Lecture 17: Language Generation

Instructor: Swabha Swayamdipta

USC CSCI 544 Applied NLP

Oct 24, Fall 2024

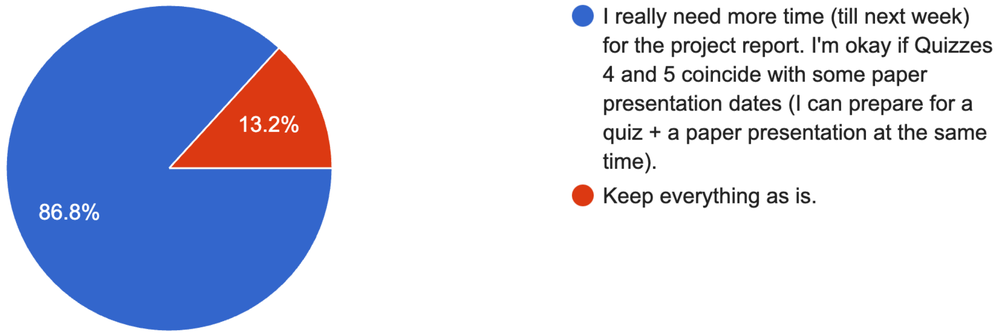
Some slides adapted from Dan Jurafsky and Chris Manning

Announcements

- Tue, 10/29 - Project proposal
- Thu, 11/7 - Quiz 4
- Tue, 11/12 Quiz 5
- Quizzes 4 and 5 - all topics after the midterms
  - Consider these as practice tests for final exams
- Thu, 11/14 Guest lecture by Prof. Willie Neiswanger on 11/14 + HW4 due
- Thu, 10/31 onwards: Paper presentations and project presentations
  - Also two remaining lectures on 10/31 and 11/5

If we delay the project proposal date deadline, this will affect the timeline for quizzes, and paper presentations. Which option is more preferable to you?

121 responses



Lecture Outline

- Announcements
- Recap: The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
- Pre-training Encoder-Decoder Models
- Tokenization
- Natural Language Generation

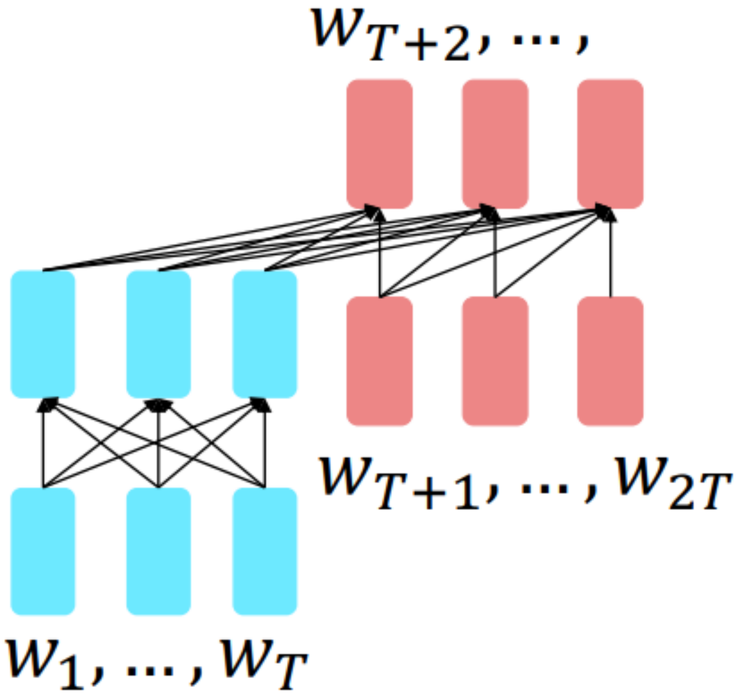
Recap: Pre-training Encoder-Decoder Models

Pretraining Encoder-Decoder Models

- For encoder-decoders, we could do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = Decoder(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$

The encoder portion benefits from bidirectional context; the decoder portion is used to train the whole model through language modeling.



T5: A Pretrained Encoder-Decoder Model

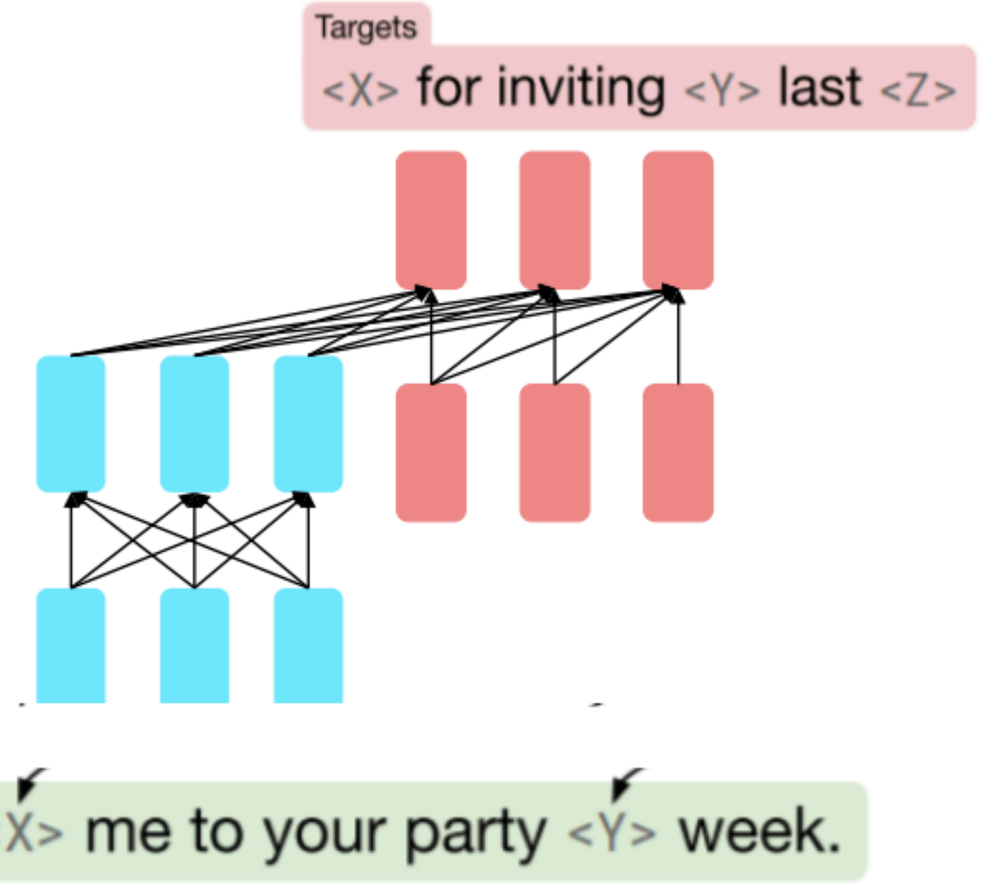
- Raffel et al., 2018 built T5, which uses as a span corruption pretraining objective

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

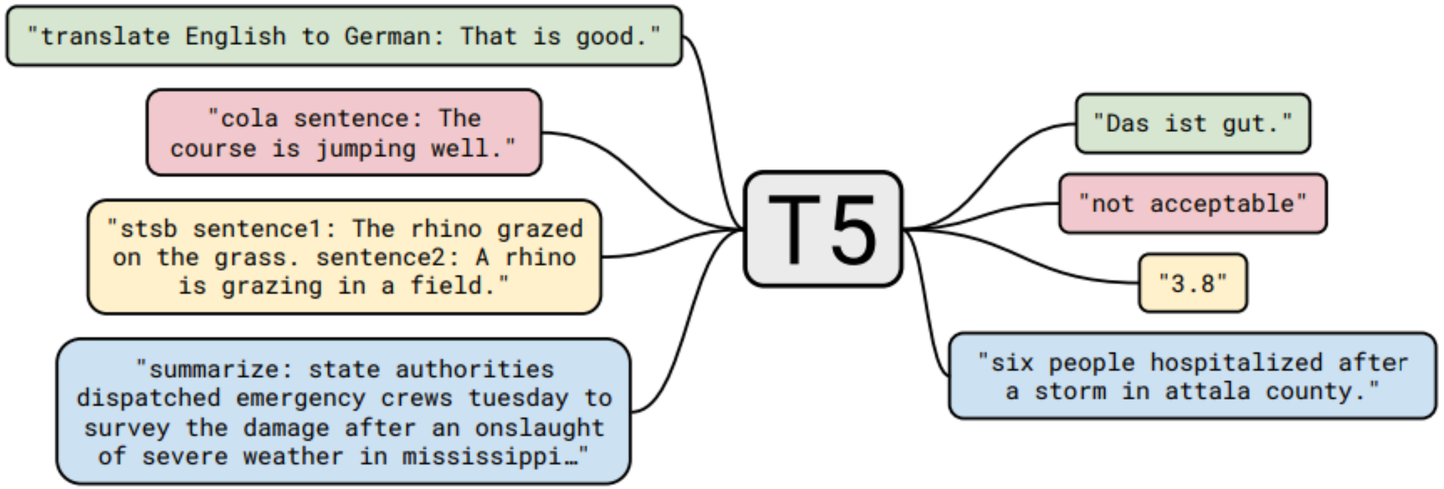
This is implemented in text preprocessing: it's still an objective that looks like language modeling at the decoder side.

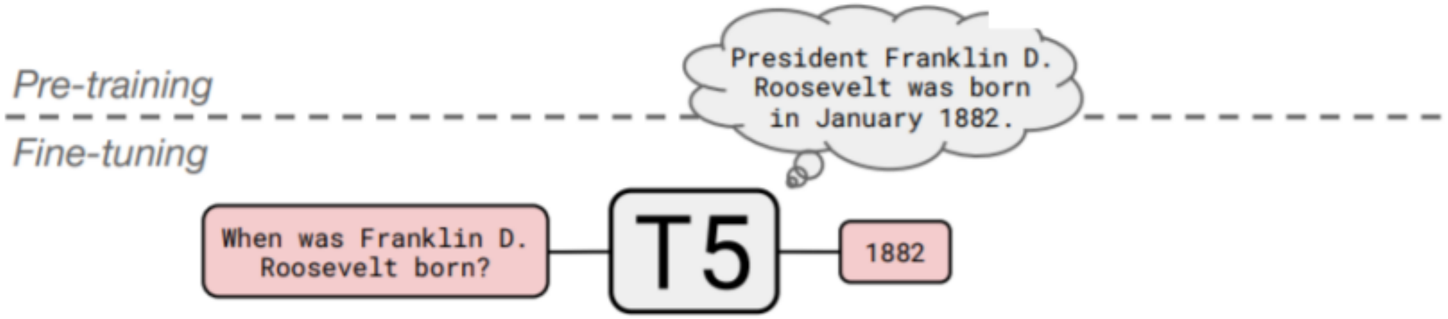


T5: Task Preparation

Pre-training task objective is very different from fine-tuning task objectives!

A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.





Recap: Tokenization in Transformers

Byte-pair encoding

- Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary
- Adapted for word segmentation from data compression technique (Gage, 1994)
  - Instead of merging frequent pairs of bytes, we merge characters or character sequences
- Algorithm:
  1. Start with a vocabulary containing only characters and an "end-of-word" symbol.
  2. Using a corpus of text, find *the most common adjacent characters* "a,b"; add "ab" as a subword
     - This is a learned operation! However, not a parametric function
     - Only combine pairs (hence the name!)
  3. Replace instances of the character pair with the new subword; repeat until desired vocabulary size.
- At test time, first split words into sequences of characters, then apply the learned operations to merge the characters into larger, known symbols
- Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es								

Frequency

d-e (3)	l-o (7)	t-</w> (8)
e-r (2)	n-e (5)	w-</w> (5)
e-s (8)	o-w (7)	w-e (7)
e-w (5)	r-</w> (2)	w-i (3)
i-d (3)	s-t (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>
l o w </w>	w i d es t</w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est							

Frequency

d-es (3)	l-o (7)	w-</w> (5)
e-r (2)	n-e (5)	w-es (5)
e-w (5)	o-w (7)	w-e (2)
es-t (8)	r-</w> (2)	w-i (3)
i-d (3)	t-</w> (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est	est</w>	lo	low	low</w>	ne	new	newest</w>

After 10 merges

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

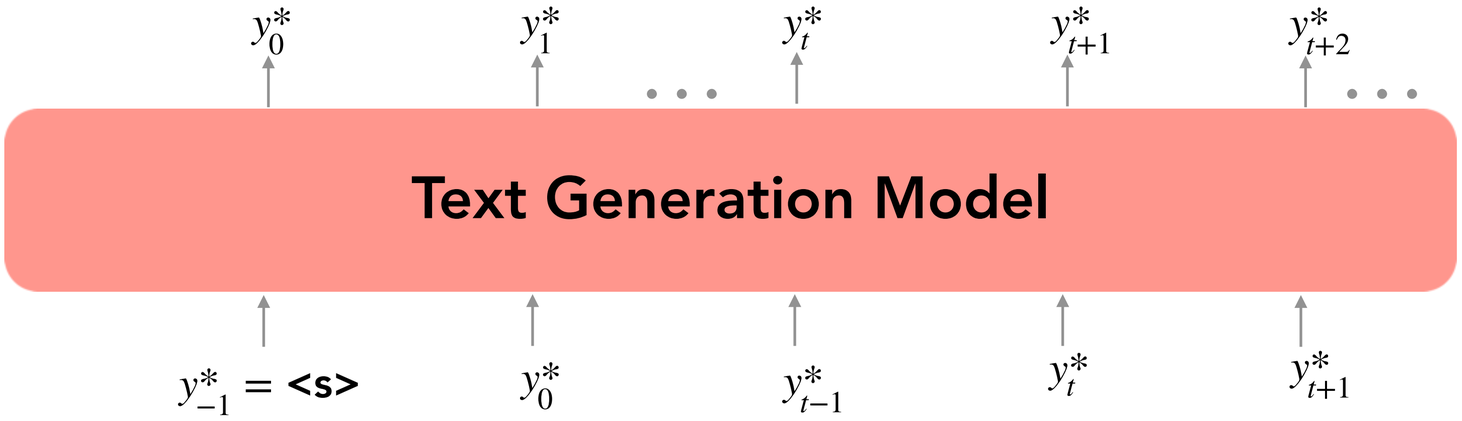
Natural Language Generation

Language Generation: Training

- Trained one token at a time to maximize the probability of the next token y_t^* given preceding words $y_{<t}^*$

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(y_t \,|\, y_{<t}) = -\sum_{t=1}^{T} \log \frac{\exp(S_{y_t|y_{<t}})}{\sum_{v \in V} \exp(S_{v|y_{<t}})}$$

- Classification task at each time step trying to predict the actual word y_t^* in the training data
- "Teacher forcing" (reset at each time step to the ground truth)



Teacher Forcing

- Strategy for **training** decoders / language models
- At each time step t in decoding we force the system to use the gold target token from training as the next input x_{t+1}, rather than allowing it to rely on the (possibly erroneous) decoder output $\hat{y}_t$
- Runs the risk of **exposure bias**!
  - During training, our model's inputs are gold context tokens from real, human-generated texts
  - At generation time, our model's inputs are previously–decoded tokens
- To avoid:
  - Allow the decoder at training times to occasionally condition on its own outputs

Language Generation: Inference

- At inference time, our decoding algorithm defines a function to select a token from this distribution:

Inference / Decoding Algorithm

$$\hat{y}_t = g(P(y_t \mid y_{<t}))$$

- The "obvious" decoding algorithm is to greedily choose the highest probability next token according to the model at each time step

$$g = \arg\max$$

$$\hat{y}_t = \arg\max_{w \in V}(P(y_t = w \mid y_{<t}))$$

Classic Inference Algorithms: Greedy and Beam Search

Greedy Decoding: Issues

- Greedy decoding has no wiggle room for errors!
  - Input: the green witch arrived
    - Output: llego
    - Output: llego la
    - Output: llego la verde
- How to fix this?
  - Need a lookahead strategy / longer-term planning

Exhaustive Search Decoding

- Ideally, we want to find a (length T) translation y that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

- We could try computing all possible sequences y
  - This means that on each step t of the decoder, we're tracking V^tpossible partial translations, where $V$is vocab size
  - This $O(V^T)$complexity is far too expensive!

Beam Search Decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - k is the beam size (in practice around 5 to 10, in NMT)
- A hypothesis has a score which is its log probability:

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

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

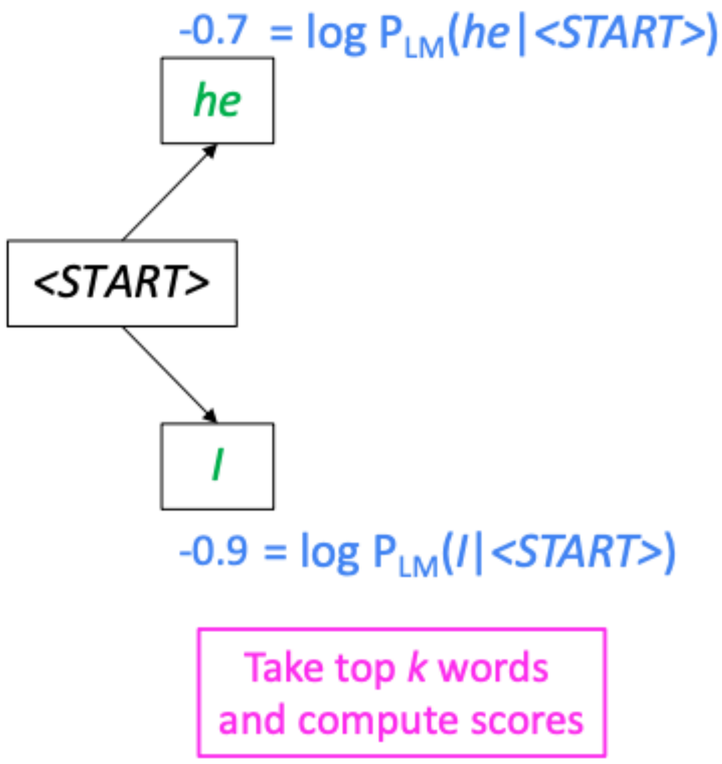
<START>

Calculate prob dist of next word

Slide credit: Chris Manning

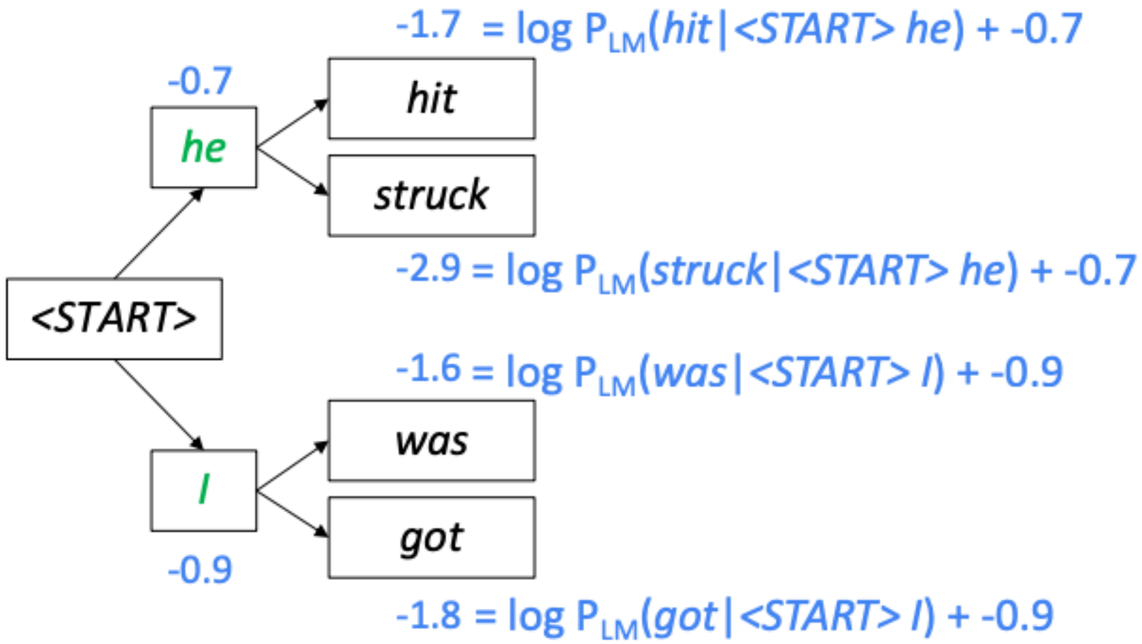
Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

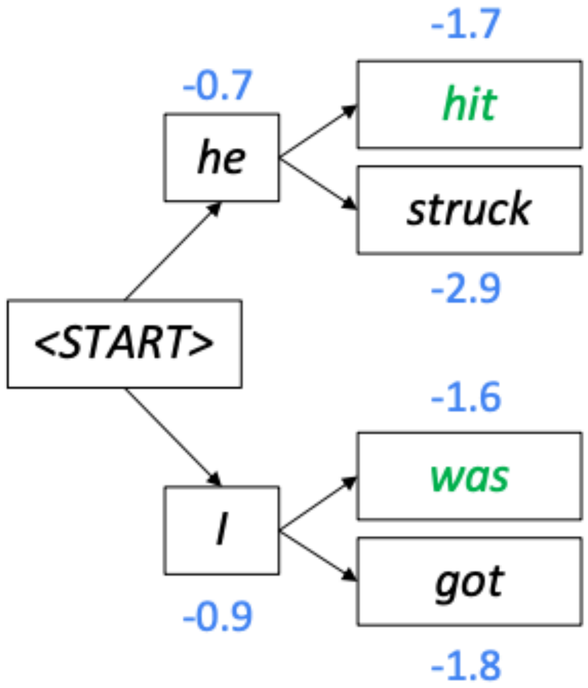


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

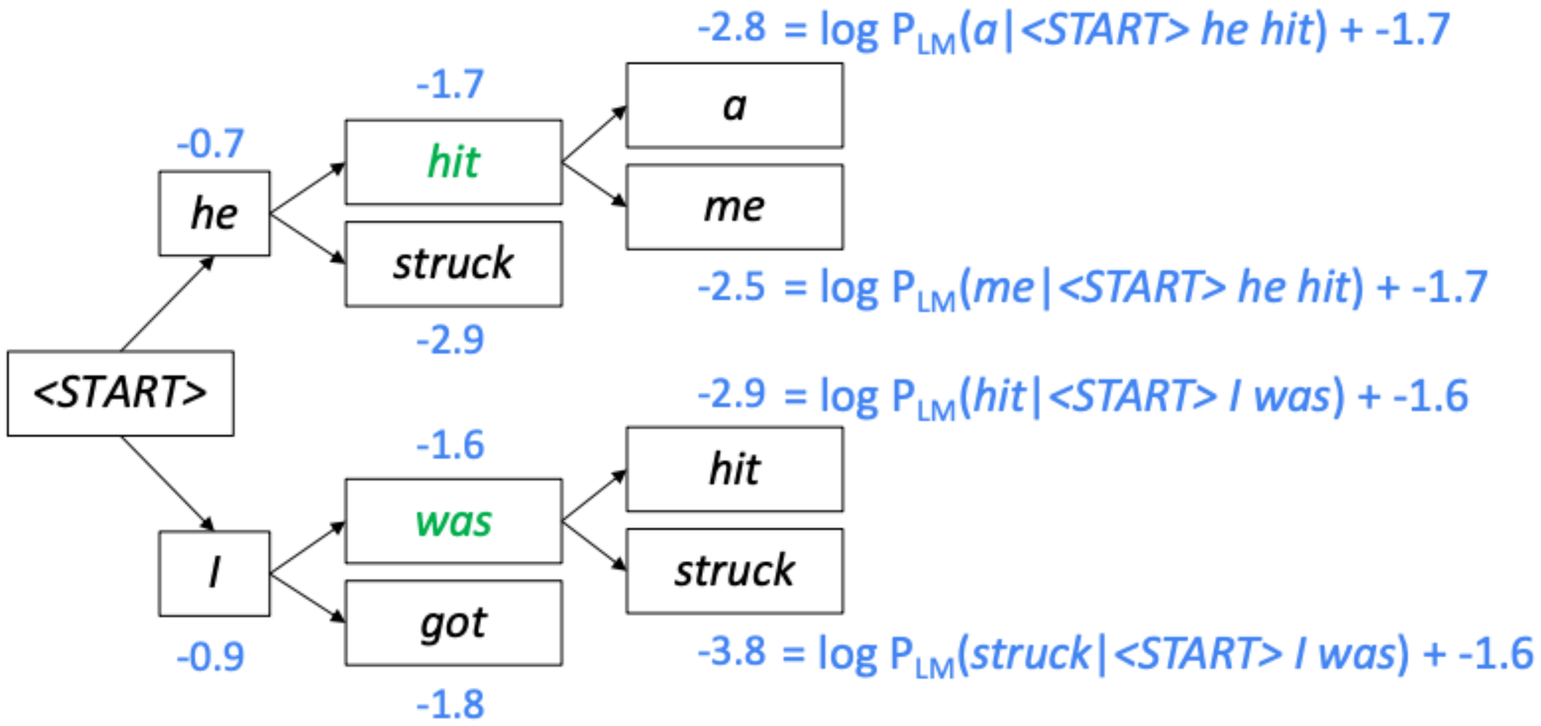


Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

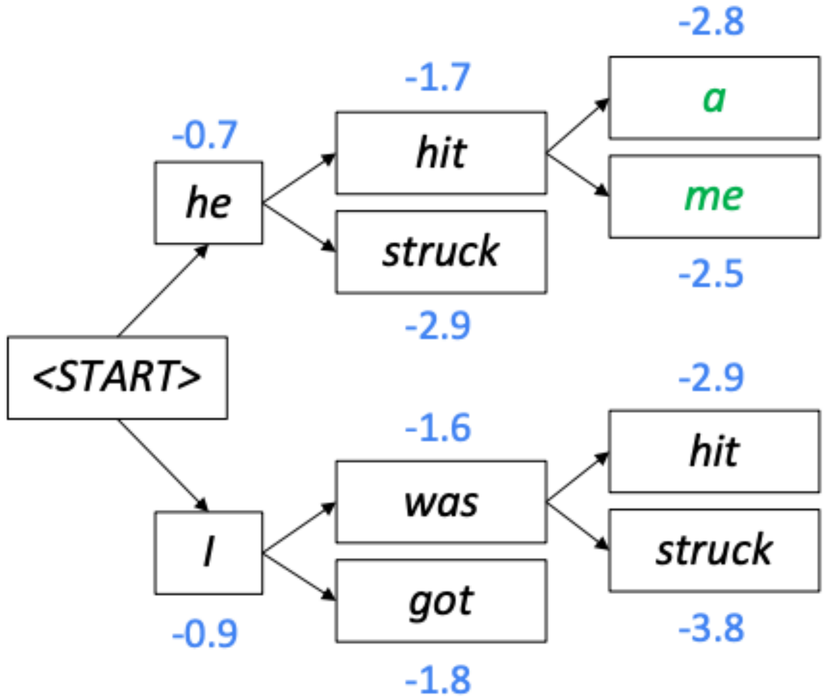


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

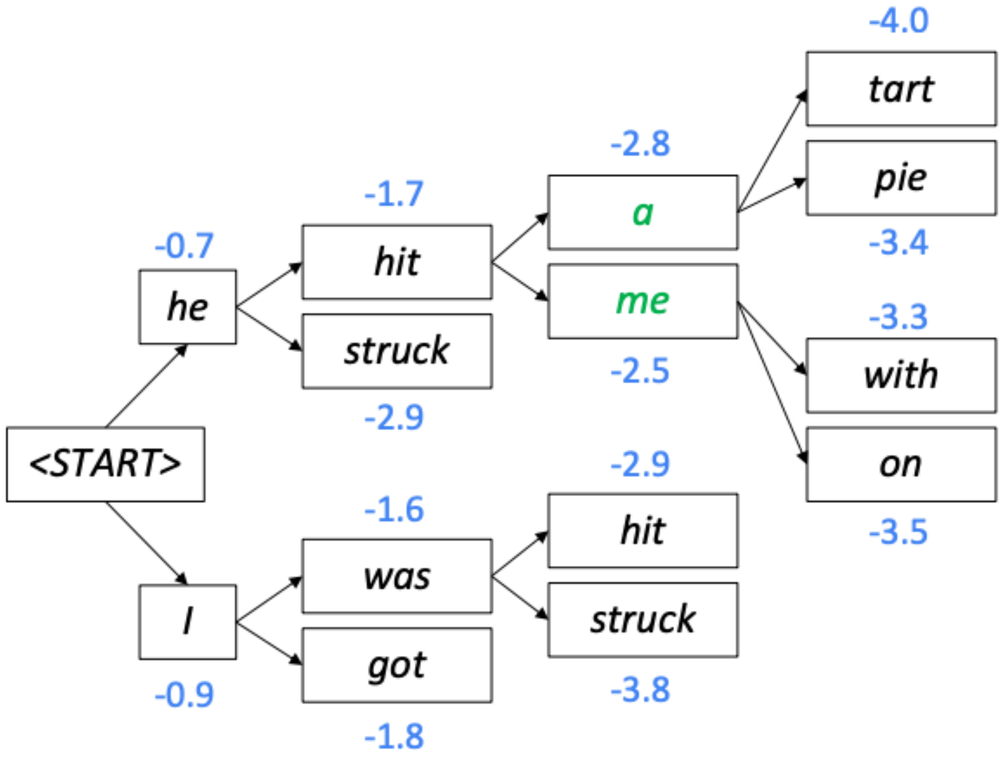


Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

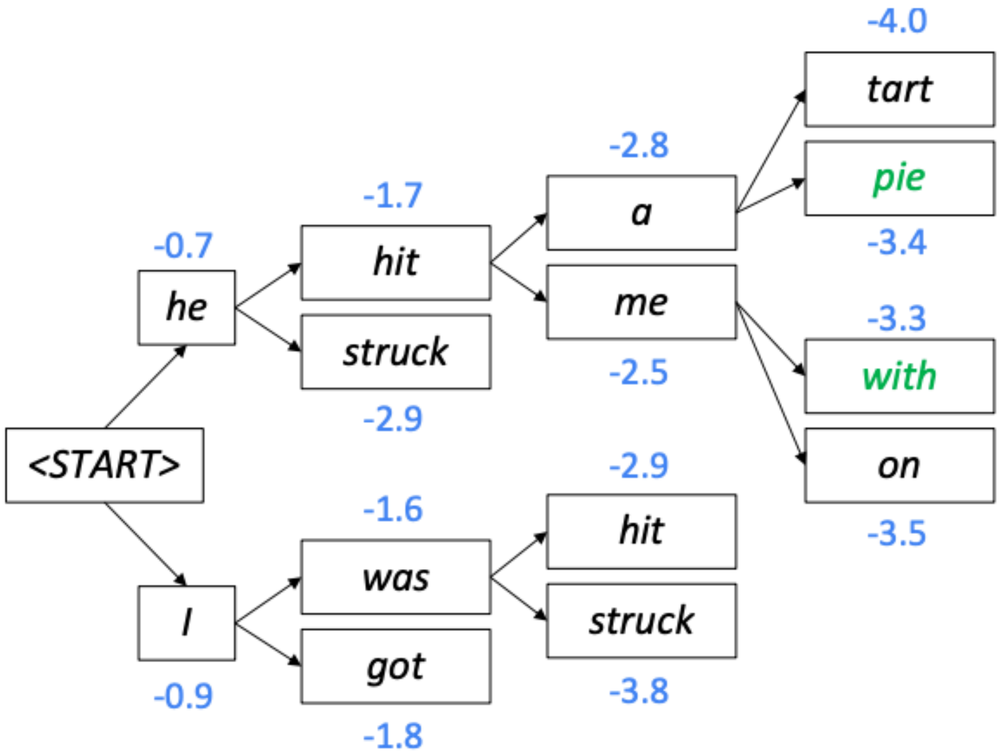


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

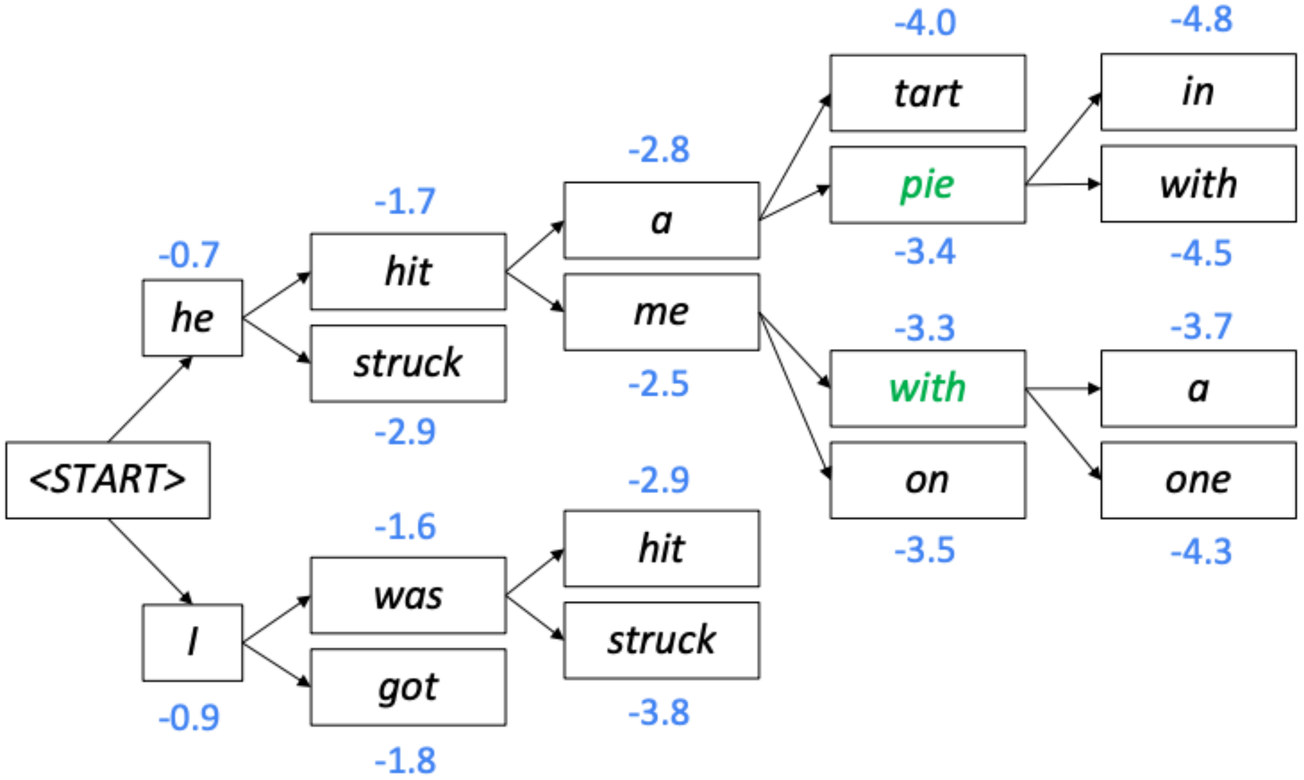


Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

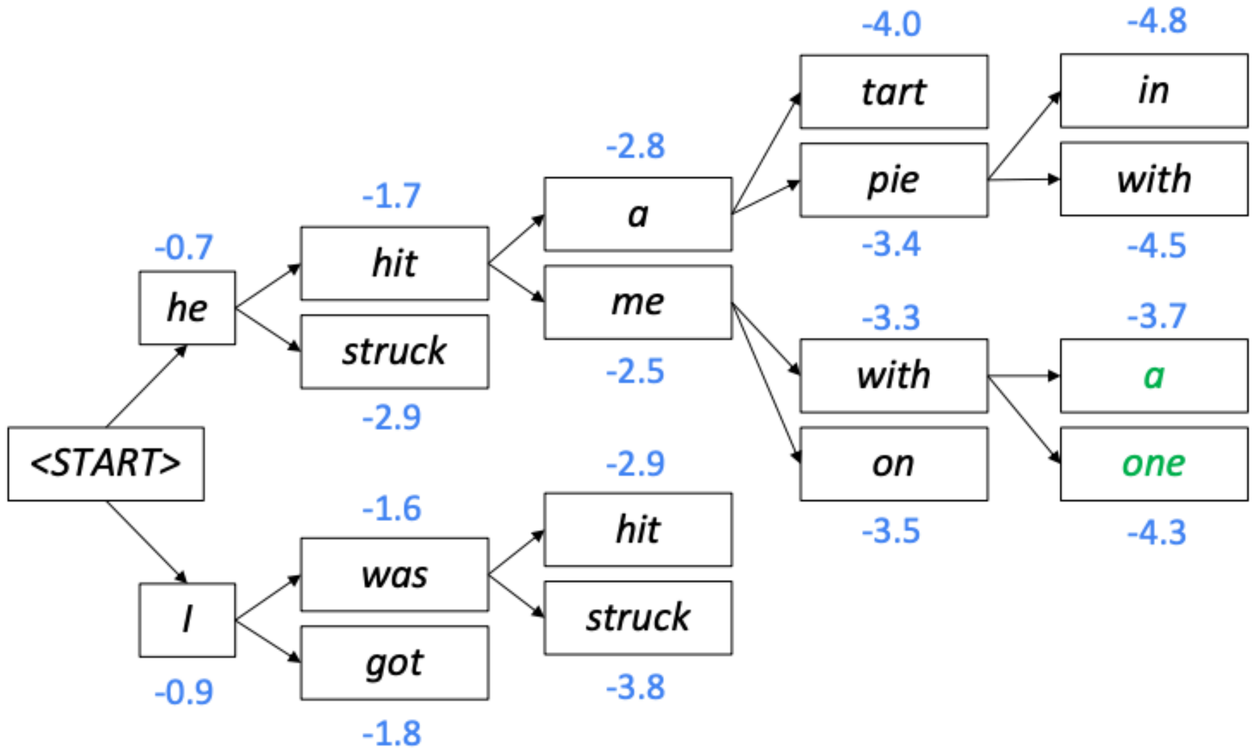


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

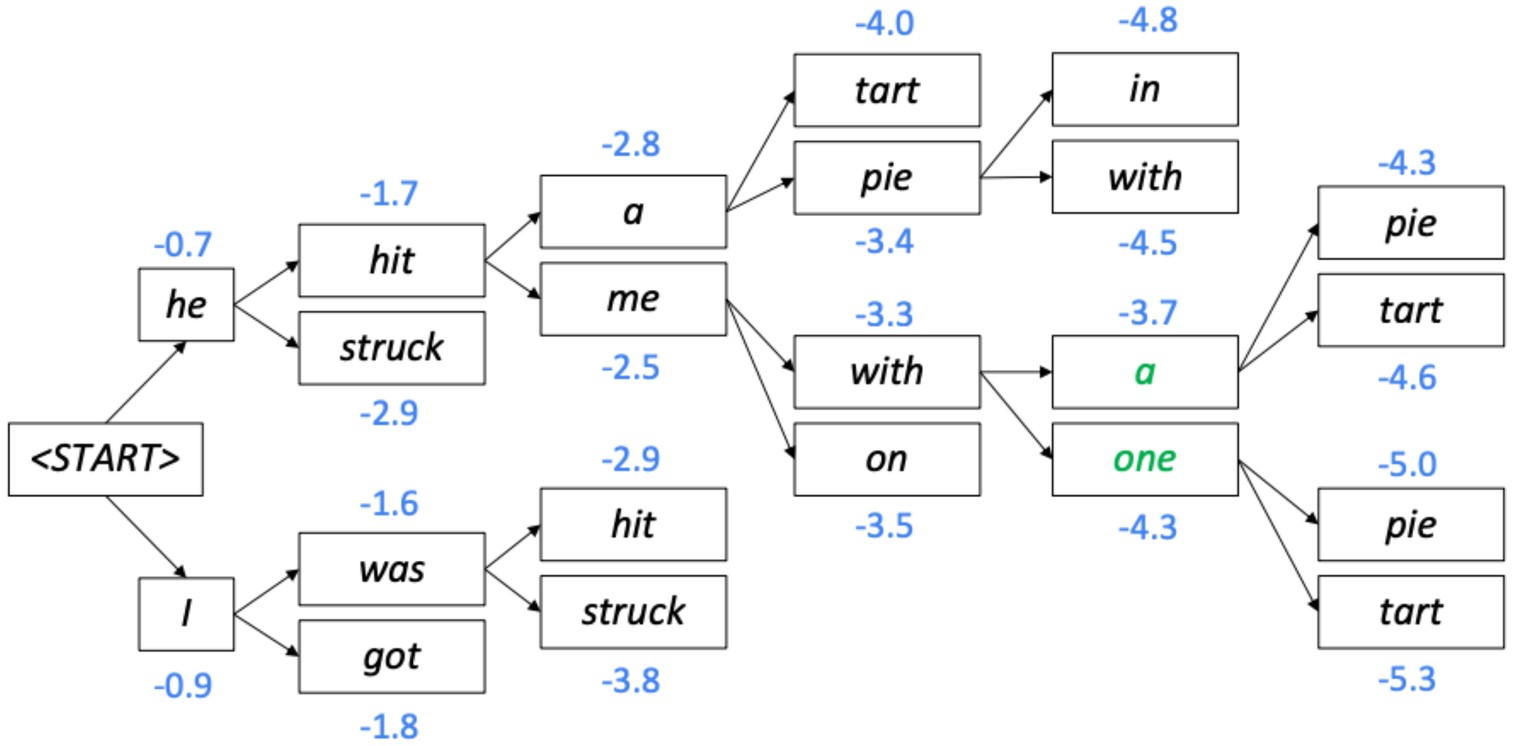


Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

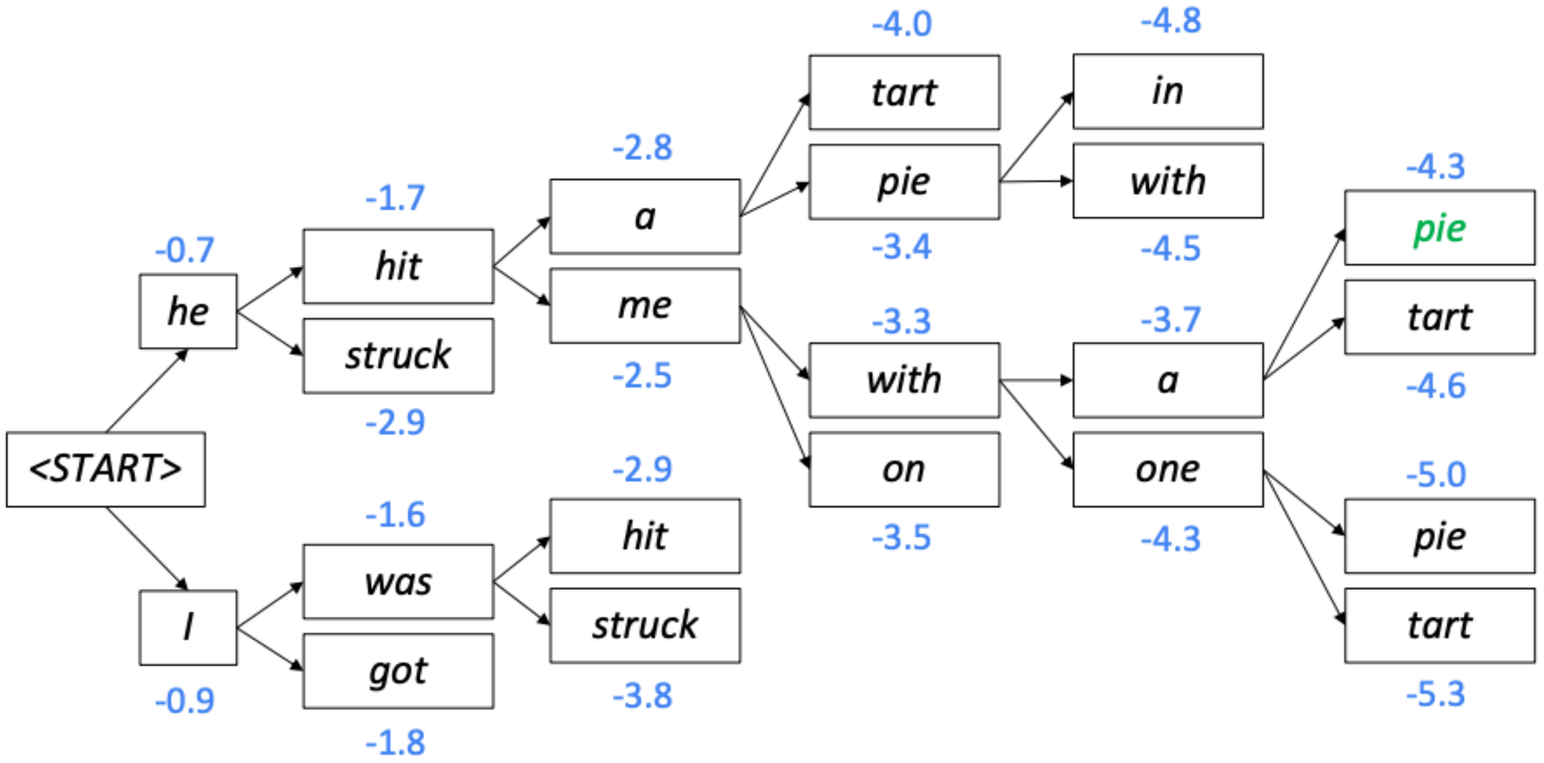


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

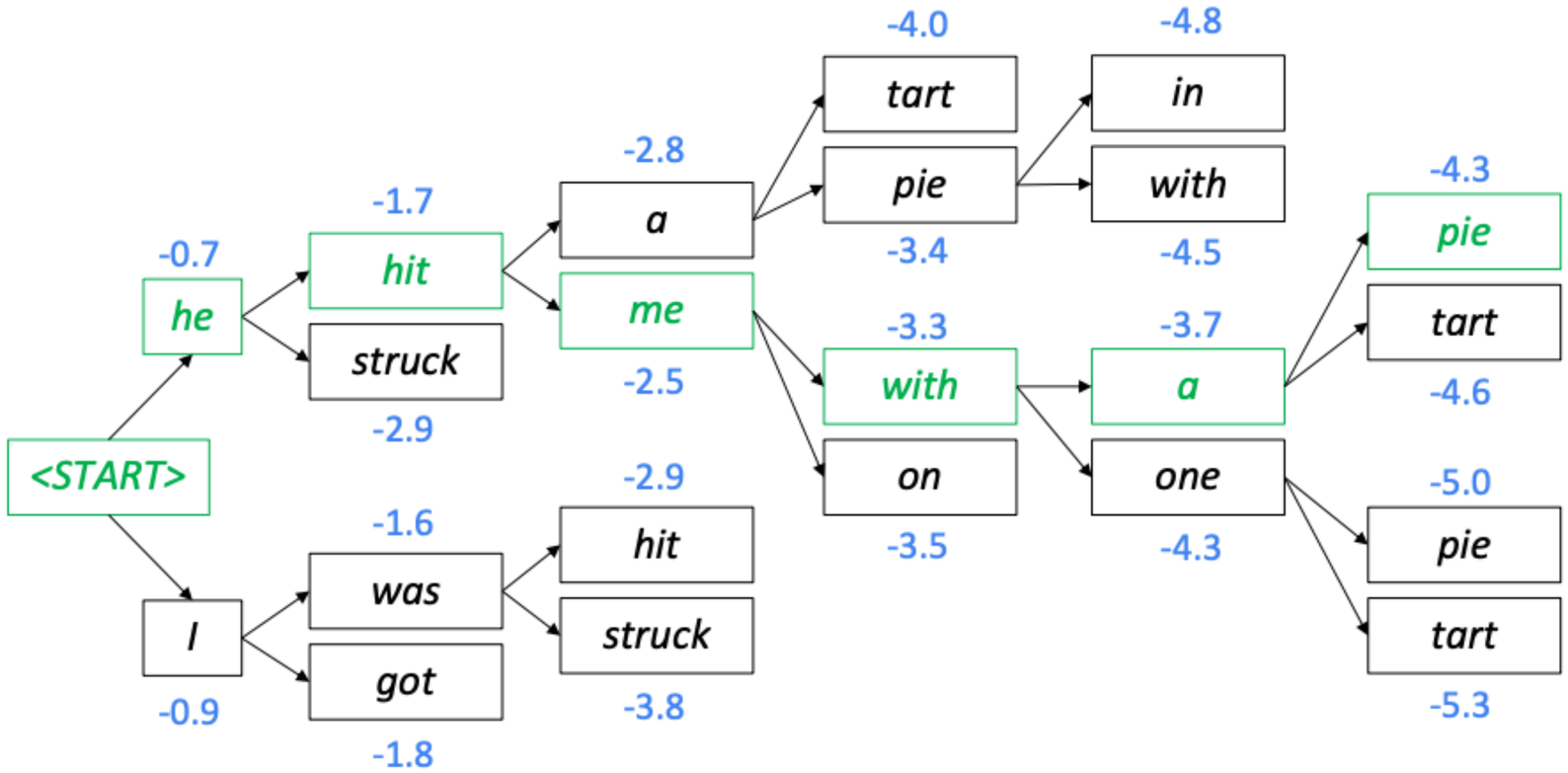


This is the top-scoring hypothesis!

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Backtrack to obtain the full hypothesis

Slide credit: Chris Manning

Beam Search Decoding: Stopping Criterion

- Greedy Decoding is done until the model produces an </s> token
  - For e.g. <s> he hit me with a pie </s>
- In Beam Search Decoding, different hypotheses may produce </s> tokens at different time steps
  - When a hypothesis produces </s>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
  - We reach time step T (where T is some pre-defined cutoff), or
  - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

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

- Problem with this: longer hypotheses have lower score
- Fix: Normalize by length. Use this to select top one instead

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

But this is expensive!

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive
- Another key issue:

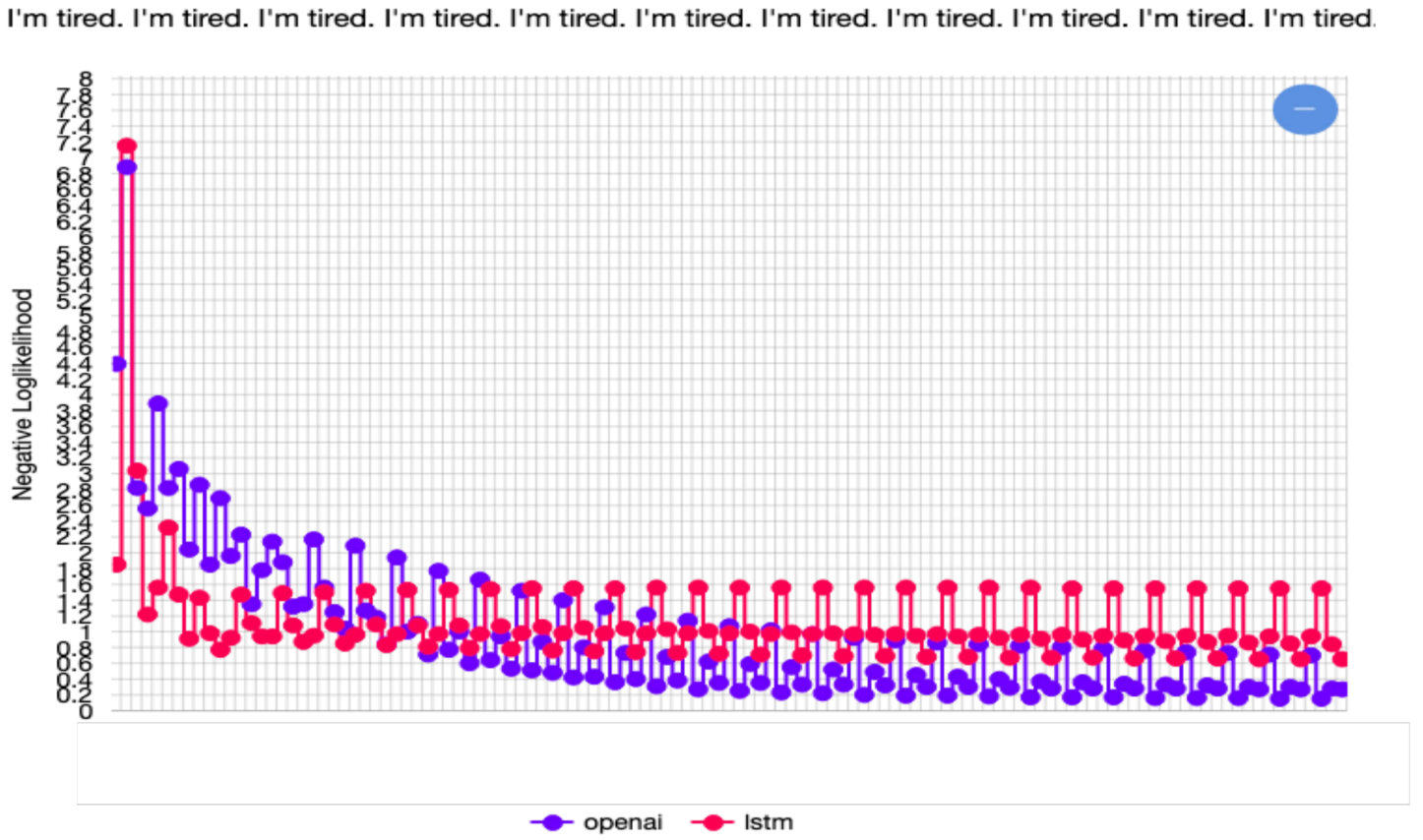
Generation can be bland or repetitive (also called degenerate)

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...**

Holtzmann et al., 2020

Degenerate Outputs

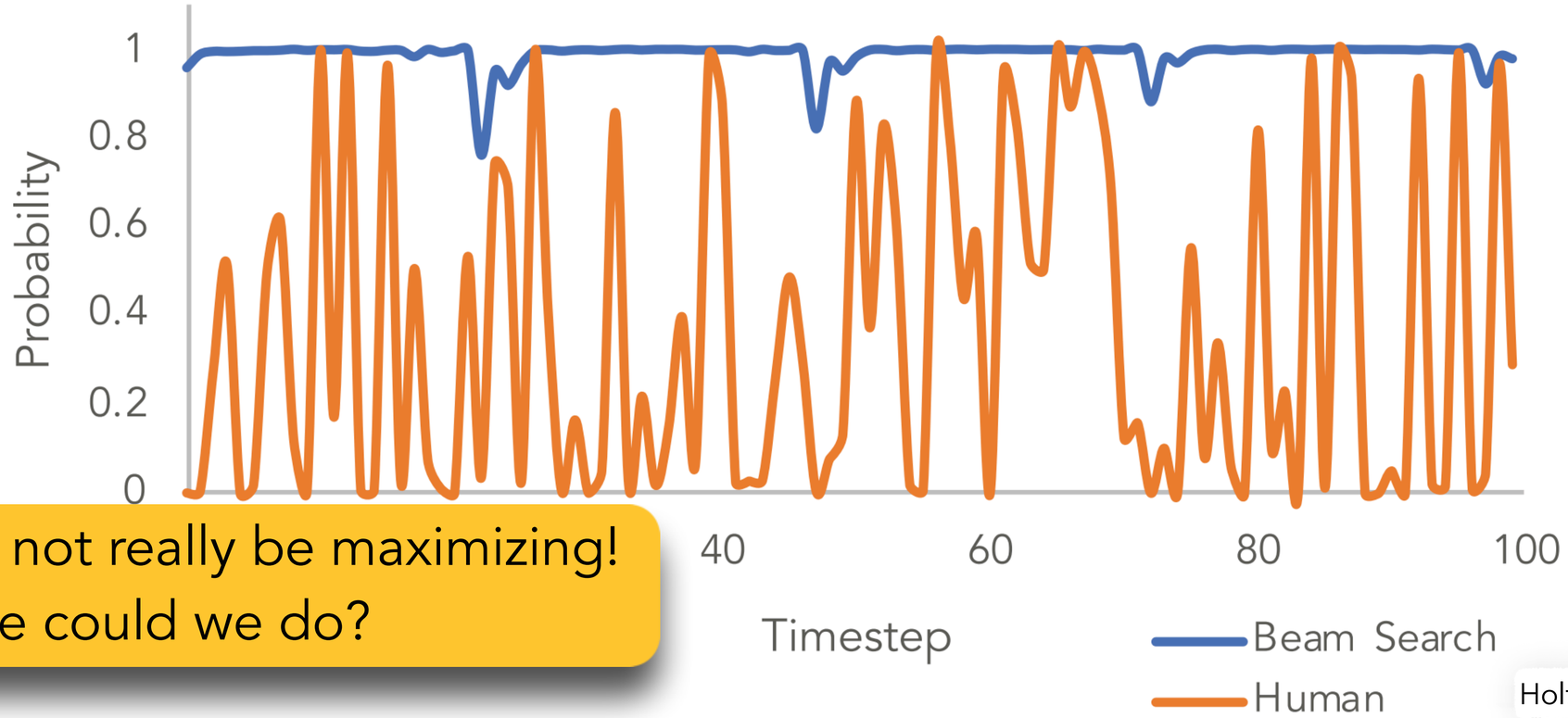


Holtzmann et al., 2020

However, the problem goes away under extreme-scale language models, such as GPT-4 and Llama-3

Why does repetition happen?

- Probability amplification due to maximization based decoding
- Generation fails to match the uncertainty distribution for human written text

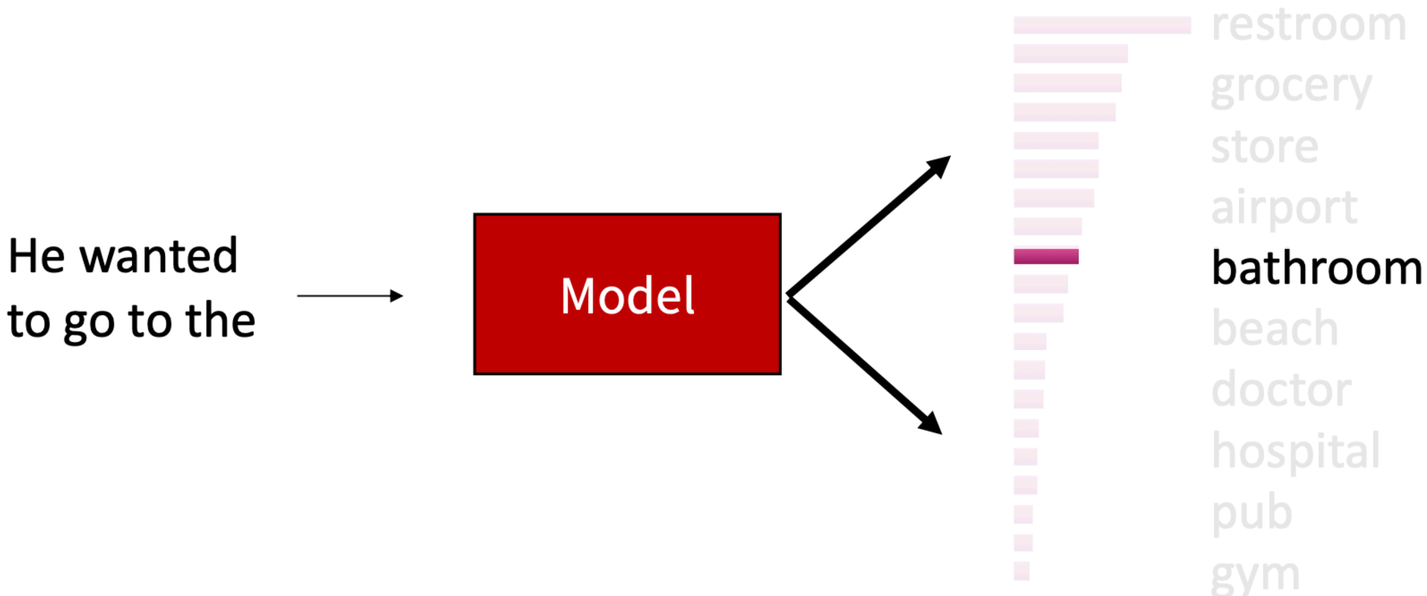


Perhaps we should not really be maximizing!
What else could we do?

Holtzmann et al., 2020

Solution: Don't Maximize, Pick a Sample

- Sample a token from the distribution of tokens.
- But this is not a random sample, it is a sample for the learned model distribution
  - Respects the probabilities, without going just for the maximum probability option
  - Or else, you would get something meaningless
  - Many good options which are not the maximum probability!

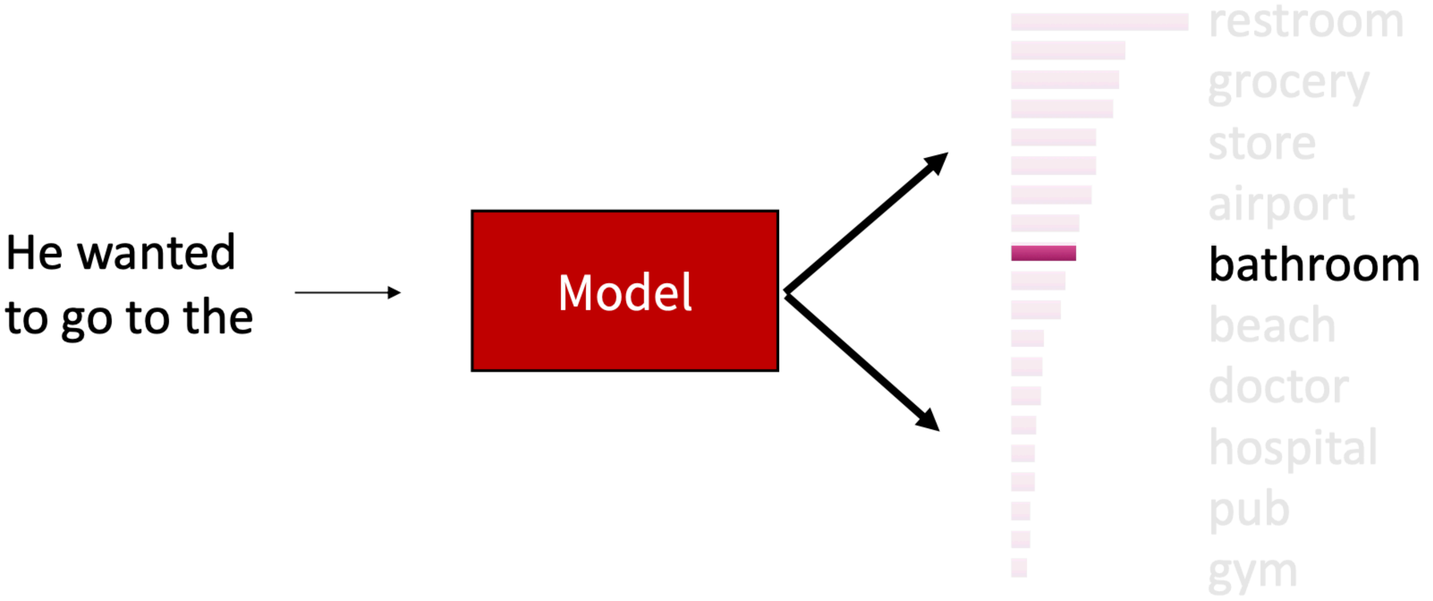


Modern Generation: Sampling

Pure / Ancestral Sampling

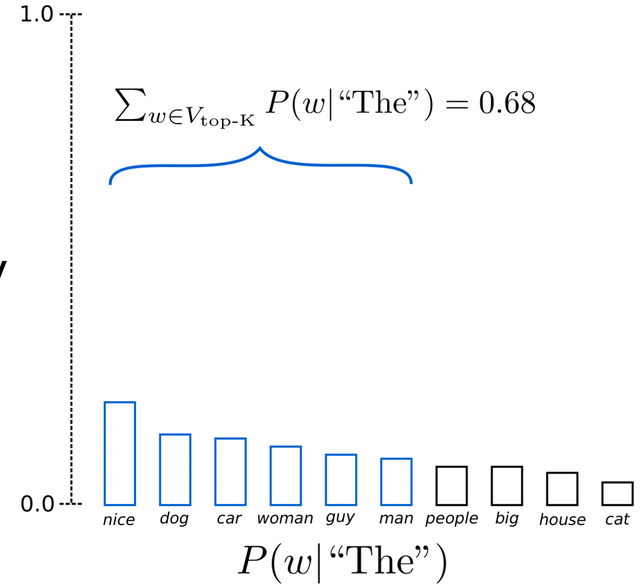
$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

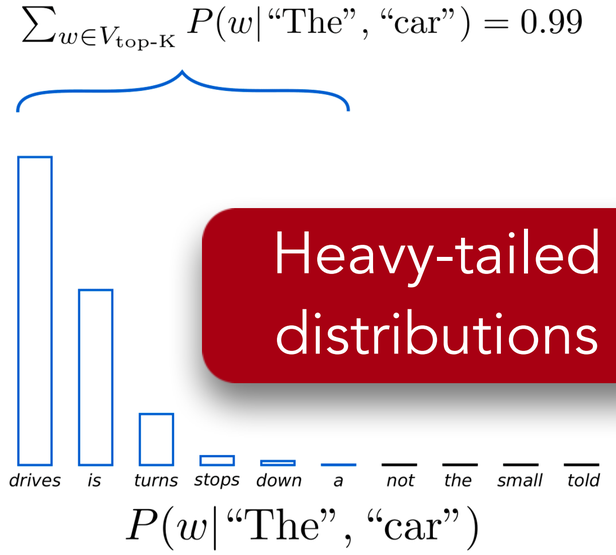
- Sample directly from P_t
- Still has access to the entire vocabulary
- But if the model distributions are of low quality, generations will be of low quality as well
- Often results in ill-formed generations
  - No guarantee of fluency



Top-K Sampling

- Problem: Ancestral sampling makes every token in the vocabulary an option
  - Even if most of the probability mass in the distribution is over a limited set of options, the tail of the distribution could be very long and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.
  - But because there are many of them, we still give them as a group a high chance to be selected.
- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution





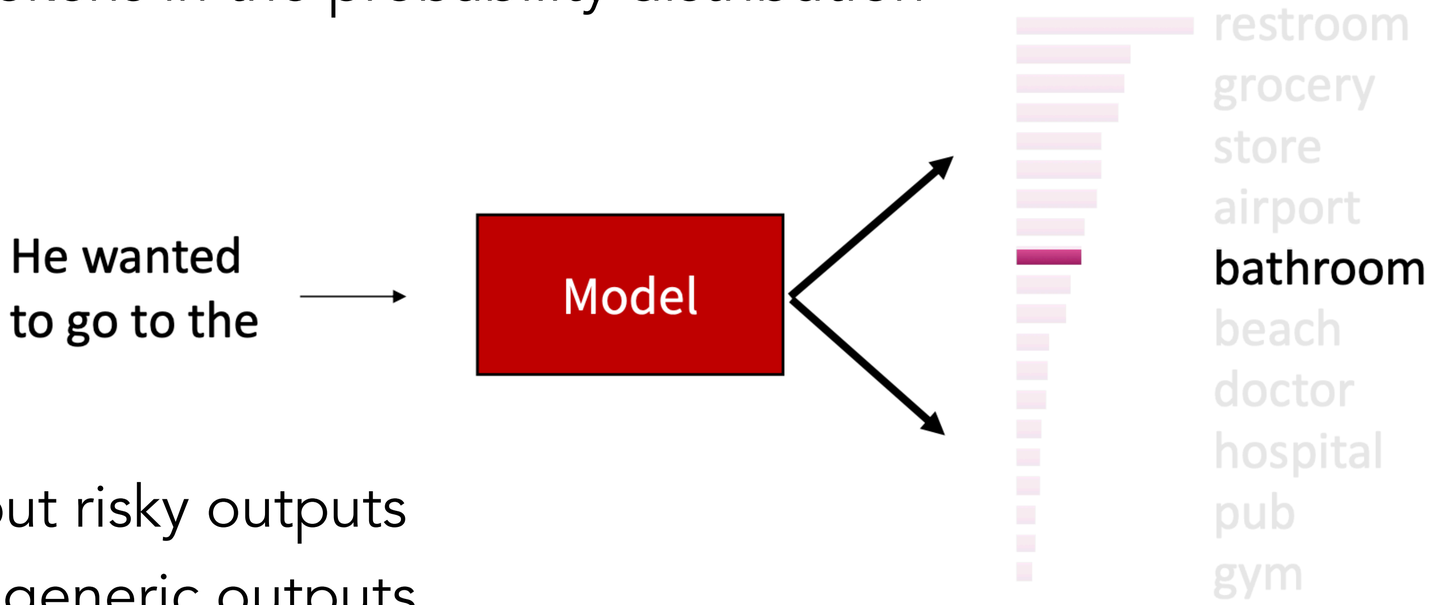
Heavy-tailed distributions

Fan et al., ACL 2018; Holtzman et al., ACL 2018

Image Source: Huggingface

Top-K Sampling: Value of K

- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution
  - Common values are K = 50



- Increase K yields more diverse, but risky outputs
- Decrease K yields more safe but generic outputs

Top-K Sampling: Issues

Top-K sampling can cut off too quickly

Top-K sampling can also cut off too slowly!

We can do better than having one-size-fits-all: a fixed K for all contexts

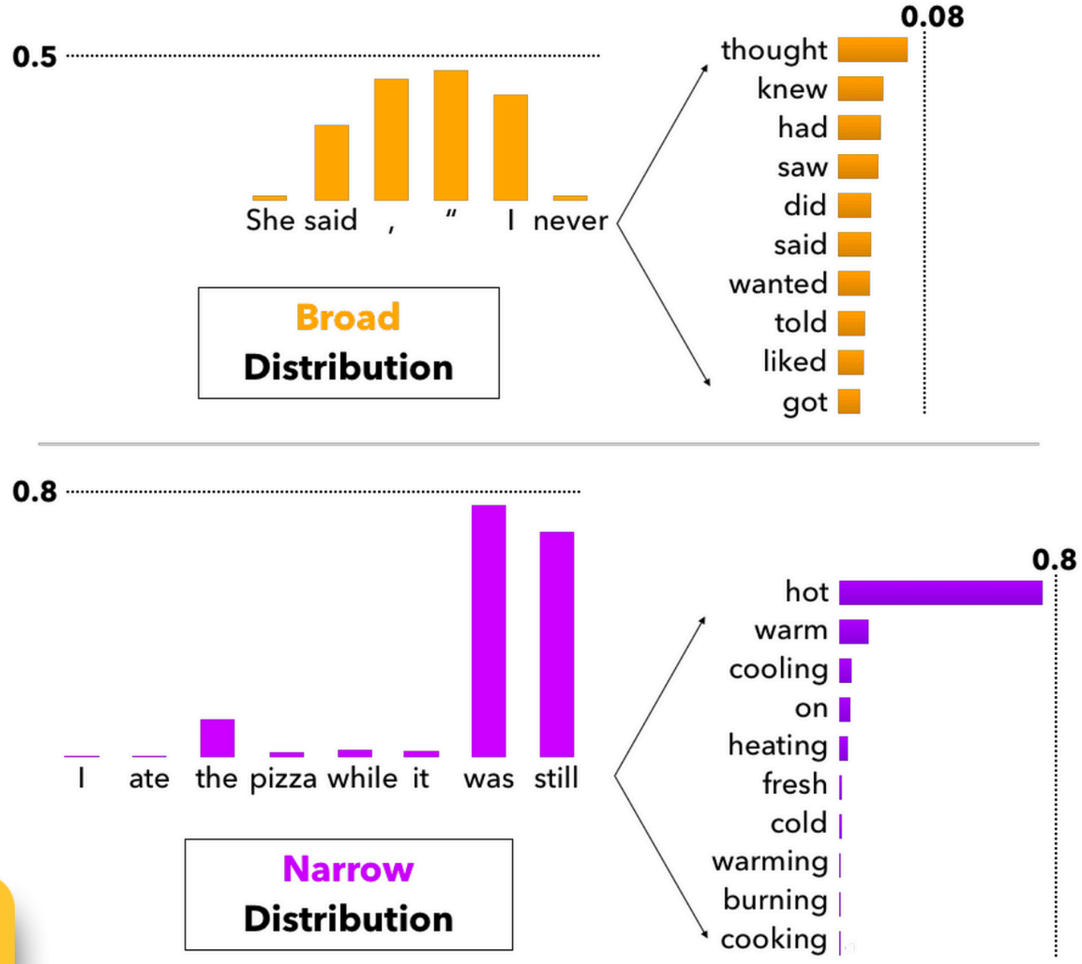


Image Source: Holtzmann et al., 2019

Modern Decoding: Nucleus Sampling

- Problem: The probability distributions we sample from are dynamic
  - When the distribution P_t is flatter, a limited K removes many viable options
  - When the distribution P_t is peakier, a high K allows for too many options to have a chance of being selected

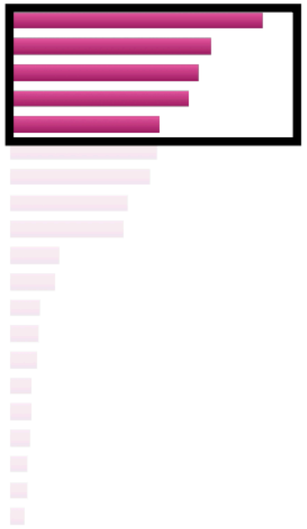
- Solution: Nucleus Sampling / Top-P sampling
  - Sample from all tokens in the top P cumulative probability mass (i.e., where mass is concentrated)
  - Varies K depending on the uniformity of P_t

Holtzman et al., ICLR 2020

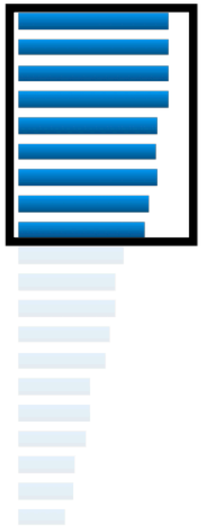
Nucleus (Top-P) Sampling

- Solution: Top-P sampling
  - Sample from all tokens in the top P cumulative probability mass (i.e., where mass is concentrated)
  - Varies K depending on the uniformity of P_t

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



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



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

Holtzman et al., ICLR 2020

Temperature Scaling

Originally, $P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$

- Recall: On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$
- We can apply a temperature hyperparameter τ to the softmax to rebalance P_t

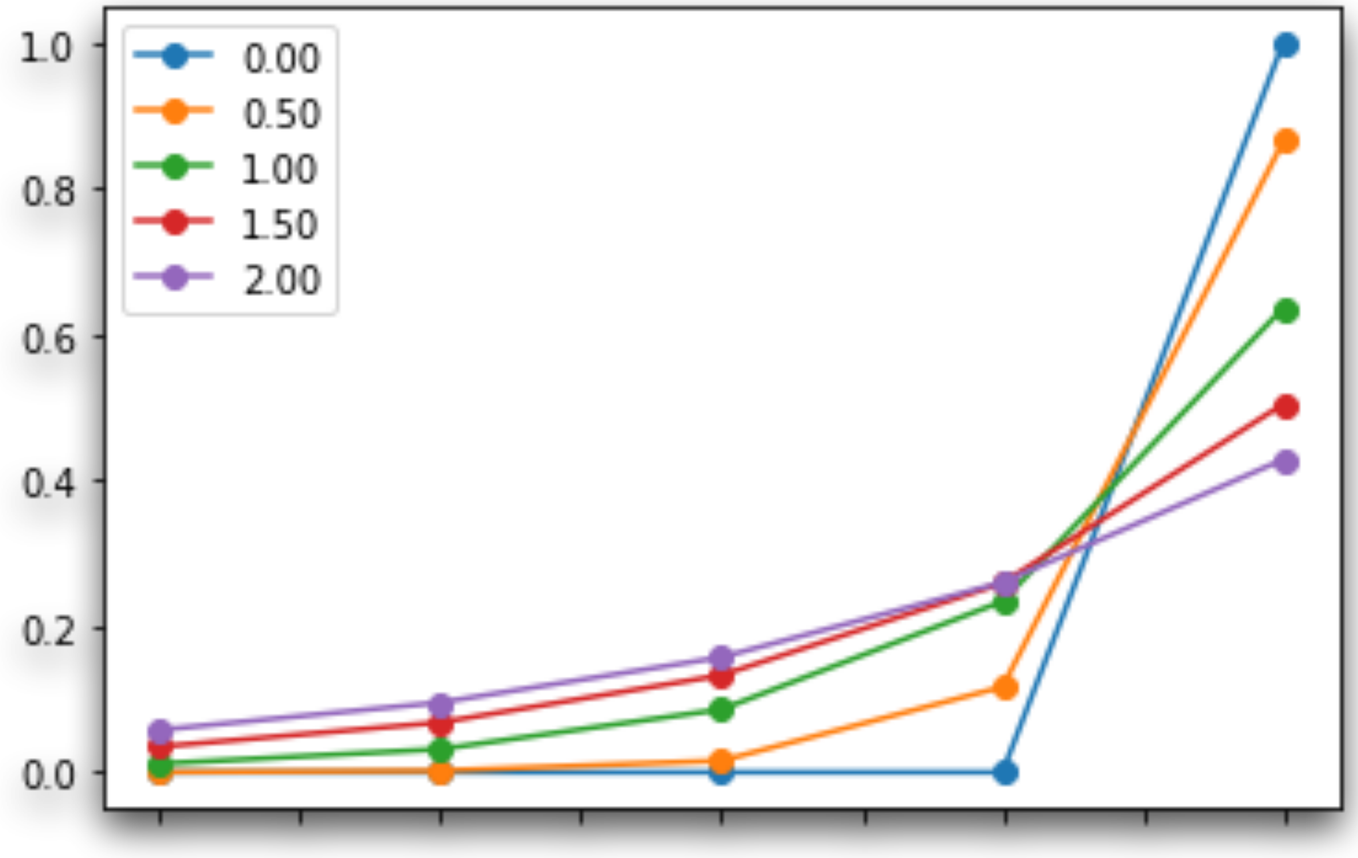
$$P(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{v \in V} \exp(S_v/\tau)}$$

- Let's say initial scores, S_w: (remember these are real-valued)
  - 0.1912, 0.7492, 0.5966, 0.5528, 0.8324, **0.9409**
- After softmax, p:
  - 0.1031, 0.1802, 0.1547, 0.1480, 0.1958, **0.2182**
- S_w/τ when $\tau = 0.01$:
  - 19.12, 74.92, 59.66, 55.28, 83.24, **94.09**
- After softmax, p
  - 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, **1.0000**

Temperature is a hyperparameter for decoding: It can be tuned for both beam search and sampling.

Sampling after Temperature Scaling

- Raise the temperature $\tau > 1$: P_t becomes more uniform
  - More diverse output (probability is spread around vocab)
- Lower the temperature $\tau < 1$: P_t becomes more spiky
  - Less diverse output (probability is concentrated on top words)



Comparing different decoding algorithms

- Generate text to continue a given context
  - Open-ended generation
- Same decoding algorithms are also useful for close-ended generation tasks

WebText

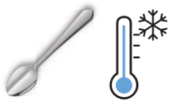
So what's new in my life?
09/11/18 - Just got back from vacation.

Beam Search, *b*=16

Just got back from vacation.
09/10/18 - Just got back from vacation. Just got back from vacation.
09/09/18 - Just got back from vacation. Just got back from vacation.
09/08/18 - Just got back from vacation. Just got back from vacation.

Pure Sampling

01/05/19 - Visited The Subaru Canada Heritage15th Star for my vehicle research. That was a wonderful experience. Thanks again for your knowledge and support.
05/23/19 - I was thrilled to receive my Subaru Legacy character! It fits me perfectly and makes me feel like a kid again!
06/15/19 - Just regained my licence for my different coloured 2016 WRX / STI...



Sampling, *t*=0.9

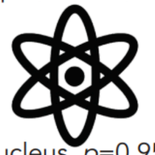
Just got back from vacation.
09/07/18 - Happy 9th birthday to me! Happy 9th birthday to me!
09/01/18 - Sofwidden also! Sofwidden also!
08/25/18 - Happy birthday to me! Happy birthday to me!
08/17/18 - Night shift! □□Night shift! □□
08/17/18 - Lucky me! Lucky me!

Top-*k*, *k*=640

Just got back from vacation.
09/03/18 - After more than 40 years, I might have an apartment in Virginia Beach.
After more than 40 years, I might have an apartment in Virginia Beach.
08/20/18 - Going for a hike at Mount Eerie in Northeast Virginia Spring
Going for a hike at Mount Eerie in Northeast Virginia Spring

Top-*k*, *k*=640, *t*=0.7

Just got back from vacation.
09/08/18 - I've gotten really sick. - I've gotten really sick.
09/07/18 - My wife and I are getting married in February.
- My wife and I are getting married in February.
09/06/18 - I'm so excited to go back to college this fall.
- I'm so excited to go back to college this fall.



Nucleus, *p*=0.95

Just got back from vacation.
07/12/18 - Happy birthday to Swingu, who is nearly 5 years old. I would like to say hi to him on the road as well as when I ride with him. You cannot go to work without feeling physically sick or psychologically exhausted because you can barely breathe. Even if you ride on rollercoaster even once, it is easy to recover from the physical side of it.

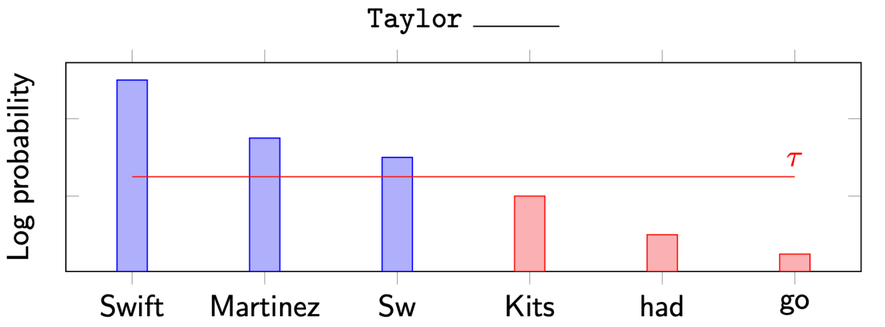
WebText

I just got back from a much needed and really great nine day vacation to my remote Arizona property. It was a really restful and relaxing visit. I got a lot accomplished while I was there, but still found time to just goof off and have fun too. I got to do some astronomy, even though the weather was pretty cloudy most of the time. Here is a 50 minute exposure of M101. It turned out pretty good.

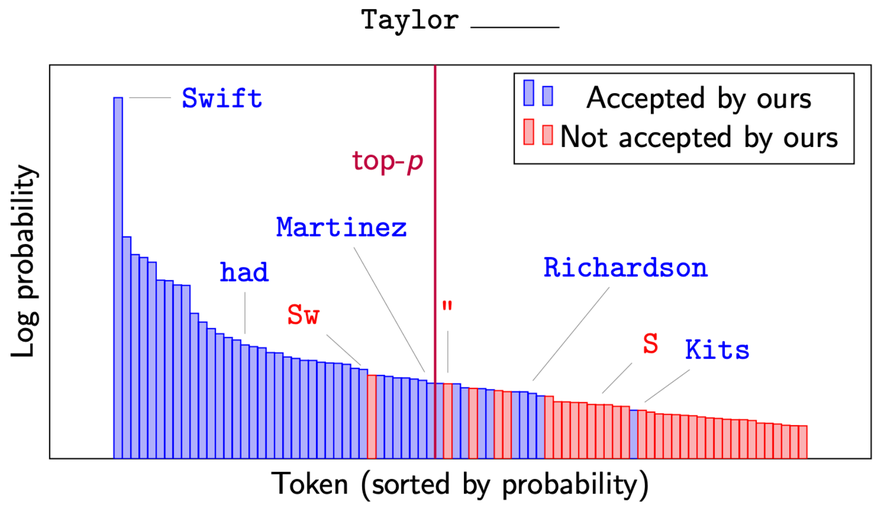
Holtzman et al., ICLR 2020

Truncation-based Sampling

- Nucleus Sampling is an example of truncation sampling
  - Certain properties of language models (mismatch between vocabulary size and hidden dimensionality) make threshold sampling a great choice!
  - [Finlayson et al., 2024]
- Locally-Typical Sampling: Similar to Nucleus Sampling, but based on conditional entropy (entropy of a distribution determines its randomness) [Meister et al., 2022]
- η-Sampling: Entropy dependent threshold that also takes into account absolute probabilities [Hewitt et al., 2022]
- BAT Sampling: More flexible than truncation [Finlayson et al., 2024]

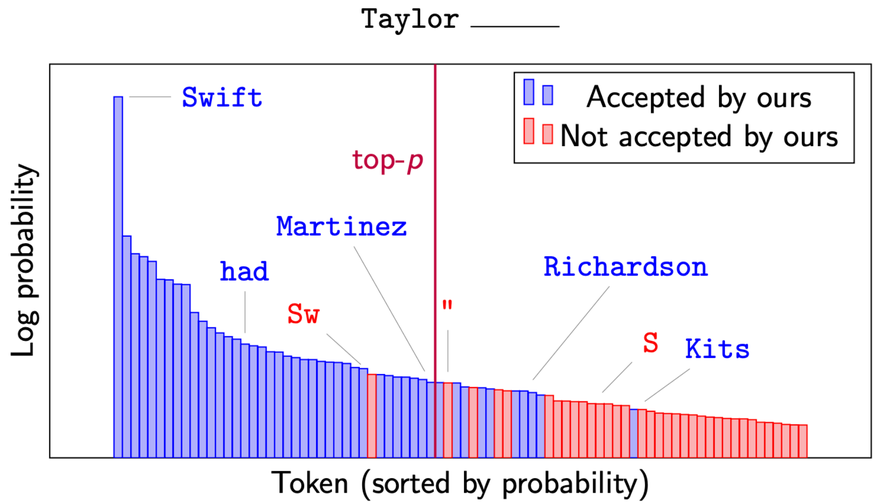


Choose a threshold τ and only sample tokens with probability greater than τ.



Modern Decoding: Takeaways

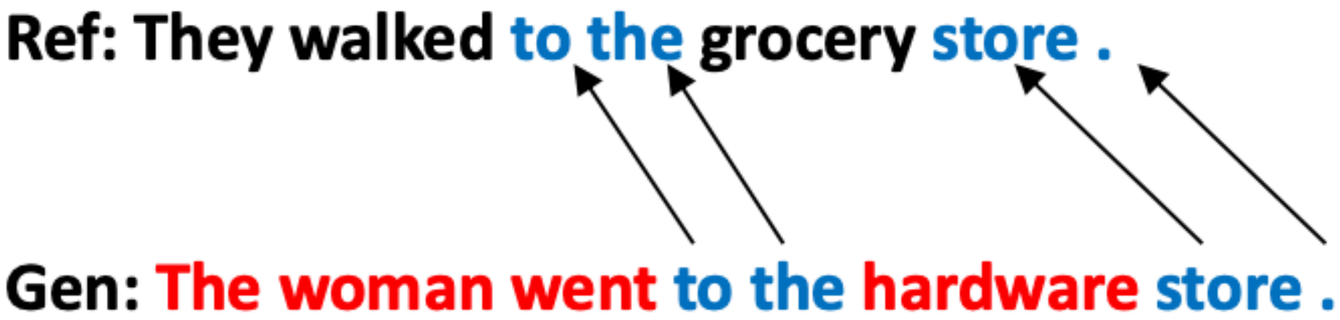
- Natural language distributions are very peaky but the softmax function assigns probabilities to all tokens in the vocabulary
- Hence we need approaches to truncate / modify the softmax distribution
  - Ancestral, Top-k, Top-p (Nucleus), Temperature
- Some properties of the softmax function make truncation based decoding necessary



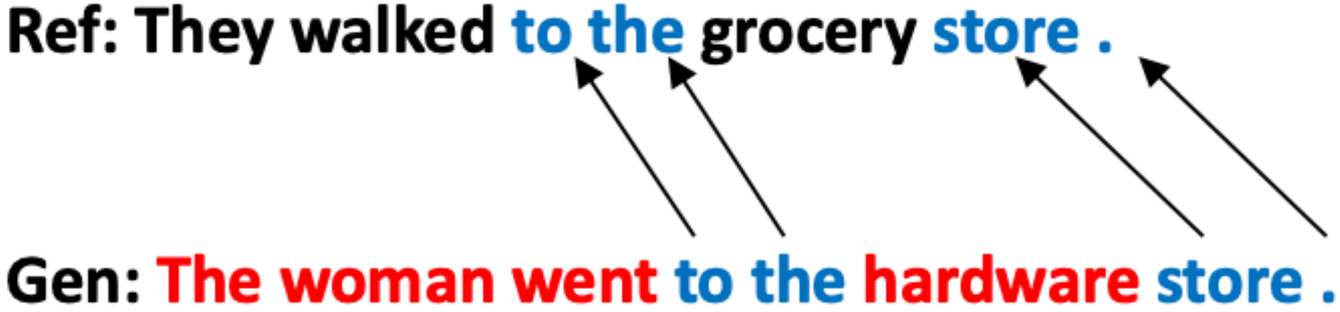
Evaluating Generations

Evaluation Strategies

- With Reference
  - Lexical Matching
  - Semantic Matching
- Without Reference
  - Perplexity
  - Model-Based Metrics
  - Advanced: Distributional Matching
  - Simplest, Most Reliable Strategy to-date: Human Evaluation
  - Even simpler and least reliable: Auto Evaluation



Reference-Based Metrics



- Only possible for close-ended generation tasks
- Compute a score that indicates the lexical similarity between generated and gold-standard (human-written) text
- Fast and efficient and widely used
- n-gram overlap metrics (e.g., BLEU, ROUGE, etc.)

BLEU

- Stands for Bilingual Evaluation Understudy
- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - Geometric mean of n-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations
- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low n-gram overlap with the human translation
- Precision-based metric

Papineni et al., 2002

Precision, Recall and F-1

- True Positives, True Negatives, False Positives and False Negatives

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Of all the items in the prediction, how many match the ground truth

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

Of all the items in the ground truth, how many are correctly predicted

$$F_1 = \frac{2 * PR}{P + R}$$

Harmonic Mean of Precision and Recall

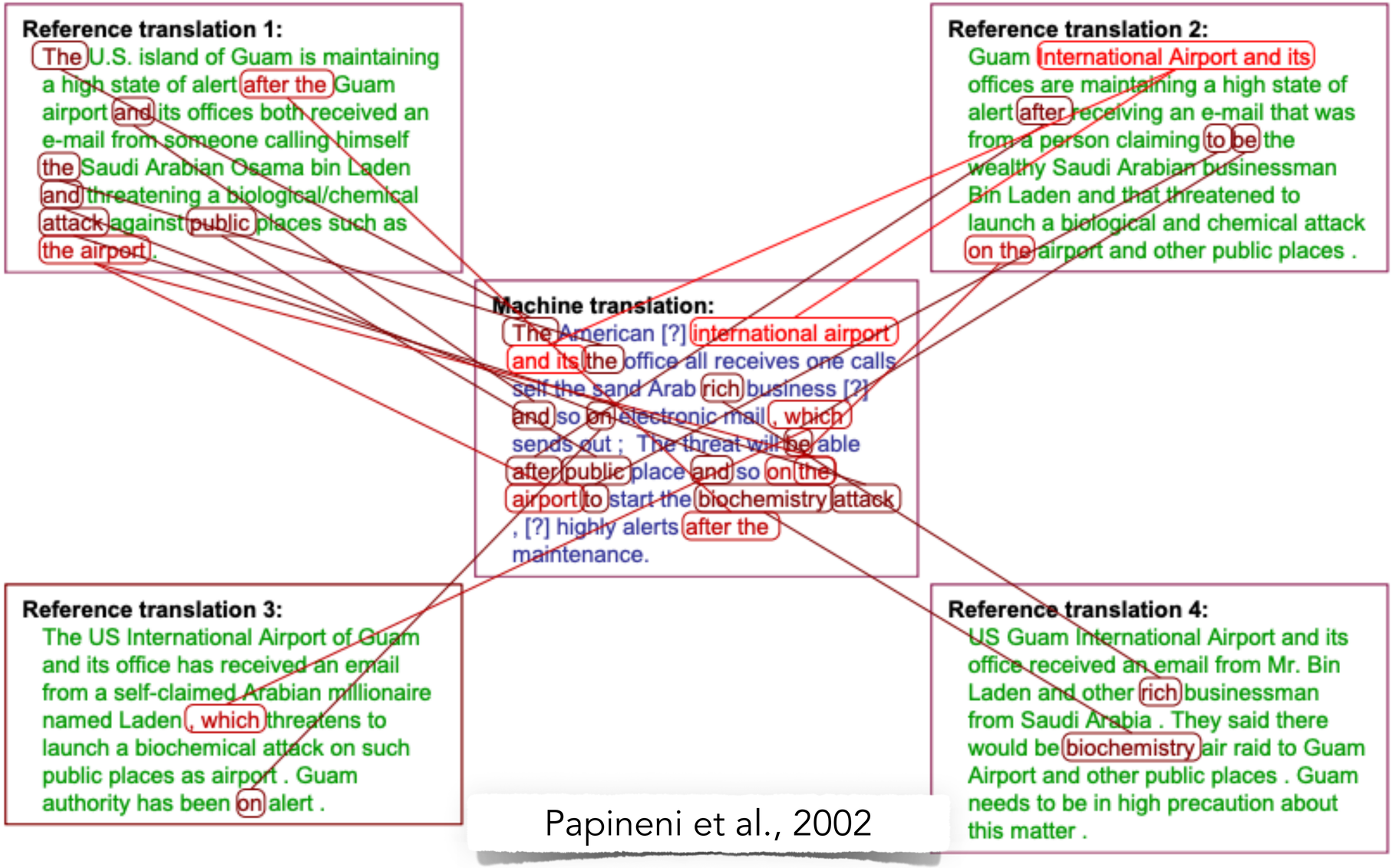
Different value for different classes!

BLEU: Details

- Purely precision-based rather than combining precision and recall.
- BLEU score for a corpus of candidate translation sentences is a function of
  - the n-gram word precision over all the sentences
  - combined with a brevity penalty computed over the corpus as a whole.
- Consider a corpus composed of a single sentence
  - The unigram precision for this corpus is the percentage of unigram tokens in the candidate translation that also occur in the reference translation, and ditto for bigrams and so on, up to 4-grams
  - It computes this n-gram precision for unigrams, bigrams, trigrams, and 4-grams and takes the geometric mean
- Because BLEU is a word-based metric, it is very sensitive to word tokenization, making it impossible to compare different systems if they rely on different tokenization

Papineni et al., 2002

BLEU: Example



Papineni et al., 2002

ROUGE

- Stands for “Recall-Oriented Understudy for Gisting Evaluation”
- Originally created for evaluating automatic summarization as well as machine translation
- Comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced)
- Four variants:
  - ROUGE-N
  - ROUGE-L
  - ROUGE-S
  - ROUGE-W

ROUGE: A Package for Automatic Evaluation of Summaries (Lin, 2004)

ROUGE: Details

- **ROUGE-N**: measures **unigram**, **bigram**, **trigram** and higher order n-gram overlap
  - n-gram recall between a candidate summary and a set of reference summaries
- **ROUGE-L**: measures **longest matching sequence** of words using LCS.
  - Does not require consecutive matches but in-sequence matches that reflect sentence level word order.
  - Since it automatically includes longest in-sequence common n-grams, you don't need a predefined n-gram length.
- **ROUGE-S**: Is any pair of words in a sentence in order, allowing for arbitrary gaps.
  - Also be called skip-gram concurrence.
  - For example, **skip-bigram** measures the overlap of word pairs that can have a maximum of two gaps in between words. As an example, for the phrase *"cat in the hat"* the skip-bigrams would be *"cat in, cat the, cat hat, in the, in hat, the hat"*.
- **ROUGE-W**: Weighted Longest Common Subsequence

ROUGE: A Package for Automatic Evaluation of Summaries (Lin, 2004)

Evaluating Generation: Other Options

- Perplexity!
- Model-based Metrics (BERTScore, BARTScore, Word Mover's Distance, BLEURT)
  - Use learned representations of words and sentences to compute semantic similarity between generated and reference texts
  - No more n-gram bottleneck because text units are represented as embeddings!
  - The embeddings are pretrained, distance metrics used to measure the similarity can be fixed
- Automatic metrics fall short of matching human decisions
- So, Human Evaluation!

Human Evaluation

- Ask humans to evaluate the quality of generated text
  - Along specific axes: fluency, coherence / consistency, factuality and correctness, commonsense, etc.
  - Mostly done via crowdsourcing
- Human judgments are regarded as the gold standard
- Of course, we know that human eval is slow and expensive
- Beyond the cost of human eval, it's still far from perfect:
  - Humans Evaluation is hard:
    - Results are inconsistent / not reproducible
    - Can be subjective!
    - Misinterpret your question
    - Precision not recall

Rising Popularity: Automatic Evaluation

AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback

Yann Dubois* Stanford
Xuechen Li* Stanford
Rohan Taori* Stanford
Tianyi Zhang* Stanford
Ishaan Gulrajani Stanford
Jimmy Ba University of Toronto
Carlos Guestrin Stanford
Percy Liang Stanford
Tatsunori B. Hashimoto Stanford

Cheap and theoretically consistent with human evaluation. BUT… reliability? Models evaluating their own generations may lead to weird mode collapsing effect

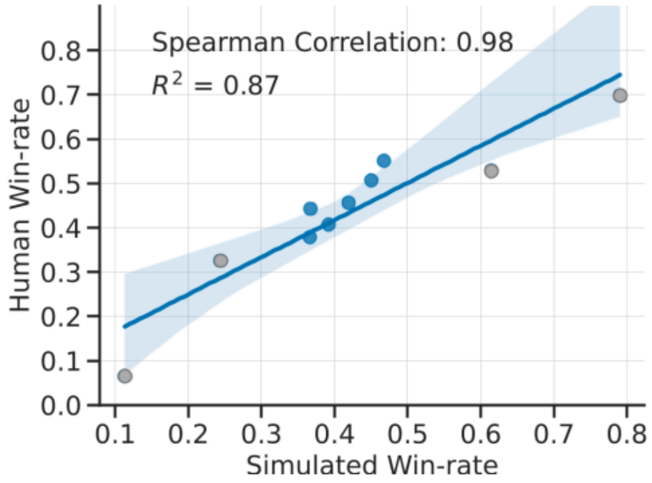


Figure 3: The ranking of methods trained and evaluated in AlpacaFarm matches that of methods trained and evaluated in the human-based pipeline. Each point represents one method M (e.g. PPO). The x-axis shows the simulated evaluation (win-rates measured by $p_{\text{sim}}^{\text{eval}}$) on methods trained in simulation M_{sim}. The y-axis shows human evaluation (win-rates measured by p_{human}) on methods trained with human feedback M_{human}. Gray points show models that we did not train, so their x and y values only differ in the evaluation (simulated vs human). Without those points, we have $R^2 = 0.83$ and a Spearman Correlation of 0.94.

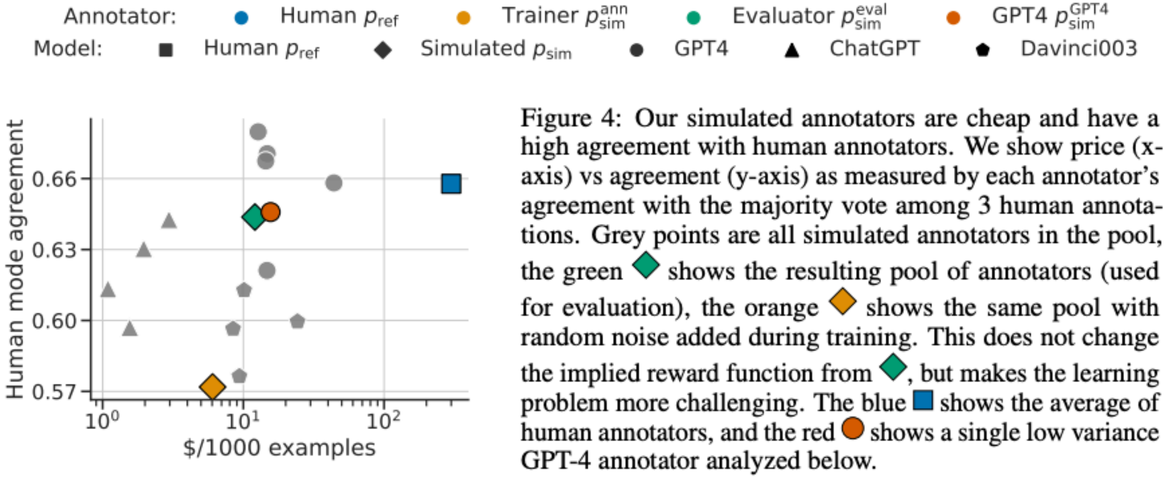


Figure 4: Our simulated annotators are cheap and have a high agreement with human annotators. We show price (x-axis) vs agreement (y-axis) as measured by each annotator's agreement with the majority vote among 3 human annotations. Grey points are all simulated annotators in the pool, the green ◆ shows the resulting pool of annotators (used for evaluation), the orange ◆ shows the same pool with random noise added during training. This does not change the implied reward function from ◆, but makes the learning problem more challenging. The blue ■ shows the average of human annotators, and the red ● shows a single low variance GPT-4 annotator analyzed below.

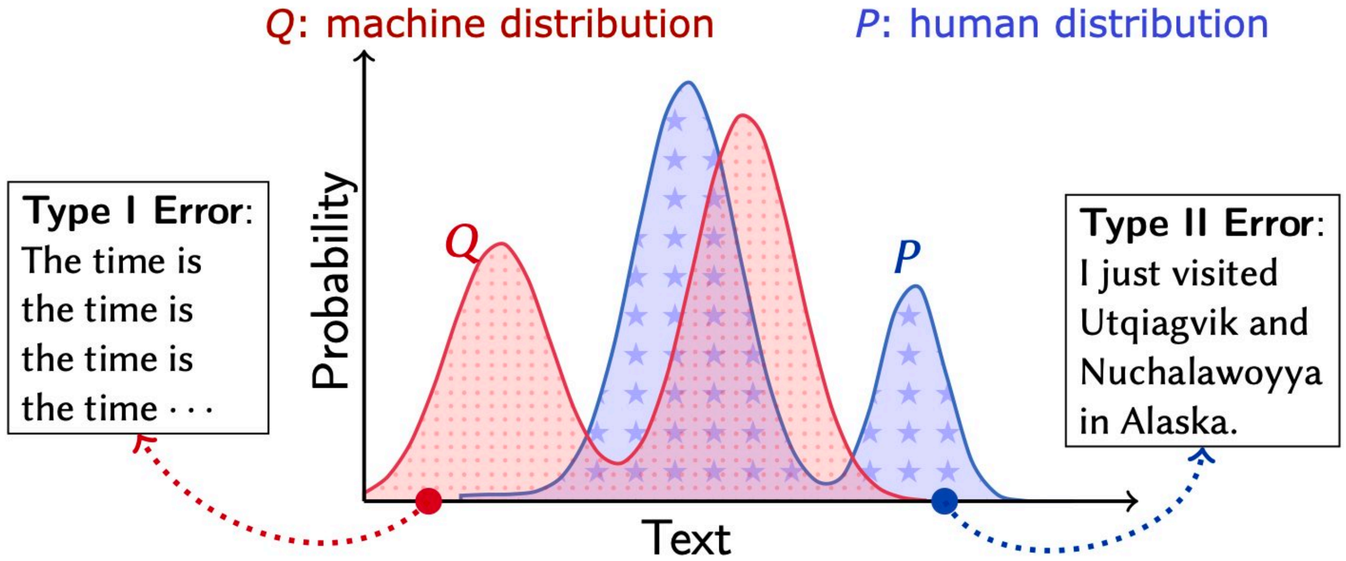
Evaluating Systems without References

- Compare human / natural language distributions to model-generated language distributions
- Divergence between these two distributions can be measured by MAUVE

MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers

Krishna Pillutla[1] **Swabha Swayamdipta**[2] **Rowan Zellers**[1] **John Thickstun**[3]
Sean Welleck[1,2] **Yejin Choi**[1,2] **Zaid Harchaoui**[4]

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence
[3]Department of Computer Science, Stanford University
[4]Department of Statistics, University of Washington



Natural Language Generation: Parting Thoughts

- Once trained, language models can be very powerful
  - The power only increases with scale
- So much so that most of our tasks in natural language can be seen as sequence completion tasks
  - Decoding Algorithms thus play a critical role
- How can you make LLMs do tasks (follow instructions)? **Instruction-Tuning and Preference-Tuning**
- **Prompting (or In-Context / Few-Shot Learning):** the ability to do many tasks with no gradient updates and no / a few examples, by simply:
  - Specifying the right sequence prediction problem
  - You can get interesting zero-shot behavior if you're creative enough with how you specify your task!

Next: Finetuning LLMs and Prompting