KYUNG HEE UNIVERSITY

Efficient Multimodal Intelligence Lab

2026S-NLP Study

Pretraining & Natural Language Generation

July 26, 2026

Dongwon Lee

Agenda

- Language Modeling & Pre-trained Architectures
- Decoder, Training
- Autoregressive Generation

Word Structure and Subword Models

The Finite Vocabulary Problem

- (Word2Vec, GloVe)
- (**vocab**) -> **UNK(unknown token)**

word		vocab mapping	
hat	→	pizza (index)	
learn	→	tasty (index)	
taaaaasty	→	UNK (index)	- (misspelling)
laern	→	UNK (index)	- (variation)
Transformerify	→	UNK (index)	- (novel item)

UNK

Word Structure and Subword Models

(morphology)

- (morphology) → ,

스와힐리어 동사 ambia (= to tell)

수백 개의 활용형이 존재하며, 각 활용형이 시제·서법·한정성·부정·목적어 정보까지 한 단어에 담는다.

nilikuambia	sikuambia
tutaambia	hatungeliambia
wangeambia	mngalimwambia

... 전부 같은 어근 -ambia 의 활용형

Problem

-
-
- UNK

Solution - The Byte-Pair Encoding Algorithm (BPE)

- (, ,) -> subword
- **(subword token)** vocabulary
- **Byte-pair encoding**: subword vocabulary

Solution - The Byte-Pair Encoding Algorithm (BPE)

1. " (end-of-word)" vocabulary

1. (corpus) **'a, b'** **'ab'** **subword**

1. **vocab** , subword

- vocab **subword**

word		vocab mapping
hat	→	pizza (index)
learn	→	tasty (index)
taaaaasty	→	UNK (index)
laern	→	UNK (index)
Transformerify	→	UNK (index)

word		vocab mapping
hat	→	hat
learn	→	learn
taaaaasty	→	taa## aaa## sty
laern	→	la## ern##
Transformerify	→	Transformer## ify

pretrained word embeddings

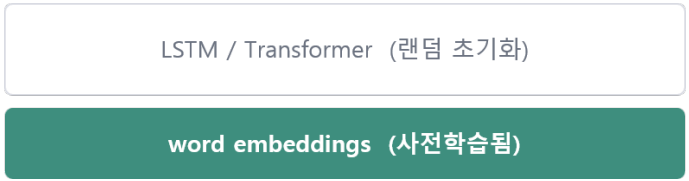
- (word2vec).

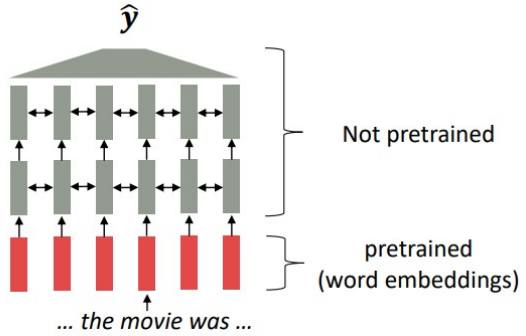
LSTM Transformer task

Problem

-
(LSTM Transformer —)

- **downstream task**

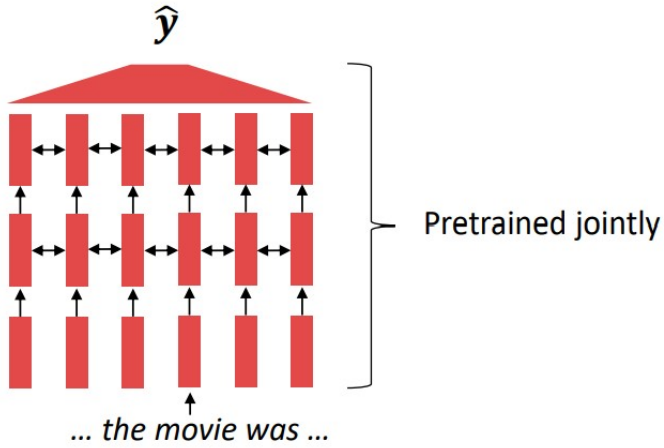




pretraining whole models

- **pretraining**
- Pretraining , **(reconstruct)**
- (representation)
-
-

전체 네트워크 + 임베딩
한꺼번에 사전학습



What can we learn from reconstructing the input?

- I went to the ocean to see the fish, turtles, seals, and _____.

  Stanford University is located in _____, California.

  … the value I got was the sum total of the popcorn and the drink. The movie was ___.

  I put ___ fork down on the table.

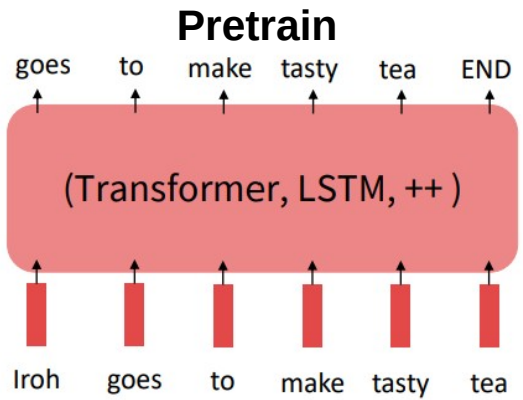
  The woman walked across the street, checking for traffic over ___ shoulder.

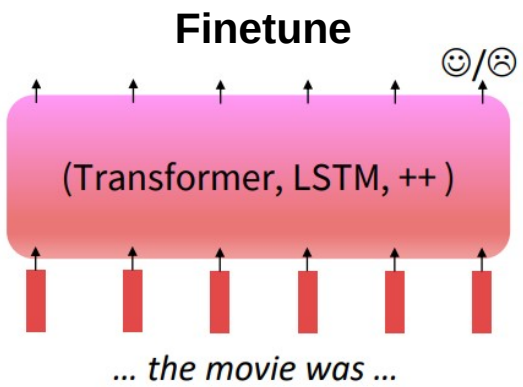
  Iroh went into the kitchen to make some tea. … Zuko left the _____.

, , , , ,

Pretraining / Finetuning

Pretraining NLP





- ,
- task
- 1 , 2 task

Stochastic gradient descent and pretrain/finetune

pretraining finetuning ?

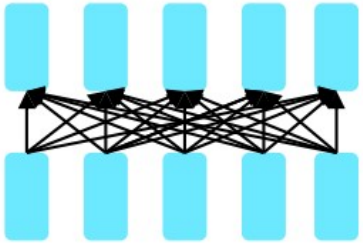
Pretrain → $\min_\theta \mathcal{L}_{\text{pretrain}}(\theta)$ → $\hat{\theta}$

Finetune → $\hat{\theta}$ → $\min_\theta \mathcal{L}_{\text{finetune}}(\theta)$

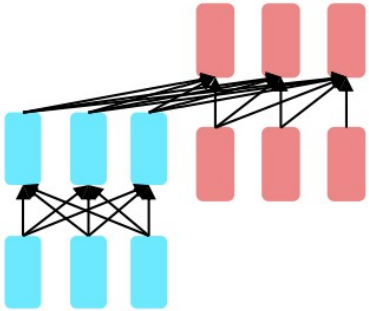
- , -
- finetuning SGD , local minima

Pretraining for three types of architectures

•

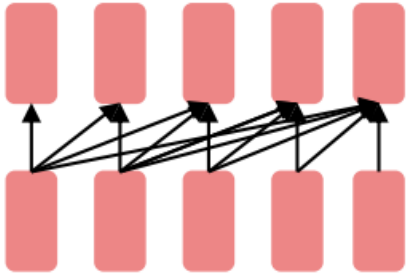


Encoders



Encoder-Decoders

,



Decoders

( )

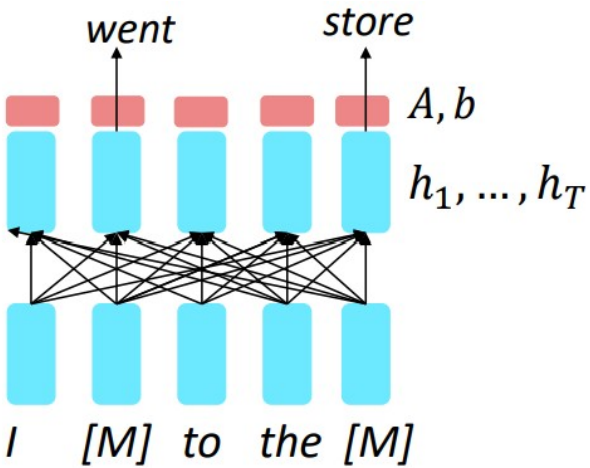
Pretraining encoders: what pretraining objective to use?

Masked LM -

- (bidirectional) -> language modeling
- : **[M SK]** ,
  .
- x

$$p_\theta(x|\tilde{x})$$

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$

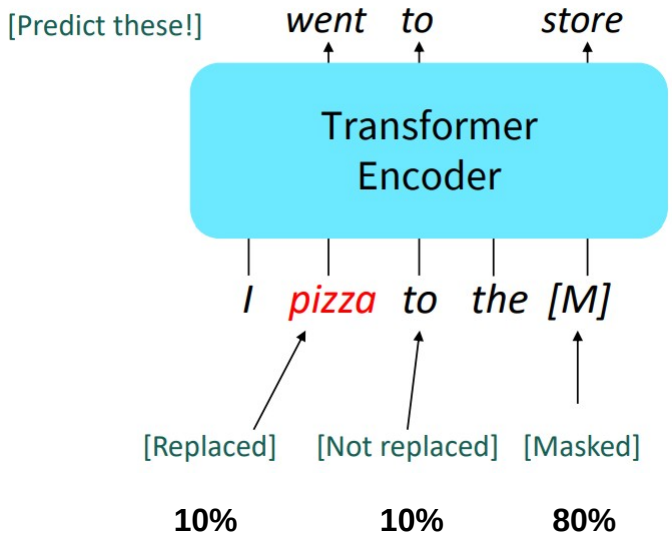


BERT: Bidirectional Encoder Representations from Transformers

- Masked LM objective ,
  Transformer BERT

step1. 15% .

step2. 80% [MASK]
10%
10%



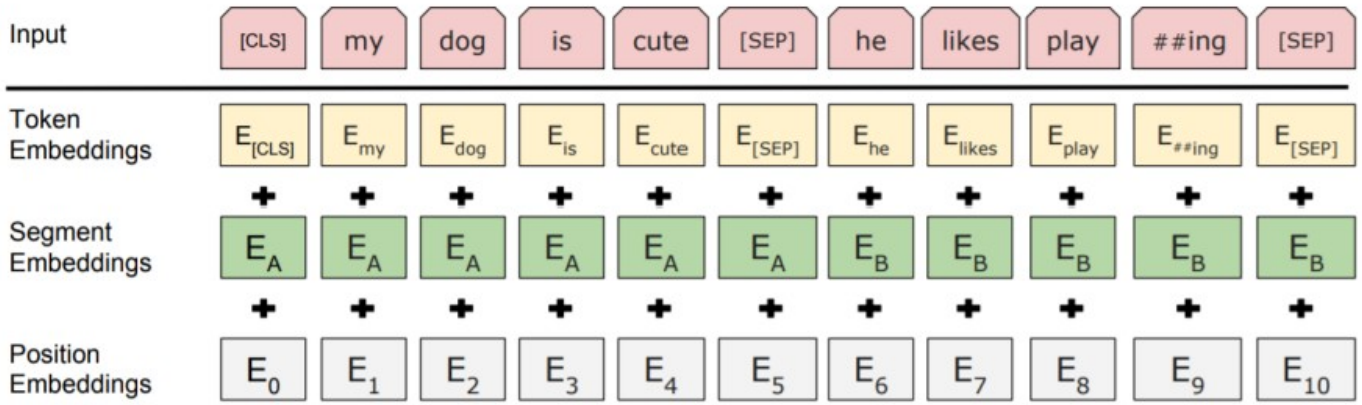
If only mask: [MASK]
.

[MASK] ,

BERT: Bidirectional Encoder Representations

- chunk [SEP]
- = **Token Embedding + Segment Embedding + Position Embedding**

chunk(A)
chunk(B)



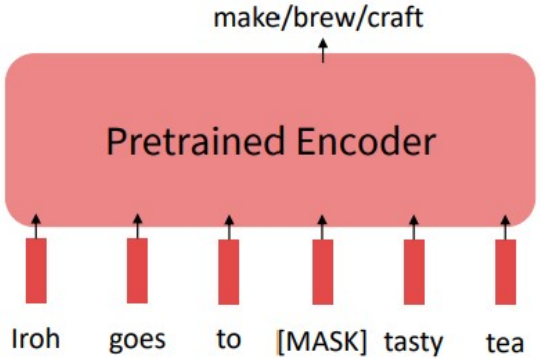
BERT —

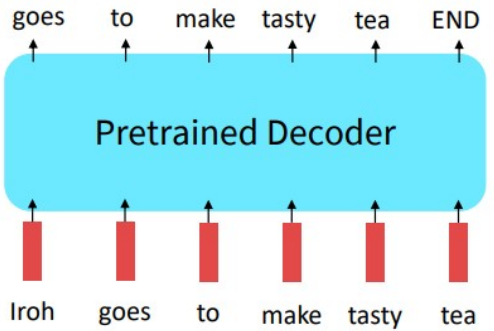
- **BERT finetuning task**

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Limitations of pretrained encoders

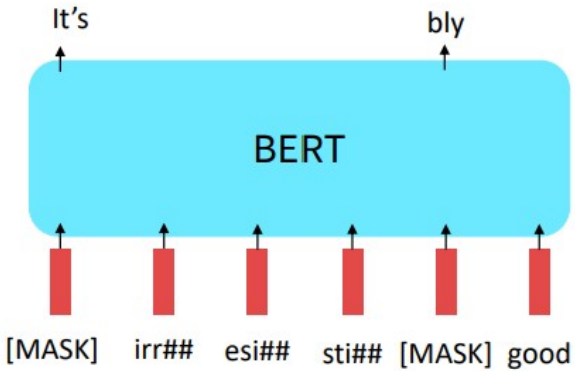
- task -> **pretrained decoder**

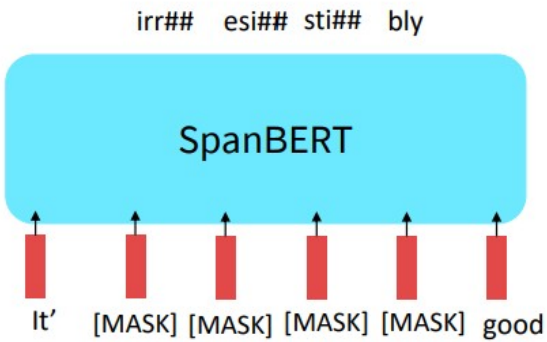




- , [MASK]
-
-
-

Extensions of BERT





RoBERTa

- ,

SpanBERT

- (span)
-

Pretraining encoder-decoders

- encoders, decoders (hidden state)

$$h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$$

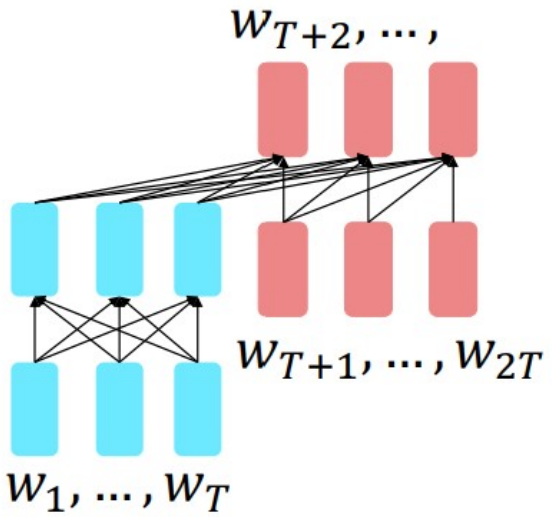
step1. hidden state

$$h_{T+1}, \dots, h_2 = Decoder(w_1, \dots, w_T, h_1, \dots, h_T)$$

step2. 1~t hidden state

$$y_i \sim Ah_i + b, i > T$$

step3. hidden state



Pretraining encoder-decoders,

span corruption

- , span corruption. - Model: T5

1 :

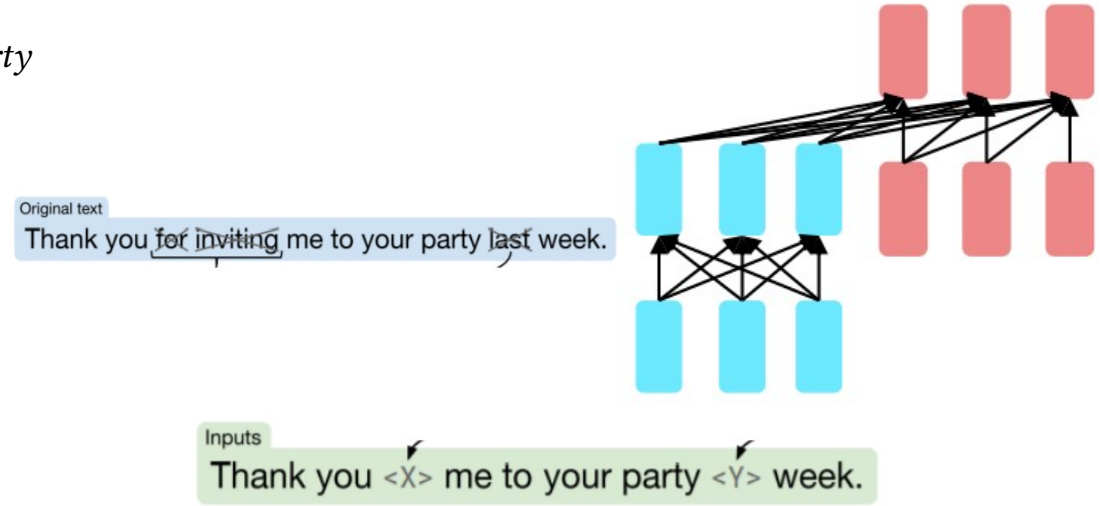
: *"Thank you for inviting me to your party last week."*

2 : span( )

3 : span placeholder→

Input

Inputs: Thank you <X> me to your party <Y> week.



Pretraining encoder-decoders,

span corruption

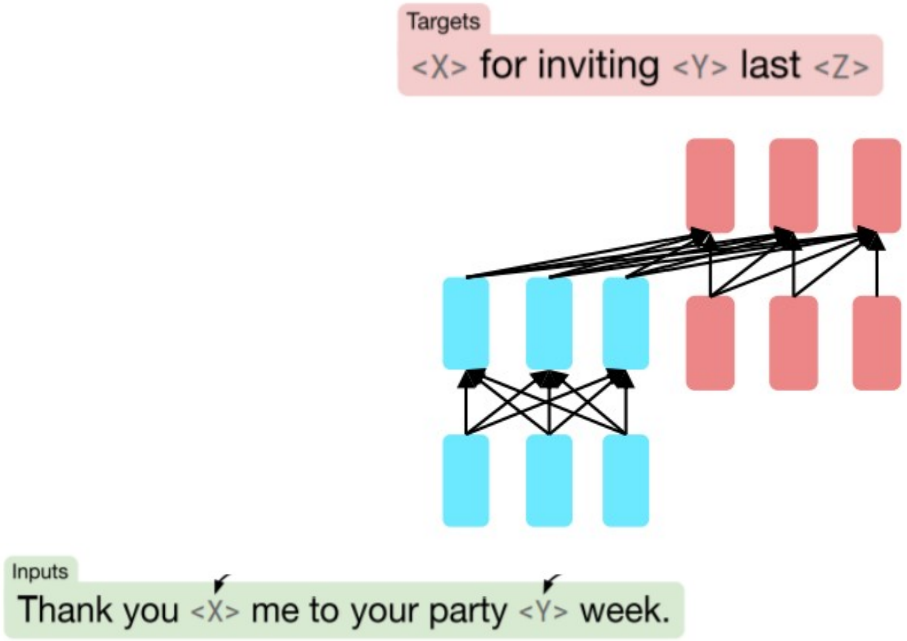
4 : Target( )

Targets: <X> for inviting <Y> last <Z>

5 : Target

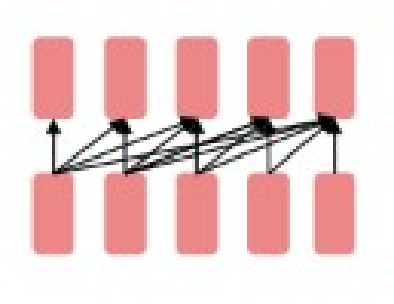
, +

6 : target Loss



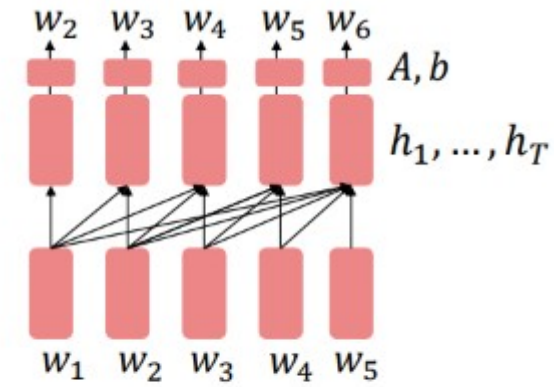
Decoder Only model

- Nice to generate from; can't condition on future words



Decoder Only model

why??



[Note how the linear layer has been pretrained.]

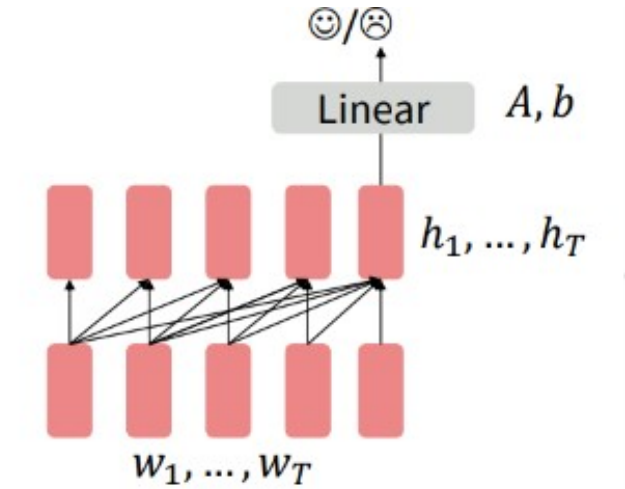
$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

- **Decoder Only Model ?**
  -
  - Pretraining
    -
      ok
  - NLP Prompt
    - ->

w_1 ~ w_t-1 hidden vector h_1~ h_t-1
h_t-1 w_t !

Decoder Only model

Fine-Tuning - classification



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$y \sim Ah_T + b$$

- ?
  - Linear Layer
  - !
    ( )
    ..

w_1 ~ w_T hidden state h_1~ h_T
extract state h_T class

GPT1

Fine-Tuning - classification

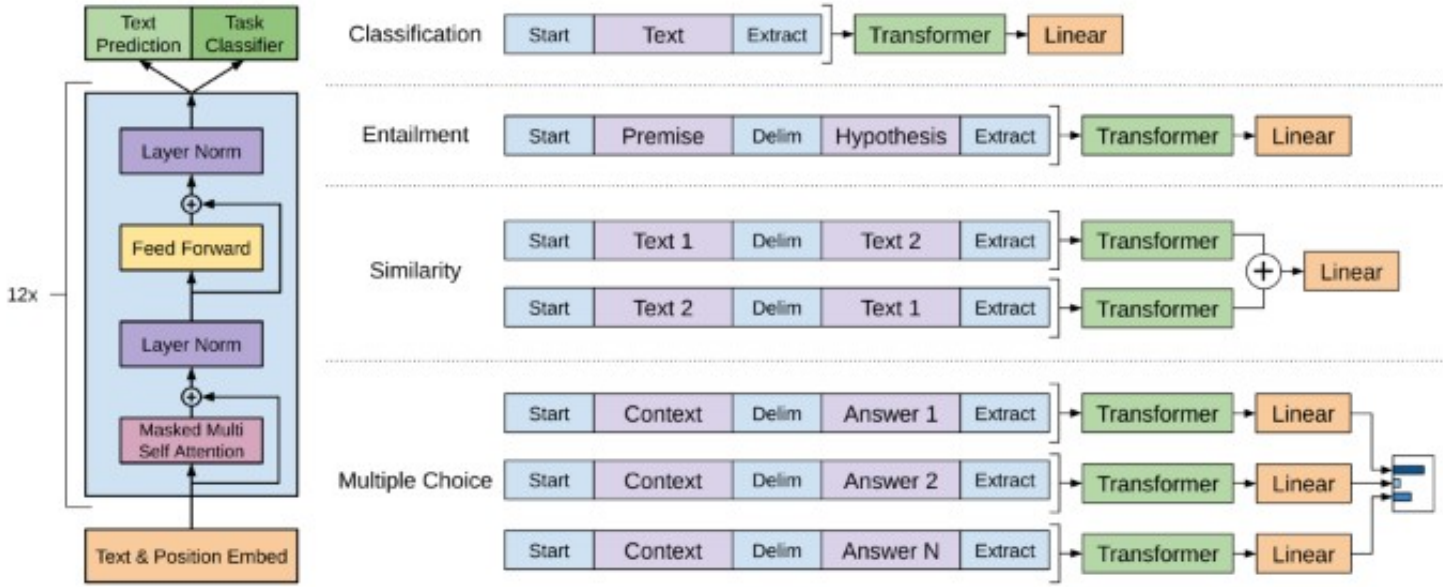


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT 3

no finetuning - prompt based In-context Learning

- **Classifier ?**
  ‘Positive’ / ‘Negative’ “ ” ?
  - - ,
    “ ” task ?
    update , gradient ?

- **In-context Learning ?**
  - , context task

Input (prefix within a single Transformer decoder context):

“ thanks -> merci
hello -> bonjour
mint -> menthe
otter -> ”

Output (conditional generations):

loutre...”

CoT Prompting

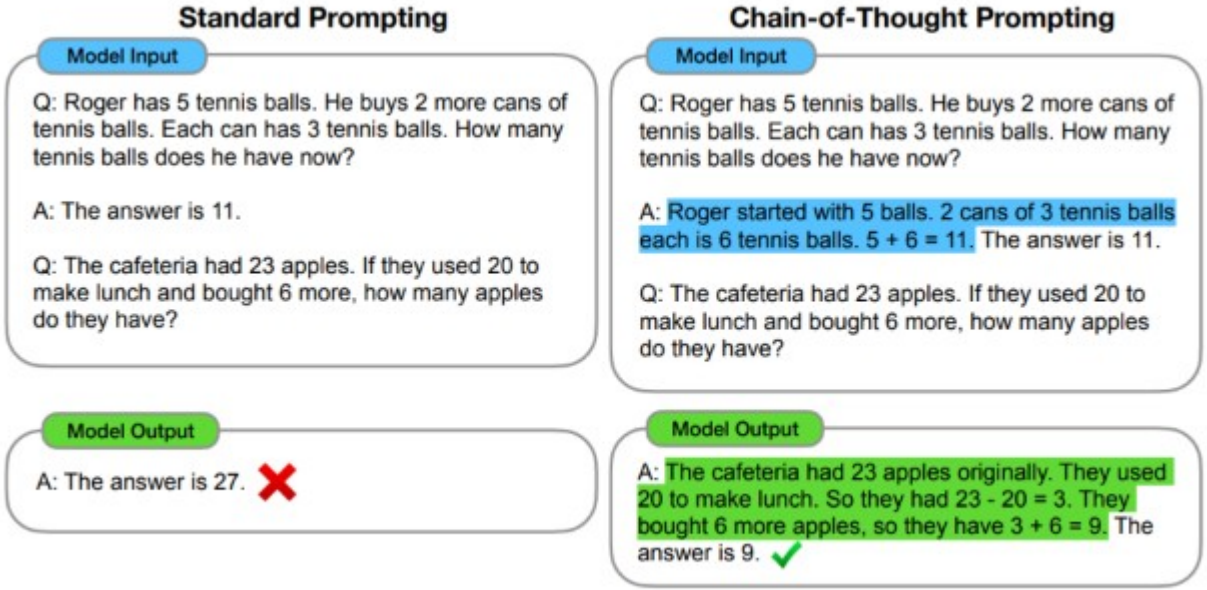
Chain of Thought

- **Chain of Thought ?**
  - -> .
    Chain of Thought .
- **(Reasoning) ?**
  -
    1.
    2.
    3.
    4.

CoT Prompting

Chain of Thought

- **Chain of T**
  - ->
-
  -
    1.
    2.
    3.
    4.



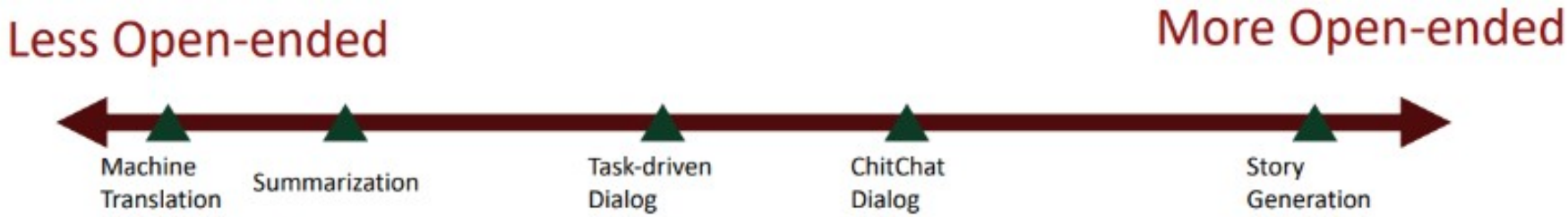
Natural Language Generation

NLP( ) = NLU( ) + **NLG( )**

-
  - Fluent, Coherent, Useful .
  - , , ,
    .

Natural Language Generation

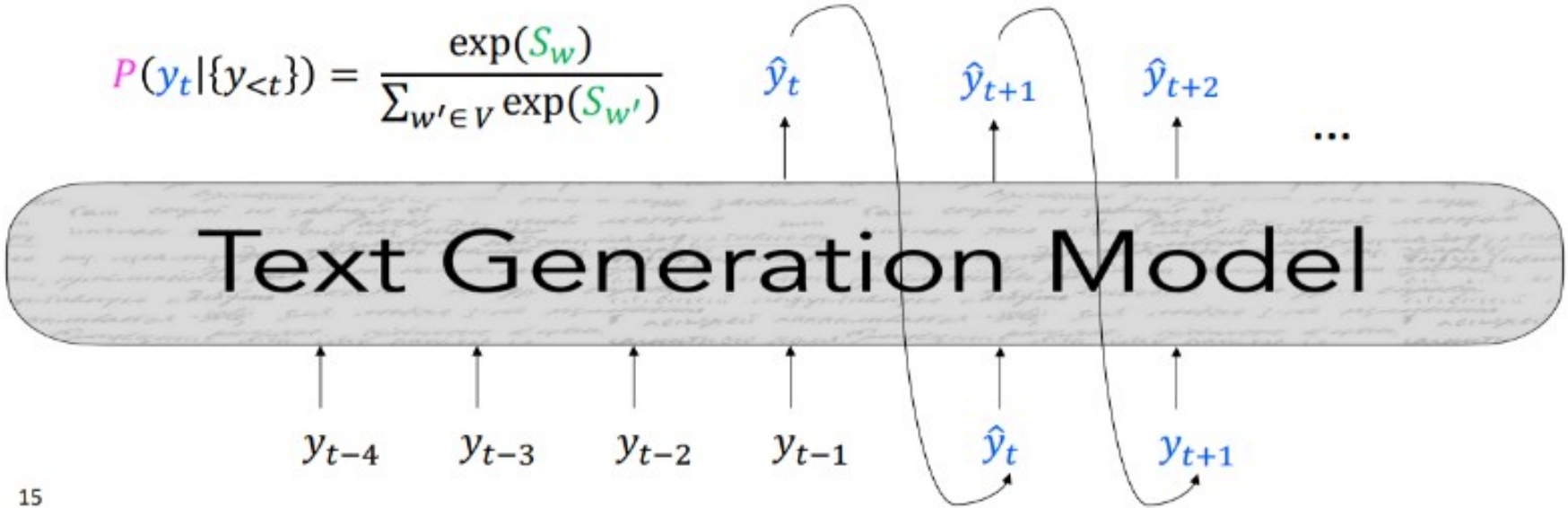
categorization



- **Open-ended?**
  -
  - - I love you ‘ ’ ‘ ’
  - - “ ?”
- **Entropy**
  - Less Open-ended
    More Opne-ended .

Train Language Model

Softmax & MLE



15

Train Language Model

Softmax & MLE

I want to eat _____

1. meat… P(meat| I want to eat)=6.2
2. pasta … P(salad | I want to eat)=5.8
3. spoon … P(spoon | I want to eat)=1.3
4. watch … P(watch | I want to eat)=-1.4

=> softmax meat

Train Language Model

Softmax & MLE

I want to eat meat with my friend.

- P(I | START)
- P(want | I)
- P(to | I want)
- …
- P(END | I want to eat meat with my friend.)

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t^* | \{y^*\}_{<t})$$

negative log-likelihood =>

Train Language Model

Softmax & MLE

I want to eat salad _____

I want to eat salad with my vegan friend. ( )

meat , vegan

-> !

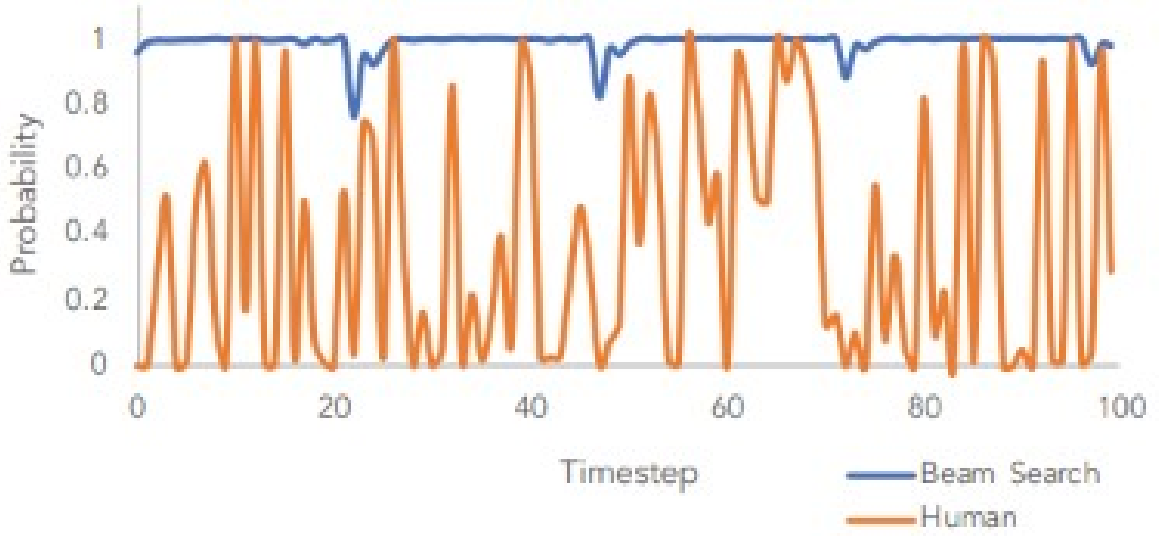
teacher forcing .

Exposure Bias

teacher forcing

-
  - :
-
  - :

-> !



Exposure Bias Solutions

Scheduled Sampling & DAgger

- **Scheduled Sampling**
  - p
  -
    -> p !
- **DAgger (Dataset Aggregation)**
  -
    .
    training set !
  -

=> .

Exposure Bias Solutions

Retrieval Augmentation & Reinforcement Learning

- **Retrieval Augmentation**
  - , reference
  - The food was excellent, but the service was slow.
    The food was good, but the service was terrible. .

- **Reinforcement Learning**
  - reward !
  - reward ?
    - BLEU - reference n-gram overlap ( )
    - ROUGE
    - RLHF -

Decoding

Intro

- **Inference time: decoding**

$$P(y_t = w|\{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

$$S = f(\{y_{<t}\})$$

$f(.)$ is your model

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

$g(.)$ is your decoding algorithm

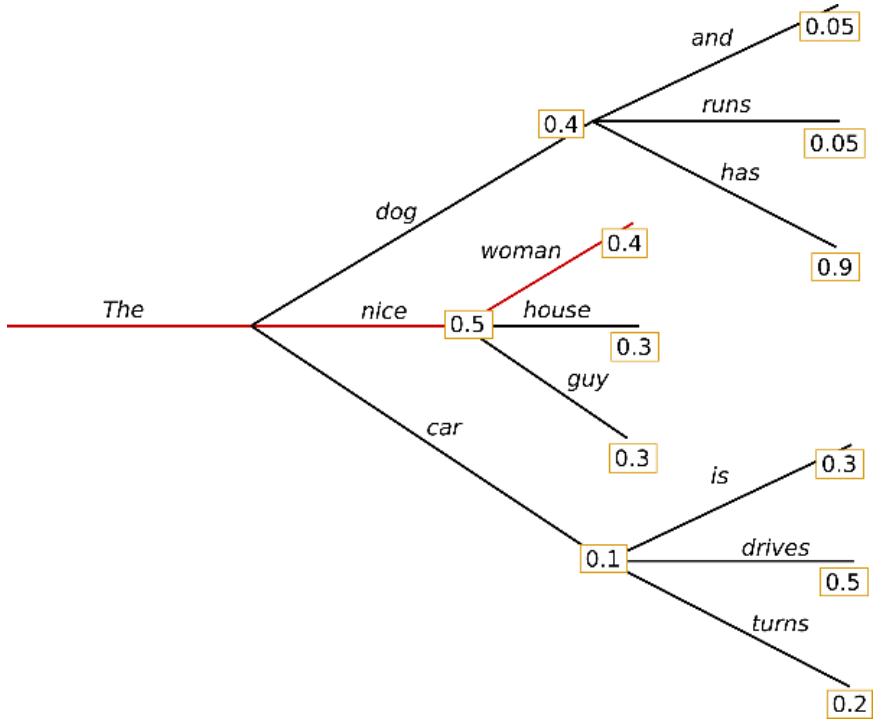
- **Decoding Strategies**
  - **Maximum Probability-based Decoding**
  - **Sampling-based Decoding**

Greedy Decoding

Maximum Probability-based Decoding

- **Take the argmax at every step — and never look back**

$$\hat{y}_t = \underset{w \in V}{\operatorname{argmax}} P(y_t = w | y_{<t})$$

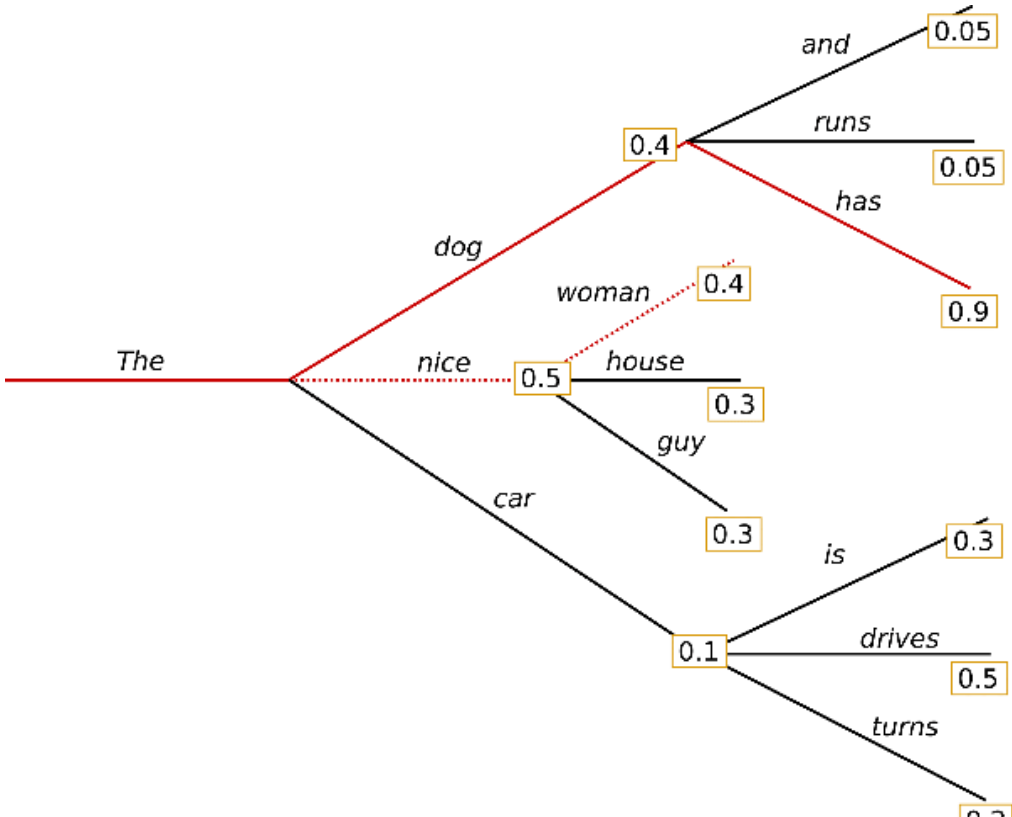


Hugging Face (huggingface.co/blog/how-to-generate)

Beam Search

Maximum Probability-based Decoding

- **Keep the k most probable partial sequences at each step**
  - greedy = beam search with k = 1



Hugging Face (huggingface.co/blog/how-to-generate)

Repetition

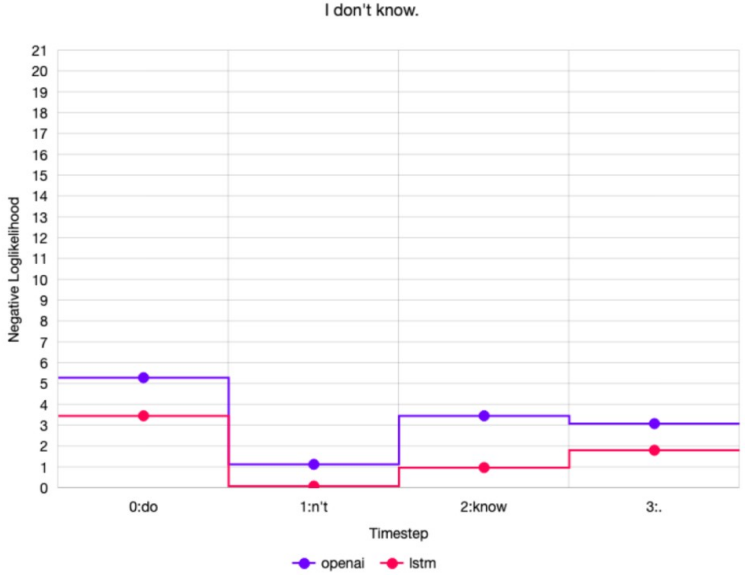
Maximum Probability-based Decoding

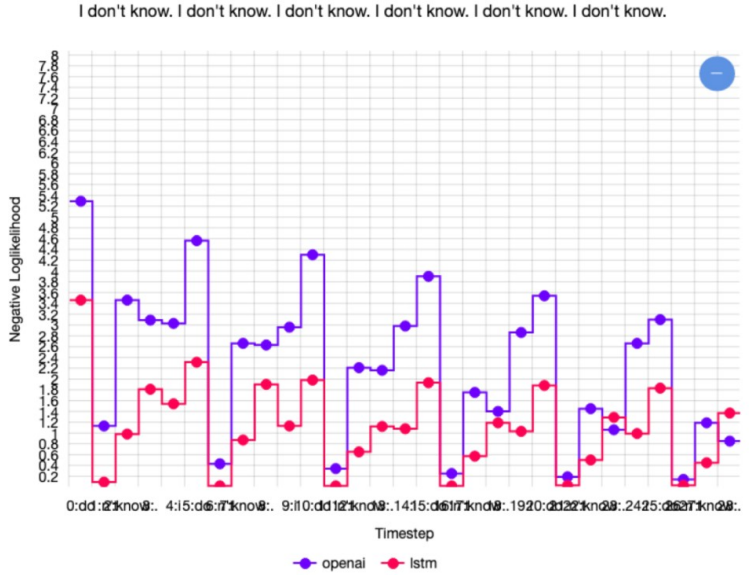
Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

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...**

- **Self-amplification effect**
  - The more a phrase repeats, the more confident the model becomes.

Negative Log Likelihood ↓ = probability ↑





Holtzman et al., *"The Curious Case of Neural Text Degeneration"* (ICLR 2020)

Repetition

Maximum Probability-based Decoding

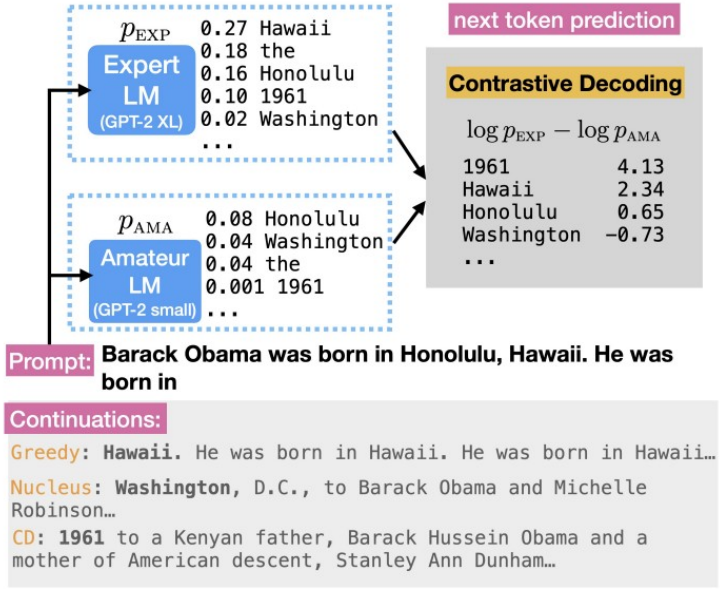
- **Problem: repetition as an artifact of the MLE objective**
  - Teacher forcing never exposes the model to its own loops
  - Token-level likelihood cannot capture sequence-level pathologies

- **Solution**
  - Heuristic: Don't repeat n-grams
  - Training Objective: Add a term to the MLE loss Unlikelihood objective (Welleck et al., 2020): penalize generation of already-seen tokens
  - Coverage loss (See et al., 2017): prevents attention mechanism from attending to the same words

Contrastive Decoding

Maximum Probability-based Decoding

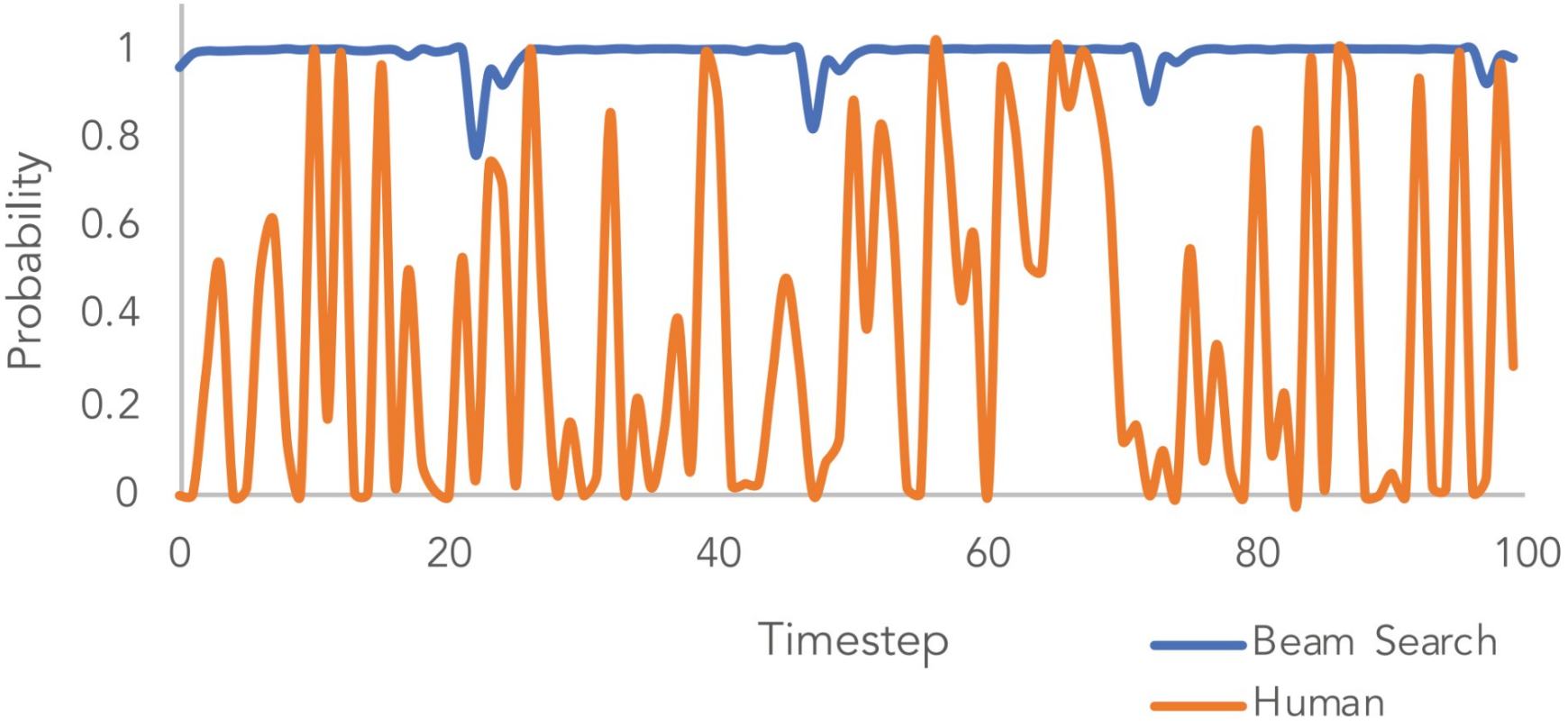
- **Contrastive decoding (Li et al, 2022)**
  - Two models, one for quality: keep what only the large model knows.
    - maximize logprob_largeLM (x) – logprob_smallLM (x)



Li et al., *"Contrastive Decoding: Open-ended Text Generation as Optimization"* (ACL 2023)

Problem of Maximum Probability-based Decoding

Maximum Probability-based Decoding



Li et al., *"Contrastive Decoding: Open-ended Text Generation as Optimization"* (ACL 2023)

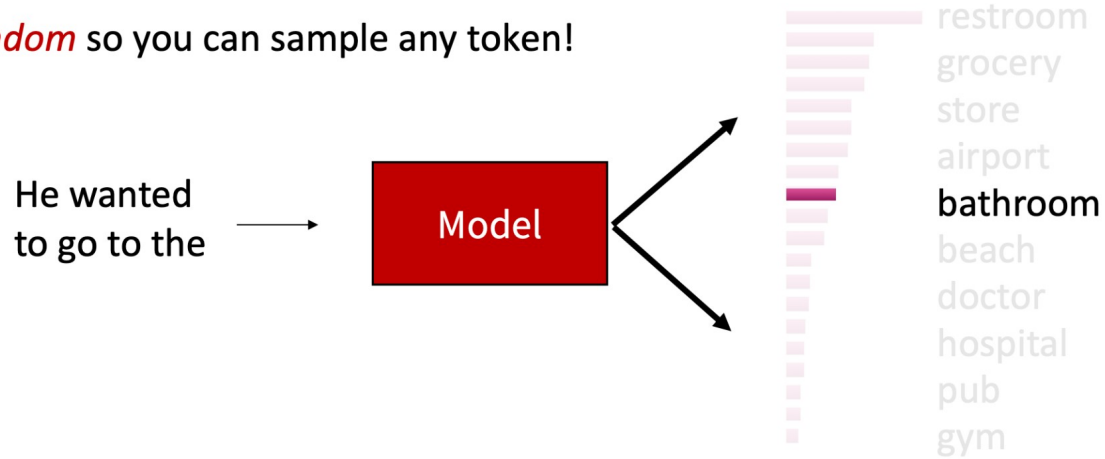
Vanilla Sampling

Sampling-based Decoding

- **Decoding as sampling, not as search**

$$\hat{y}_t = g(P(y_t|\{y_{<t}\})) \longrightarrow \hat{y}_t \sim P(y_t = w\,|\{y\}_{<t})$$

- It's *random* so you can sample any token!



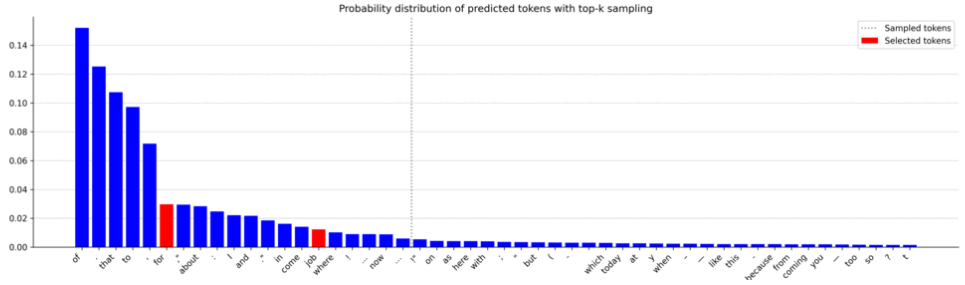
but the tail is large: rare tokens sum to real mass

Top-k Sampling

Sampling-based Decoding

Problem

- sampling from the full distribution
- the tail is unreliable

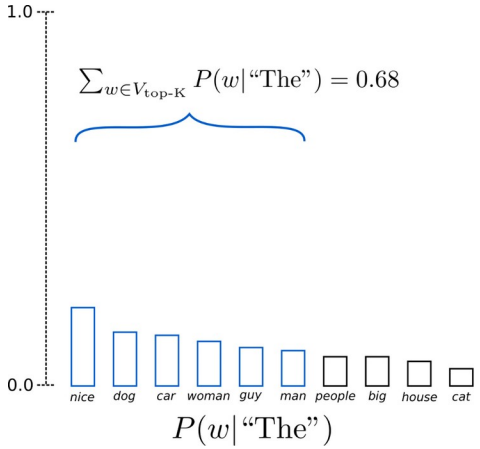


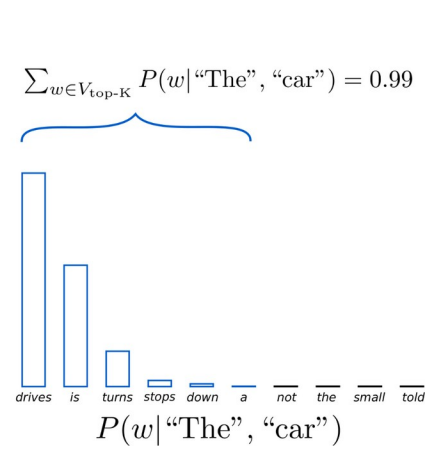
Solution

- truncate to the k most probable tokens, then renormalize

Limitation

- k is fixed, but the shape of the distribution is not





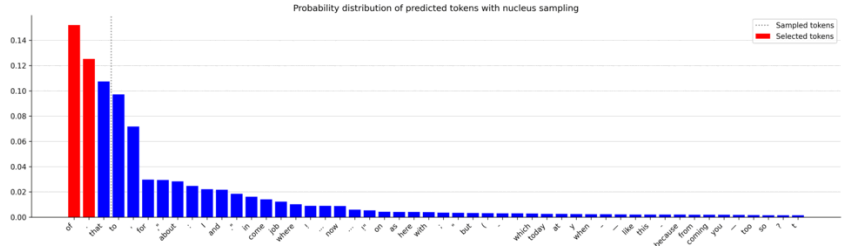
Hugging Face huggingface.co/blog/mlabonne/decoding-strategies)

Top-p (nucleus) sampling

Sampling-based Decoding

Problem

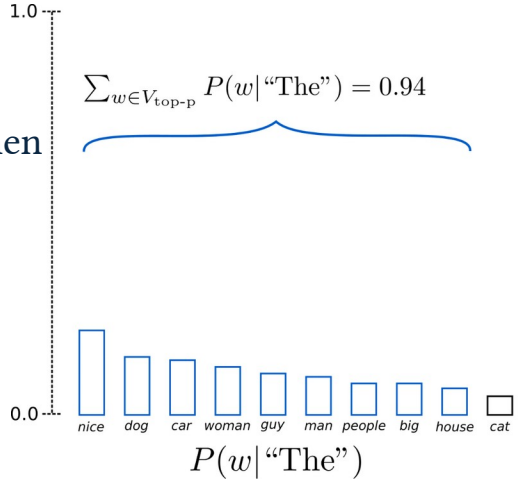
- a fixed k ignores how peaked the distribution is

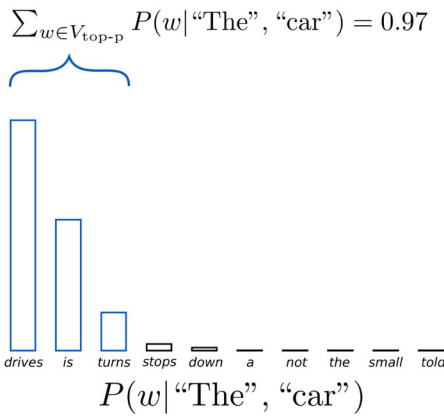


Solution

- keep the smallest set whose cumulative probability ≥ p

  ⟶ k adapts: wide when flat, narrow when peaked





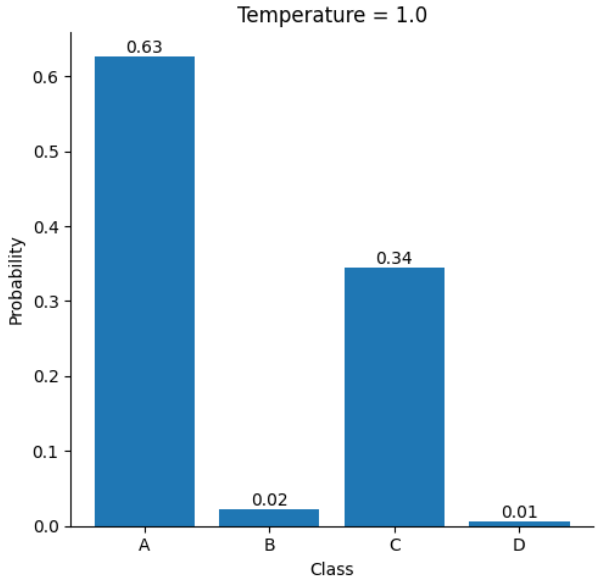
Hugging Face huggingface.co/blog/mlabonne/decoding-strategies)

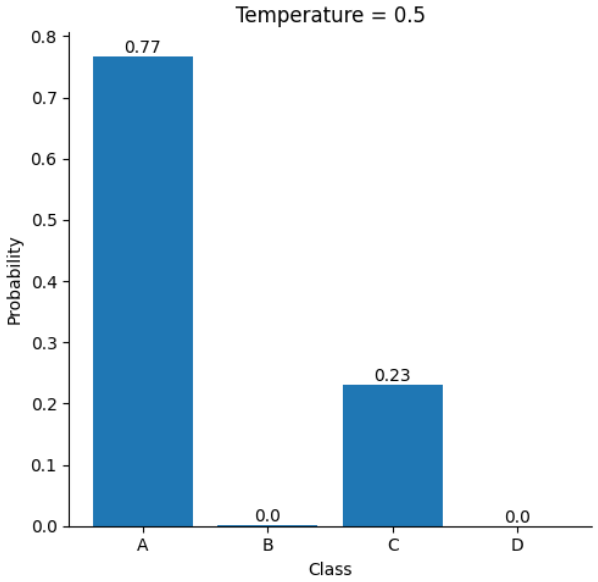
Temperature

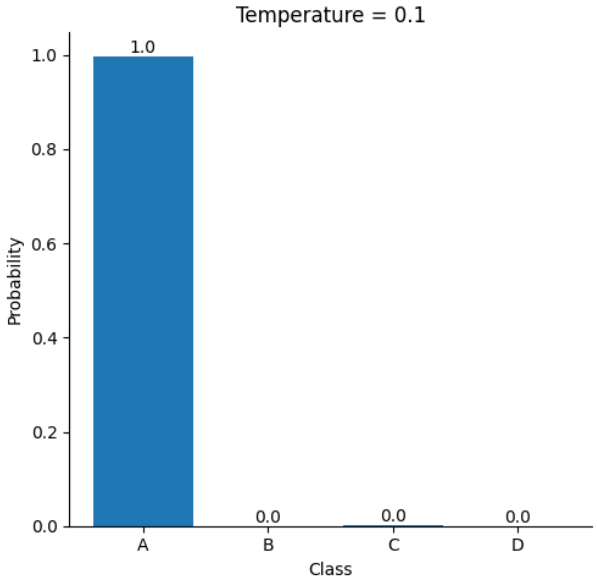
Sampling-based Decoding

- **Divide the logits by τ before the softmax**
  - τ < 1 sharpens the distribution · τ > 1 flattens it
  - Temperature reshapes the distribution before you sample from it
  - Not a truncation method — combines with top-k / top-p.

$$P_t(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$







Re-ranking

Sampling-based Decoding

Improving Decoding: Re-ranking

- Problem: What if I decode a bad sequence from my model?

- Decode a bunch of sequences
  - 10 candidates is a common number, but it's up to you
- Define a score to approximate quality of sequences and re-rank by this score
  - Simplest is to use (low) perplexity!
    - Careful! Remember that repetitive utterances generally get low perplexity.
  - Re-rankers can score a variety of properties:
    - style (Holtzman et al., 2018), discourse (Gabriel et al., 2021), entailment/factuality (Goyal et al., 2020), logical consistency (Lu et al., 2020), and many more …
    - Beware poorly-calibrated re-rankers
  - Can compose multiple re-rankers together.

Evaluating Natural Language Generation Systems

Intro

- Content overlap metrics
- Model-based Evaluation
- Human Evaluation

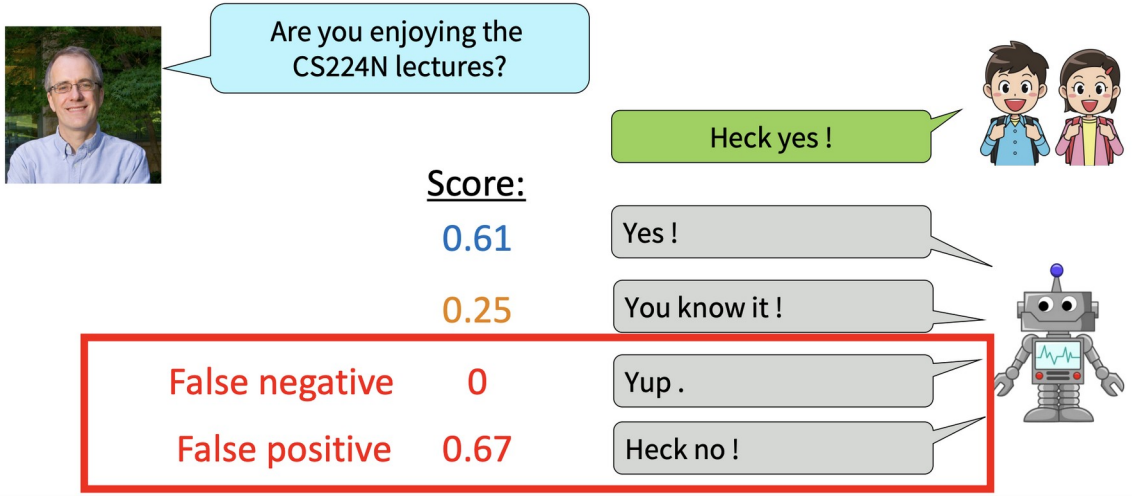
Content Overlap Metrics

- **BLEU · ROUGE — n-gram overlap with a reference**
  - No concept of semantic relatedness

    "Yup." ⟶ 0.00 (false negative)

    "Heck no!" ⟶ 0.67 (false positive)

  - Suitable for MT and summarization — the problem grows as tasks become more open-ended.



Model-based Evaluation

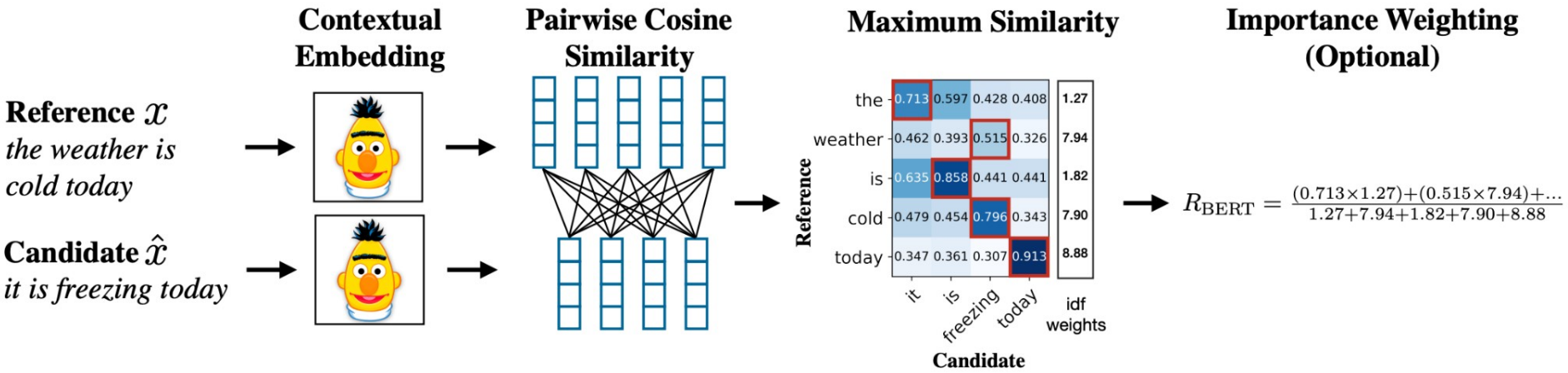
- **Problem - Content Overlap Metrics**
  - n-gram overlap has no concept of semantic relatedness
- **Solution - Model-based Evaluation**
  - Use **learned representations** of words and sentences to compute semantic similarity between generated and reference texts
  - **Embeddings -> Distance -> Score**
  - Early approaches (static embeddings)
    - Vector similarity · Word Mover's Distance · Sentence Movers Similarity → all built on word2vec / GloVe: one vector per word, regardless of context
- Question: What if the embedding itself were context-dependent?

BERTScore

Model-based Evaluation

- **Match tokens by meaning in context, not by exact string.**
  - Replace exact n-gram matching with soft alignment in contextual embedding space

Embeddings → Distance → Score



BLEURT

Model-based Evaluation

- ~~**Embeddings → Distance → Score**~~
- **(candidate, reference) → Score**

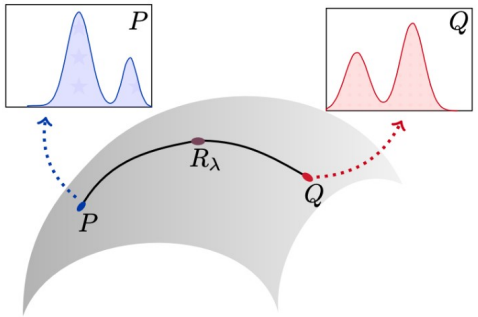
BERT ✦ regression layer

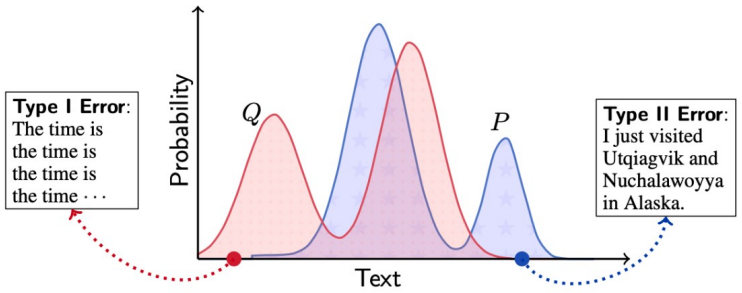
Don't design the metric — learn it.

MAUVE

Model-based Evaluation

- **Compare distributions: human text P vs. Generated text Q**
  - How: embed → k-means quantize → compare histograms (information divergence in a quantized embedding space)





  - Type I Q generates text unlikely under P → degenerate, repetitive
  - Type II Q fails to cover text likely under P → lost diversity (top-k, top-p)

Human Evaluation

- Every automatic metric so far(BLEURT, MAUVE, BERTScore) is validated against one thing:

  human judgment

So why not just ask humans?

Human Evaluation

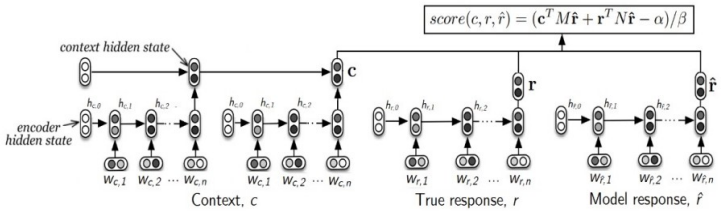
- **The gold standard**
  - slow and expensive
  - not reproducible across studies
  - measures precision, not recall

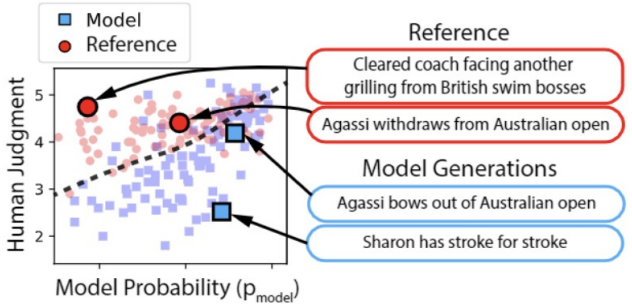
Learning from human judgment

Human Evaluation

- **ADEM: human ratings as training data**
- **HUSE: human (precision) ✦ model (recall)**

→ HUSE's two axes = MAUVE's Type I / II





ADEM:

A learned metric from human judgments for dialog system evaluation in a chatbot setting.

(Lowe et.al., 2017)

HUSE:

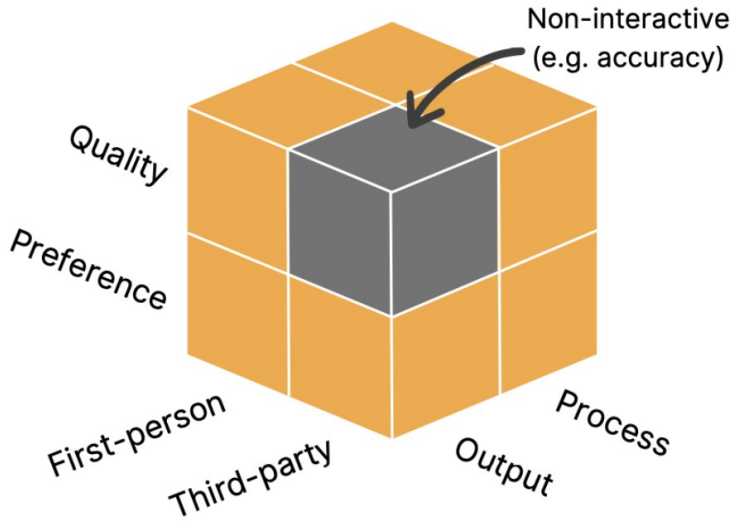
Human Unified with Statistical Evaluation (HUSE), determines the similarity of the output distribution and a human reference distribution.

(Hashimoto et.al. 2019)

Evaluating LMs by interacting with them

Human Evaluation

- **Prior work**
  - third party evaluates the output
- **Evaluating LMs by interacting with them**
  - all the other axes
    - process, not just the final output
    - first-person, not third-party
    - the human is a co-author, not a judge



Evaluating Natural Language Generation

Takeaways

- content overlap → model-based → distributional
- open-ended generation has no single correct answer
- human judgment stays the anchor

Thank You!