CSCI 5541: Natural Language Processing

Lecture 7: Language Models: Search and Decoding Algorithms

Dongyeop Kang (DK), University of Minnesota

dongyeop@umn.edu | twitter.com/dongyeopkang | dykang.github.io

computer science & engineering

MINNESOTA NLP EST. 2021

By ModelScope Text to Video Synthesis

AI model generating different outputs and searching for a good answer

University of Minnesota

Driven to Discover®

Outline

- Search
  - Basics
  - Greedy Search
  - Beam Search
  - Fixing Model Errors in Search
- Sampling
  - Top-k Sampling
  - Top-p Sampling
- Search in Training
- HW3 out -> Feb 27 (Tues)
- Demo

- *greedy decoding* by calling greedy_search() if num_beams=1 and do_sample=False.
- *multinomial sampling* by calling sample() if num_beams=1 and do_sample=True.
- *beam-search decoding* by calling beam_search() if num_beams>1 and do_sample=False.
- *beam-search multinomial sampling* by calling beam_sample() if num_beams>1 and do_sample=True.
- *diverse beam-search decoding* by calling group_beam_search(), if num_beams>1 and num_beam_groups>1.
- *constrained beam-search decoding* by calling constrained_beam_search(), if constraints!=None or force_words_ids!=None.

https://huggingface.co/docs/transformers/main_classes/text_generation

Notation

$$P(x_j \mid x_1, \ldots x_{j-1})$$

Context given and previous text generated

$$P(Y_j \mid X, y_1, \ldots y_{j-1})$$

Context given in seq2seq setup

Previous text generated

Search

Generation Problem

- We have a language model of $P(Y|X)$ trained on text corpora, how do we use it to generate a sentence?
- Two methods:
  - We want the best possible single output

    Search (Argmax): Try to generate the sentence with the highest probability.

$$Y_j = argmax\ P(Y_j | X, y_1 \dots y_{j-1})$$

  - We want to observe multiple outputs according to the probability distribution

    Sampling: Try to generate a random sentence according to the probability distribution.

$$Y_j = \text{sampling from } P(Y_j | X, y_1 \dots y_{j-1})$$

Ancestral Sampling

- Randomly generate words one-by-one
  - $Y_j = P(Y_j \mid X, y_1 \dots y_{j-1})$
  - Until <STOP> is generated

- An exact method for sampling from $P(X)$, no further work needed.

https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3

Search Basics

We want to find the **best** output

- The **most accurate** output

$\longrightarrow$ impossible! we don't know the reference

$$\hat{Y} = \underset{\tilde{Y}}{\operatorname{argmin}}\ \operatorname{error}(Y, \tilde{Y})$$

- The **most probable** output according to the model

$\longrightarrow$ simple, but not necessarily tied to accuracy

$$\hat{Y} = \underset{\tilde{Y}}{\operatorname{argmax}}\ P(\tilde{Y}|X)$$

- The output with the lowest **Bayes risk**

$\longrightarrow$ which output looks like it has the lowest error?

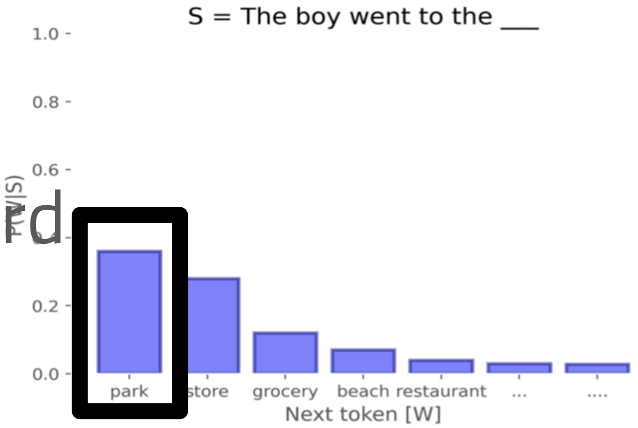
$$\hat{Y} = \underset{\tilde{Y}}{\operatorname{argmin}} \sum_{Y'} P(Y'|X)\operatorname{error}(Y', \tilde{Y})$$

Greedy Search

- One by one, pick the single highest-probability word

$$While\, Y_{j-1} \;!= < STOP >$$
$$Y_j = \color{red}{argmax}\; P(Y_j \mid X, y_1, \ldots y_{j-1})$$

- Not exact, real problems:
  - Will often generate the easy words first
  - Will prefer multiple common words to one rare word



Greedy methods get repetitive

$$Y_j = argmax\ P(Y_j \mid X, y_1, \ldots y_{j-1})$$

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Continuation: The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México…**

Problems w/ Disparate Search Difficulty

$$Y_j = argmax\ P(Y_j \mid X, y_1, \ldots y_{j-1})$$

- Sometimes need to cover specific content, some easy some hard

I	saw	the escarpment
watashi	*mita*	*dangai? zeppeki? kyushamen? iwa?*

- Can cause the search algorithm to select the easy thing first, then hard thing later

watashi wa dangai wo mita (I saw the escarpment)	*watashi ga mita dangai* (the escarpment I saw)

Problems w/ Multi-word Sequences

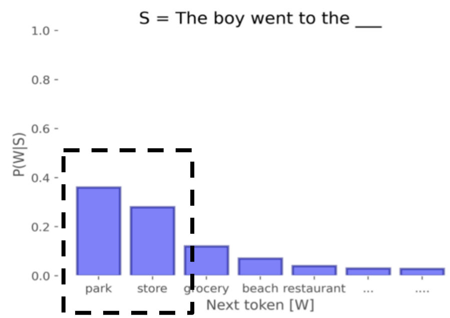
$$Y_j = argmax\ P(Y_j \mid X, y_1, \ldots y_{j-1})$$

Next word	P(next word)
Pittsburgh	0.4
New York	0.3
New Jersey	0.25
Other	0.05

P(Pittsburgh|...) = 0.4

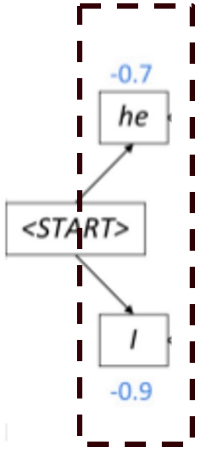
P(New|...) = 0.55

Beam Search



- Instead of picking the highest probability/score, maintain multiple paths (beam size)

- At each time step
  - Expand each path until <STOP>
  - Choose a subset paths from the expanded set



Beam size (k) = 2

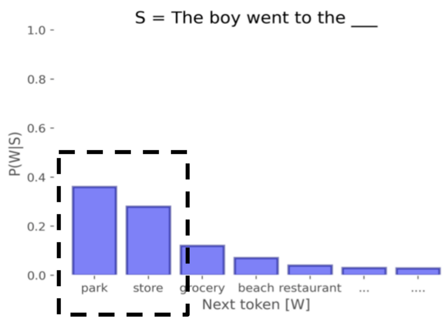
Blue numbers = $score\ (y_1 \dots y_t)$

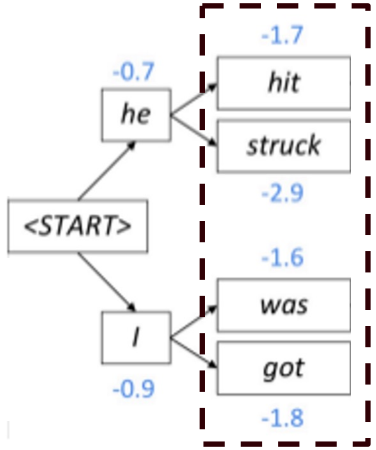
$$= \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \dots y_{i-1}, x)$$

Beam Search

- Instead of picking the highest probability/score, maintain multiple paths (beam size)

- At each time step
  - Expand each path until <STOP>
  - Choose a subset paths from the expanded set





Beam size (k) = 2

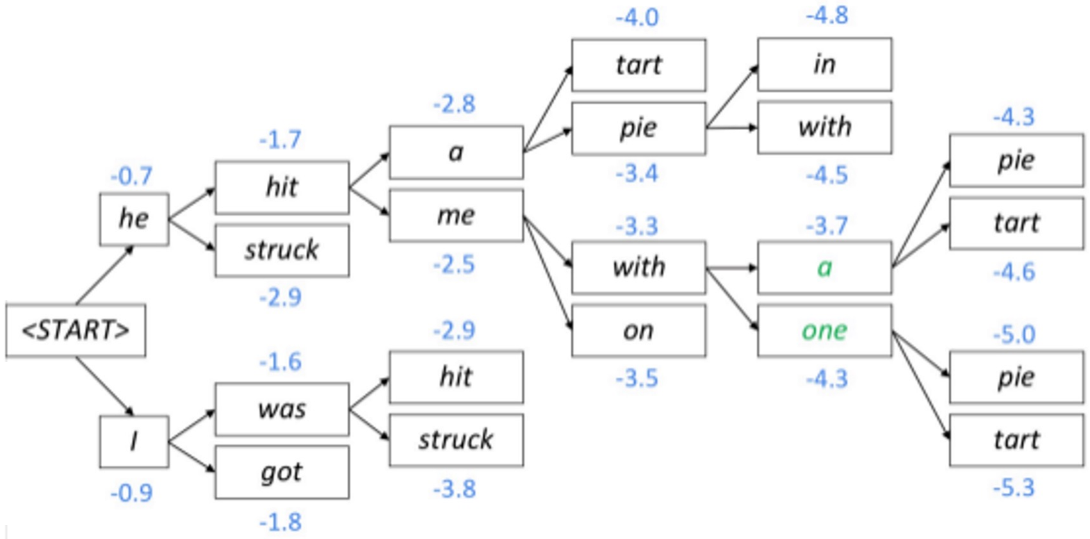
Blue numbers = $score\ (y_1 \dots y_t)$

$$= \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \dots y_{i-1}, x)$$

Beam Search

- Instead of picking the highest probability/score, maintain multiple paths (beam size)

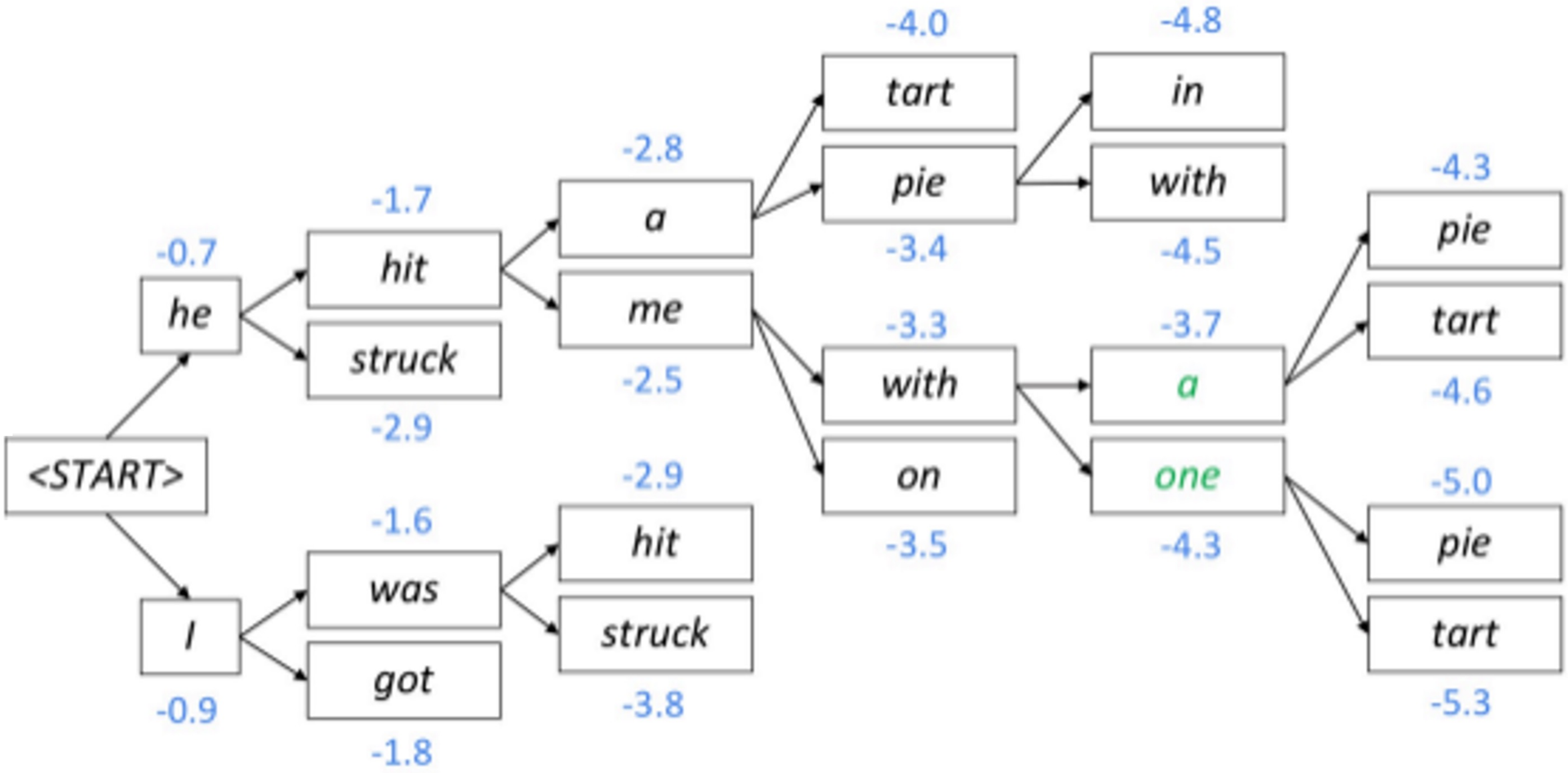
- At each time step
  - Expand each path until <STOP>
  - Choose a subset paths from the expanded set

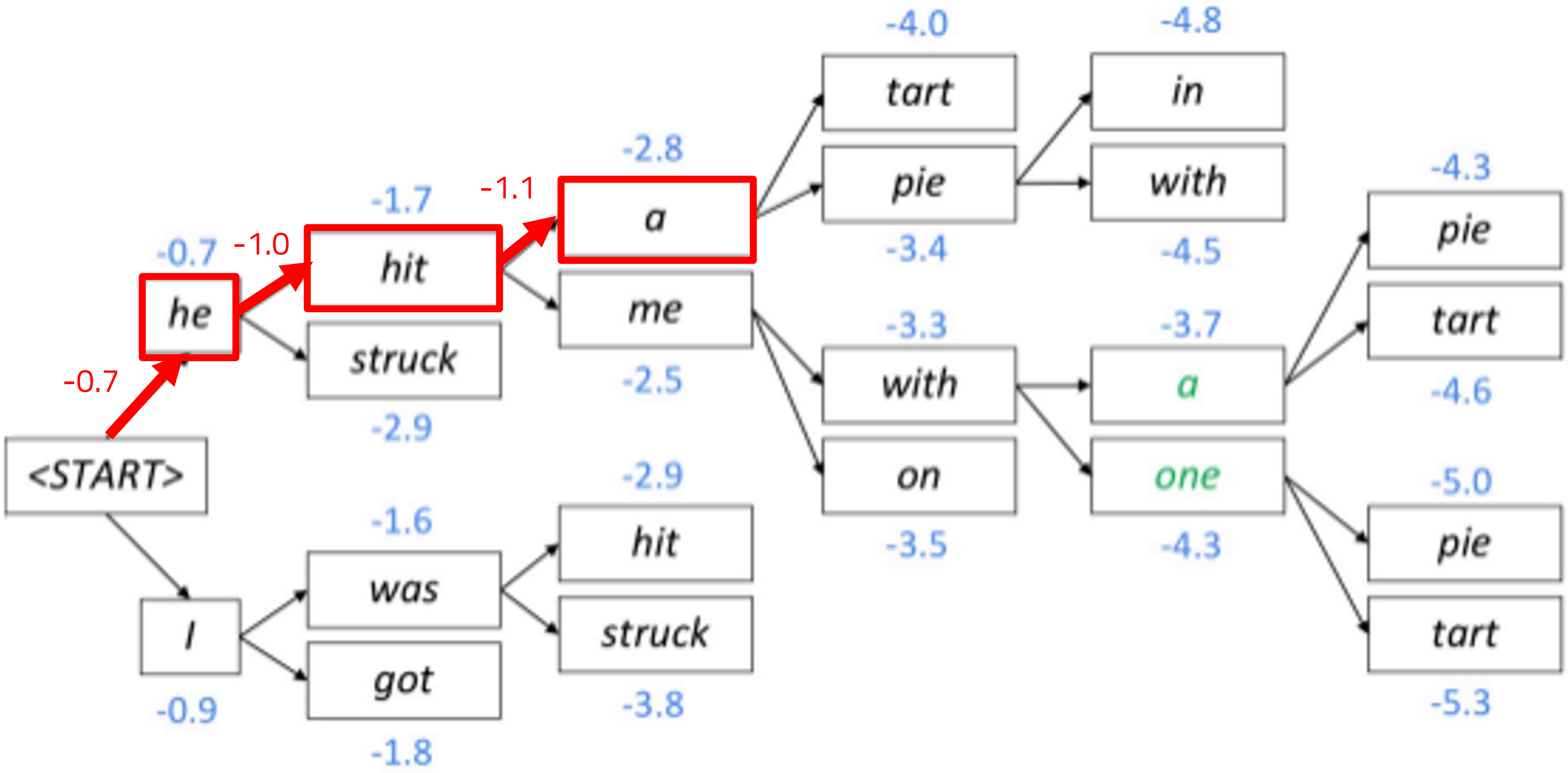


Beam size (k) = 2

Blue numbers = $score\ (y_1 \ldots y_t)$

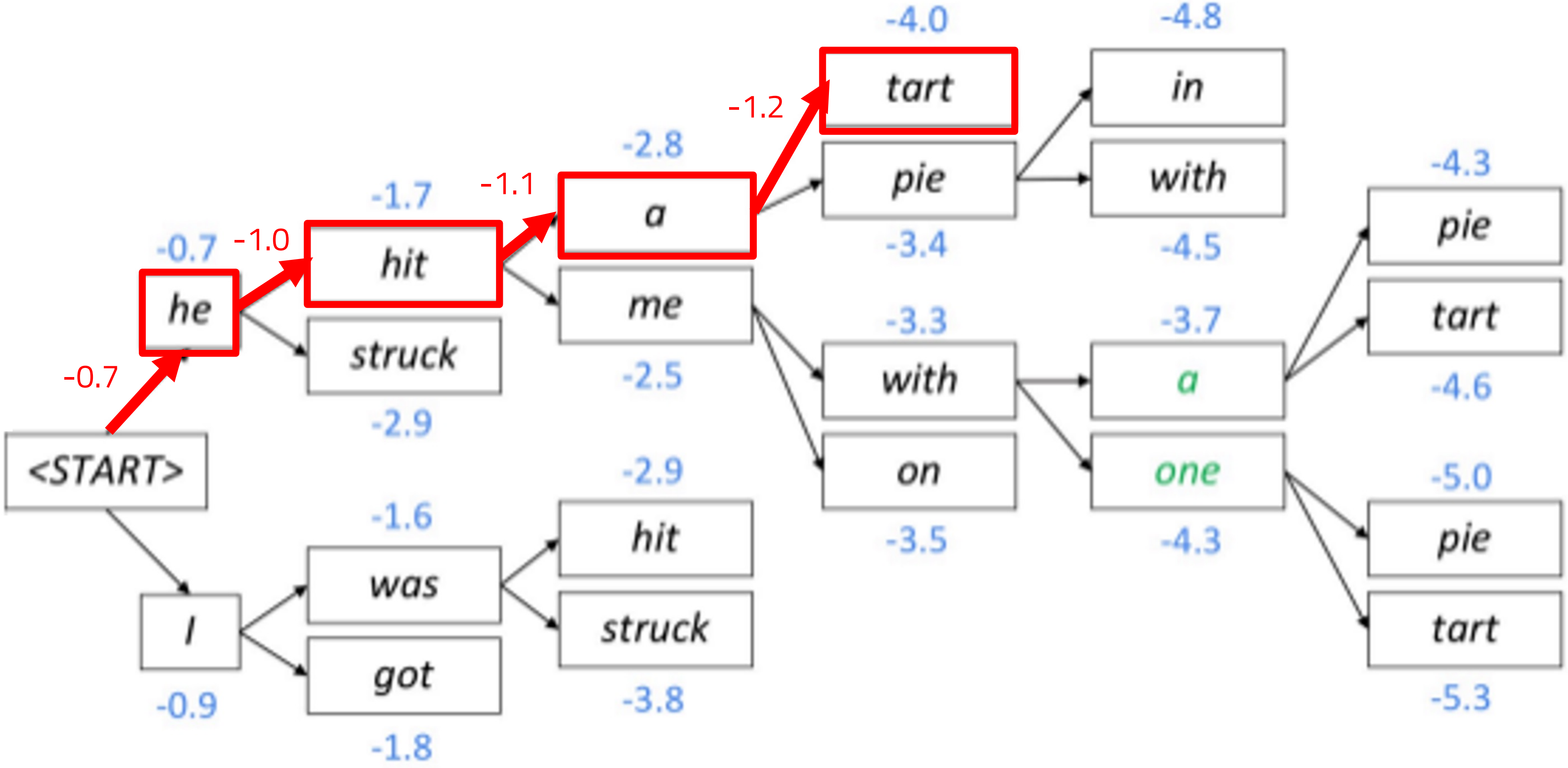
$$= \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \ldots y_{i-1}, x)$$

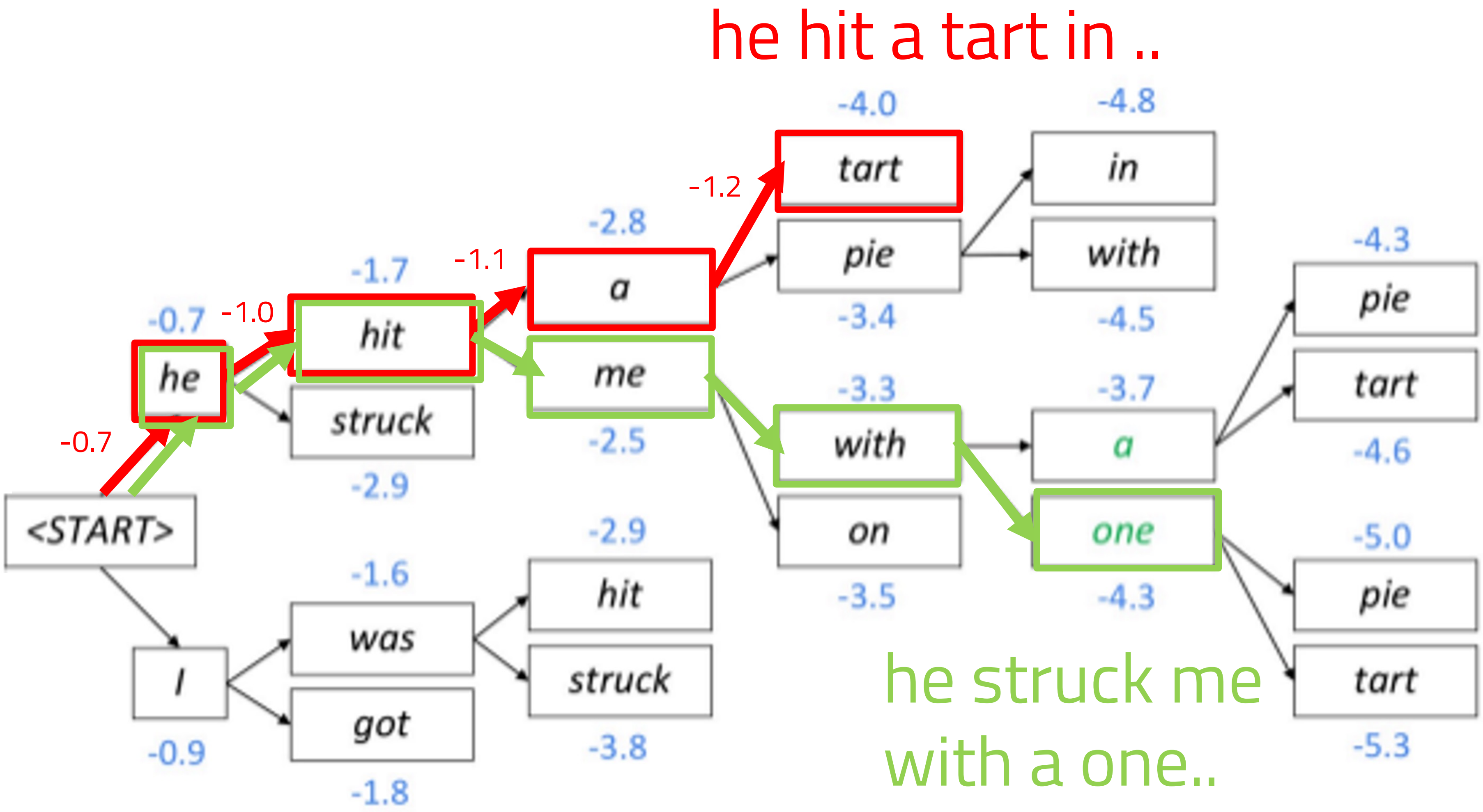

<START>
-0.7
he
I
-0.9
-1.7
hit
struck
-2.9
-1.6
was
got
-1.8
-2.8
a
me
-2.5
-2.9
hit
struck
-3.8
-4.0
tart
pie
-3.4
-3.3
with
on
-3.5
-4.8
in
with
-4.5
-3.7
a
one
-4.3
-4.3
pie
tart
-4.6
-5.0
pie
tart
-5.3



<START>
-0.7
he
-0.7
-1.0
hit
-1.7
struck
-2.9
-1.1
a
-2.8
me
-2.5
I
-0.9
was
-1.6
got
-1.8
hit
-2.9
struck
-3.8
tart
-4.0
pie
-3.4
with
-3.3
on
-3.5
in
-4.8
with
-4.5
a
-3.7
one
-4.3
pie
-4.3
tart
-4.6
pie
-5.0
tart
-5.3

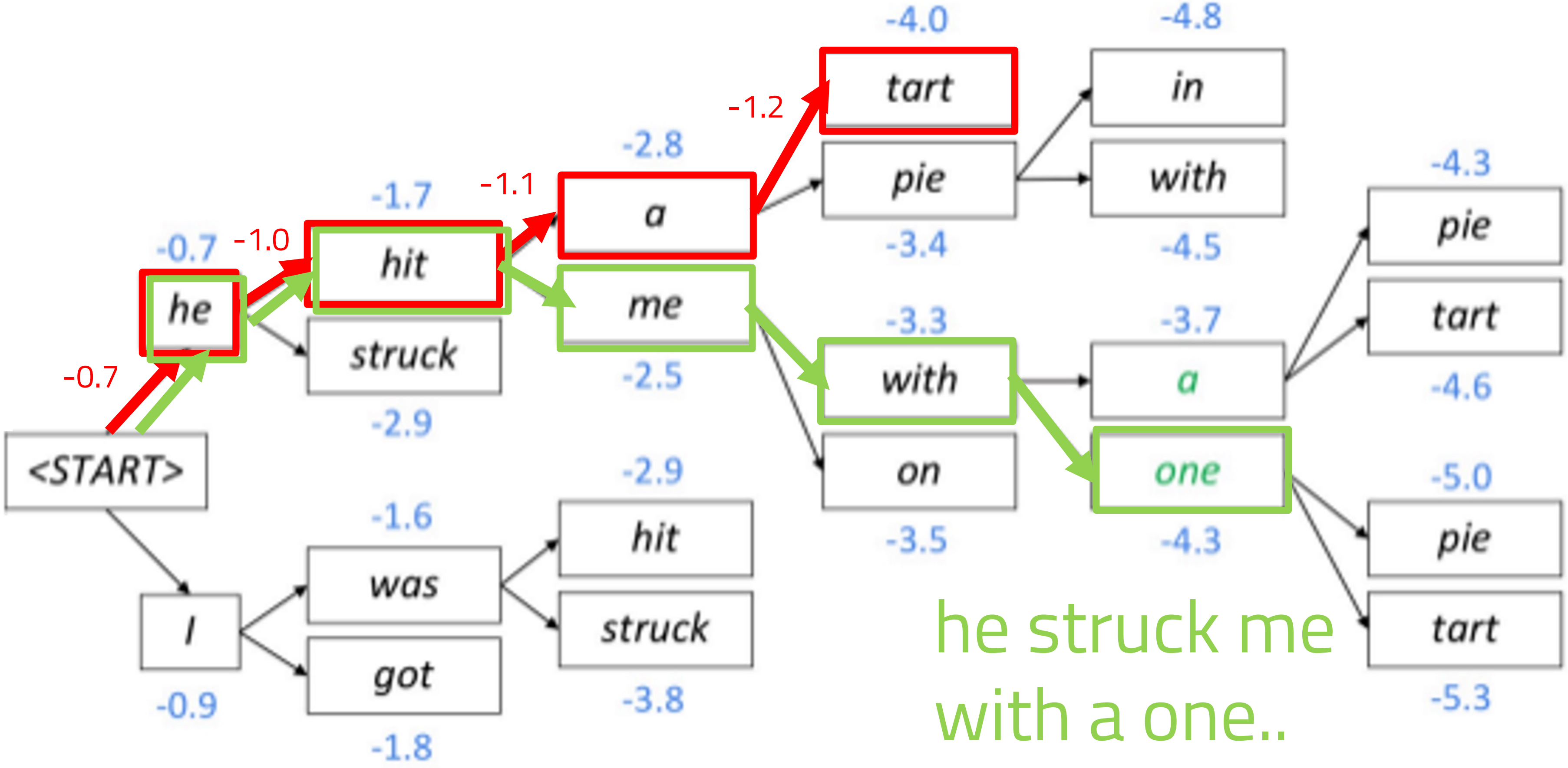
he hit a tart in ..




he hit a tart in ..
-4.0
-4.8
-1.2
tart
in
-2.8
pie
with
-1.7
-1.1
-3.4
-4.5
-4.3
-0.7
-1.0
a
pie
hit
he
me
-3.3
-3.7
tart
-0.7
struck
-2.5
with
a
-4.6
-2.9
<START>
-2.9
on
one
-5.0
-1.6
hit
-3.5
-4.3
pie
was
struck
tart
I
got
-3.8
-5.3
-0.9
-1.8
he struck me
with a one..

$$score\ (y_1 \ldots y_t) = \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \ldots y_{i-1}, x) = -4.8$$

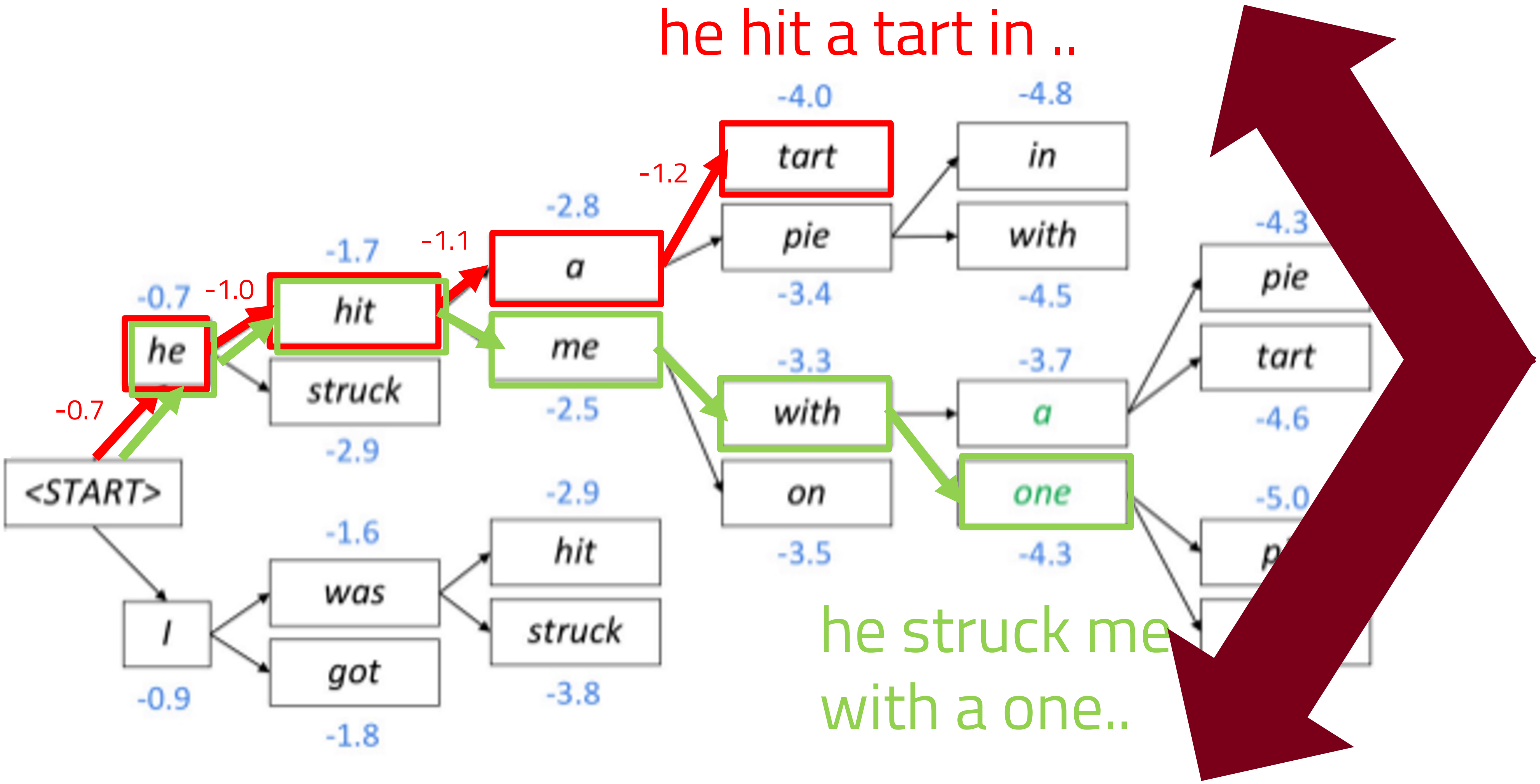
he hit a tart in ..



he struck me with a one..

$$score\ (y_1 \ldots y_t) = \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \ldots y_{i-1}, x) = -4.3$$

$$score\ (y_1 \ldots y_t) = \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \ldots y_{i-1}, x) = -4.8$$

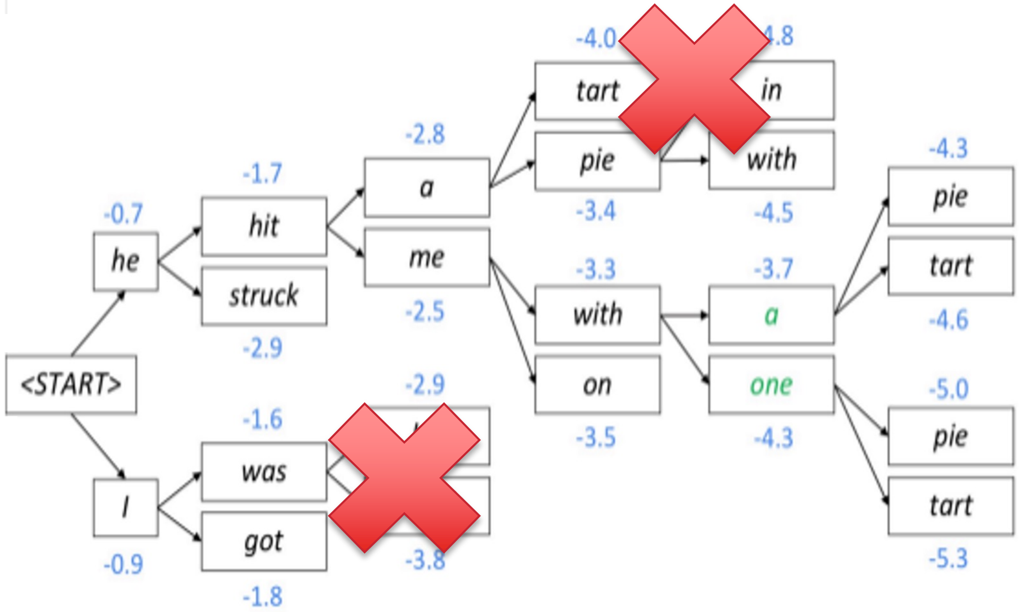


$$score\ (y_1 \ldots y_t) = \prod_{i=1}^{t} \log P_{LM}(y_i | y_1 \ldots y_{i-1}, x) = -4.3$$

Basic Pruning Methods (Steinbiss et al. 1994)

How to select which paths to keep expanding?

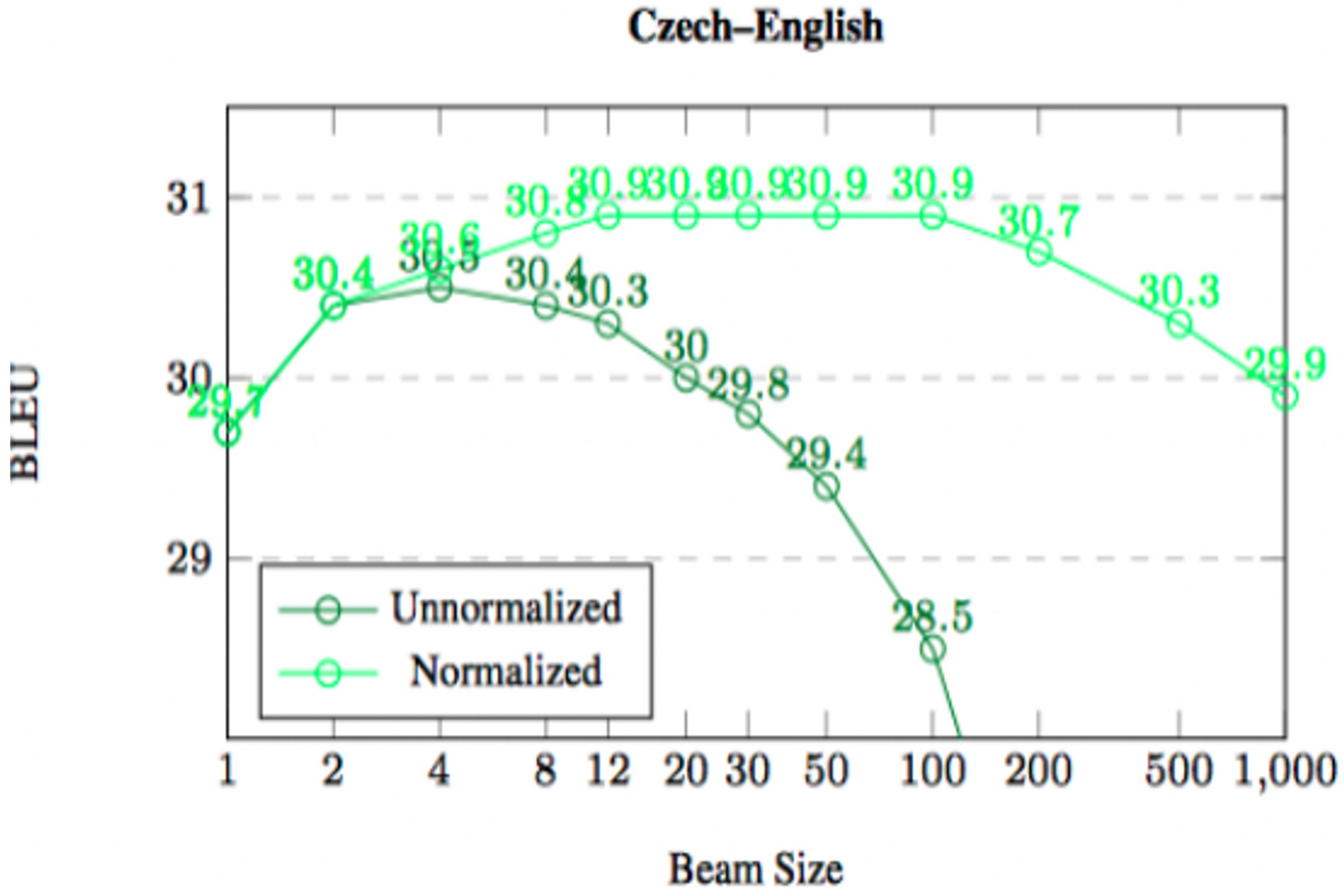
- **Histogram Pruning**: keep exactly k hypotheses at every time step
- **Score Threshold Pruning**: keep all hypotheses where score is within a threshold α of best score s_1
- **Probability Mass Pruning:** keep all hypotheses up until probability mass α



Better Search can Hurt Results! (Koehn and Knowles 2017)

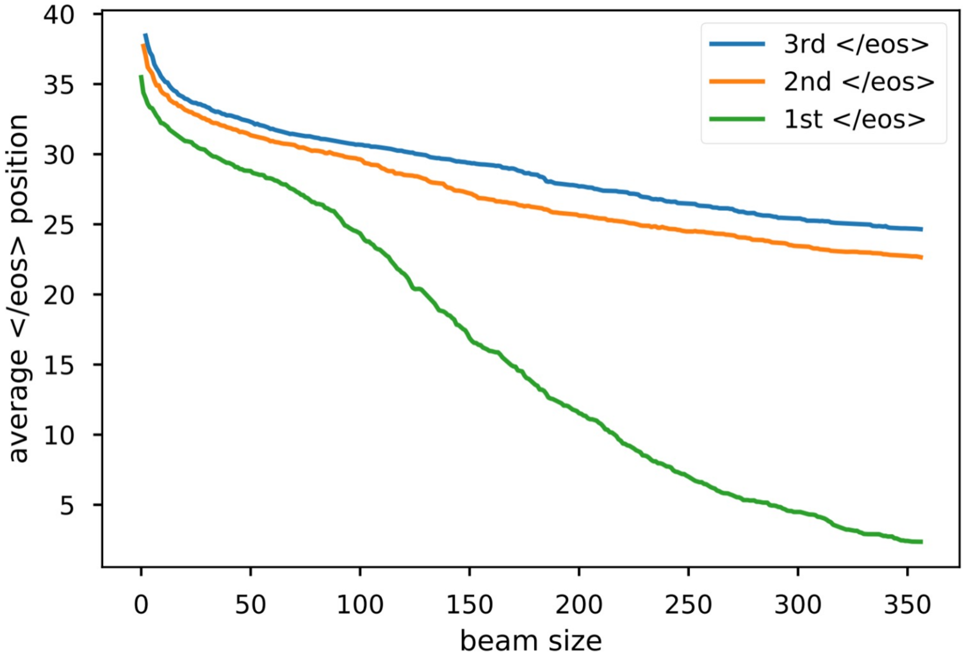
- Better search (=better model score) can result in worse BLEU score!

- Why? Model errors!

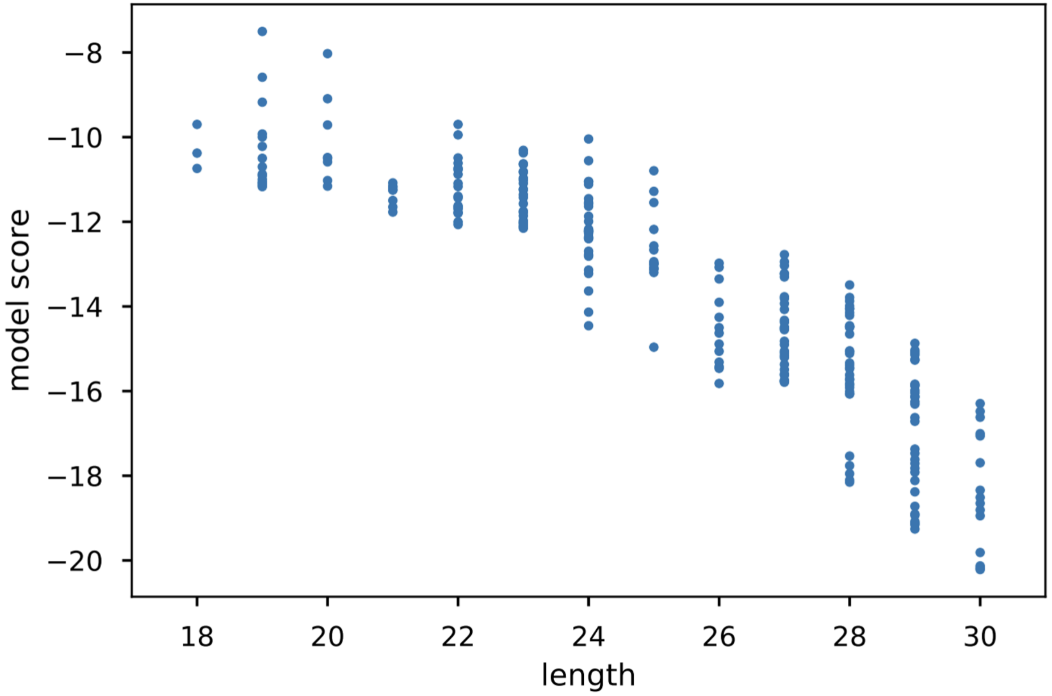


Beam Search Curse (Yang et al. 2018)

- As beam size increases, it becomes easier for the search algorithm to find the **</eos>** symbol.



- Then, shorter candidates have clear advantages w.r.t. model score.



Fixing Model Errors in Search

A Typical Model Error: Length Bias

- In many tasks (e.g. Machine translation), the output sequences will be of variable length

- Running beam search may then favor short sentences

Length Normalization

- ❑ Beam search may then favor short sentences
- ❑ Normalize by the length, dividing by |Y| to prioritize longer sentences.

(Cho et al. 2014)

$$\frac{1}{T_y^{\alpha}} \quad argmax_y \sum_{j=1}^{T_y} \log P(y_j \,|X, y_1, \ldots y_{j-1})$$

$$\alpha = [0, 1.0]$$

(Wu et al. 2016)

$$\frac{(5+1)^{\alpha}}{(5+|Y|)^{\alpha}}$$

Sampling

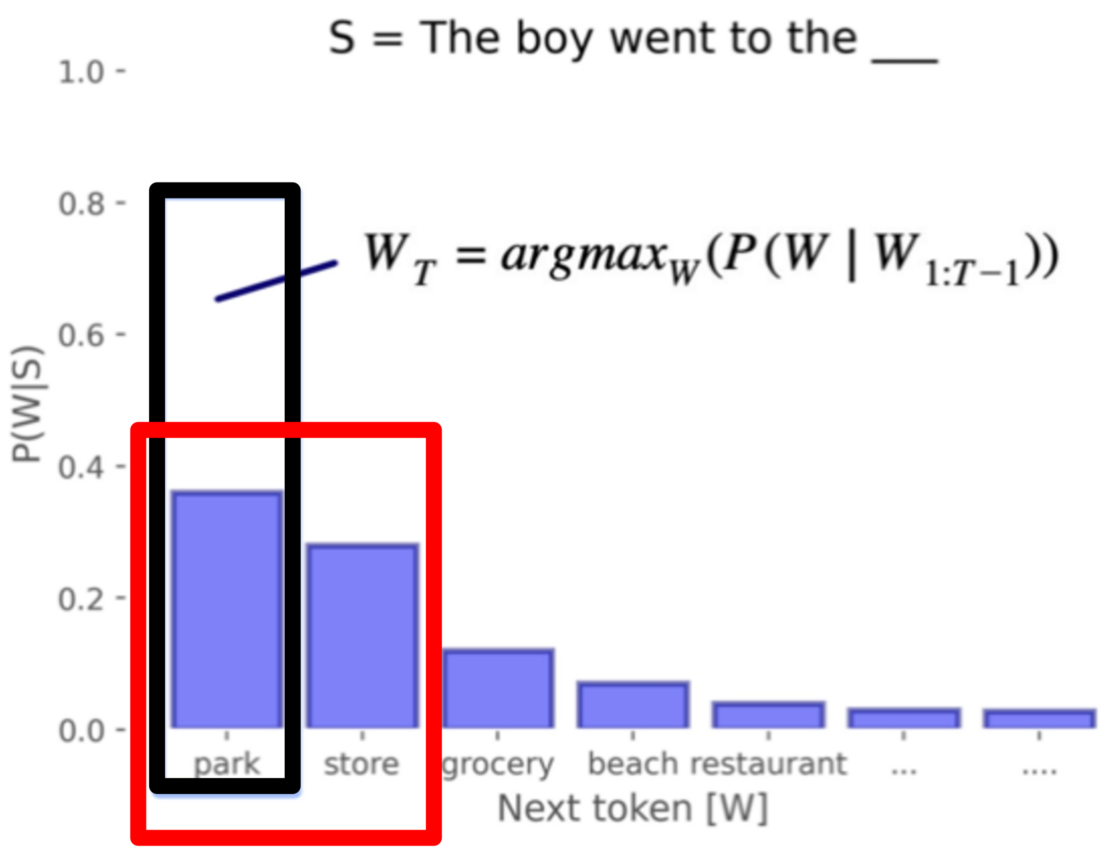
How are you doing?

I'm good! How about you?

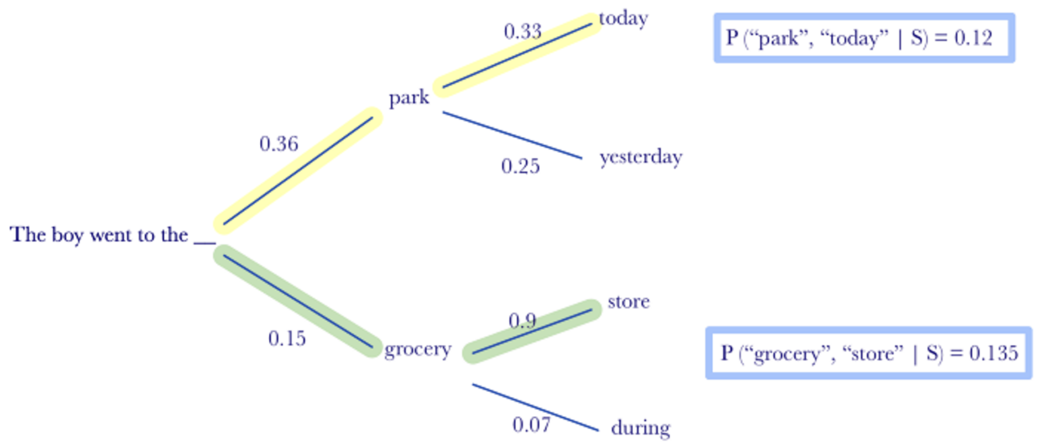
So so..

It was a hard day for me.

Recap: Greedy/Beam Search (w/o Sampling)

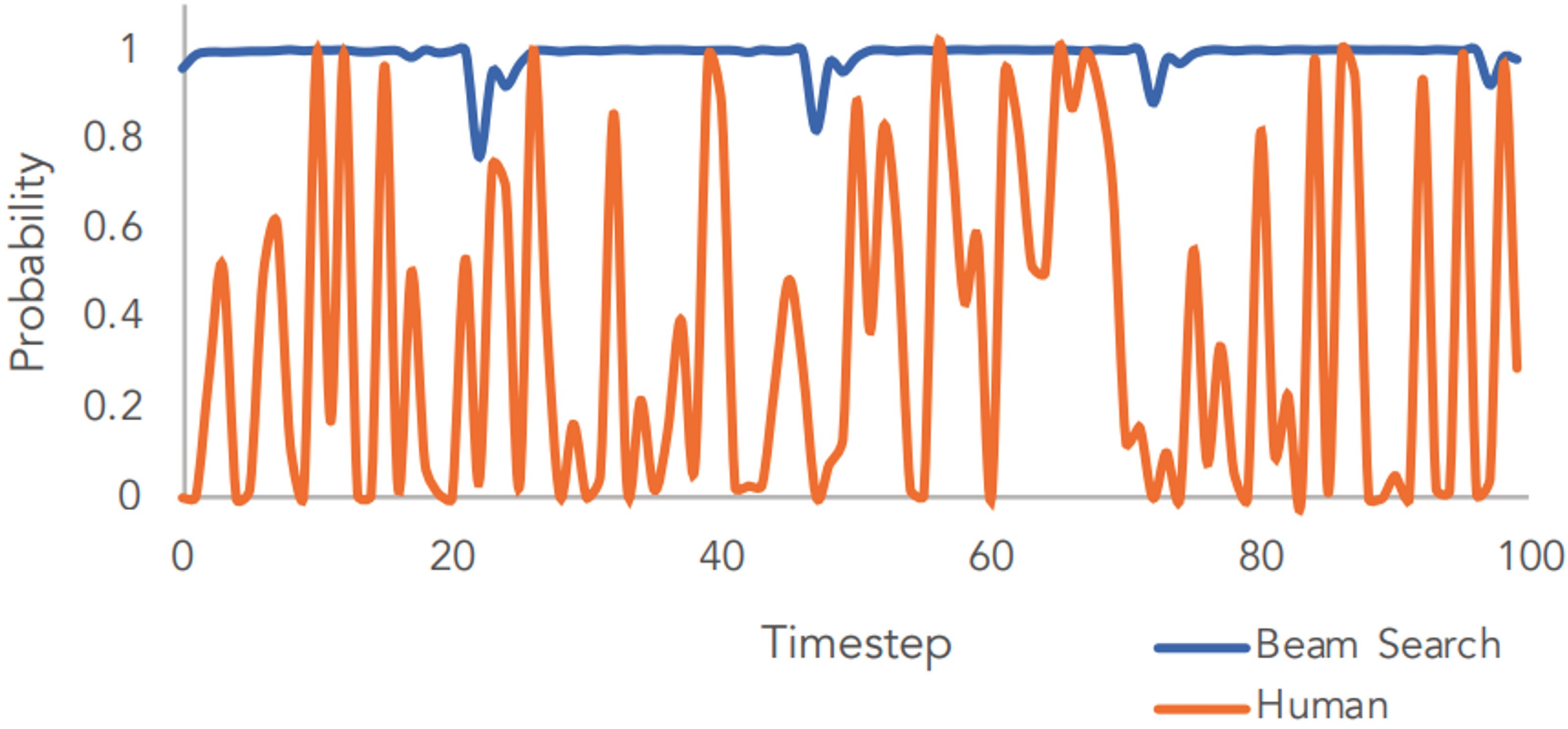


Beam size (k) = 2



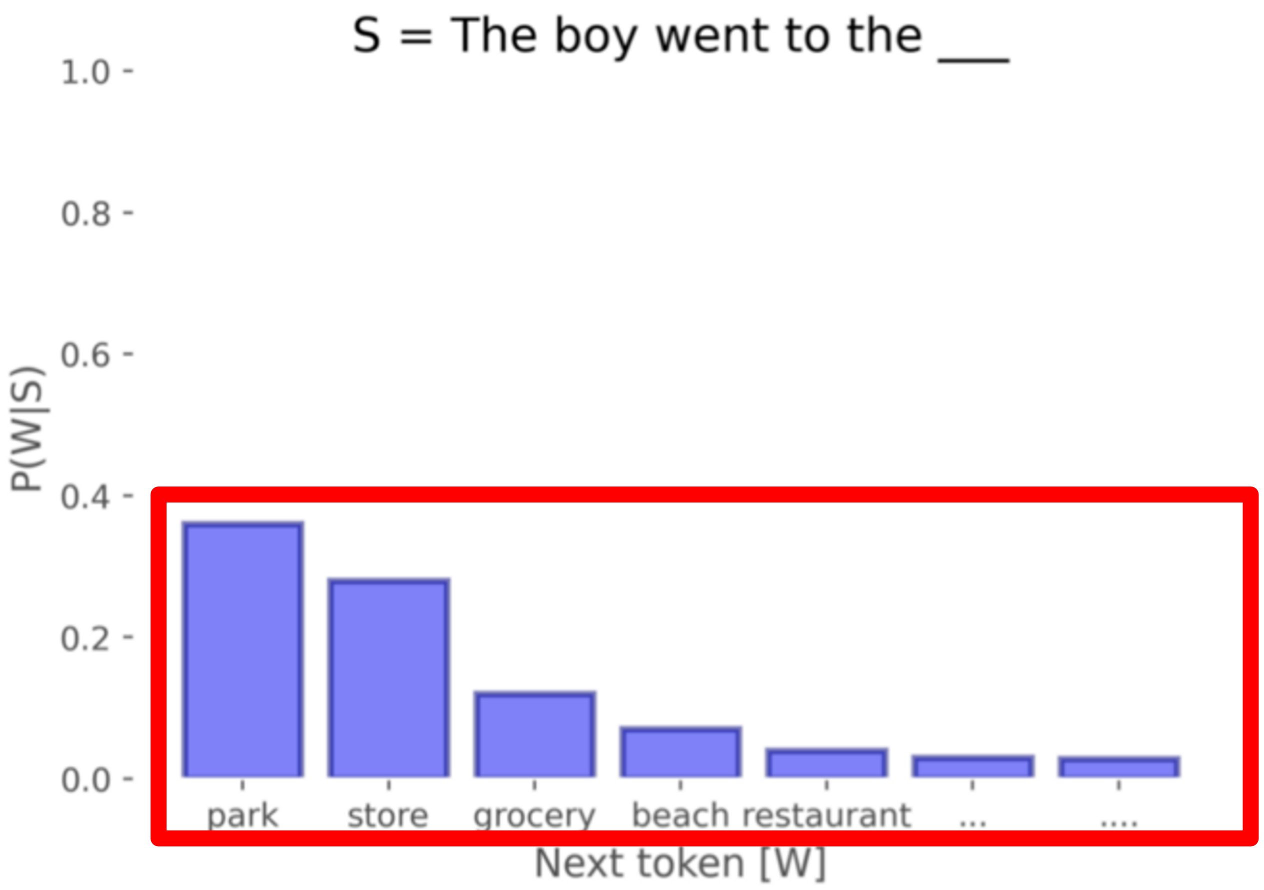
Deterministic beam search:
I went into town on Saturday morning because...
-> I was going to go to the gym and I was going to go to the gym and I was going to go to the ..

https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3



(Holtzman et al. 2020)

Decoding with Ancestral/Multinomial Sampling



Multinomial Sampling:
I went into town on Saturday morning because...
-> I have to wear suits and collared in the South Bay. This was shocking!" "This is our city. First of all, I'm strange in the name of Santa, Howard Daniel, and

https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Repetition

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

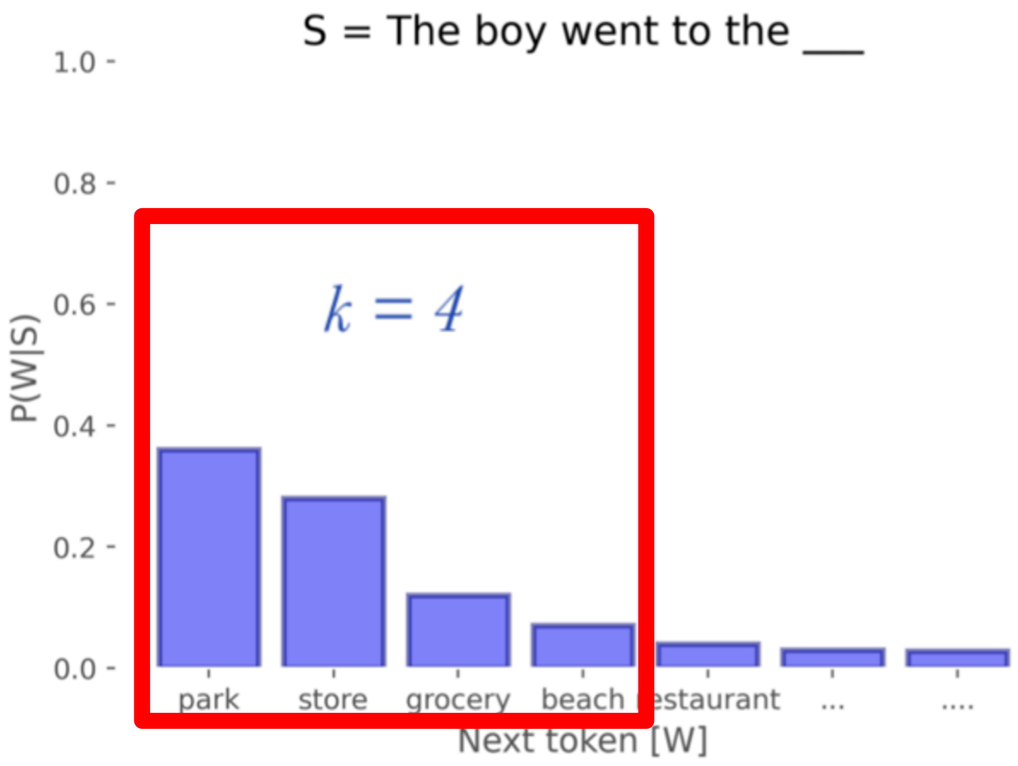
Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Repetition

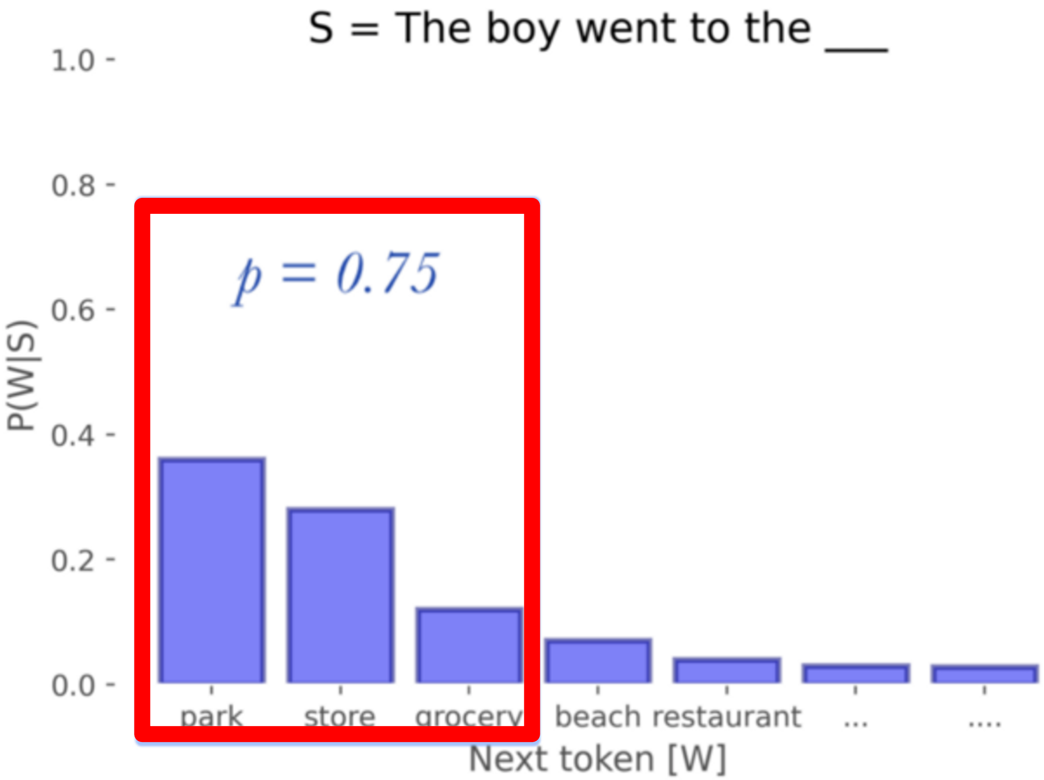
Incoherence

Top-k Sampling



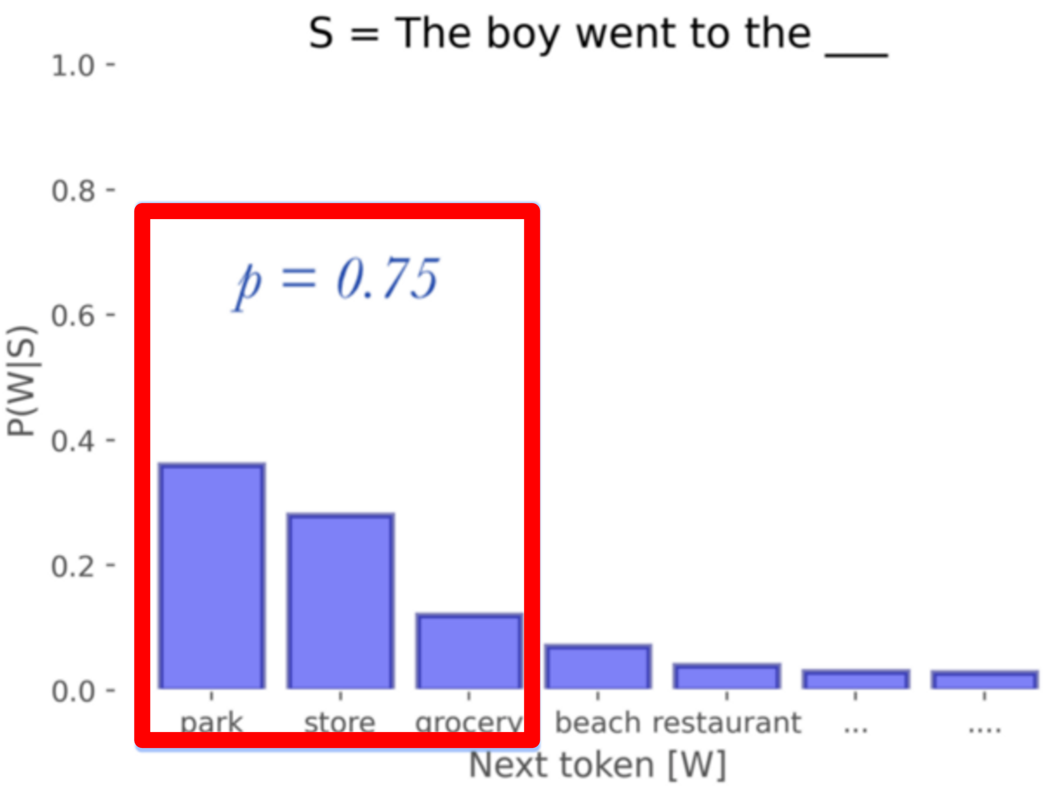
- ❑ Only sample from the *k* most probable tokens, by redistributing the PMF over the top-k tokens
- ❑ But, picking a good value of *k* can be difficult as the distribution or words is different for each step.
  - Increase k for more diverse/risky outputs
  - Decrease k for more generic/safe outputs

Top-p Sampling (or Nucleus Sampling) (Holtzman et al. 2020)



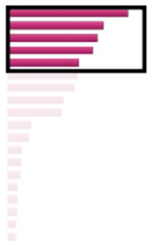
- Another way to exclude very low probability tokens is to include the most probable tokens that make up the "nucleus" of the PMF
  - the sum of the most probable tokens just reaches P

Top-p Sampling (or Nucleus Sampling) (Holtzman et al. 2020)



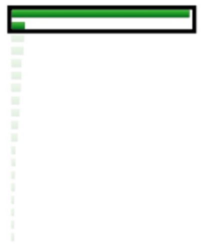
- Flexible as the distribution changes, allowing the size of the filtered words to expand and contract when it makes sense.

$P_t^1(y_t = w \mid \{y\}_{<t})$



$P_t^2(y_t = w \mid \{y\}_{<t})$

$P_t^3(y_t = w \mid \{y\}_{<t})$



Cautions about Sampling-based Search

- Is sampling necessary for diversity?
  - questionable, we could do diverse beam search instead.
- Results are inconsistent from run-to-run:
  - need to consider variance from this in reporting
  - (in addition to variance in training and data selection)
- Conflates model and search errors:
  - if you make a better model you might get worse results, because the search algorithm can't find the outputs your model likes

Decoding: Takeaways

- Many problems in neural NLG are not really problems with our learned language model probability distribution, but problems with the decoding algorithm
- Human language production is a subtle presentation of information and can't be modeled by simple properties like probability maximization.
- Different decoding algorithms can allow us to inject biases that encourage different properties of coherent natural language generation
- Some of the most impactful advances in NLG of the last few years have come from simple but effective modifications to decoding algorithms

HW3 Teaser

Rubric (25 points)

- **Task 1 : Implementation of Decoding Algorithms (12 points)**
  - Full Marks
    All 4 decoding algorithms are not implemented: (-2 per algorithm not implemented)
  - Parameters of the algorithms not mentioned (-1)
  - Prompt is not constant across all algorithms (-1)
  - Perplexity or Likelihood of each output not calculated (-2)
  - No Marks
- **Task 2: Decoding for extrinsic evaluation (2 points)**
  - Full Marks, the correct implementation of models (loading and decoding) and Nx6 spreadsheet correctly generated
  - XSUM dataset is used (-1)
  - Output summary generated from the train set (-1)
  - Minor mistakes in results or code
  - Major mistakes in results or code
  - No Marks
- **Task 3.1 Automatic Evaluation (4 points)**
  - Full Marks
  - If only content overlap metrics or only model-metric metrics are implemented (-2)
  - Metrics not calculated between reference text and decoded outputs (-1)
  - Average metric score across all samples not reported (-1)
  - No Marks (no automatic evaluation done)
- **Task 3.2 Human Evaluation (5 points)**
  - Full Marks
  - At least 2-3 aspects of human evaluation for the target task devised (-1)
  - Reasoning not given behind choice of aspects (-1)
  - Majority/Average voting is not implemented (-1)
  - Difference between human and automatic evaluation is not highlighted (-1)

The auto-regressive language models (e.g., GPT3 [BMR+20]) trained on human-written text can produce natural text as humans do. In this homework, you will implement and use various decoding algorithms, generate text using the pre-trained large language models (LLMs) on different generation tasks, evaluate the output text, and justify the limitations of current decoding methods. The lead TA for this assignment is Karin de Langis (dento019@umn.edu). Please communicate with the lead TA via Slack, email, or during office hours.

This assignment assumes that you have covered most of the search algorithms and evaluation metrics in text generation on Language Models: Search Algorithms and Language Models: Search in Training, Evaluation. Please read the reading materials and lecture notes if you missed class.

In this homework, you don't actually need to implement anything from scratch; instead, you will make a complete pipeline of text generation research including task selection, decoding, automatic evaluation, human evaluation, and analysis of output text. Please follow the steps below, report outputs from the **Tasks** of each step, and submit the spreadsheet, codebase, and report.

Step 1: Trying out different decoding algorithms using HuggingFace

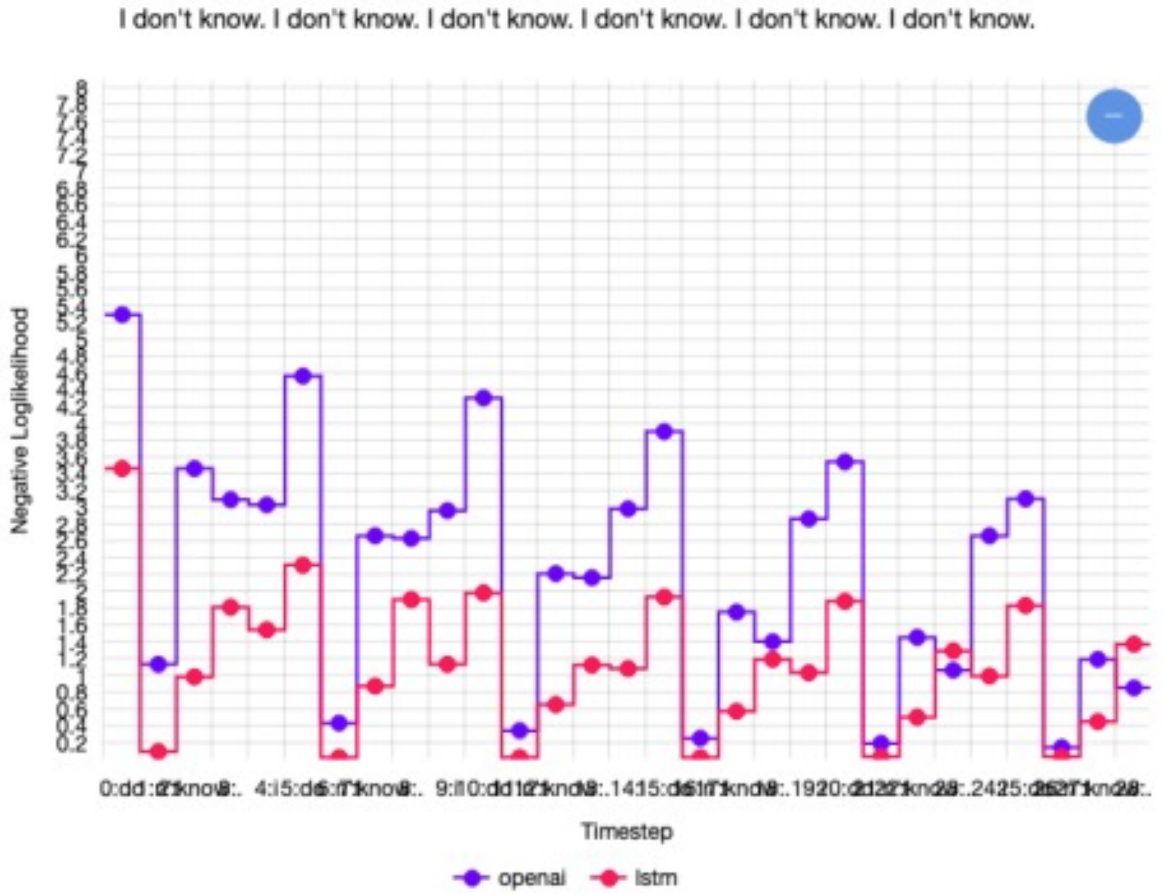
```
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("gpt2")
model = AutoModelForCausalLM.from_pretrained("gpt2")

prompt = "Today I believe we can finally"
input_ids = tokenizer(prompt, return_tensors="pt").input_ids

/* generate up to 30 tokens */
outputs = model.generate(input_ids, do_sample=False, max_length=30)
tokenizer.batch_decode(outputs, skip_special_tokens=True)

/* step 1 */
outputs1 = model.YourDecodingAlgorithmToImplement1(input_ids)
outputs2 = model.YourDecodingAlgorithmToImplement2(input_ids)
..

```

You can first go to (HuggingFace API on text generation) and run an example script to generate text. For instance in the example above, once you load pre-trained autoregressive language models like GPT2 [RWC+19], the HuggingFace library allows you to select a variety of decoding algorithms.

Search in Training

Diversity Issues (Holtzman et. al., 2020)

- ❑ Maximum Likelihood Estimation discourages diverse text generation



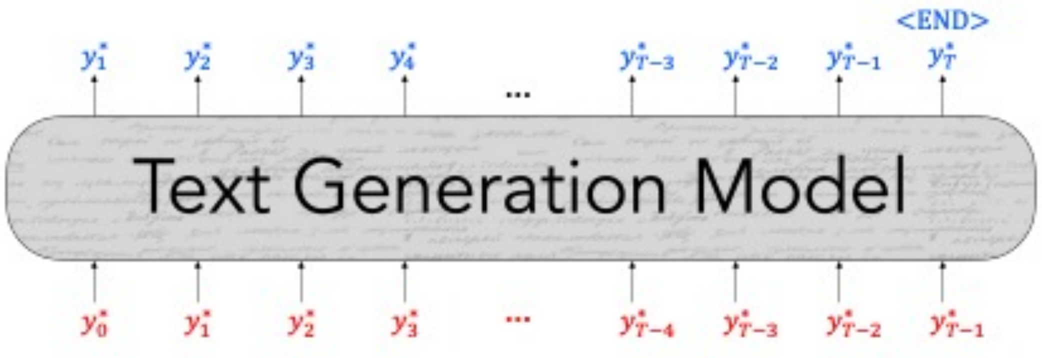
Why? Exposure Bias

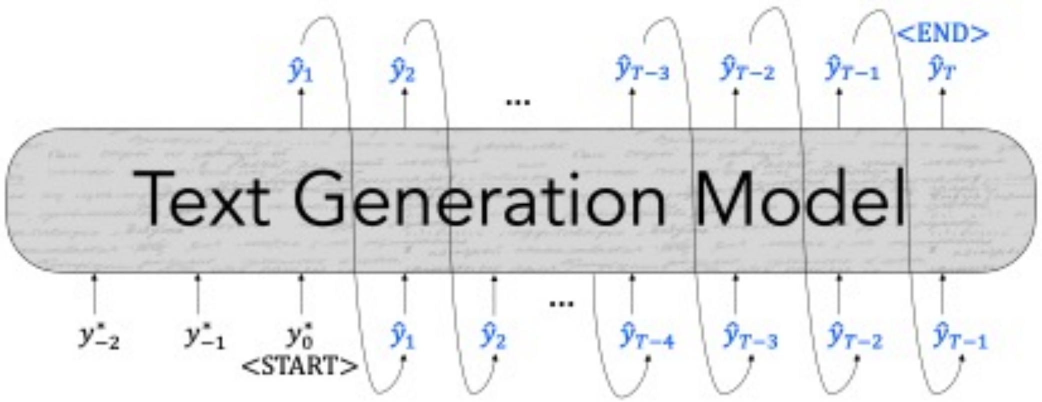
- Training with teacher forcing leads to exposure bias at generation time
  - During training, our model's inputs are gold context tokens from real, human-generated texts

$$\mathcal{L}_{MLE} = -\log P(y_t^* | \{y^*\}_{<t})$$

  - At generation time, our model's inputs are previously–decoded tokens

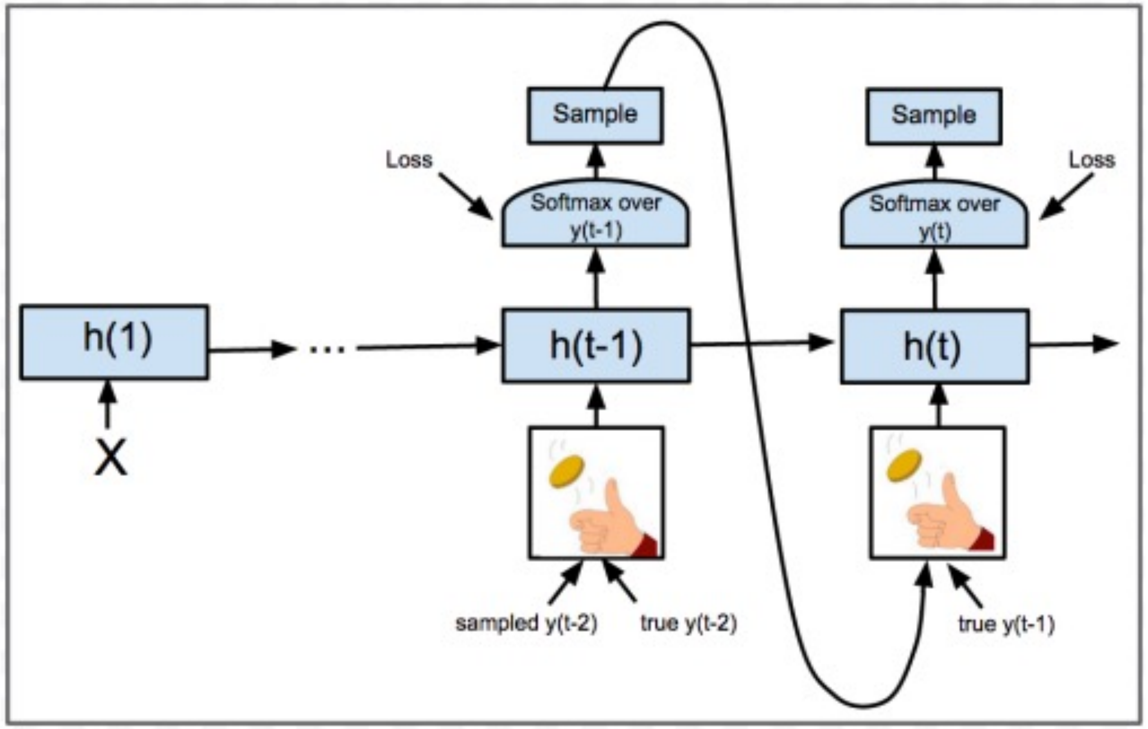
$$\mathcal{L}_{dec} = -\log P(\hat{y}_t | \{\hat{y}\}_{<t})$$





Fix Exposure Bias: Scheduled sampling

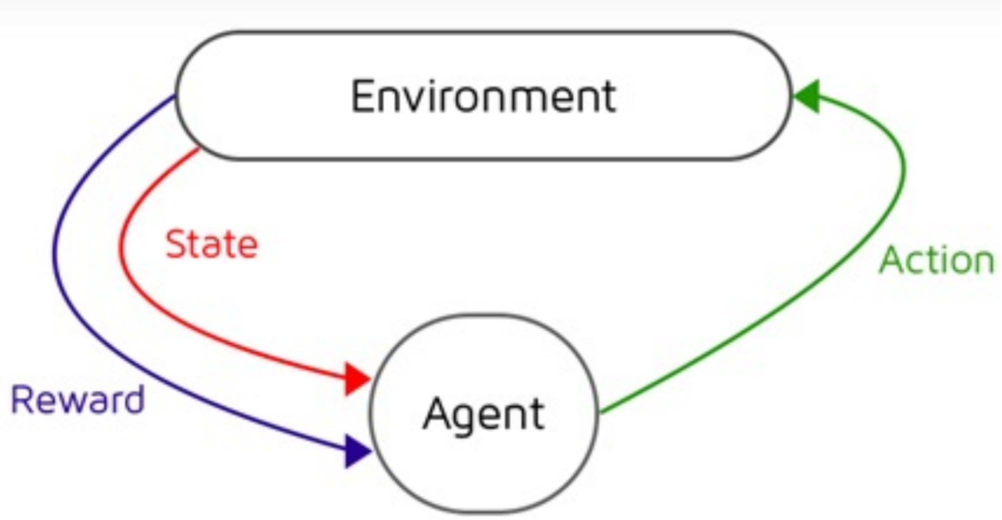
- With some probability p, decode a token and feed that as the next input, rather than the gold token.
- Increase *p* over the course of training
- Leads to improvements in practice, but can lead to strange training objectives
- Also called teacher forcing



(Bengio et al., 2015)

Fix Exposure Bias: Reinforcement Learning

- Cast your text generation model as a Markov decision process
  - **State** s is the model's representation of the preceding context
  - **Actions** a are the words that can be generated
  - **Policy** π is the decoder
  - **Rewards** r are provided by an external score
- Learn behaviors by rewarding the model when it exhibits them
- Use REINFORCE or similar; it's difficult because huge branching factor/search space



MIXER: Sequence-level training with REINFORCE

Ranzato et al., 2016

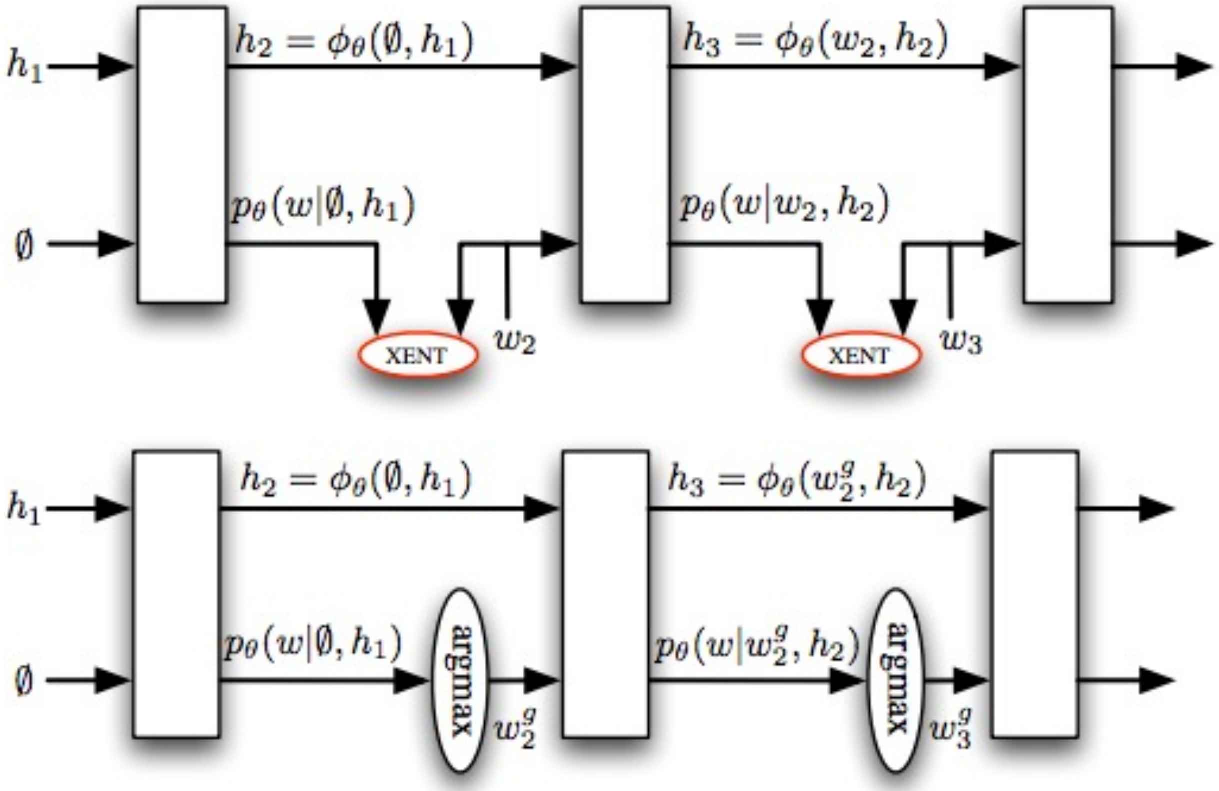
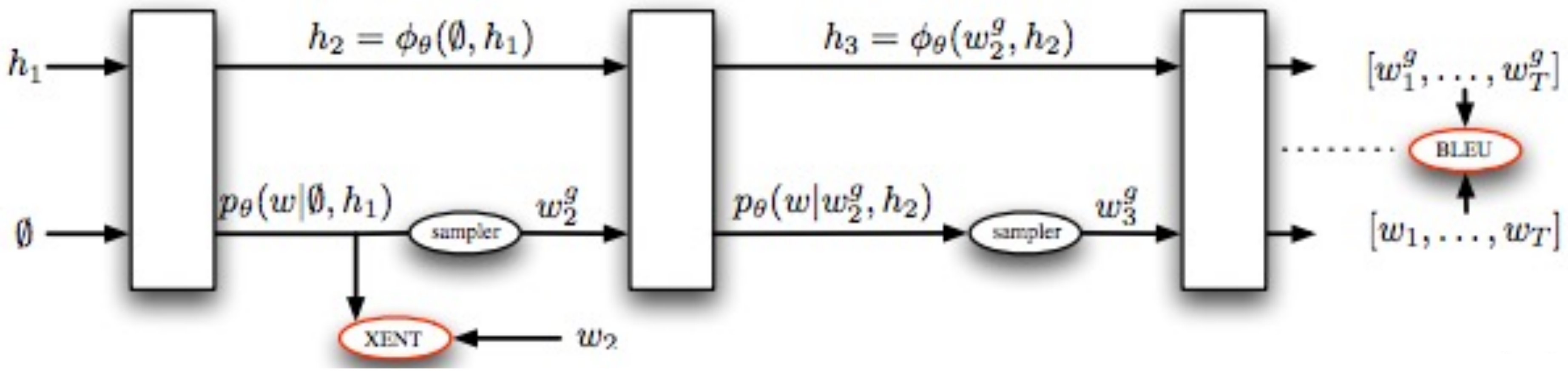


Figure 1: RNN training using XENT (top), and how it is used at test time for generation (bottom).

MIXER: Sequence-level training with REINFORCE

Ranzato et al., 2016



$$\mathcal{L} = \mathcal{L}_{MLE} + \alpha \mathcal{L}_{RL}$$

$$\mathcal{L}_{RL} = -\sum_{t=1}^{T} (r(\hat{y}_t) - b) \log P(\ldots)$$

TASK	XENT	DAD	E2E	MIXER
summarization	13.01	12.18	12.78	**16.22**
translation	17.74	20.12	17.77	**20.73**
image captioning	27.8	28.16	26.42	**29.16**

MIXER seems to be a useful, agnostic trick to improve MT results, but did not see wide usage ~ perhaps due to unstability of REINFORCE

Reward Estimation

$$\mathcal{L}_{RL} = -\sum_{t=1}^{T} (r(\hat{y}_t) - b) \log P(\ldots)$$

- How should we define a reward function? Just use your evaluation metric!
  - BLEU (machine translation; Ranzato et al., ICLR 2016; Wu et al., 2016)
  - ROUGE (summarization; Paulus et al., 2018; Celikyilmaz et al., 2018)
  - CIDEr (image captioning; Rennie et al., CVPR 2017)
  - SPIDEr (image captioning; Liu et al., ICCV 2017)
- Be careful about optimizing for the task as opposed to "gaming" the reward!
  - Evaluation metrics are merely proxies for generation quality!
  - "*even though RL refinement can achieve better BLEU scores, it barely improves the human impression of the translation quality*" – Wu et al., 2016

Reward Estimation

- What behaviors can we tie to rewards?
  - Sentence simplicity (Zhang and Lapata, EMNLP 2017)
  - Temporal Consistency (Bosselut et al., NAACL 2018)
  - Cross-modality consistency in image captioning (Ren et al., CVPR 2017)
  - Utterance Politeness (Tan et al., TACL 2018)
  - Paraphrasing (Li et al., EMNLP 2018)
  - Sentiment (Gong et al., NAACL 2019)
  - Formality (Gong et al., NAACL 2019)
- If you can formalize a behavior as a Python function (or train a neural network to approximate it!), you can train a text generation model to exhibit that behavior!

Search in Training: Takeaways

- Teacher forcing is still the main algorithm for training text generation models
- Diversity is an issue with sequences generated from teacher forced models
- Exposure bias causes text generation models to lose coherence easily
- Training with RL can allow models to learn behaviors that are challenging to formalize
  - But learning can be very unstable!
  - chatGPT: advanced RL algorithms (e.g., PPO) for better human alignment with human feedback

Other techniques not covered

- ❑ Decoding time control for controllable text generation (e.g., PPLM)
- ❑ Multi-attribute control using RL (will be covered)
- ❑ Unlikelihood training
- ❑ Data augmentation for reducing the exposure bias
- ❑ Retrieval-augmented Generation (RAG)
- ❑ Retrieval based generation (e.g., KNN Language Models)
- ❑ Instruction tuning and human feedback learning (will be covered)

…

Questions?