+0.4)

MTTT Fr-En

k	λ	BLEU
5	1.0	34.1 (+0.8)
10	1.2	34.1 (+1.1)

To conclude...

To conclude...

- We propose best-first beam search, an algorithm that allows for faster decoding while still guaranteeing the same set of results as standard beam search.
- We provide results on several sequence-to-sequence transduction tasks that show the speed-ups our algorithm provides over standard beam search for decoding neural models.
- We provide a general algorithm parameterized by a few choices that let us recover many standard search algorithms.
- We adapt popular alternate scoring functions to best- first beam search and find they offer competitive performance

If Beam Search is the Answer, What was the Question?

Decoding Neural Language Generators

Neural probabilistic language generators are models of a (conditional) probability distribution over all sequences of text $\mathbf{y}$ given some input $\mathbf{x}$. We then "generate" text according to this distribution.

Our model, e.g., a neural seq-to-seq model: $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$

Decoding Neural Language Generators

Neural probabilistic language generators are models of a (conditional) probability distribution over all sequences of text $\mathbf{y}$ given some input $\mathbf{x}$. We then "generate" text according to this distribution.

Our model, e.g., a neural seq-to-seq model: $\boxed{p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})}$

Decoding Neural Language Generators

Neural probabilistic language generators are models of a (conditional) probability distribution over all sequences of text $\mathbf{y}$ given some input $\mathbf{x}$. We then "generate" text according to this distribution.

Our model, e.g., a neural seq-to-seq model: $p_\theta(\mathbf{y} \mid \mathbf{x})$

$\mathbf{x}$ = < El café negro me gusta mucho > $\longrightarrow p_\theta(\mathbf{y} \mid \mathbf{x}) \longrightarrow$

0.1	The coffee black me pleases much
….	….
….	….
….	….
0.8	I like black coffee

Decoding Neural Language Generators

Neural probabilistic language generators are models of a (conditional) probability distribution over all sequences of text $\mathbf{y}$ given some input $\mathbf{x}$. We then "generate" text according to this distribution.

Our model, e.g., a neural seq-to-seq model: $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$

In the case of *neural* generators, we typically model locally normalized distributions over words at each time step:

$$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$$

Decoding Neural Language Generators

The decoding problem (a.k.a. maximum a posteriori (MAP) inference):

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$

Think: What's the most probable translation $\mathbf{y}$ for some source sentence $\mathbf{x}$?

$\mathbf{x}$ = < El café negro me gusta mucho > $\longrightarrow p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \longrightarrow$

0.1	The coffee black me pleases much
….	….
….	….
….	….
0.8	I like black coffee

Decoding Neural Language Generators

The decoding problem (a.k.a. maximum a posteriori (MAP) inference):

$$\mathbf{y}^{\star} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$

Think: What's the most probable translation $\mathbf{y}$ for some source sentence $\mathbf{x}$?

$\mathbf{x}$ = < El café negro me gusta mucho > $\longrightarrow p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \longrightarrow$

0.1	The coffee black me pleases much
....	....
....	....
....	....
0.8	$\boxed{\text{I like black coffee}}$

Decoding Neural Language Generators

The decoding problem (a.k.a. maximum a posteriori (MAP) inference):

$$\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{Y}} \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$

usually really big; like the set of all sentences in MT!!

Think: What's the most probable translation $\mathbf{y}$ for some source sentence $\mathbf{x}$?

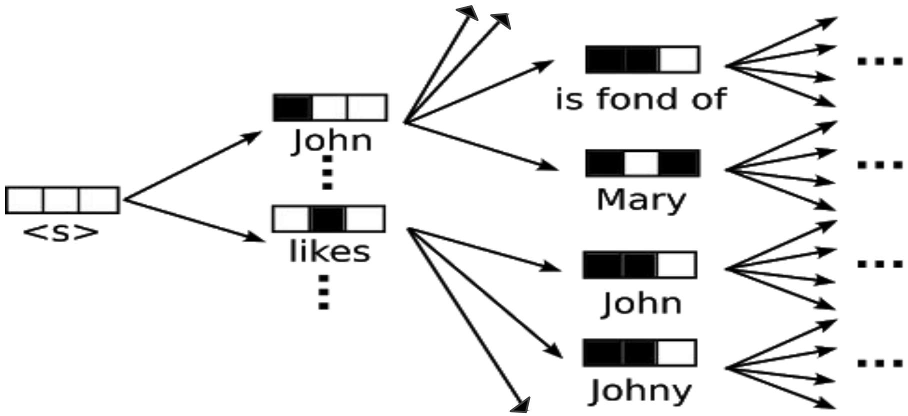
$\mathbf{x}$ = < El café negro me gusta mucho > $\longrightarrow p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \longrightarrow$

0.1	The coffee black me pleases much
....	....
....	....
....	....
0.8	I like black coffee

Decoding Neural Language Generators

How do we generally solve this?

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$



Most neural text generators have non-Markovian structure. This means we have none of the structural independences that allow dynamic programming methods for search to be so efficient.

Decoding Neural Language Generators

How do we generally solve this?

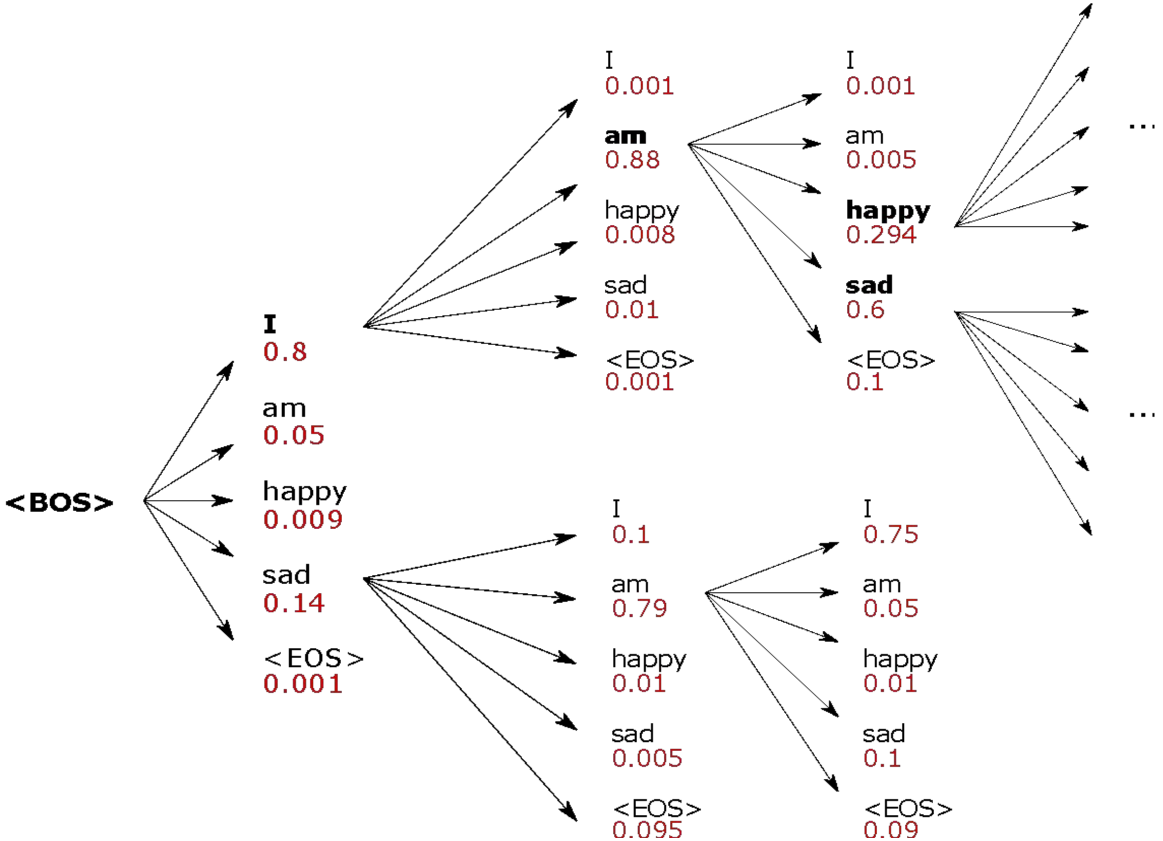
$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$

In short: we use heuristic search methods, like beam search!

Decoding Neural Language Generators

Beam Search

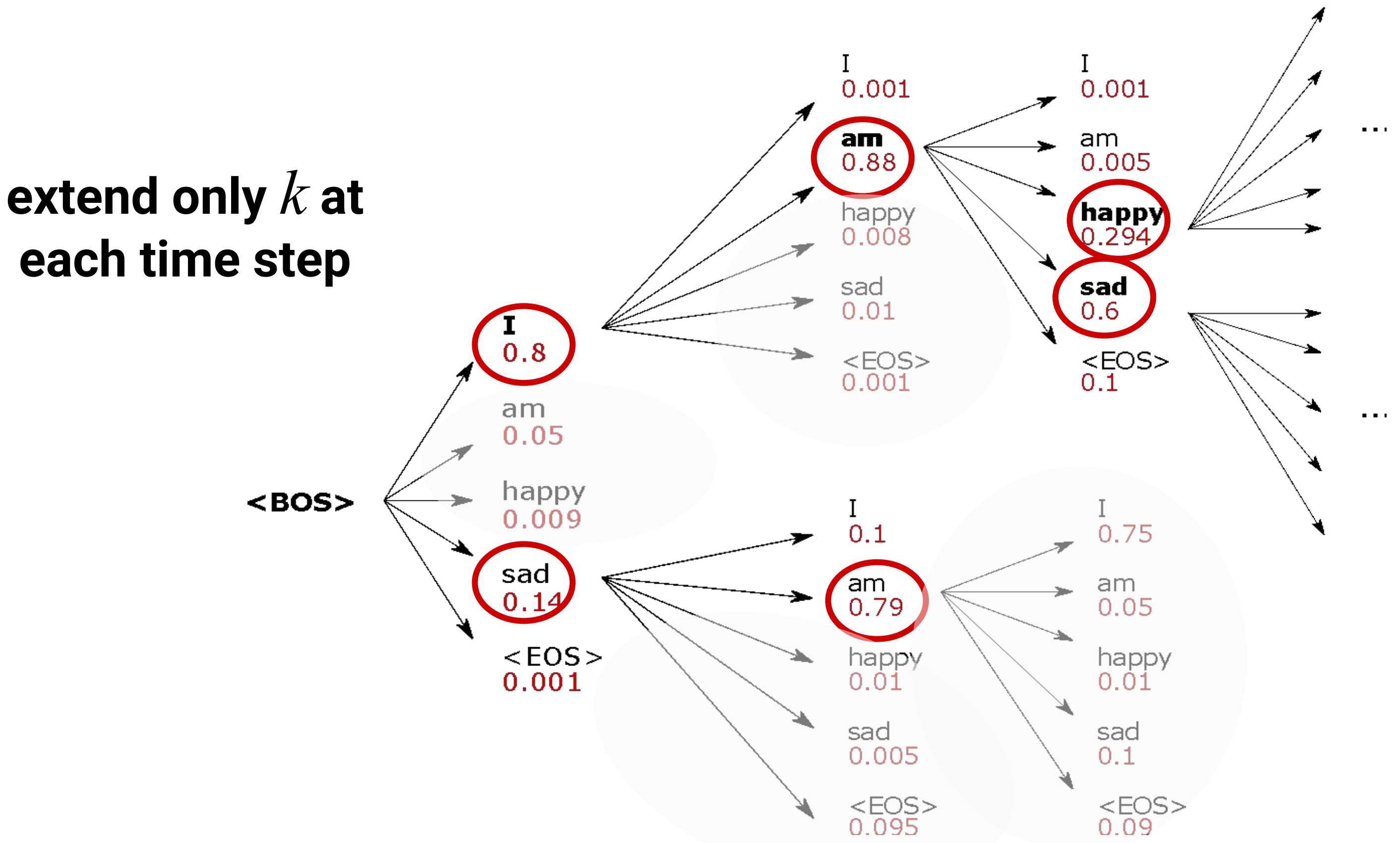
- Pruned breadth-first search where the breadth is limited to size k



Decoding Neural Language Generators

Beam Search

- Pruned breadth-first search where the breadth is limited to size k

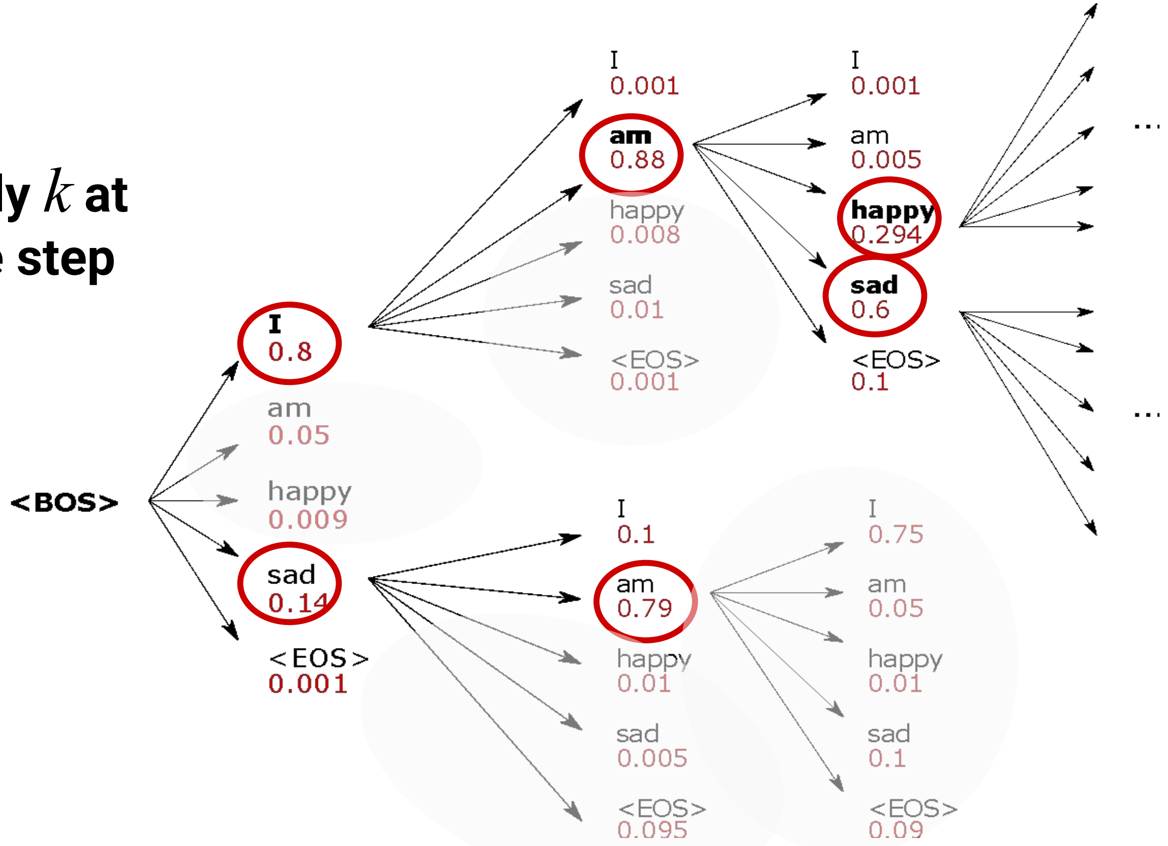


Decoding Neural Language Generators

Beam Search

- Pruned breadth-first search where the breadth is limited to size k

extend only k at each time step



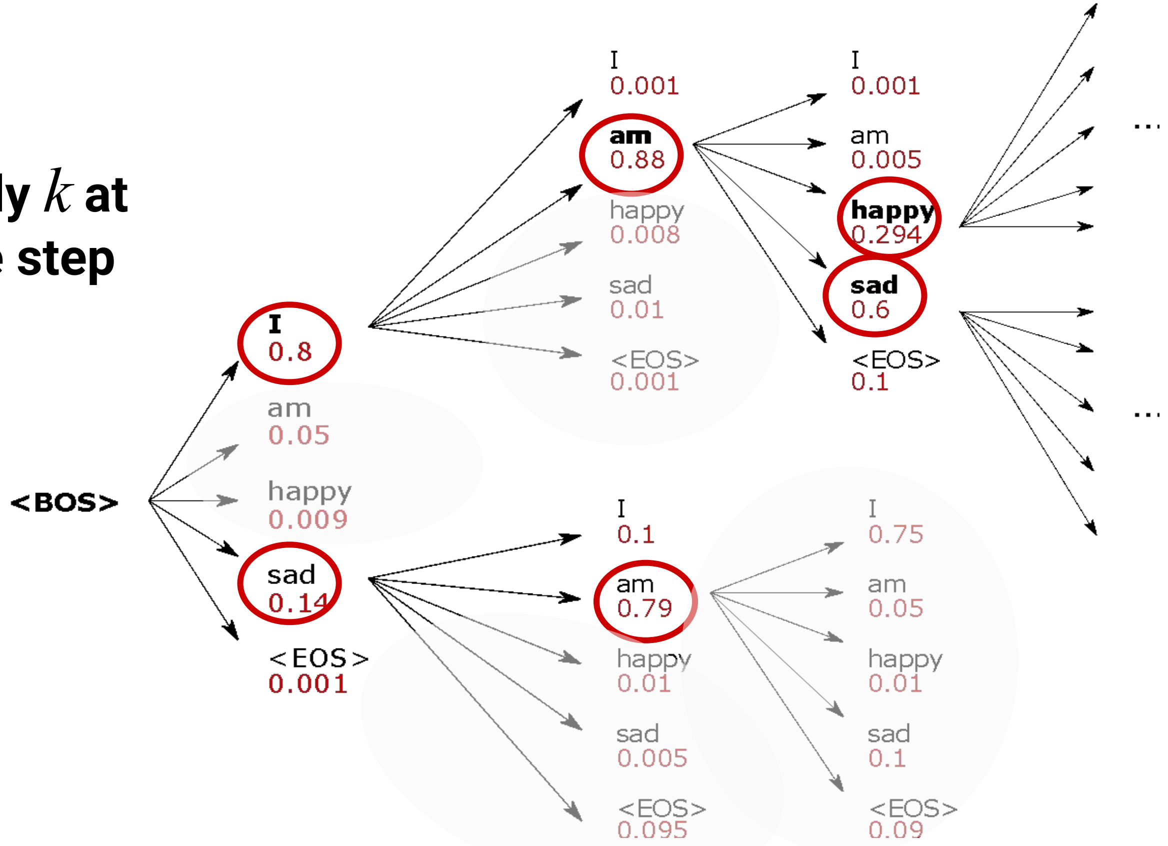
stop at a pre-determined max sequence length

Decoding Neural Language Generators

Beam Search

- Pruned breadth-first search where the breadth is limited to size k

extend only k at each time step



stop at a pre-determined max sequence length

No guarantee that we'll find the most probable sequence under the model!

Decoding Neural Language Generators

Beam Search

- How often *does* beam search find the global optimum in language generation tasks?

Decoding Neural Language Generators

Beam Search

- How often *does* beam search find the global optimum in language generation tasks?

  Answer: Not often.*

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

*At least not for language generation tasks for which this question has been studied

Decoding Neural Language Generators

Beam Search

- Yet how come it does so well?

  Answer: ????

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

Decoding Neural Language Generators

Beam Search

- And how come it does **so much** better than exact search?

  Answer: ???????????????

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

Decoding Neural Language Generators

Beam Search

- The solution to MAP inference is clearly not desirable text...

$$\mathbf{y}^{\star} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$$

Search	BLEU	#Search Errors	#Empty
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

Decoding Neural Language Generators

Beam Search

- But the solution provided by beam search is...

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \ ?$$

Search	**BLEU**	**#Search Errors**	**#Empty**
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

Decoding Neural Language Generators

Beam Search

- But the solution provided by beam search is...

$$\boxed{\mathbf{y}^{\star} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \ ?}$$

Search	**BLEU**	**#Search Errors**	**#Empty**
Greedy	29.3	73.6%	0.0%
Beam-10	30.3	57.7%	0.0%
Exact	2.1	0.0%	51.8%

Results on NMT systems decoded with different search strategies from Stahlberg and Byrne (2019)

Decoding Neural Language Generators

Our (clunky) algorithm for beam search

$$Y_0 = \{\textsc{bos}\}$$

$$Y_t = \underset{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}}{\operatorname{argmax}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x})$$

$$\mathcal{B}_t = \left\{ \mathbf{y}_{t\text{-}1} \circ y \mid y \in \bar{\mathcal{V}} \textbf{ and } \mathbf{y}_{t\text{-}1} \in Y_{t\text{-}1} \right\}$$

Return $Y_{n_{\max}}$

Decoding Neural Language Generators

Our (clunky) algorithm for beam search

$$Y_0 = \{\textsc{bos}\}$$

$$Y_t = \underset{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}}{\operatorname{argmax}} \log p_{\theta}(Y' \mid \mathbf{x})$$

$$\mathcal{B}_t = \left\{ \mathbf{y}_{t\text{-}1} \circ y \mid y \in \bar{\mathcal{V}} \textbf{ and } \mathbf{y}_{t\text{-}1} \in Y_{t\text{-}1} \right\}$$

Return $Y_{n_{\max}}$

$$\rightarrow \quad Y^{\star} = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

Can we write this as a (sleek) optimization problem?

Decoding Neural Language Generators

Our (clunky) algorithm for beam search

$$Y_0 = \{\textsc{bos}\}$$

$$Y_t = \underset{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}}{\operatorname{argmax}} \log p_{\boldsymbol{\theta}}(Y' \mid \mathbf{x})$$

$$\mathcal{B}_t = \left\{ \mathbf{y}_{t\text{-}1} \circ y \mid y \in \bar{\mathcal{V}} \textbf{ and } \mathbf{y}_{t\text{-}1} \in Y_{t\text{-}1} \right\}$$

Return $Y_{n_{\max}}$

$$\longrightarrow \quad Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

Can we write this as a (sleek) optimization problem? **Spoiler alert: Yes!**

What problem does beam search solve?

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \quad ?$$

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \boxed{?}$$

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{Y}} \; ?$$

$$\mathbf{y}^\star = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} \left(\boxed{\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})} - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

original objective

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; ?$$

"regularizer"

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \left(\boxed{\log p_{\boldsymbol\theta}(\mathbf{y} \mid \mathbf{x})} - \boxed{\lambda \cdot \mathcal{R}(\mathbf{y})} \right)$$

original objective

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \quad ?$$

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}}{\mathrm{argmax}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

$$\mathcal{R}_{\mathrm{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(-\log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) + \min_{y' \in \mathcal{V}} \log p_{\boldsymbol{\theta}}(y' \mid \mathbf{x}, \mathbf{y}_{<t}) \right)^2$$

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \ ?$$

$$\Downarrow$$

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) - \boxed{\lambda} \cdot \mathcal{R}(\mathbf{y}) \right)$$

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(-\log p_{\theta}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) + \min_{y' \in \mathcal{V}} \log p_{\theta}(y' \mid \mathbf{x}, \mathbf{y}_{<t}) \right)^2$$

The optimum of our regularized decoding problem as λ →∞ is the same as the solution found by greedy search!

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^{\star} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; ?$$

$$\mathbf{y}^{\star} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \Big(\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \Big)$$

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \Big(-\log p_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) + \min_{y' \in \mathcal{V}} \log p_{\boldsymbol{\theta}}(y' \mid \mathbf{x}, \mathbf{y}_{<t}) \Big)^2$$

surprisal:

$$u_0(\textsc{bos}) = 0$$
$$u_t(y) = -\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}, \mathbf{y}_{<t}), \;\; \textbf{for } t \geq 1$$

What problem does beam search solve?

Easy Case: k = 1 (greedy search)

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; ?$$

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - \min_{y' \in \mathcal{V}} u_t(y') \right)^2$$

surprisal:

$$u_0(\text{BOS}) = 0$$

$$u_t(y) = -\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}, \mathbf{y}_{<t}), \;\; \textbf{for } t \geq 1$$

What problem does beam search solve?

General Case: k > 1

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y},\\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

⬇

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y},\\ |Y|=k}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(Y) \right)$$

Now we're dealing with sets!

What problem does beam search solve?

General Case: k > 1

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \quad ?$$

$$\Downarrow$$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y|=k}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(Y) \right)$$

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)$$

What problem does beam search solve?

General Case: k > 1

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y| = k}}{\operatorname{argmax}} \quad ?$$

$$\Downarrow$$

$$Y^\star = \underset{\substack{Y \subseteq \mathcal{Y}, \\ |Y| = k}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{x}) - \boxed{\lambda \cdot} \mathcal{R}(Y) \right)$$

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'| = k}} u_t(Y') \right)$$

The optimum of our regularized decoding problem as λ →∞ is the same as the solution found by beam search!

Beam search as a cognitively motivated search heuristic

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)^2$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

surprisal

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)^2$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

surprisal

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(\underbrace{u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t,\\ |Y'|=k}} u_t(Y')}_{\text{(squared) distance from lowest surprisal choice}} \right)^2$$

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \Big(\overbrace{u_t(\mathbf{Y}_t)}^{\text{surprisal}} - \min_{\substack{\mathbf{Y}' \subseteq \mathcal{B}_t, \\ |\mathbf{Y}'| = k}} u_t(\mathbf{Y}') \Big)^2$$

Beam search enforces low surprisal choices at each time step

(squared) distance from lowest surprisal choice

Beam search as a cognitively motivated search heuristic

What do these optimization problems tell us about beam search?

surprisal

$$\mathcal{R}_{\text{beam}}(Y) = \sum_{t=1}^{n_{\max}} \left(u_t(Y_t) - \min_{\substack{Y' \subseteq \mathcal{B}_t, \\ |Y'|=k}} u_t(Y') \right)^2$$

Beam search enforces **low surprisal choices** at each time step

(squared) distance from lowest surprisal choice

Beam search as a cognitively motivated search heuristic

Great! Why does that matter?

The **uniform information density hypothesis** (Levy, 2005; Levy and Jaeger, 2007; Jaeger 2010):

“Within the bounds defined by grammar, speakers prefer utterances that distribute information uniformly across the signal (information density). Where speakers have a choice between several variants to encode their message, they prefer the variant with more uniform information density (ceteris paribus)”

Beam search as a cognitively motivated search heuristic

Great! Why does that matter?

The **uniform information density hypothesis** (Levy, 2005; Levy and Jaeger, 2007; Jaeger 2010):

"Within the bounds defined by grammar, speakers prefer utterances that distribute information uniformly across the signal (information density). Where speakers have a choice between several variants to encode their message, they prefer the variant with more uniform information density (ceteris paribus)"

TL;DR: Humans prefer sentences that evenly distribute information across the sentence. We don't like moments of high surprisal!

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is $[_{NP}$ the family$_i$ $[_{RC}$ (that) you cook for $\_{}_i]]$?

This sentence is also grammatically correct (and relays the same message) without the word “that.”

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is [$_{NP}$ the family$_i$ [$_{RC}$ (~~that~~) you cook for $\_{i}$]]?

This sentence is also grammatically correct (and relays the same message) without the word “that.”

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is [$_{NP}$ the family$_i$ [$_{RC}$ (~~that~~) you cook for $\_{i}$]]?

This sentence is also grammatically correct (and relays the same message) without the word “that.”

But it just sounds better with it….

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is $[_{NP}$ the family$_i$ $[_{RC}$ ~~(that)~~ you cook for $\_{}_i]]$?

Information-theoretic explanation:

- Without “that,” the word “you” conveys two pieces of information: the onset of a relative clause and part of its internal contents.

Beam search as a cognitively motivated search heuristic

The uniform information density hypothesis in action:

How big is [$_{\text{NP}}$ the family$_i$ [$_{\text{RC}}$ (~~that~~) you cook for $\_{i}$]]?

Information-theoretic explanation:

- Without "that," the word "you" conveys two pieces of information: the onset of a relative clause and part of its internal contents.
- Including the relativizer spreads information across two words, thereby **distributing information across the sentence** more uniformly and **avoiding instances of high surprisal**

Beam search as a cognitively motivated search heuristic

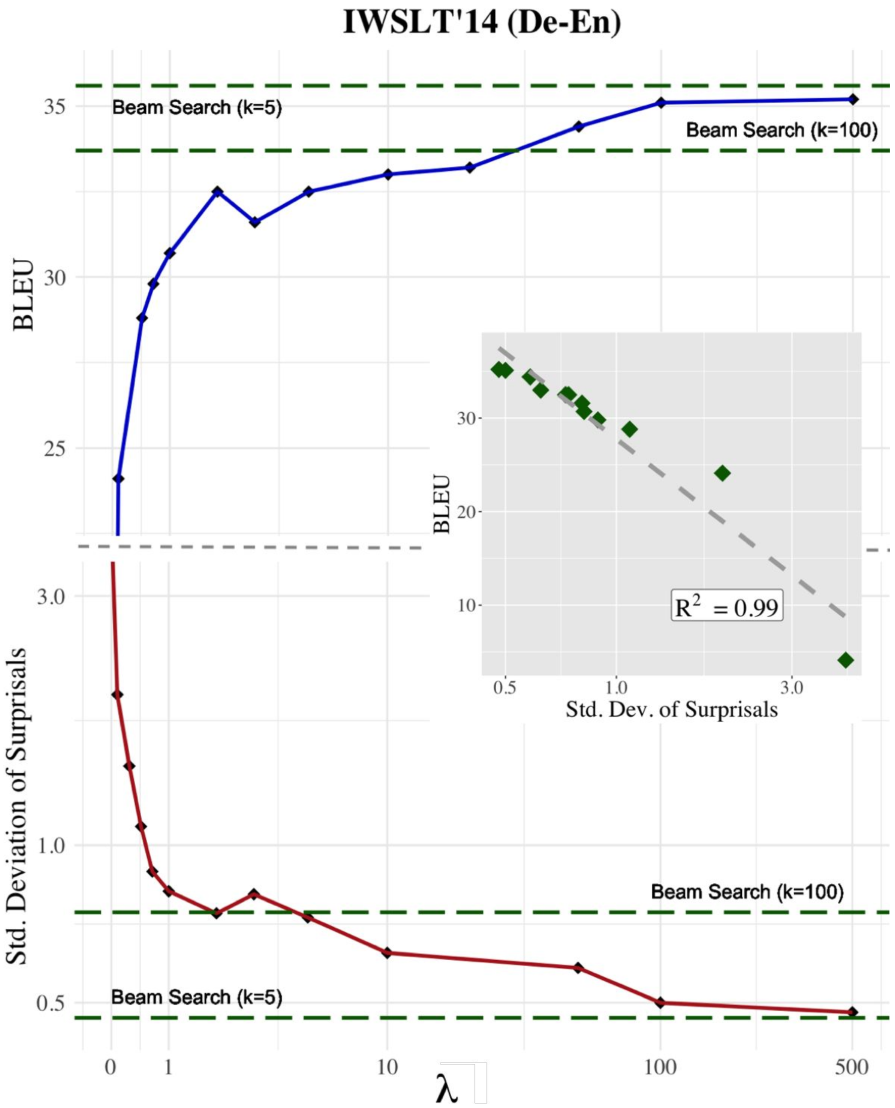
Nice hypothesis

but where’s the proof?

Beam search as a cognitively motivated search heuristic

Nice hypothesis

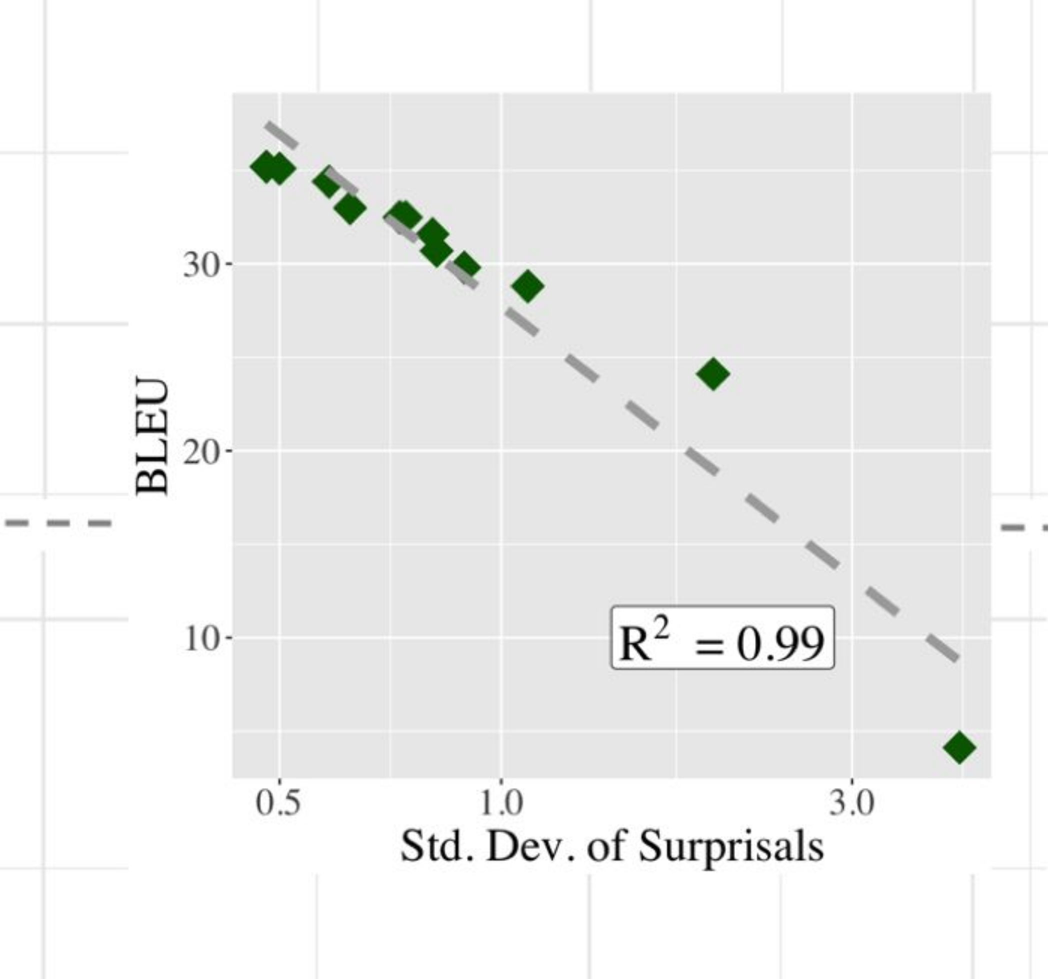
but where's the proof?



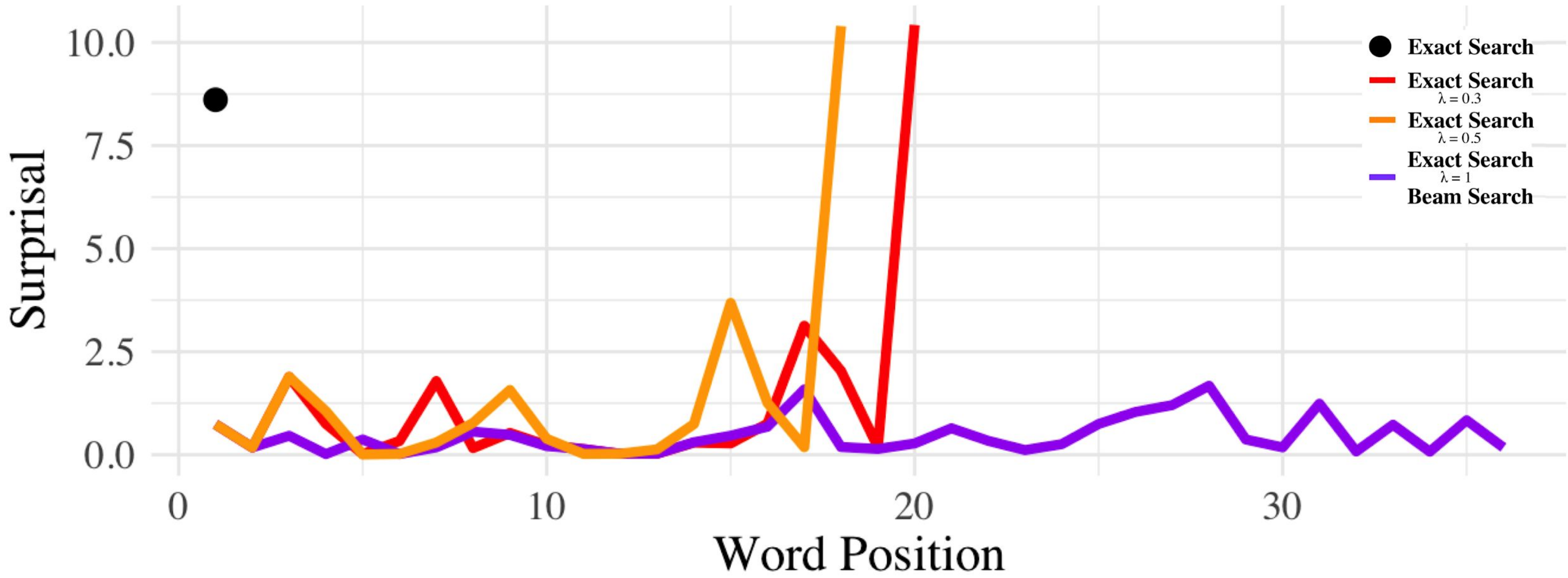
Beam search as a cognitively motivated search heuristic

Nice hypothesis

but where's the proof?



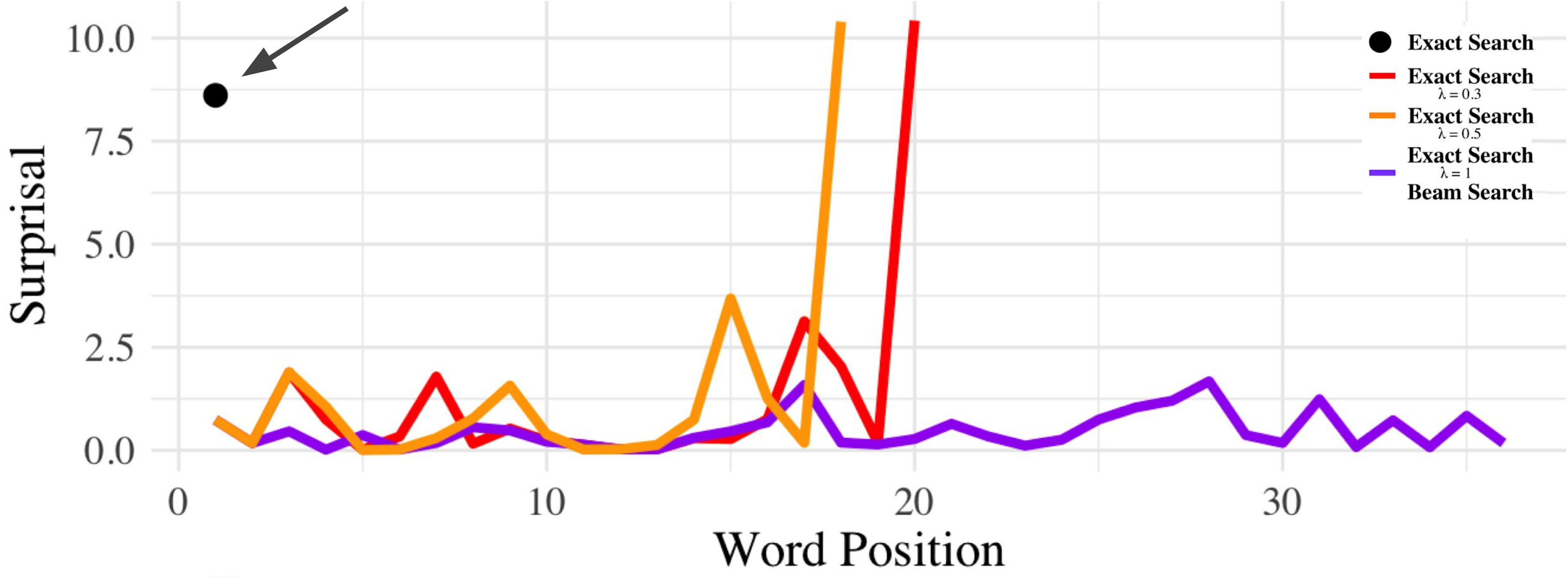
Beam search as a cognitively motivated search heuristic



Recall: our standard decoding objective explicitly minimizes the sum of surprisals, i.e., maximizes log-probability.

Beam search as a cognitively motivated search heuristic

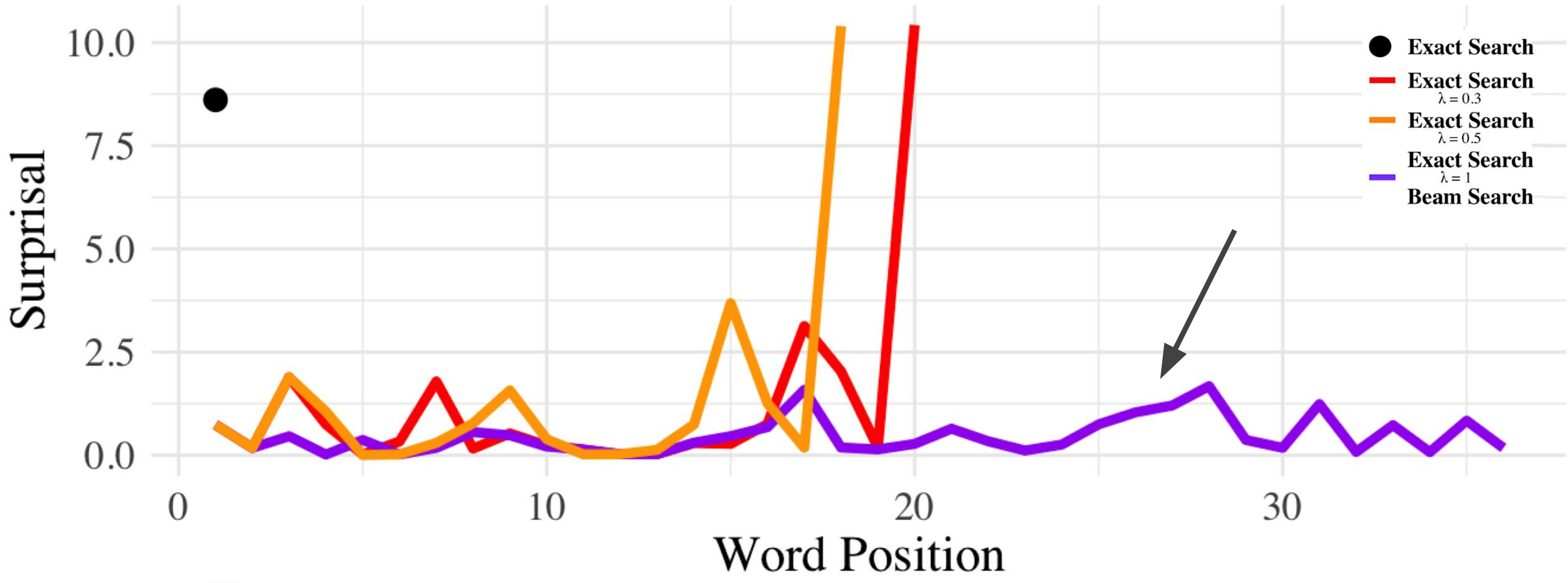
Exact search finds the highest probability sequence, regardless of local decisions



Recall: our standard decoding objective explicitly minimizes the sum of surprisals, i.e., maximizes log-probability.

Beam search as a cognitively motivated search heuristic

Beam search enforces UID!



Recall: our standard decoding objective explicitly minimizes the sum of surprisals, i.e., maximizes log-probability. Therefore, the only way the distribution of a solution can become distinctly non-uniform is when there are several high-surprisal decisions

Uniform information density (UID) regularization

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- Recall our regularized decoding objective:

$$\mathbf{y}^{\star} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- Recall our regularized decoding objective:

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) - \lambda \cdot \mathcal{R}(\mathbf{y}) \right)$$

- In practice, it’s not practical do set optimization...

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- We started with a "greedy" regularizer that mimics beam search (k=1)

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - \min_{y' \in \mathcal{V}} u_t(y') \right)^2$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages high variance in surprisals**?

$$\mathcal{R}_{\text{var}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - \mu \right)^2$$

define:

$$\mu = 1/|\mathbf{y}| \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages high variance in surprisals *locally***?

$$\mathcal{R}_{\text{local}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|}\sum_{t=1}^{|\mathbf{y}|}\Big(u_t(y_t) - u_{t-1}(y_{t-1})\Big)^2$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages instances of high surprisal**?

$$\mathcal{R}_{\max}(\mathbf{y}) = \max_{t=1}^{|\mathbf{y}|} u_t(y_t)$$

Uniform information density (UID) regularization

Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that **discourages consistently high surprisal**?

$$\mathcal{R}_{\text{square}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)^2$$

Uniform information density (UID) regularization

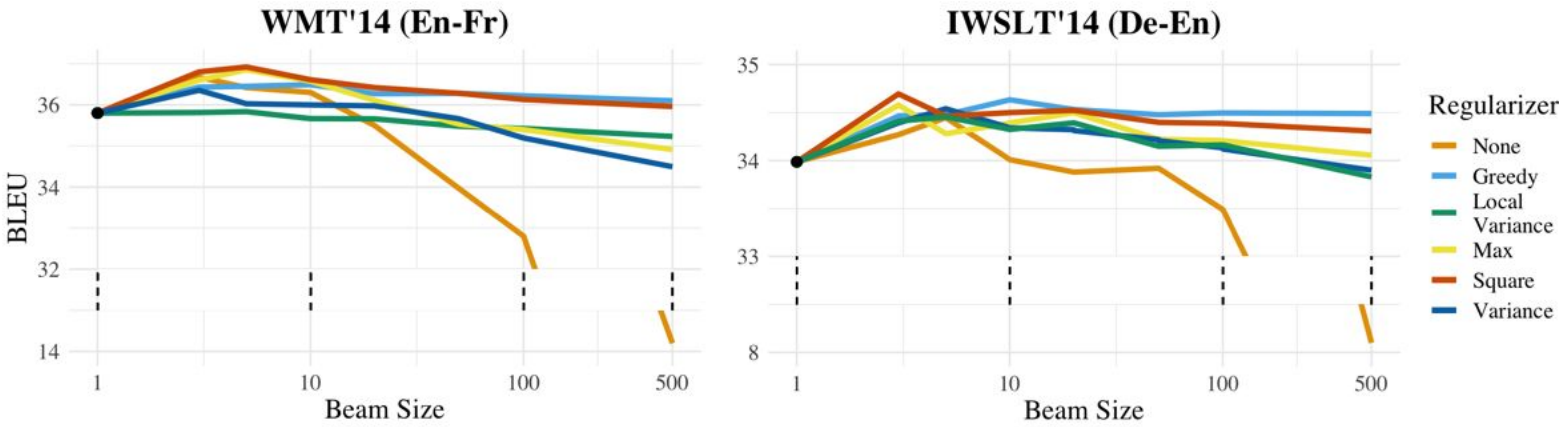
Luckily for us, our favorite search heuristic has been enforcing UID for years. Can we explicitly encourage UID in generated text?

- How about a regularizer that discourages consistently high surprisal?

Let's see 'em in action!

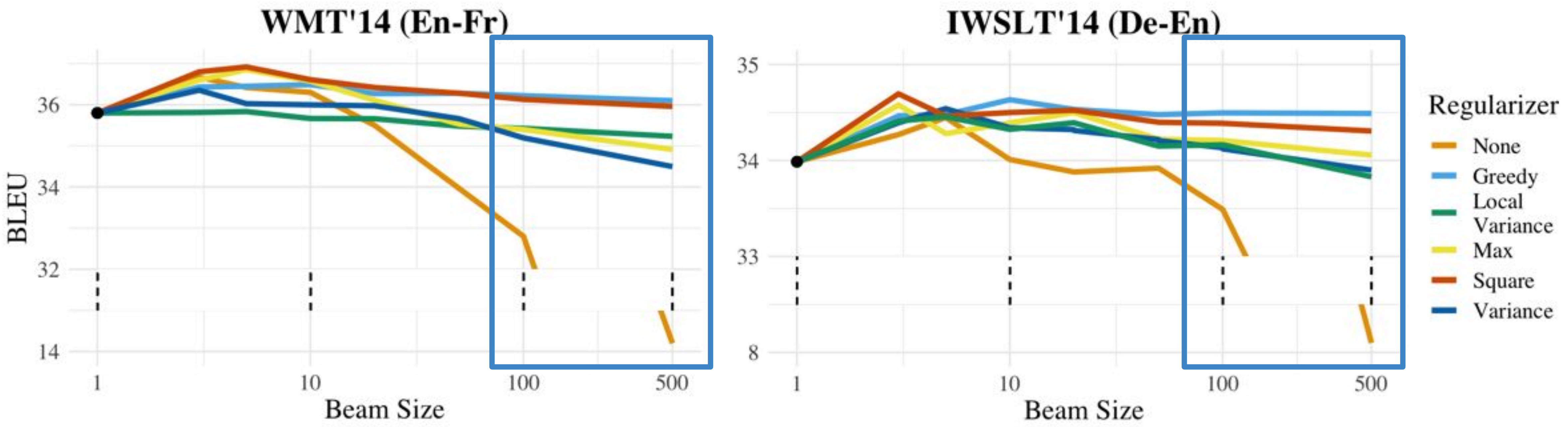
$$\mathcal{R}_{\text{square}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)^2$$

Uniform information density (UID) regularization



Experiments on NMT systems; decoding of models with various objectives for different beam sizes

Uniform information density (UID) regularization



Experiments on NMT systems; decoding of models with various objectives for different beam sizes

Uniform information density (UID) regularization

	$k=5$	$k=10$	$k=100$	$k=500$
No Regularization	36.42	36.30	32.83	14.66
Squared Regularizer	**36.92**	36.42	36.13	35.96
Greedy Regularizer	36.45	36.49	36.22	36.15
Combined Regularizers	36.69	**36.65**	**36.48**	**36.35**
Length Normalization	36.02	35.94	35.80	35.11

Table 1: BLEU scores on first 1000 samples of Newstest2014 for predictions generated with various decoding strategies. Best scores per beam size are bolded.

Uniform information density (UID) regularization

	$k=5$	$k=10$	$k=100$	$k=500$
No Regularization	36.42	36.30	32.83	14.66
Squared Regularizer	**36.92**	36.42	36.13	35.96
Greedy Regularizer	36.45	36.49	36.22	36.15
Combined Regularizers	36.69	**36.65**	**36.48**	**36.35**
Length Normalization	36.02	35.94	35.80	35.11

Luckily, not a huge difference! One regularizer seems good enough

Table 1: BLEU scores on first 1000 samples of Newstest2014 for predictions generated with various decoding strategies. Best scores per beam size are bolded.

To conclude...

To conclude...

- We frame beam search, a search heuristic that does strangely well at finding good text under neural probabilistic text generators, as the solution to an exact decoding problem.
- We provide evidence that beam search has an inductive bias which can be linked to the promotion of uniform information density (UID), a theory from cognitive science regarding even distribution of information in linguistic signals.
- We observe a strong relationship between variance of surprisals (an operationalization of UID) and BLEU in our experiments with NMT models.
- We design a set of objectives to explicitly encourage UID in text generated from neural probabilistic models and find that they alleviate the quality degradation typically seen with increased beam widths.

Thank you!

Title: If beam search is the answer, what was the question?

Authors: Clara Meister, Tim Vieira, and Ryan Cotterell

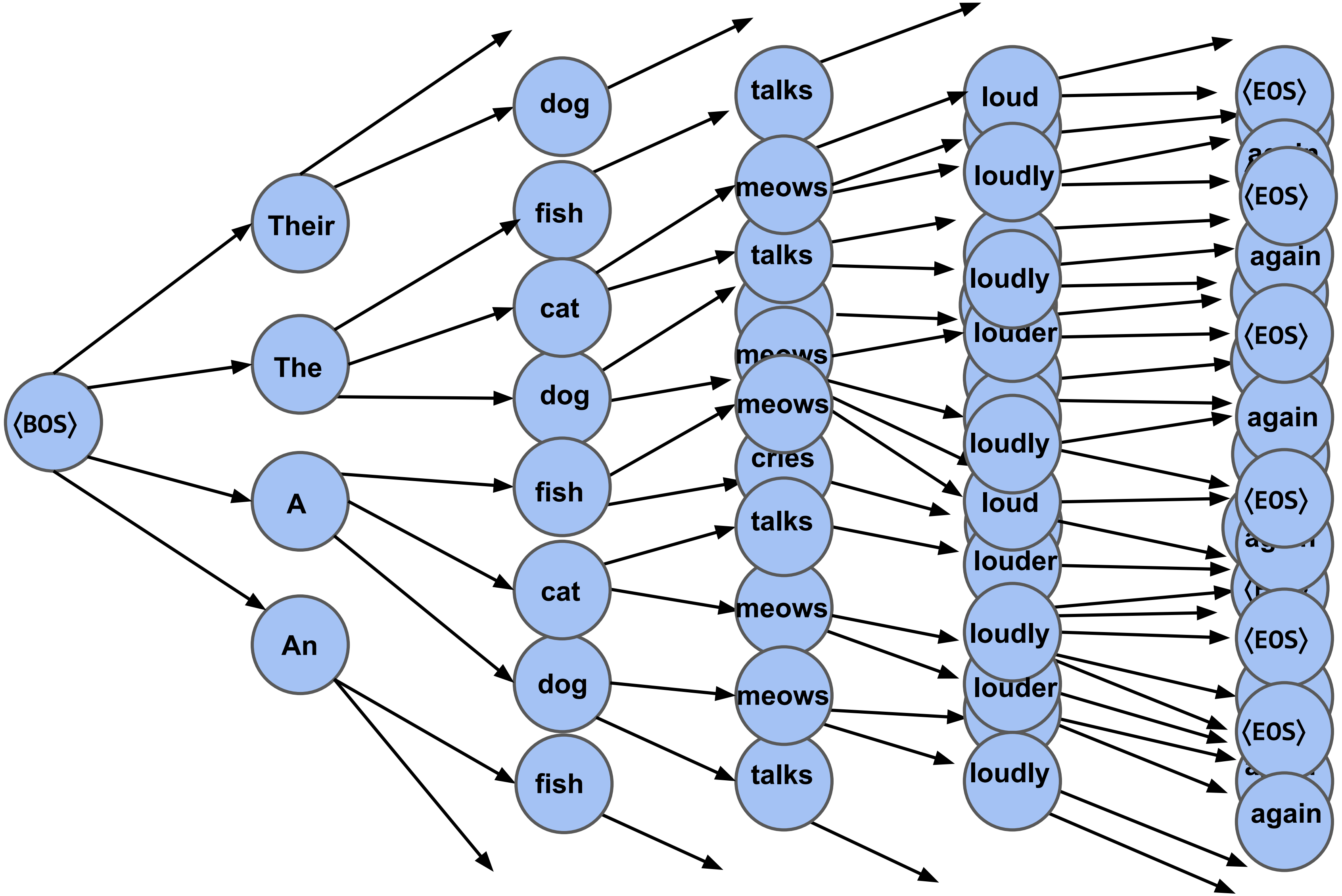
Link to Paper

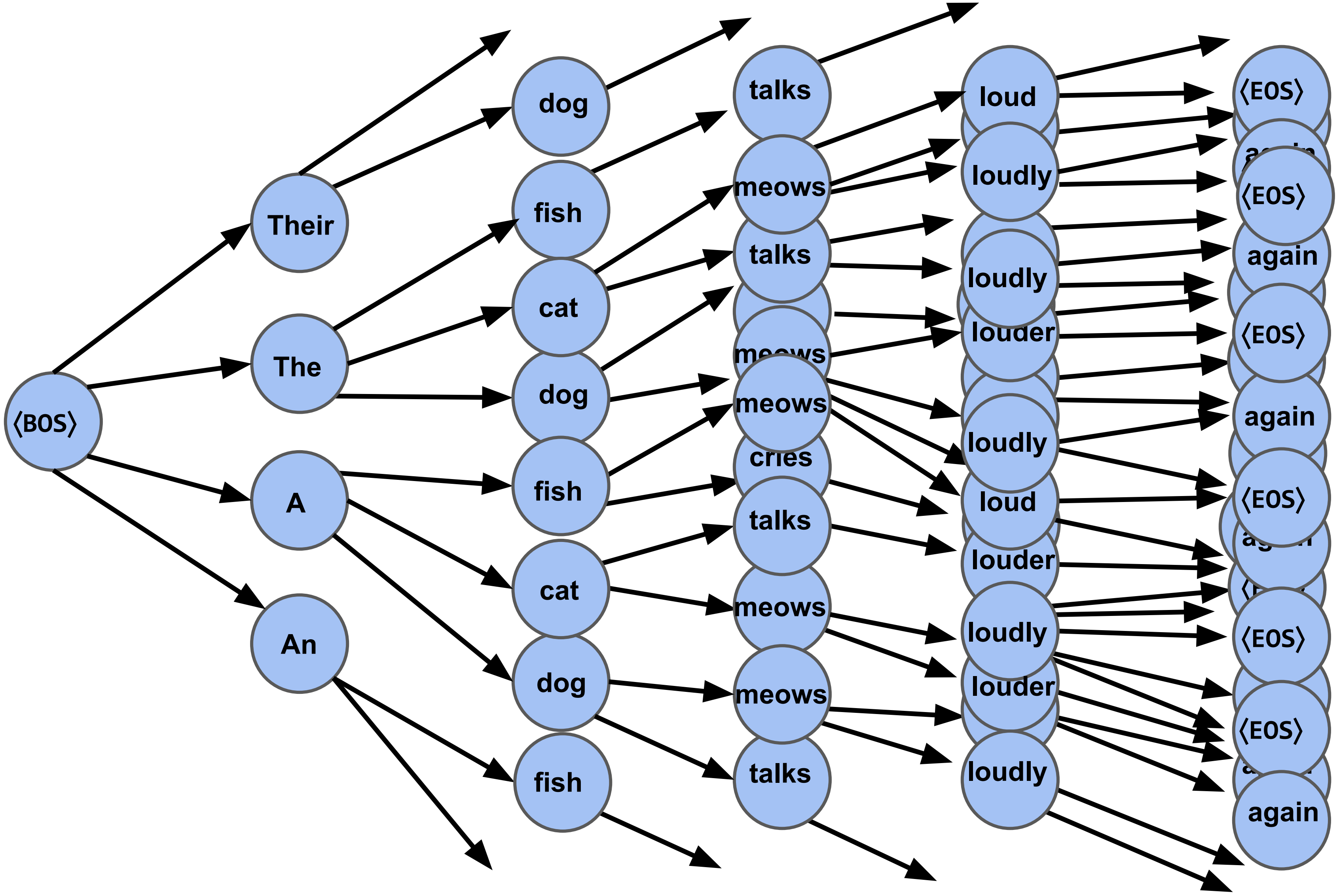
Thank you!

Title: Best-First Beam Search

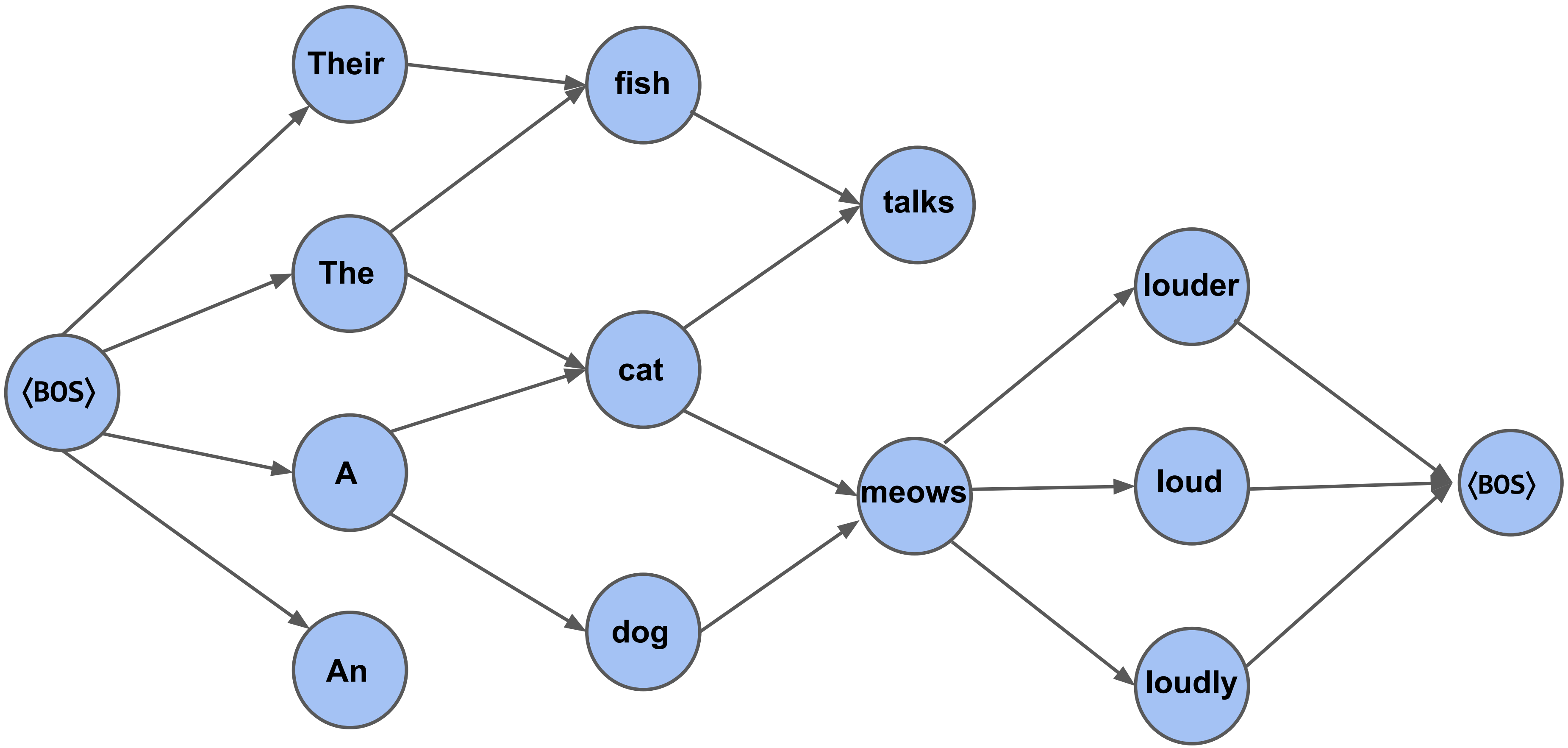
Authors: Clara Meister, Tim Vieira, and Ryan Cotterell

Link to Paper


⟨BOS⟩
Their
The
A
An
dog
fish
cat
dog
fish
cat
dog
fish
talks
meows
talks
meows
cries
talks
meows
meows
talks
loud
loudly
loudly
louder
loudly
loud
louder
loudly
louder
loudly
⟨EOS⟩
⟨EOS⟩
again
⟨EOS⟩
again
⟨EOS⟩
⟨EOS⟩
⟨EOS⟩
again



⟨BOS⟩
Their
The
A
An
dog
fish
cat
dog
fish
cat
dog
fish
talks
meows
talks
meows
cries
talks
meows
meows
talks
loud
loudly
loudly
louder
loudly
loud
louder
loudly
louder
loudly
⟨EOS⟩
⟨EOS⟩
again
⟨EOS⟩
again
⟨EOS⟩
⟨EOS⟩
⟨EOS⟩
again



⟨BOS⟩
Their
The
A
An
fish
cat
dog
talks
meows
louder
loud
loudly
⟨BOS⟩