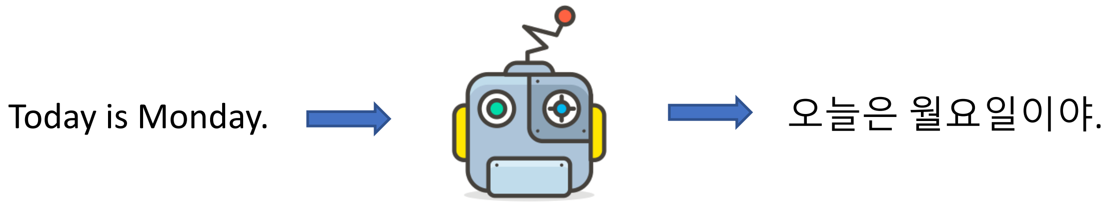
NATURAL LANGUAGE PROCESSING

Text Generation

Contents

What is Text Generation?

- Text Generation (or Natural Language Generation (NLG))
  - Given some inputs, a model generates new texts.



- Applications
  - Machine Translation
  - Open-ended Generation
  - Summarization
  - Dialogue System

Applications

- Open-ended Generation

Prompt: The Mage, the Warrior, and the Priest

Story: A light breeze swept the ground, and carried with it still the distant scents of dust and time-worn stone. The Warrior led the way, heaving her mass of armour and muscle over the uneven terrain. She soon crested the last of the low embankments, which still bore the unmistakable fingerprints of haste and fear. She lifted herself up onto the top the rise, and looked out at the scene before her. [...]

Applications

- Document Summarization

(a) Extractive Summarization

Source Text: Peter and Elizabeth took a taxi to attend the night party in the city.

While in the party, Elizabeth collapsed and was rushed to the hospital.

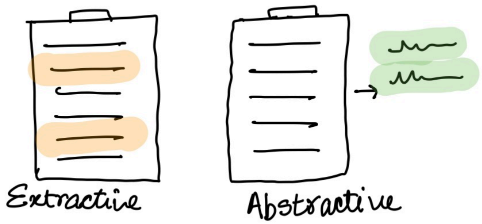
Summary: Peter and Elizabeth attend party city. Elizabeth rushed hospital.

(b) Abstractive Summarization

Source Text: Peter and Elizabeth took a taxi to attend the night party in the city.

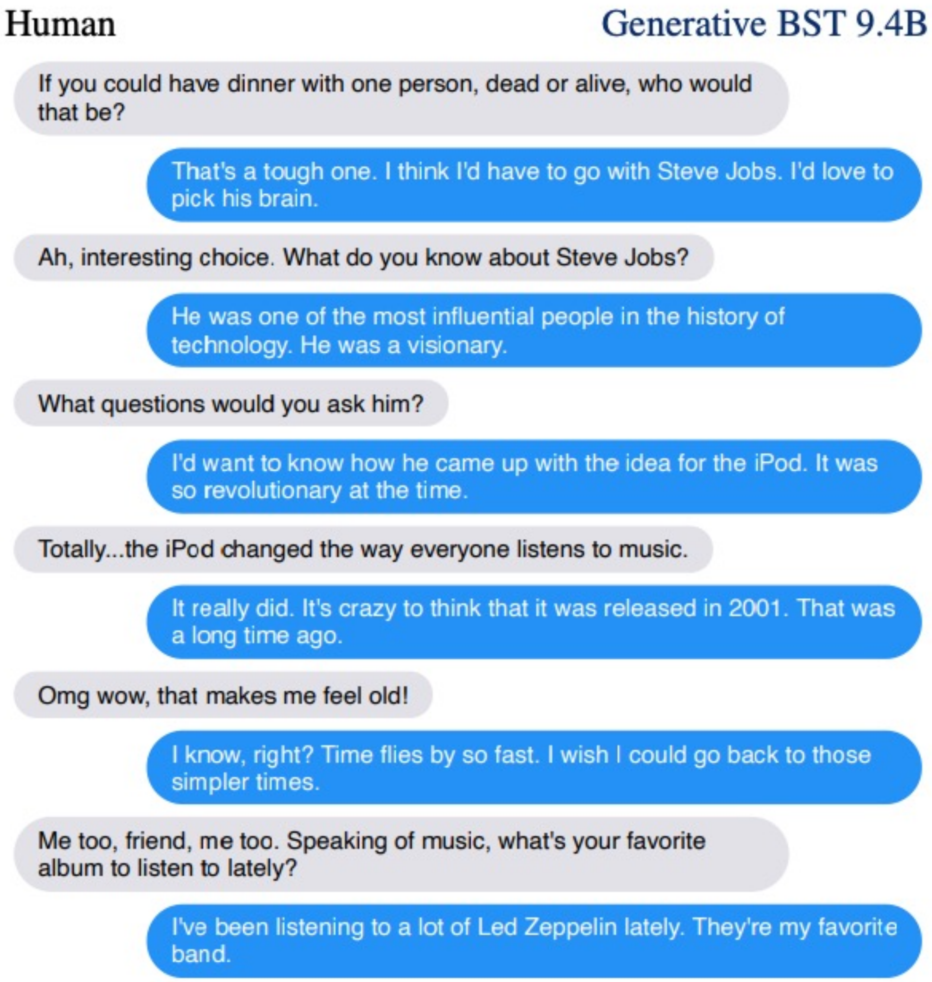
While in the party, Elizabeth collapsed and was rushed to the hospital.

Summary: Elizabeth was hospitalized after attending a party with Peter.



Applications

- Open-domain Dialogue System



Recipes for building an open-domain chatbot, 2020, Facebook

Contents

What is Text Generation?

Formulation, Training

Inference (=Testing), Decoding Strategy

Evaluation Metrics

Formulation

- Text classification: Only a few prediction is required.
  - Ex) For movie review sentiment classification task, we only need to make a single prediction.

- **Text generation**: The probability space of text generation is incredibly large.
  - when vocab size (V)= 10000 and sequence length (T) = 30, the number of cases for possible text is 10000^30=**10^120**.

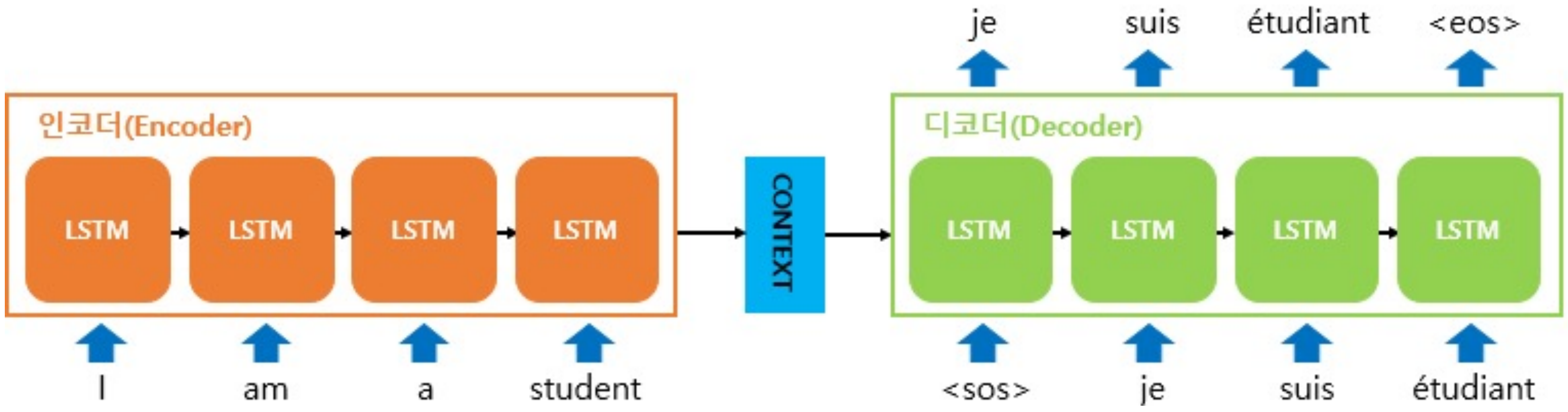
- Solution: Apply conditional probability with chain rule

$$P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \dots * P(y_T|y_{1:t-1}, X)$$

  - X: Source Text
  - Y: Target Text (to be generated)

Formulation

- Recap) Seq2Seq Model

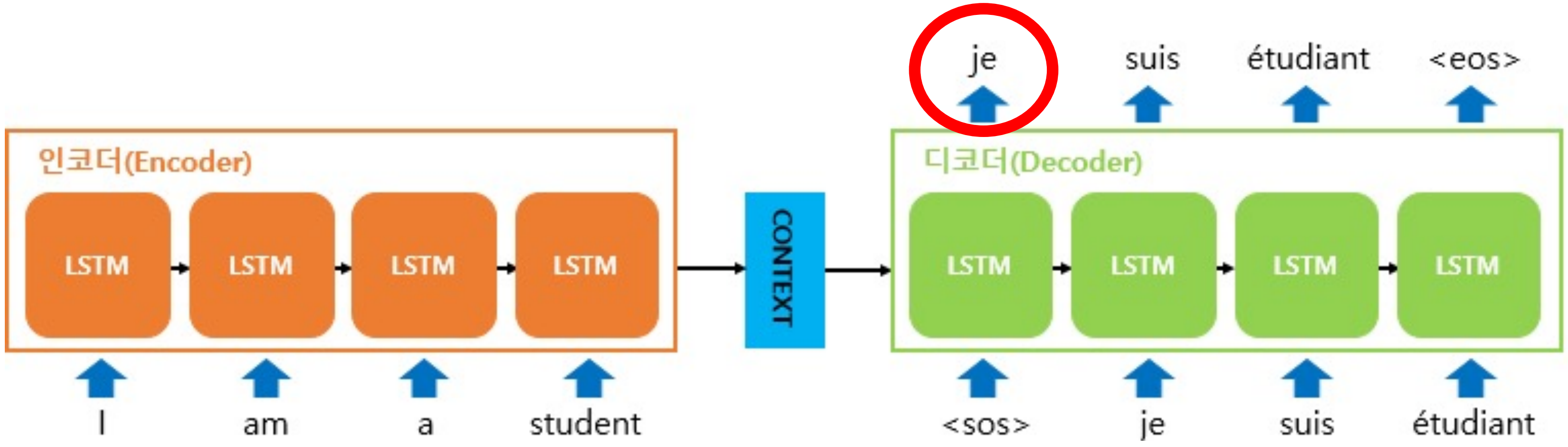


https://wikidocs.net/24996

Formulation

- Recap) Seq2Seq Model

$$\mathcal{L}_{y_1} = CE(P(y_1|X), \text{“}je\text{”})$$

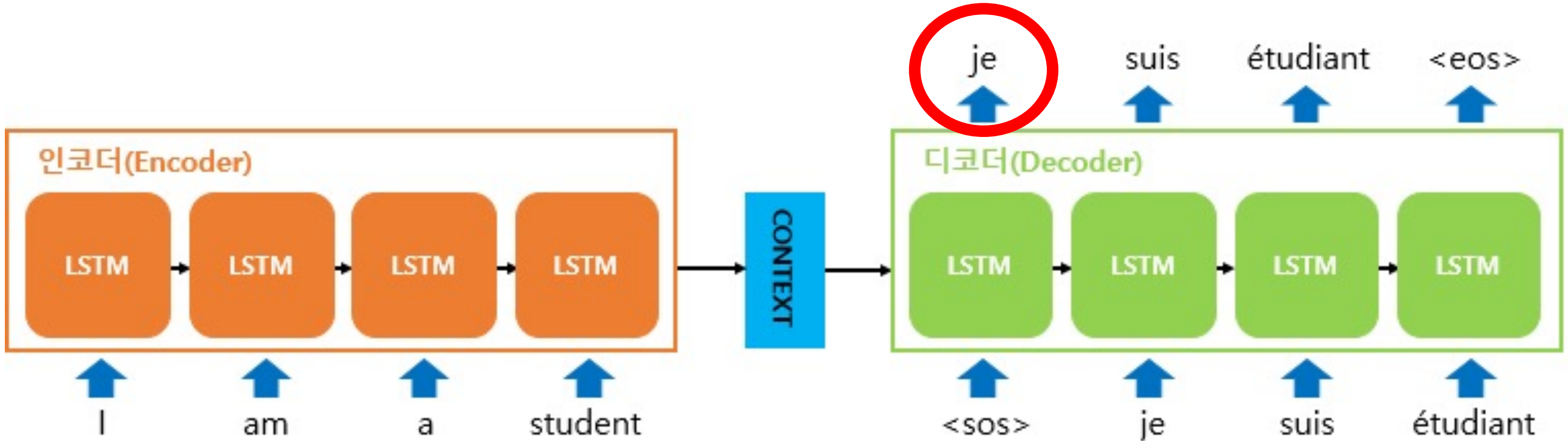


https://wikidocs.net/24996

Formulation

- Recap) Seq2Seq Model

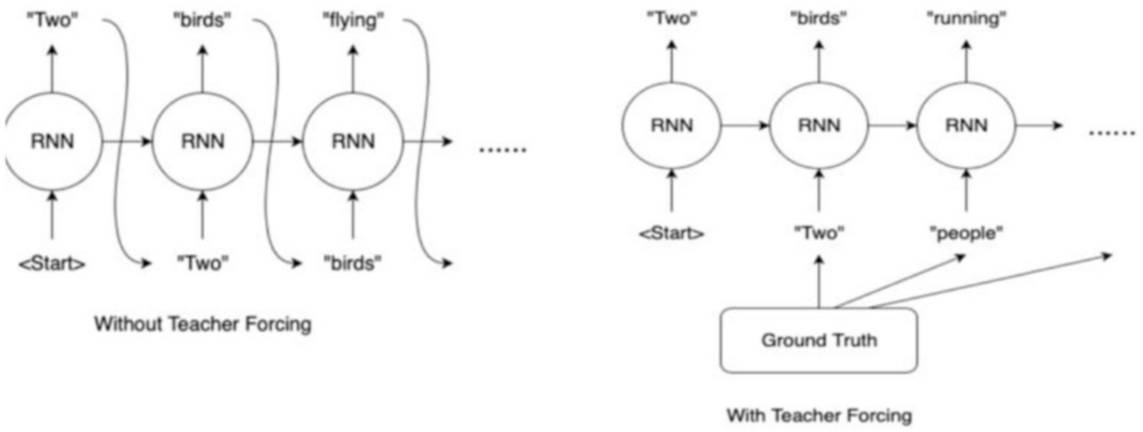
$$\mathcal{L}_{total} = (\mathcal{L}_{y_1} + \mathcal{L}_{y_2} + \mathcal{L}_{y_3} + \mathcal{L}_{y_4})/4$$



https://wikidocs.net/24996

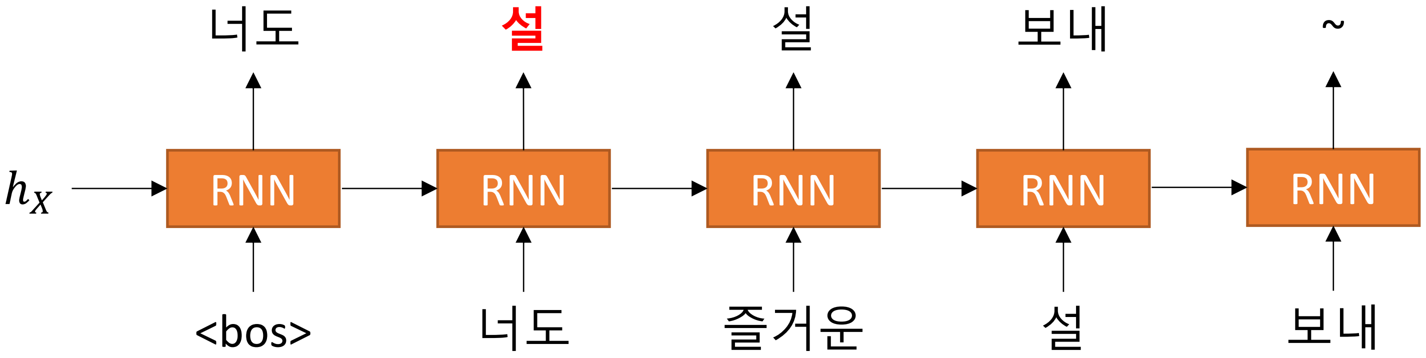
Training Text Generation Model

- How to provide input words to a decoder?
- **Teacher forcing**: The ground-truth target word is passed as the next input to the decoder.



Training Text Generation Model

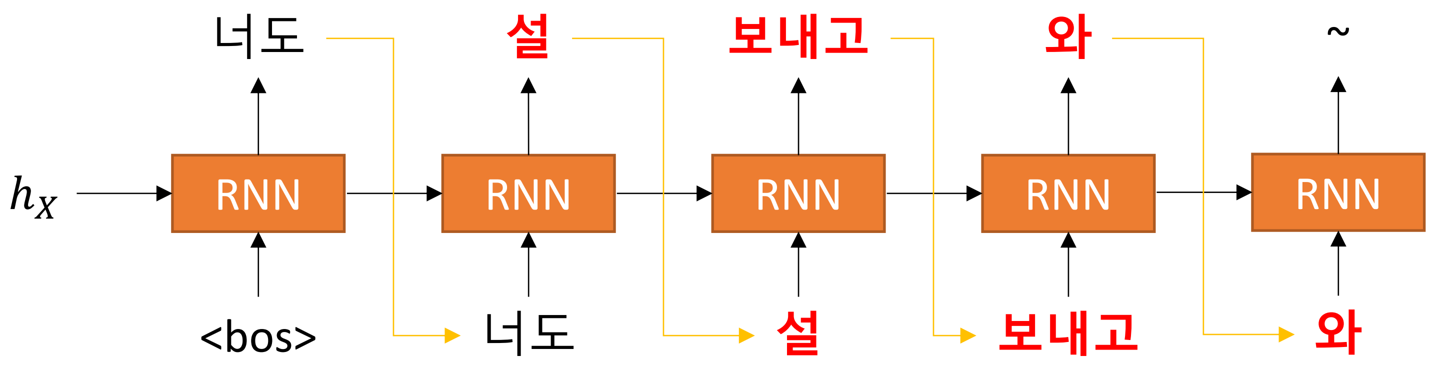
- With Teacher-forcing: 이전 timestep의 정답 단어가 다음 timestep의 입력으로 제공됨.
  - 이전 timestep의 예측은 이후 timestep의 예측에 영향을 주지 않음.



Y = [너도, 즐거운, 설, 보내,~]
X = [집에도, 잘, 다녀와, ~]

Training Text Generation Model

- Without Teacher-forcing: 이전 timestep에서 생성된 단어가 다음 timestep의 입력으로 제공됨.
  - 이전 timestep의 예측이 이후 timestep의 예측에 영향을 줌.



Y = [너도, 즐거운, 설, 보내,~]
X = [집에도, 잘, 다녀와, ~]

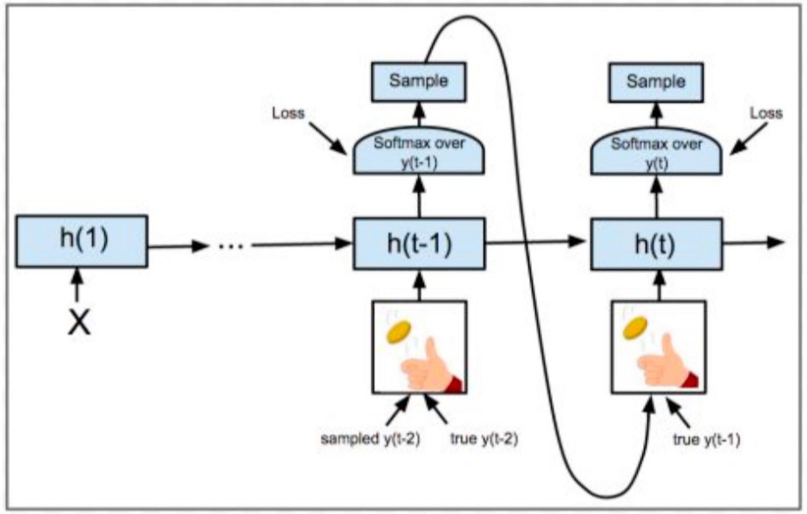
Training Text Generation Model

- **Teacher forcing**: The ground-truth target word is passed as the next input to the decoder.
  - Pros
    - Fast, stable, and efficient training
  - Cons
    - **Exposure Bias**: When we test the generation model, we cannot provide ground-truth words. This discrepancy (Train & test mismatch) degrade the performance and stability of models.

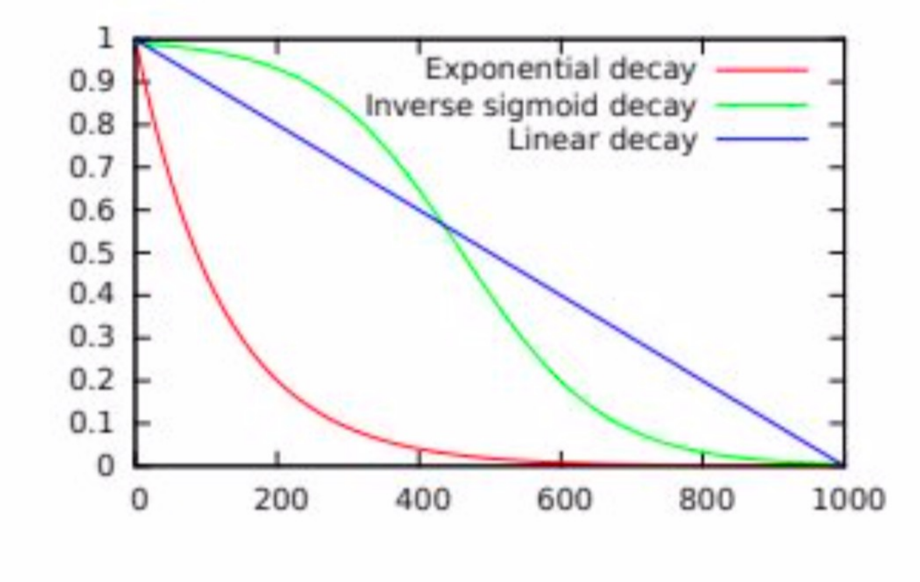
- Teacher Forcing: 여러 소문항이 있는 수학 문제에서, 이전 소문항은 못풀어도 다음 소문항은 풀 수 있게 하기

Teacher Forcing: Train & Inference Mismatch

- Scheduled Sampling (Bengio et al. 2015)



The illustration of the Scheduled Sampling approach, where one flips a coin at every time step to decide to use the true previous token or one sampled from the model itself.



Decay Function for epsilon

Contents

What is Text Generation?

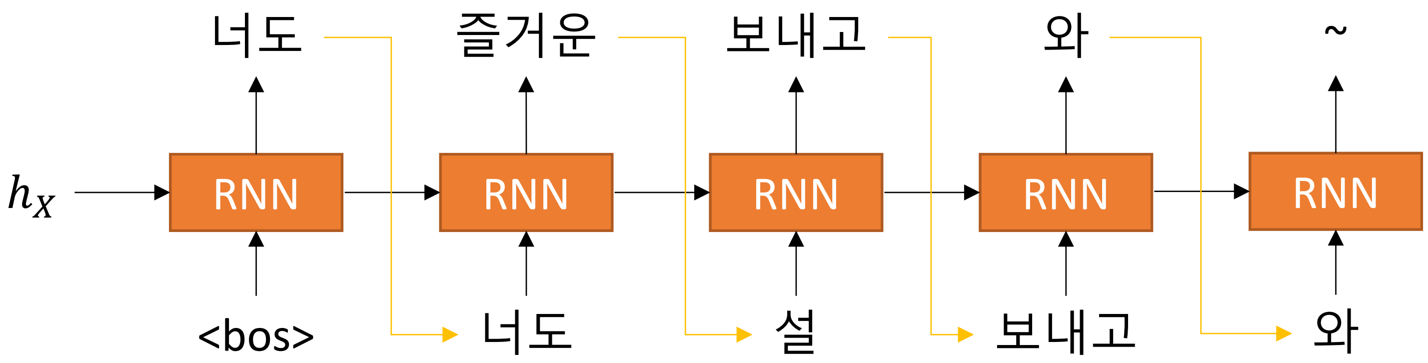
Formulation, Training

Inference (=Testing), Decoding Strategy

Evaluation Metrics

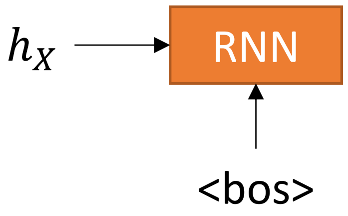
Text Generation: Inference (=Testing)

- Recap) $P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \dots * P(y_T|y_{1:t-1}, X)$
- In each timestep, the trained model generates new word y_t.
- Generated word y_t is used as an input of the next timestep.



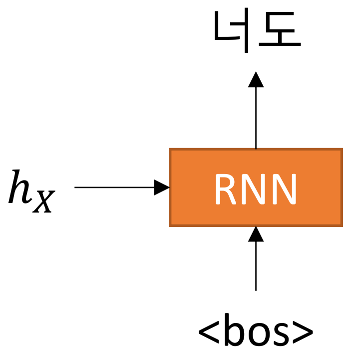
Text Generation: Inference (=Testing)

- Recap) $P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \dots * P(y_T|y_{1:t-1}, X)$
- In each timestep, the trained model generates new word y_t.
- Generated word y_t is used as an input of the next timestep.



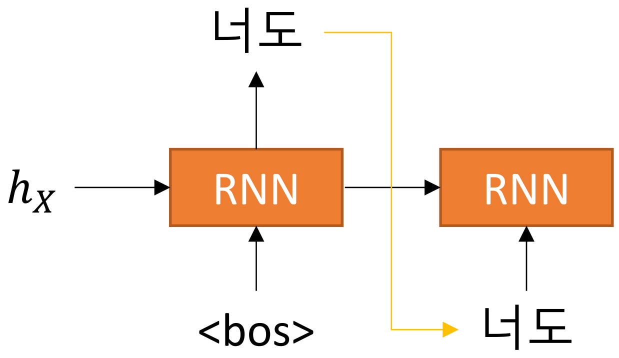
Text Generation: Inference (=Testing)

- Recap) $P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \ldots * P(y_T|y_{1:t-1}, X)$
- In each timestep, the trained model generates new word y_t.
- Generated word y_t is used as an input of the next timestep.



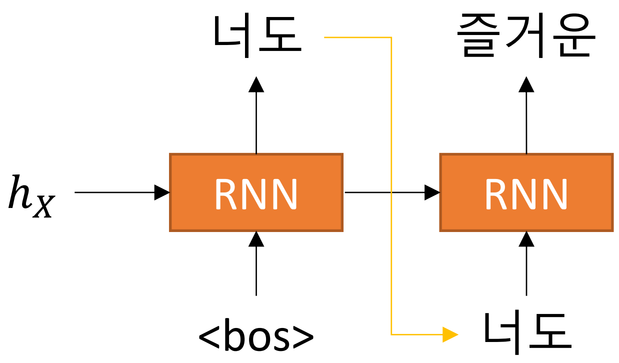
Text Generation: Inference (=Testing)

- Recap) $P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \dots * P(y_T|y_{1:t-1}, X)$
- In each timestep, the trained model generates new word y_t.
- Generated word y_t is used as an input of the next timestep.



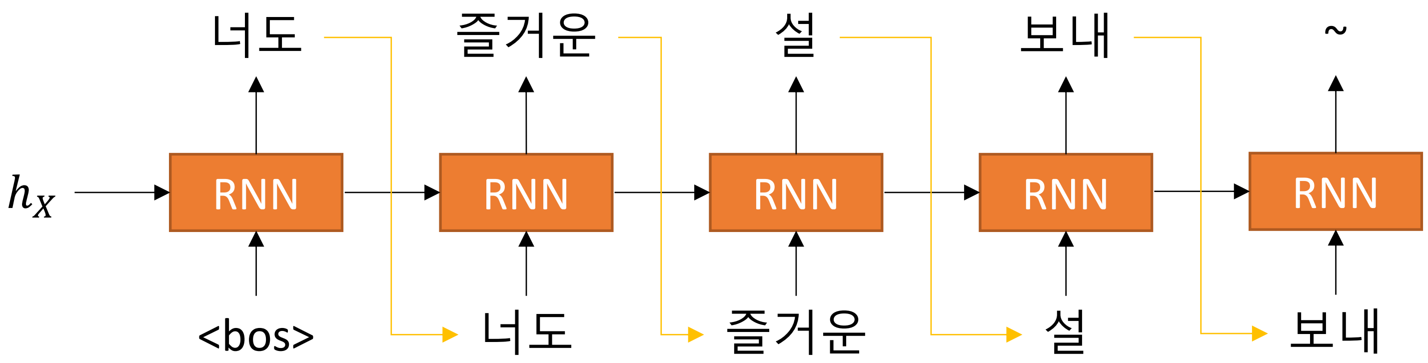
Text Generation: Inference (=Testing)

- Recap) $P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \dots * P(y_T|y_{1:t-1}, X)$
- In each timestep, the trained model generates new word y_t.
- Generated word y_t is used as an input of the next timestep.



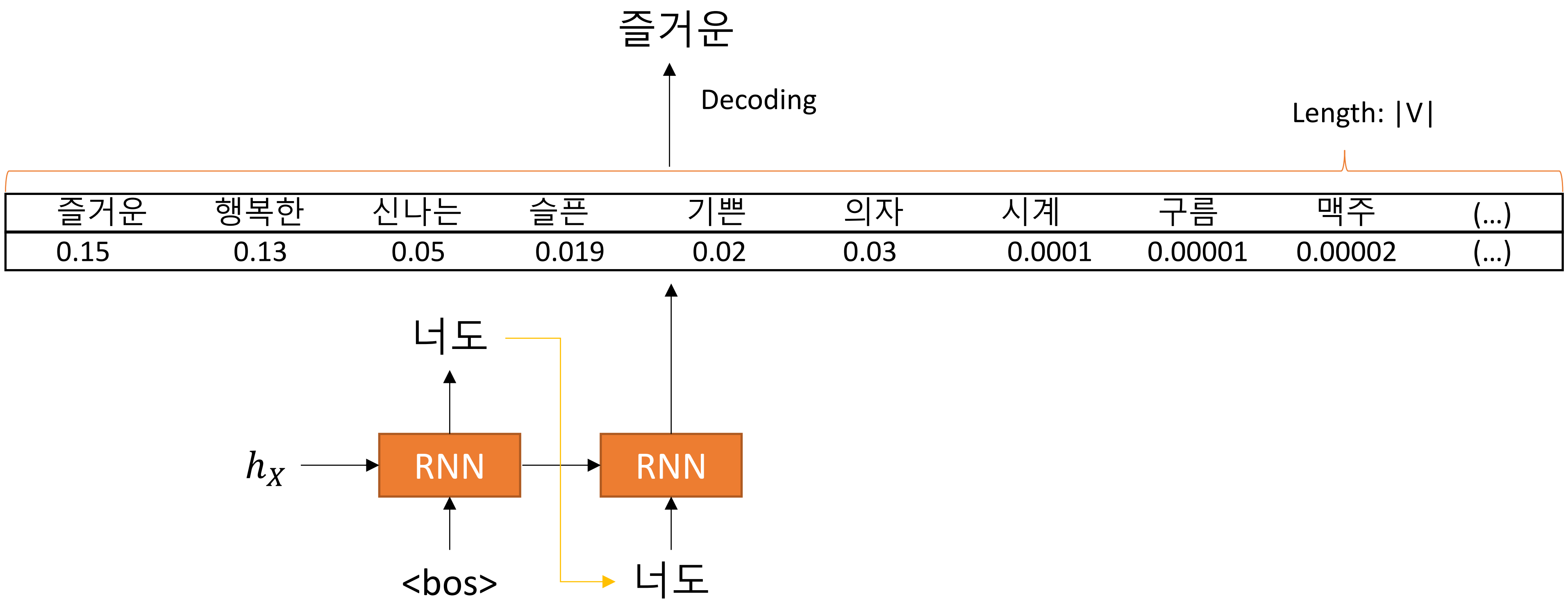
Text Generation: Inference (=Testing)

- Recap) $P(Y|X) = P(y_1|X) * P(y_2|X, y_1) \dots * P(y_T|y_{1:t-1}, X)$
- In each timestep, the trained model generates new word y_t.
- Generated word y_t is used as an input of the next timestep.



Text Generation: Inference (=Testing)

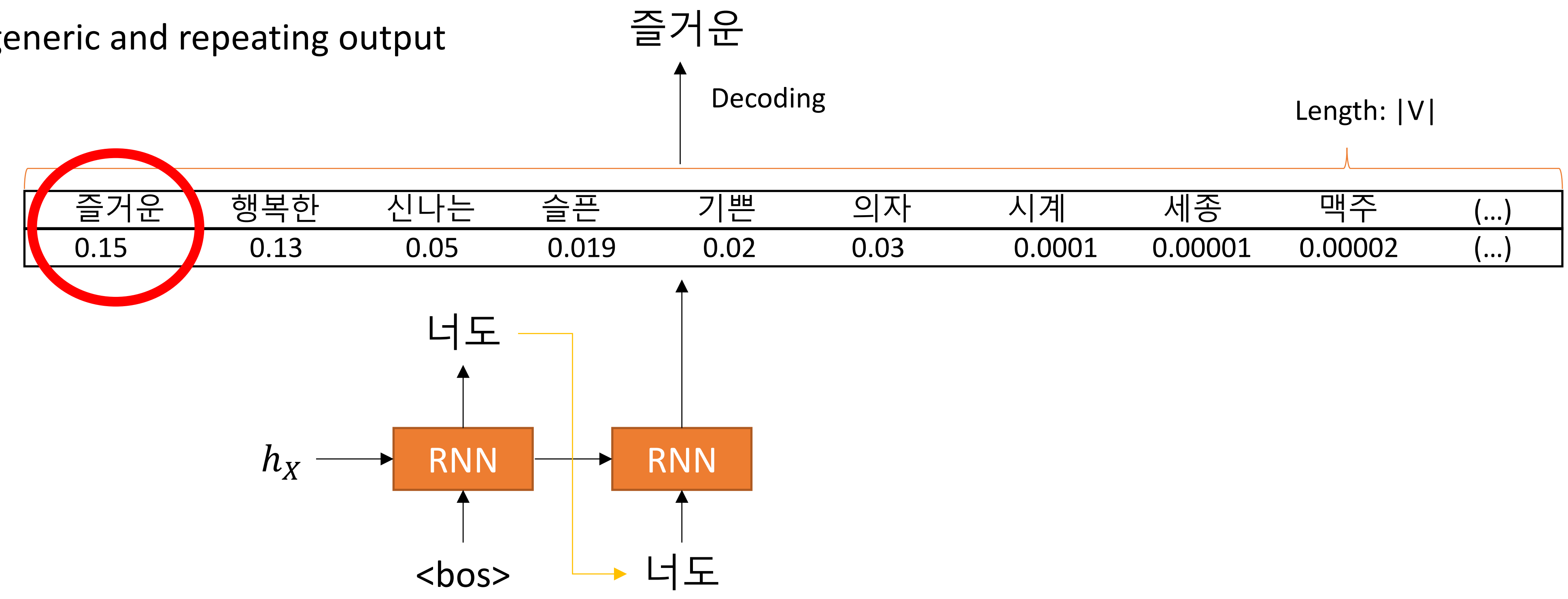
- In each timestep t in inference time, the trained model predicts $P(y_t|X, y_{1:t-1})$.
  - The size of $P(y_t|X, y_{1:t-1})$ is vocab size |V|.
- The choice of a decoded (generated) word depends on a decoding strategy.



즐거운	행복한	신나는	슬픈	기쁜	의자	시계	구름	맥주	(…)
0.15	0.13	0.05	0.019	0.02	0.03	0.0001	0.00001	0.00002	(…)

Decoding Strategy

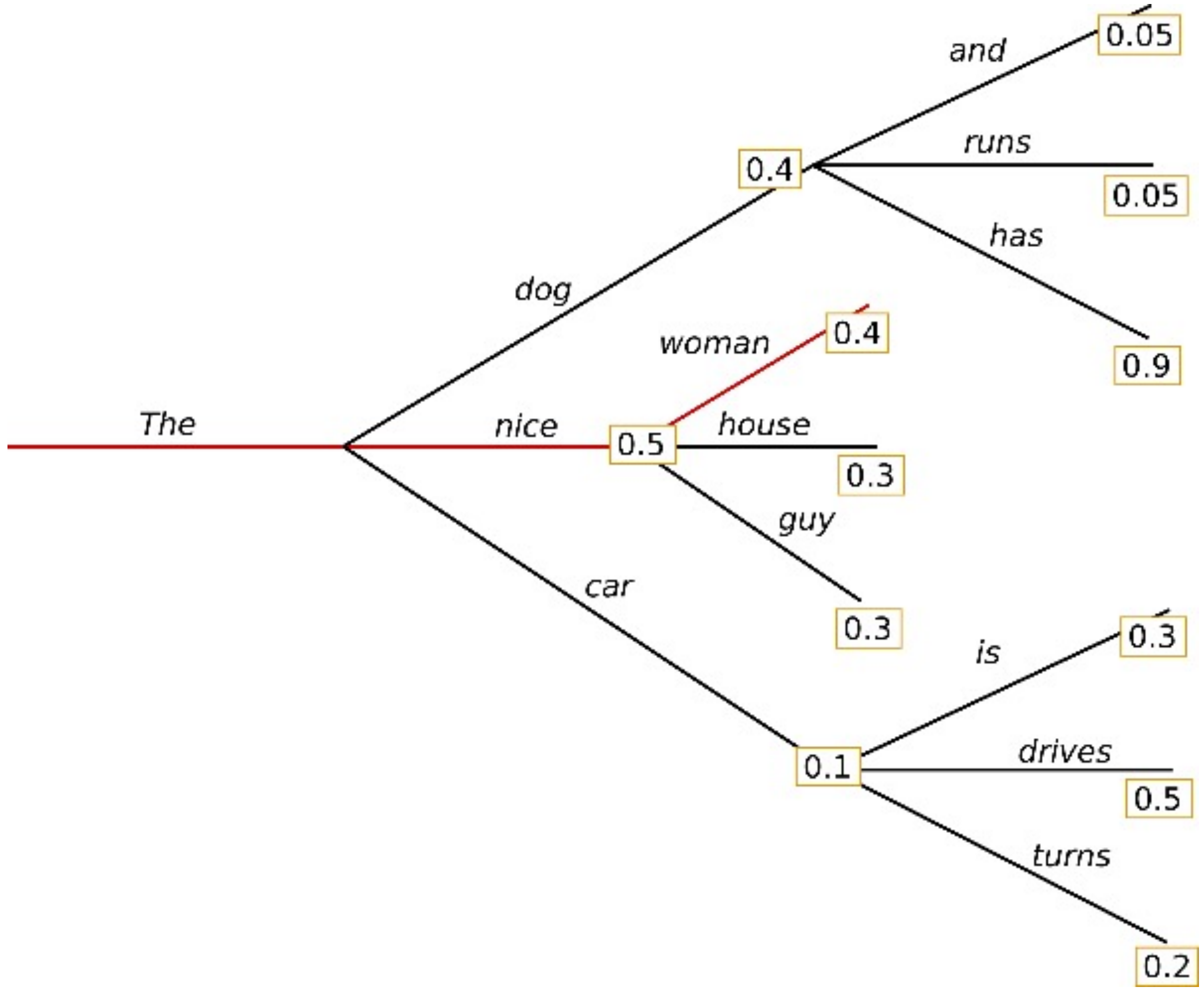
- **Greedy Decoding**
  - $argmax\ P(y_t | X, y_{1:t-1})$
- Generate the most probable word (argmax)
- Limitation: generic and repeating output



https://huggingface.co/blog/how-to-generate

Decoding Strategy

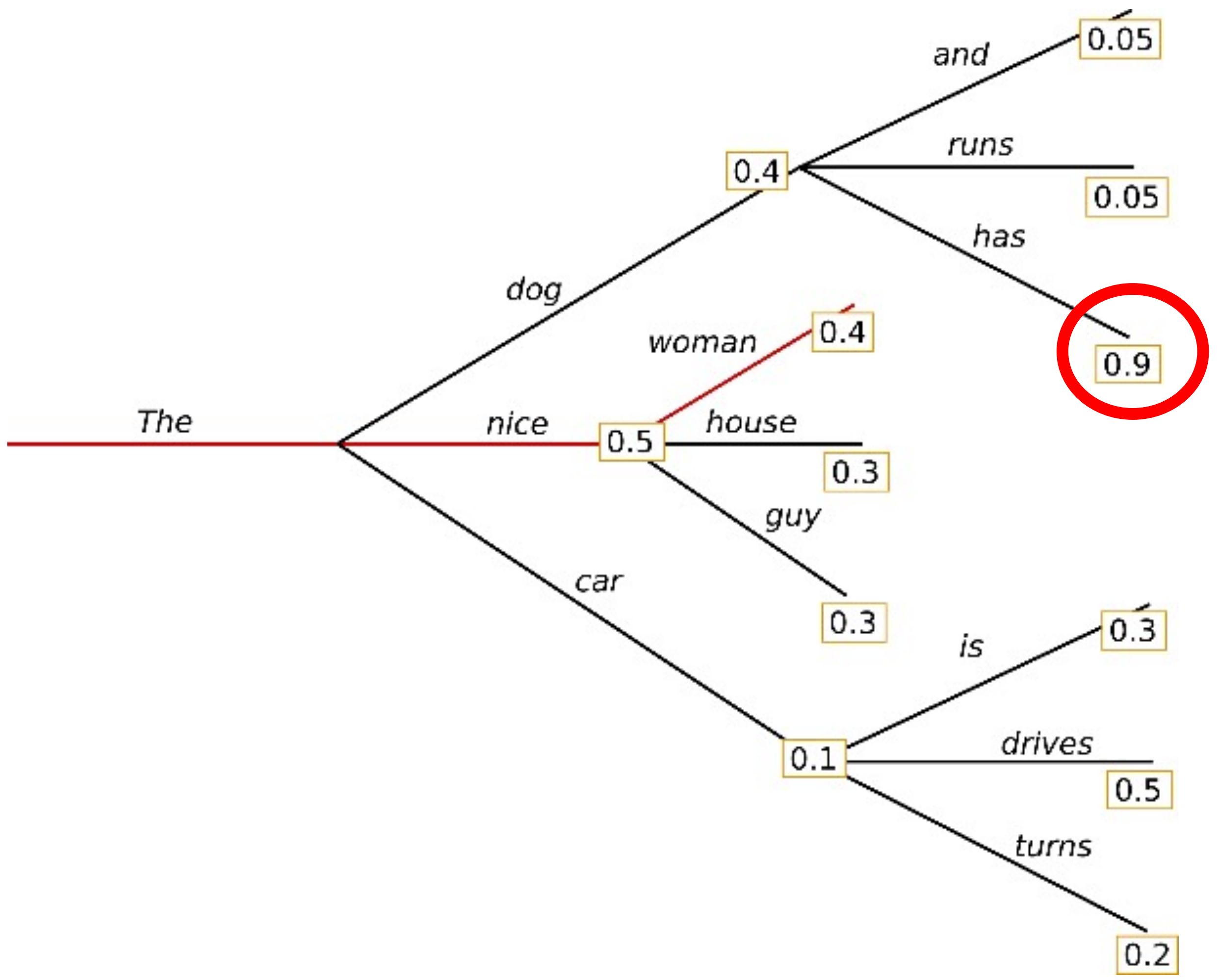
- **Greedy Decoding**
  - $argmax\ P(y_t|X, y_{1:t-1})$
- Generate the most probable word (argmax)
- Limitation: generic and repeating output



https://huggingface.co/blog/how-to-generate

Decoding Strategy

- **Greedy Decoding**
    - $argmax\ P(y_t|X, y_{1:t-1})$
- Generate the most probable word (argmax)
- Limitation: generic and repeating output



Decoding Strategy

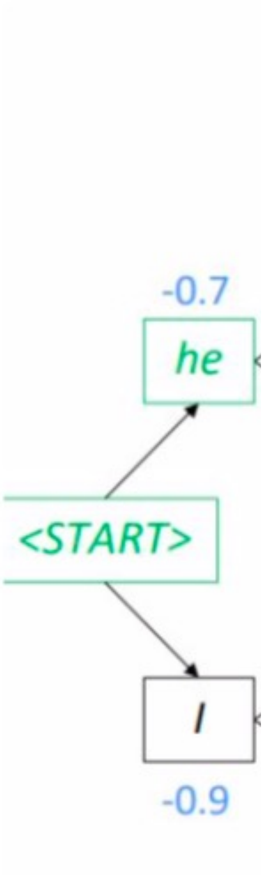
- **Beam Search**

- To compensate Greedy Decoding Error

- Idea: On each step of a decoder, keep track of the *k* most probable partial sequence (not only 1).

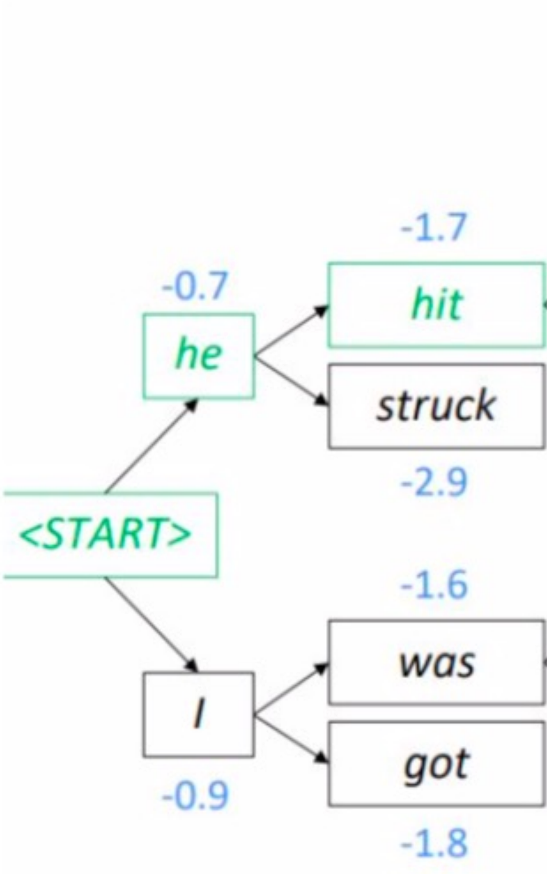
Decoding Strategy

- Beam search Decoding (*k*=2)



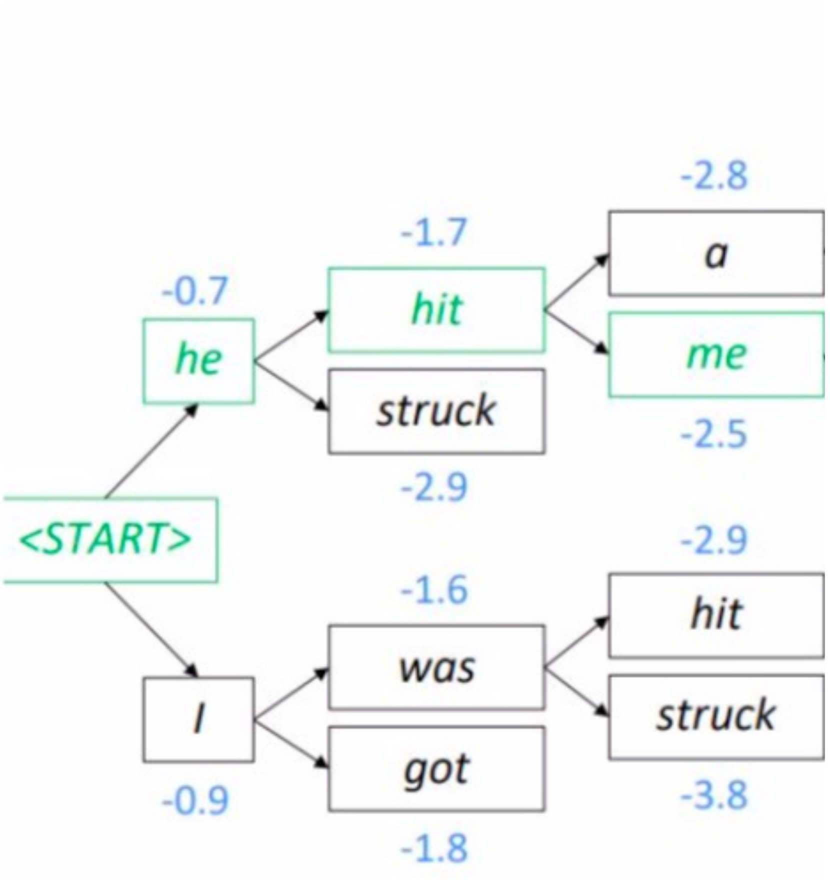
Decoding Strategy

- Beam search Decoding (*k*=2)



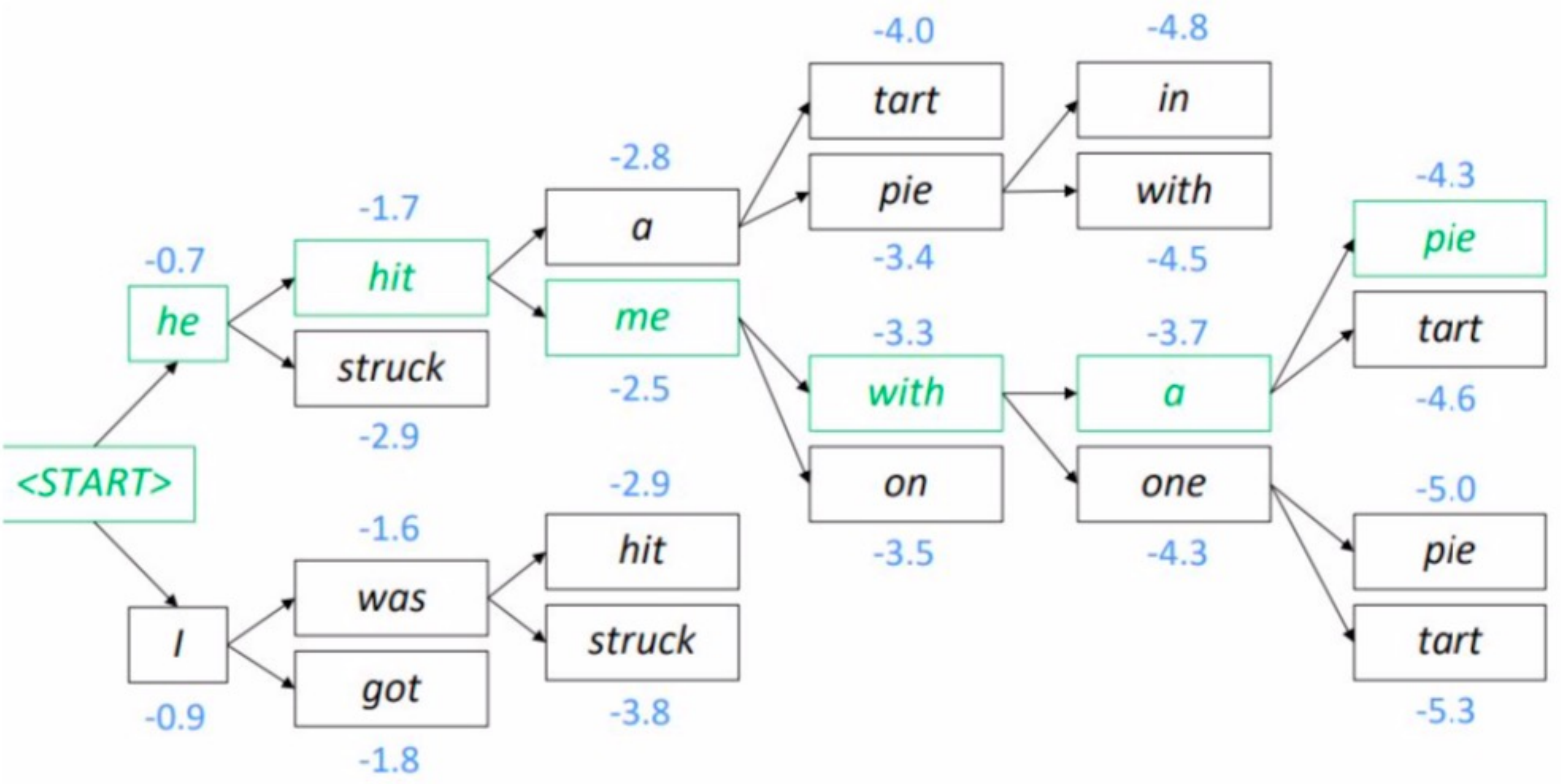
Decoding Strategy

- Beam search Decoding (*k*=2)



Decoding Strategy

- Beam search Decoding (k=2)

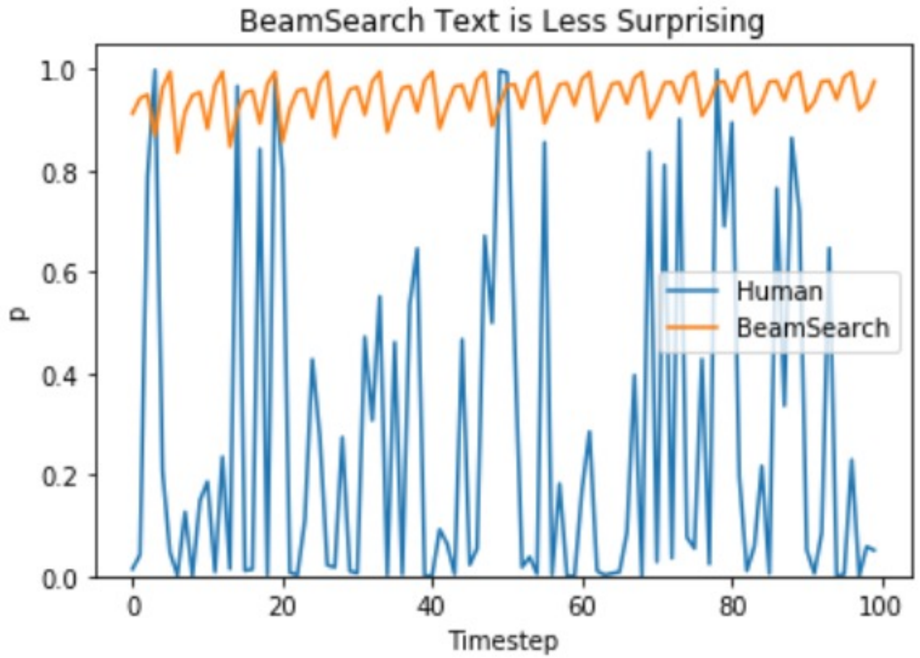


Sampling-based Decoding Strategy

- Probability of a human-written text doesn't always close to 1.

- We need to make more diverse, surprising, and not boring texts.

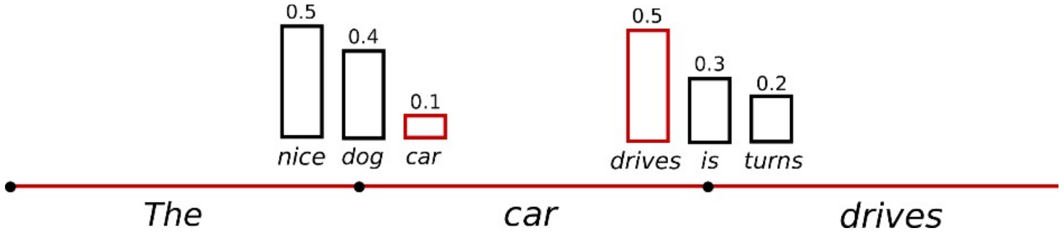
- How to?: Introduce some randomness



The Curious Case of Neural Text Degeneration, ICLR2019

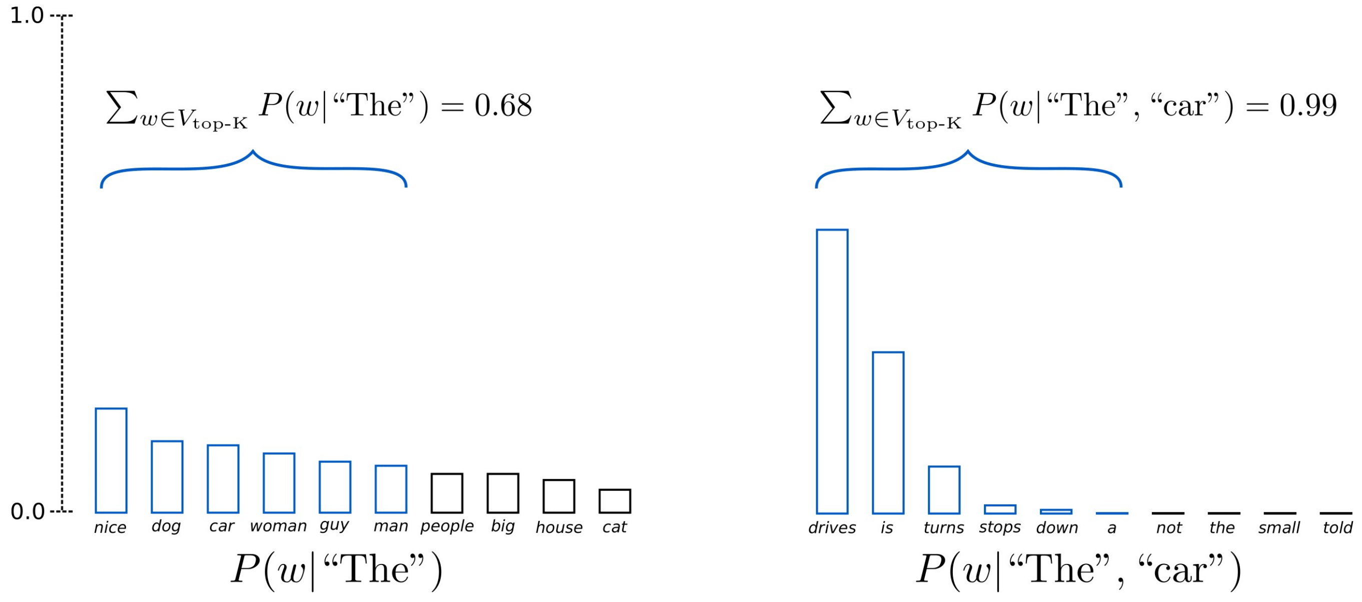
Sampling-based Decoding Strategy

- Pure Sampling
  - $y_t \sim P(y_t | X, y_{1:t-1})$



Sampling-based Decoding Strategy

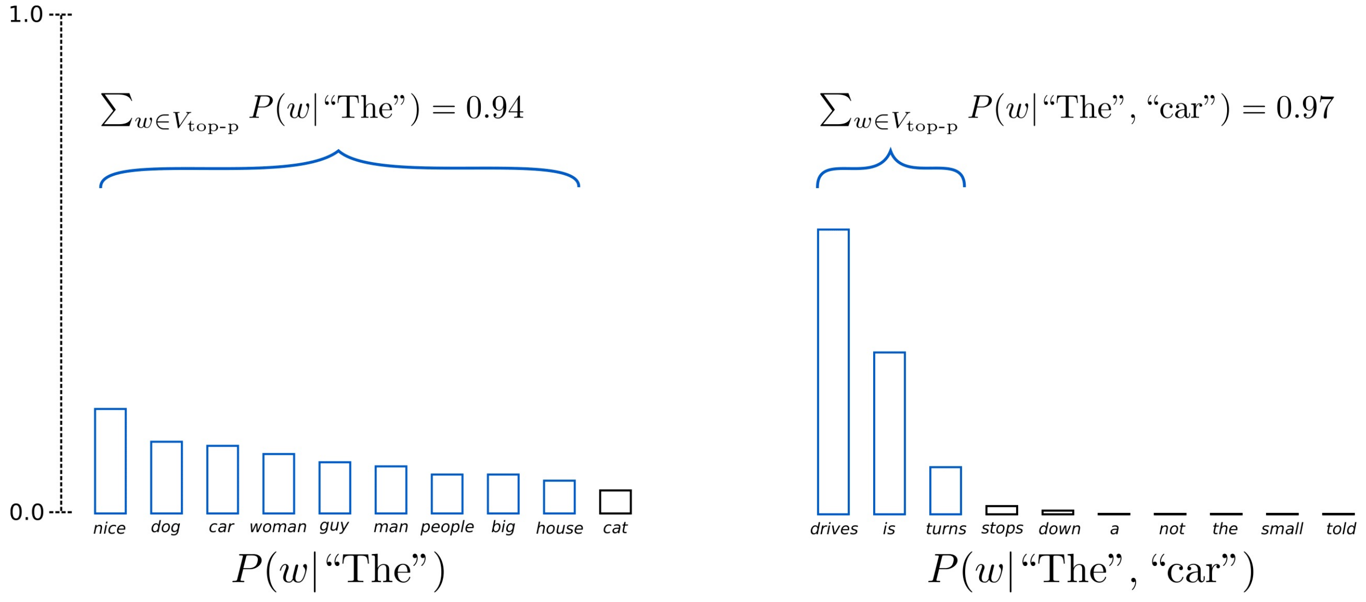
- Top-K sampling
  - Using only K most likely words for sampling



https://huggingface.co/blog/how-to-generate

Sampling-based Decoding Strategy

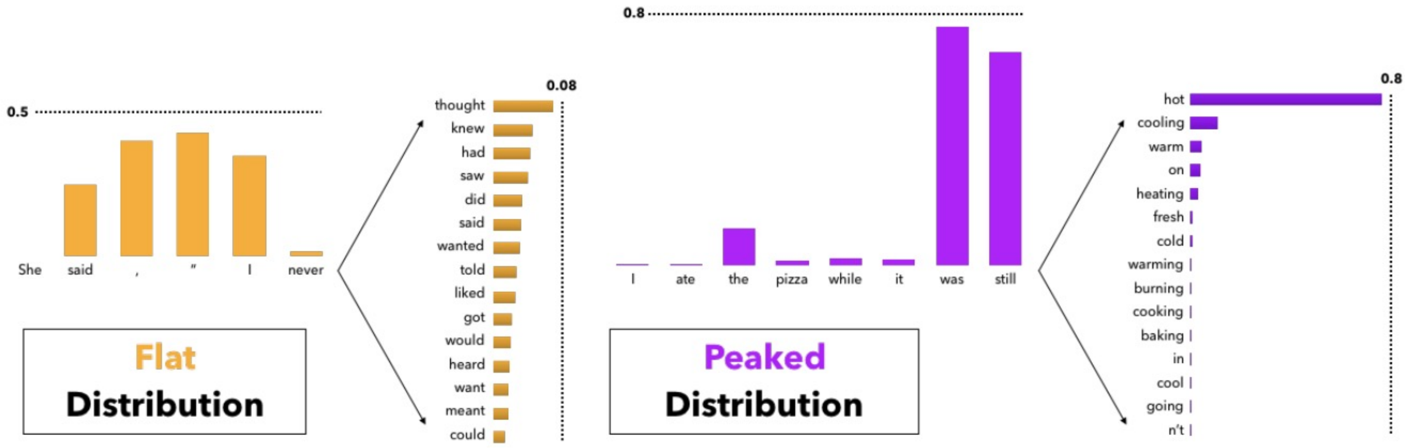
- Top-P sampling (or nucleus sampling)
  - Choose the smallest possible set of words whose cumulative probability exceeds the probability *p*.
  - The number of words in sampling pool is not fixed.



https://huggingface.co/blog/how-to-generate

Sampling-based Decoding Strategy

- Top–P vs Top-K sampling



Contents

Evaluation Metric

- Common approach: similarity between generated text and the answer text

- Word-level Similarity
  - BLEU
  - ROUGE
  - METEOR
- Embedding Similarity
  - Word Average/Extrema/Greedy
  - BERTScore

Source: 나는 너를 좋아해.

Target: I like you .

Model1: I love you .

Model2: I hate you .

Evaluation Metric

- Perplexity (PPL): Measuring the performance of a generation model (NOT a generated text).

- $PPL(W) = P(w_1, w_2, \ldots w_N)^{-\frac{1}{N}} \propto CrossEtropyLoss(W)$
    - W: Answer text

- Perplexity는 “모델이 정답 텍스트를 얼마나 잘 예측할 수 있는지”에 반비례하는 값이며, PPL이 낮을수록 일반적으로 더 좋은 text generation model입니다.

References:

- CS224n(http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture15-nlg.pdf)
- https://ai-information.blogspot.com/2019/03/scheduled-sampling.html
- Neural Text Generation with Unlikelihood Training (https://arxiv.org/abs/1908.04319)
- Nucleus Sampling (https://arxiv.org/abs/1904.09751)
- Get to the point (https://arxiv.org/abs/1704.04368)
- CTRL: A Conditional Transformer Language Model for Controllable Generation (https://arxiv.org/abs/1909.05858)