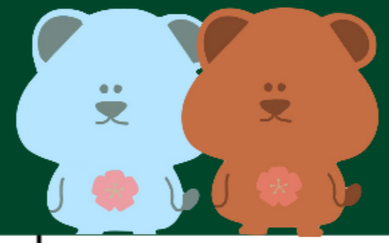


Natural Language Generation(NLG)

송민경, 김수한

EWHA EURON

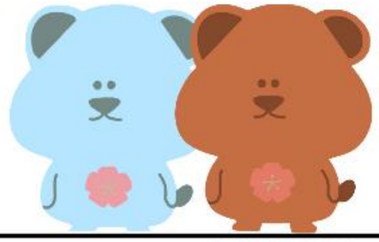
목차

#01 NLG

#02 Decoding

#03 Training

#04 Evaluating



EWHA EURON

#01 NLG

EWHA EURON

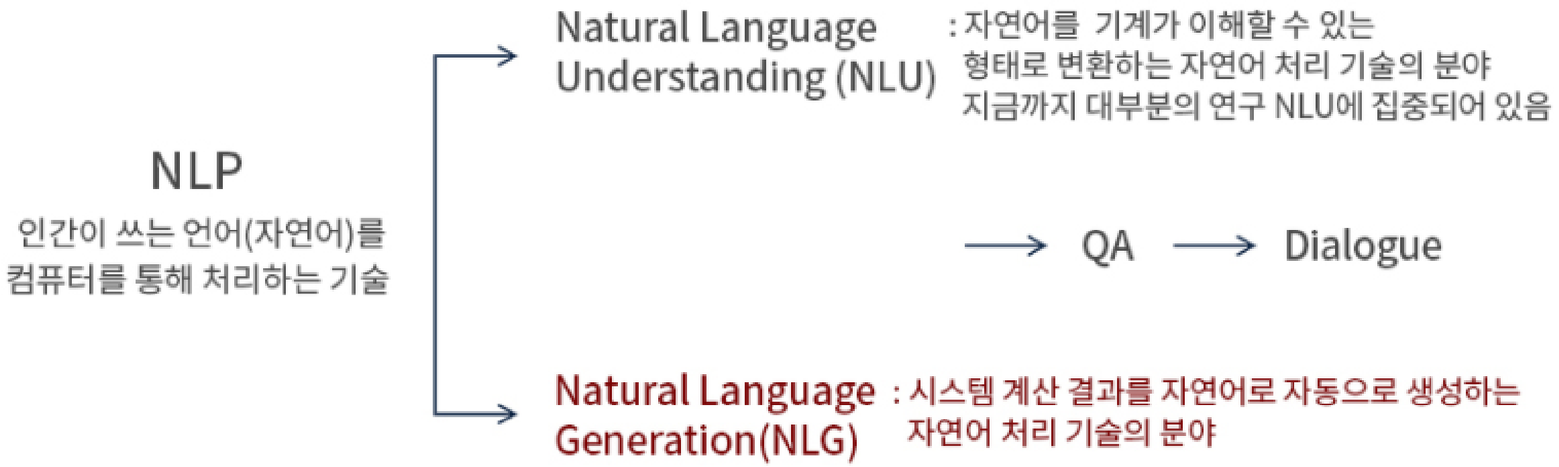
#01 Natural Language Generation(NLG)

#1 NLG의 개념과 특징

- NLG?

Any task involving text production for human consumption requires natural language generation.

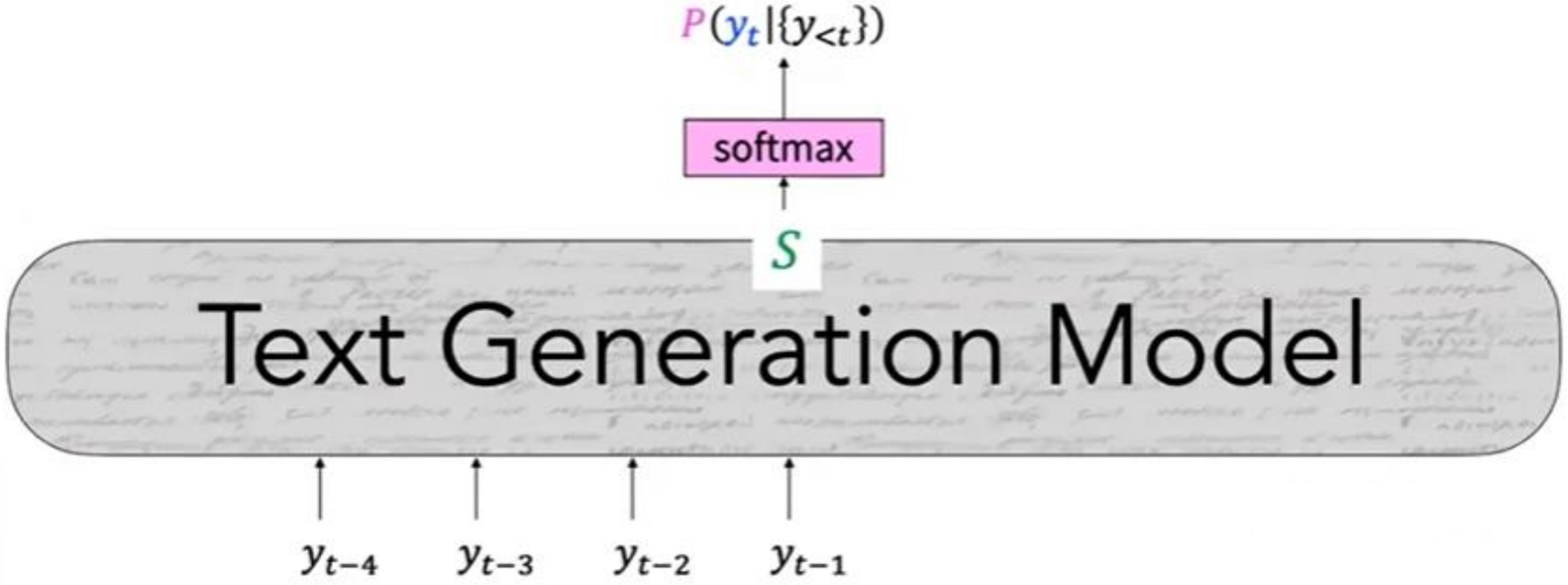
NLP의 한 분야로, 주어진 input x에 대해 새로운 text를 생성해내는 작업



#01 Natural Language Generation(NLG)

#1 NLG의 개념과 특징

- At each time step t, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$. Then, we compute a probability distribution P over $w \in V$ using these scores:



#01 Natural Language Generation(NLG)

#1 NLG의 개념과 특징

- **<NLG가 잘 된다는 것의 의미>**

- **적절성** : 생성된 문장이 모호하지 않고, 원래의 input text의 의미와 일치하는 정도

- **유창성**: 문법이 정확하고, 어휘를 적절히 사용하는 정도

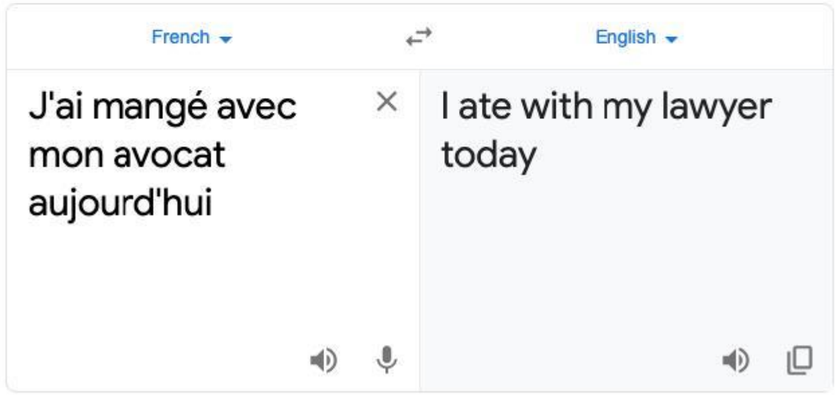
- **가독성**: 적절한 지시어, 접속사 등을 사용하여 문장의 논리 관계를 고려하여 생성하는 정도

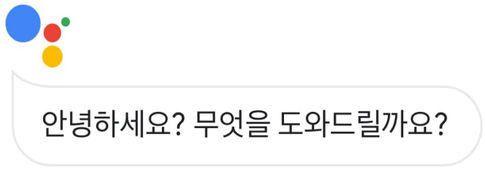
- **다양성**: 상황에 따라 혹은 대상에 따라 표현을 다르게 생성하는 정도

#01 Natural Language Generation(NLG)

#2 NLG Task

#2.1 MT(Machine Translation), Dialog Systems





#01 Natural Language Generation(NLG)

#2 NLG Task

#2.1 Summarization, Data-to-Text Generation

C: Looking at what we've got, we we want an LCD display with a spinning wheel.
B: You have to have some push-buttons, don't you?
C: Just spinning and not scrolling, I would say.
B: I think the spinning wheel is definitely very now.
A: but since LCDs seems to be uh a definite yes,
C: We're having push-buttons on the outside
C: and then on the inside an LCD with spinning wheel,

Decision Abstract (Summary):
The remote will have push buttons outside, and an LCD and spinning wheel inside.

A: and um I'm not sure about the buttons being in the shape of fruit though.
D: Maybe make it like fruity colours or something.
C: The power button could be like a big apple or something.
D: Um like I'm just thinking bright colours.

Problem Abstract (Summary):
How to incorporate a fruit and vegetable theme into the remote.

(Wang and Cardie, ACL 2013)

document/E-mail/Meeting script summarization

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AS ...
Heat	11	12	103	49	47	27
Hawks	7	15	95	43	33	20

PLAYER	AS	RB	PT	FG	FGA	CITY ...
Tyler Johnson	5	2	27	8	16	Miami
Dwight Howard	4	17	23	9	11	Atlanta
Paul Millsap	2	9	21	8	12	Atlanta
Goran Dragic	4	2	21	8	17	Miami
Wayne Ellington	2	3	19	7	15	Miami
Dennis Schroder	7	4	17	8	15	Atlanta
Rodney McGruder	5	5	11	3	8	Miami
Thabo Sefolosha	5	5	10	5	11	Atlanta
Kyle Korver	5	3	9	3	9	Atlanta
...						

The Atlanta Hawks defeated the Miami Heat , 103 - 95 , at Philips Arena on Wednesday . Atlanta was in desperate need of a win and they were able to take care of a shorthanded Miami team here . Defense was key for the Hawks , as they held the Heat to 42 percent shooting and forced them to commit 16 turnovers . Atlanta also dominated in the paint , winning the rebounding battle , 47 - 34 , and outscoring them in the paint 58 - 26.The Hawks shot 49 percent from the field and assisted on 27 of their 43 made baskets . This was a near wire - to - wire win for the Hawks , as Miami held just one lead in the first five minutes . Miami (7 - 15) are as beat - up as anyone right now and it 's taking a toll on the heavily used starters . Hassan Whiteside really struggled in this game , as he amassed eight points , 12 rebounds and one blocks on 4 - of - 12 shooting ...

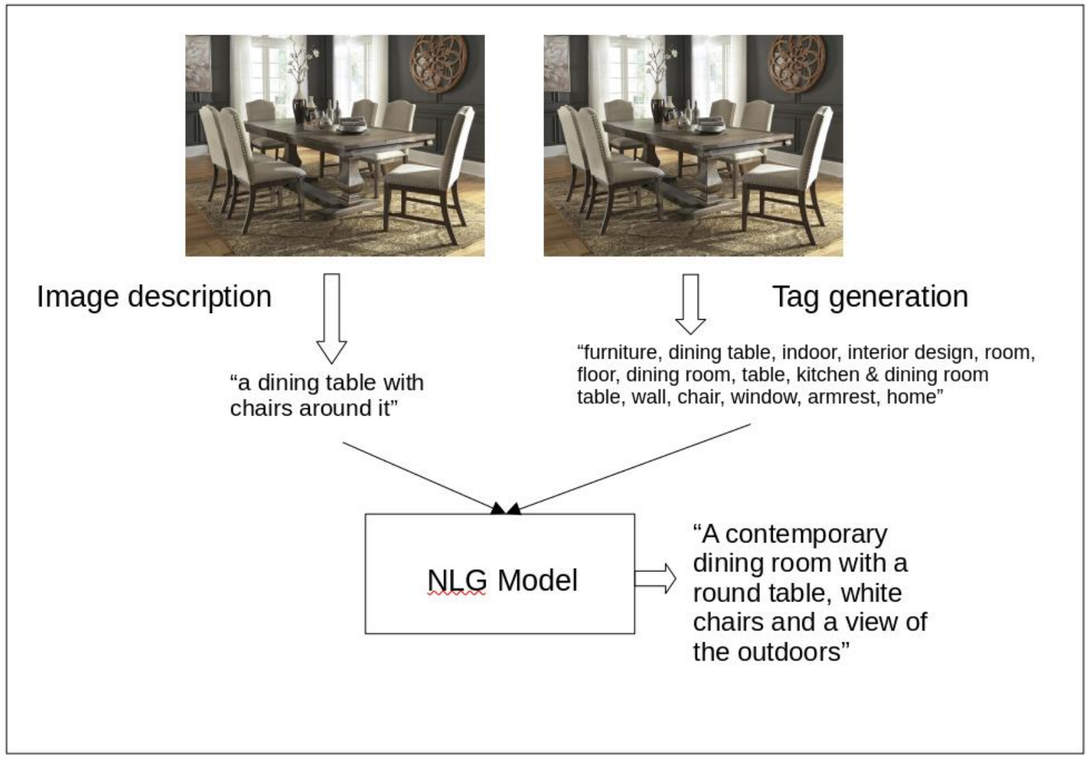
(Wiseman and Rush., EMNLP 2017)

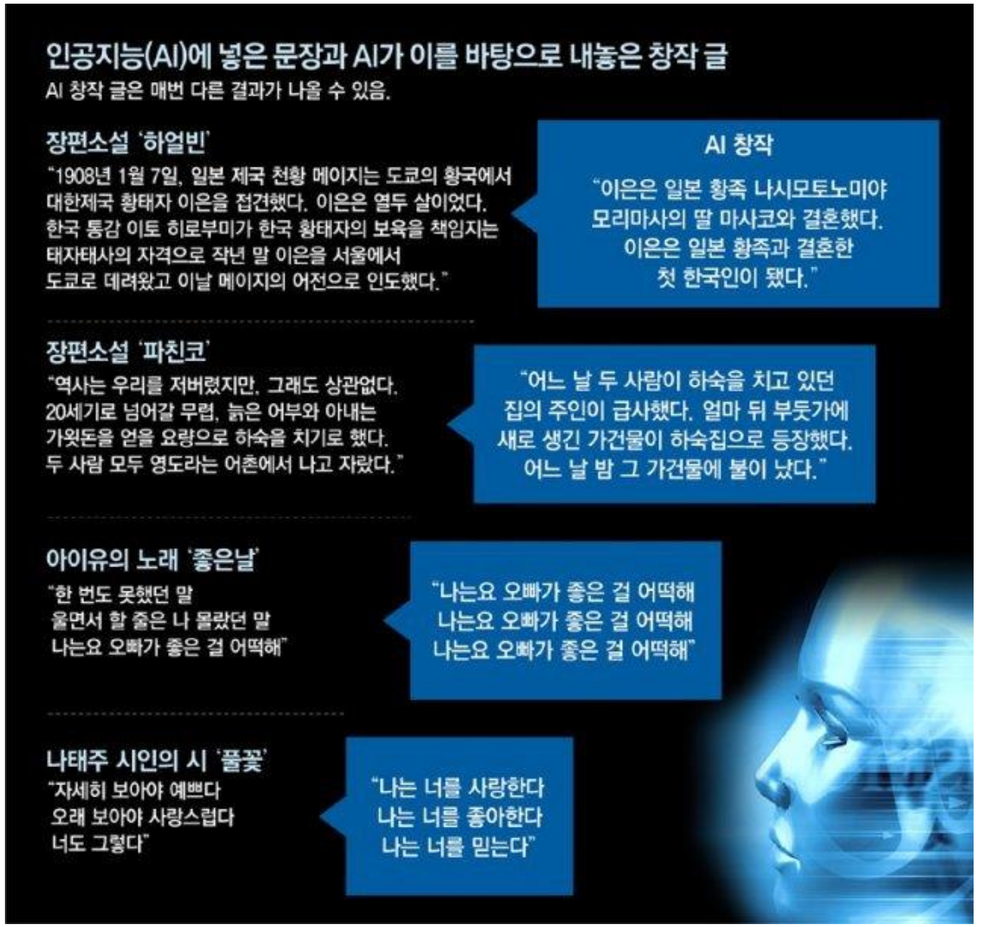
table, graph, .. interpretation

#01 Natural Language Generation(NLG)

#2 NLG Task

#2.1 Visual Description, Creative Generation





story&Narratives/ Poetry generation

#02 Decoding

EWHA EURON

#02 Decoding

At each time step t, our model computes a vector of scores for each tokens in our vocabulary,

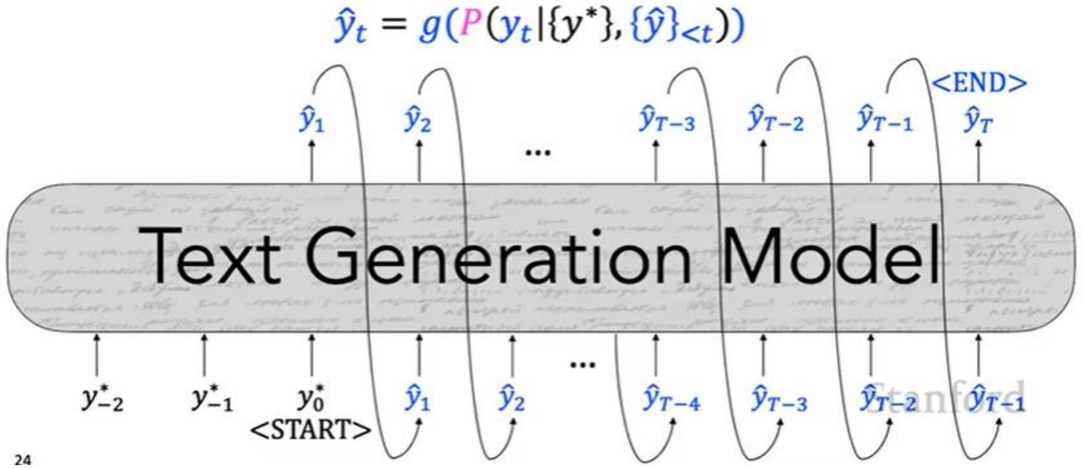
$$S = f(\{y_{<t}\})$$

$f(.)$ is your model

Then, we compute a prob distribution P over these scores(usually with a softmax function):

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

Our decoding algorithm defines a function to select a token from this distribution:



24

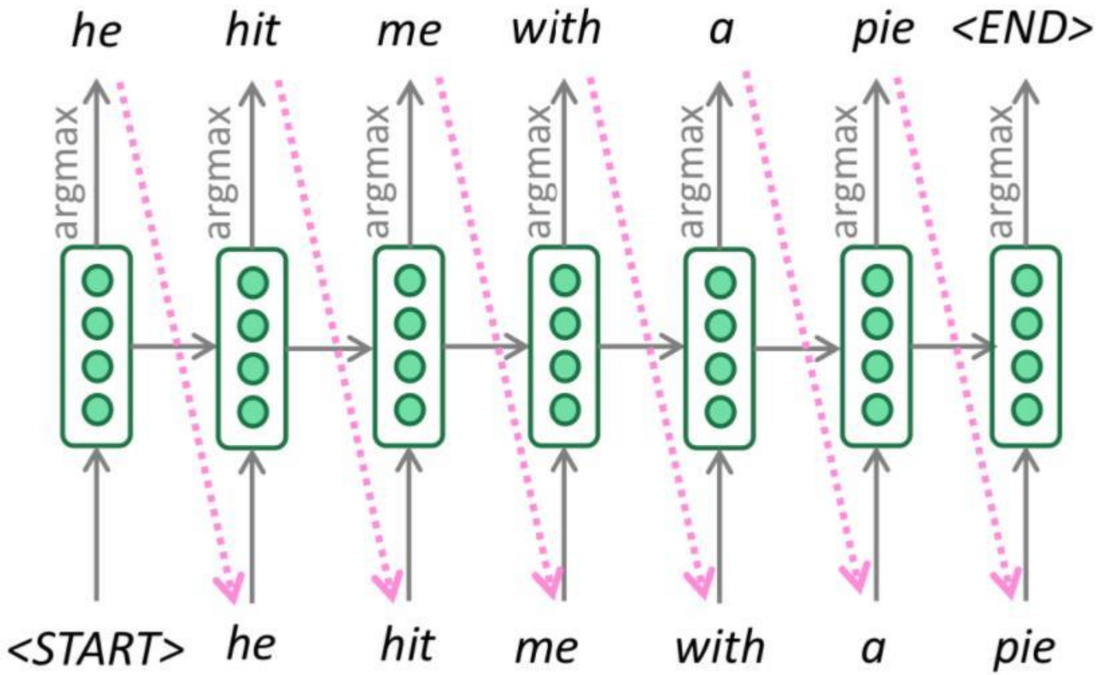
$$\hat{y}_t = g(P(y_t|\{y_{<t}\}))$$

$g(.)$ is your decoding algorithm

#02 Decoding

#1 Greedy methods : Argmax decoding vs. Beam search

- Argmax decoding: 각 출력을 예측하는데 매 스텝에서 가장 가능성이 높은 단어를 한 개 선택

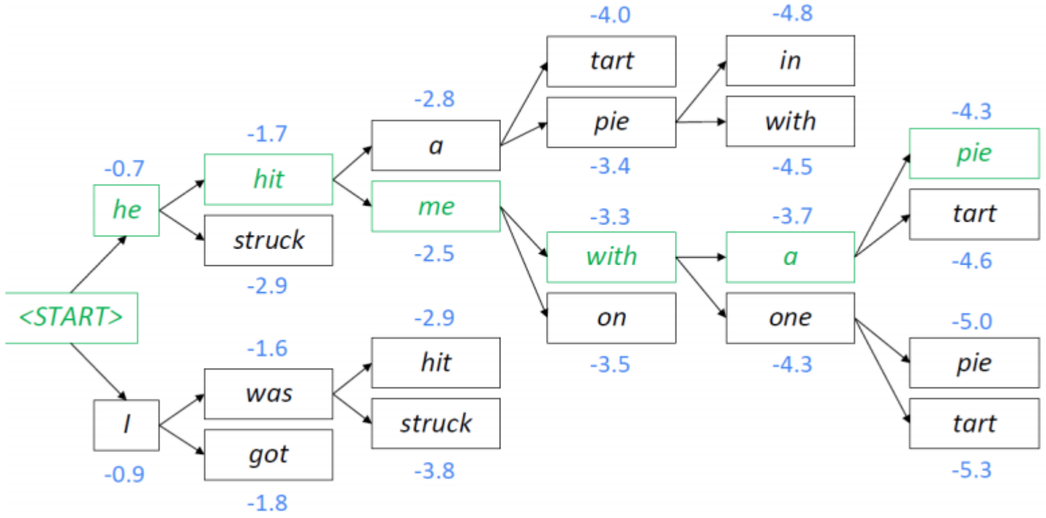


탐색하는 데 걸리는 시간이 매우 빠르나,
최종 출력 결과가 좋지 않음.

#02 Decoding

#1 Greedy methods : Argmax decoding vs. Beam search

- Beam search: argmax decoding의 확장된 형태로, k개의 가능한 가설들을 두고,
가장 높은 확률을 갖는 문장을 찾아 나가는 방식 -> hyper parameter로 Beam size(k)를 가짐.



Eg., Beam size=2

Start값부터 2개씩의 가장 확률이 높은 가설들을 두고 문장을 찾아 나감

$$\text{Blue numbers} = \text{score}(y_1, \dots, y_t)$$

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

#02 Decoding

#1 Greedy methods : Argmax decoding vs Beam search

- Beam search-> hyper parameter로 Beam size를 가짐. -> 결정 방법?

Human chit-chat partner

Beam size	Model response
1	*I love to eat healthy and eat healthy*
2	*That is a good thing to have*
3	*I am a nurse so I do not eat raw food*
4	*I am a nurse so I am a nurse*
5	*Do you have any hobbies?*
6	*What do you do for a living?*
7	*What do you do for a living?*
8	*What do you do for a living?*

Low beam size:
More on-topic but nonsensical; bad English

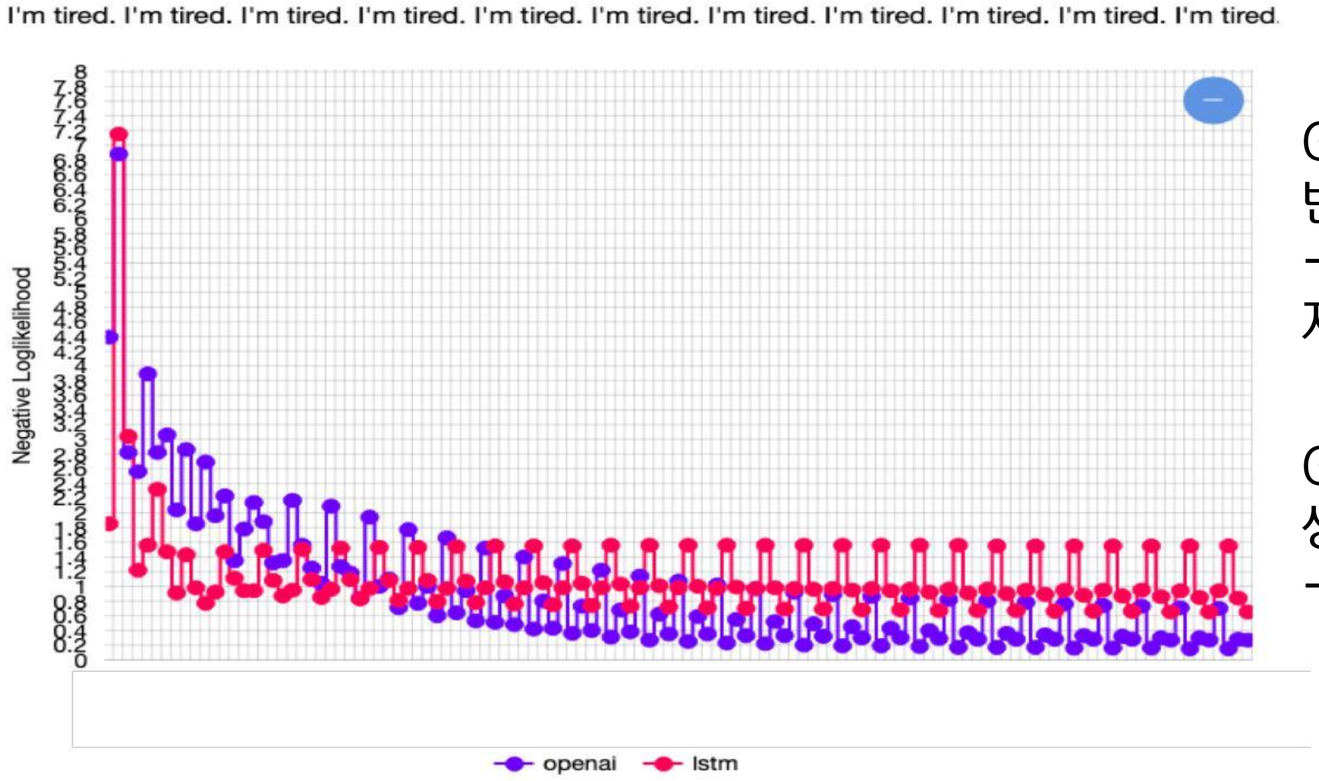
High beam size:
Converges to safe, "correct" response, but it's generic and less relevant

Beam size가 너무 작으면
주제에 더 가깝지만, nonsensical한 답

Beam size가 너무 크면
보다 일반적이고(주제로부터 먼),
짧은 답변 -> BLEU score ↓

#02 Decoding

#2 Greedy method의 단점(: Repetition)



Greedy decoding 방식은 동일한 phrase를 반복해서 생성하는 문제가 있음.
-> Dialog NLG 같은 open end 문장 생성 시 자주 발생함.

GPT에서는 특정 표현이 반복 생성될 때,
생성 결과에 대해 모델의 confidence가 높아짐.
-> 병목현상 없앤 attention 구조 때문

#02 Decoding

#2 Greedy method의 단점(Repetition) 해결방법

- Decoding 단계에서 n-grams의 반복을 막음(Simple option: Heuristic)
- Decoding 알고리즘을 변경
  - Random sampling
  - Top-k sampling
  - Top-p sampling
  - + Scaling randomness: Temperature
  - Re-balancing distribution

- 연속되는 문장의 hidden representation similarity를 낮추는 방식 -> 문장 내 반복은 막을 수 있음
- 동일한 단어가 등장하는 것을 막는 attention mechanism: Coverage loss
- 이미 등장한 토큰에 penalty를 부여: Unlikelihood objective

Cf. unlikelihood training

https://doi.org/10.48550/arXiv.1908.04319

5 THE UNLIKELIHOOD TRAINING OBJECTIVE

We now describe *unlikelihood training* for neural language models, then in Section 6 demonstrate empirically that our proposal substantially improves neural text degeneration (§4).

5.1 UNLIKELIHOOD TRAINING

The key idea behind unlikelihood training is decreasing the model's probability of certain tokens, called *negative candidates*. Given a sequence $(x_1, \ldots, x_T)$ and a set of negative candidate tokens $\mathcal{C}^t = \{c_1, \ldots, c_m\}$, where each $c_j \in \mathcal{V}$, we define the **unlikelihood loss** for step t as:

$$\mathcal{L}_{\text{UL}}^t(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t})). \tag{3}$$

The loss decreases as $p_\theta(c|x_{<t})$ decreases. We incorporate the unlikelihood loss into a **token-level unlikelihood objective** which augments each time-step of maximum likelihood training:

$$\mathcal{L}_{\text{UL-token}}^t(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\alpha \cdot \underbrace{\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p_\theta(x_t|x_{<t})}_{\text{likelihood}}. \tag{4}$$

As candidates, we use previous context tokens:

$$\mathcal{C}_{\text{prev-context}}^t = \{x_1, \ldots, x_{t-1}\} \setminus \{x_t\}. \tag{5}$$

Intuitively, minimizing the unlikelihood loss with this candidate set makes (i) incorrect repeating tokens less likely, as the previous context contains potential repeats, and (ii) frequent tokens less likely, as these tokens appear often in the previous context. These candidates are efficient to compute, without requiring additional supervision.

Gradient analysis We assume $p_\theta(x_t|x_{<t}) = \text{softmax}(a)$ and consider the gradient of (4) with respect to the softmax input $a \in \mathbb{R}^{\mathcal{V}}$. With a single negative candidate, the (negative) gradient is:

$$\nabla \mathcal{L}_a = x^* - m \odot p, \quad m_i = \begin{cases} (1 - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}) & \text{if } i \neq i_{\text{neg}} \\ (1 + \alpha) & \text{if } i = i_{\text{neg}}, \end{cases} \tag{6}$$

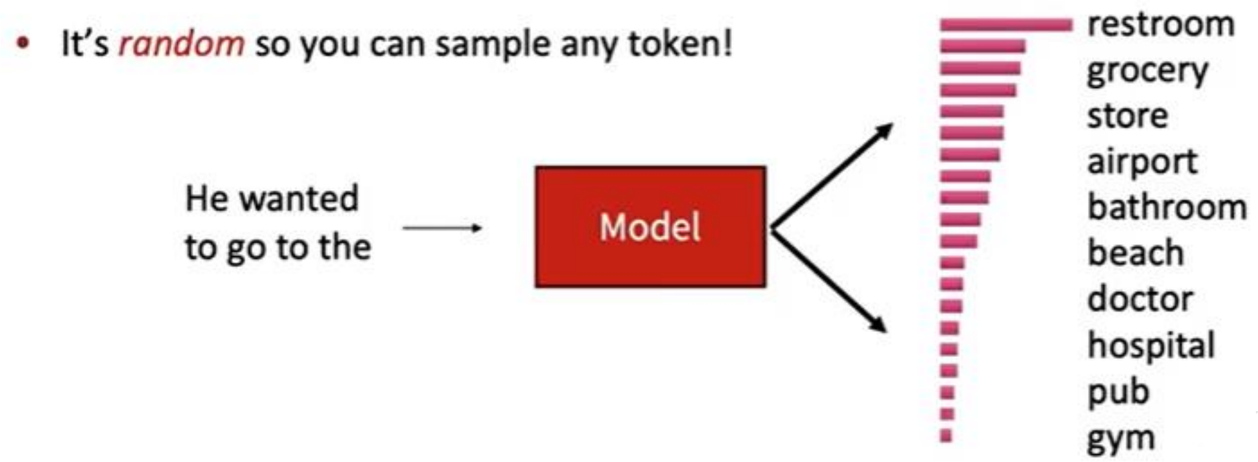
Neural Text Generation with Unlikelihood Training

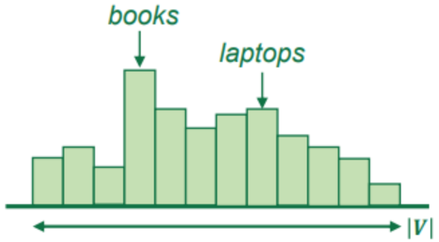
Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, Jason Weston

#02 Decoding

#3 Other methods

#3.1 Random sampling



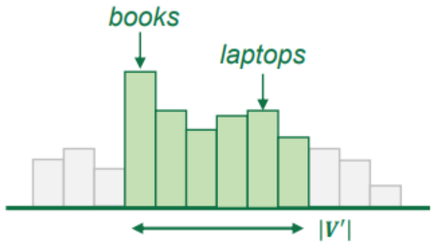


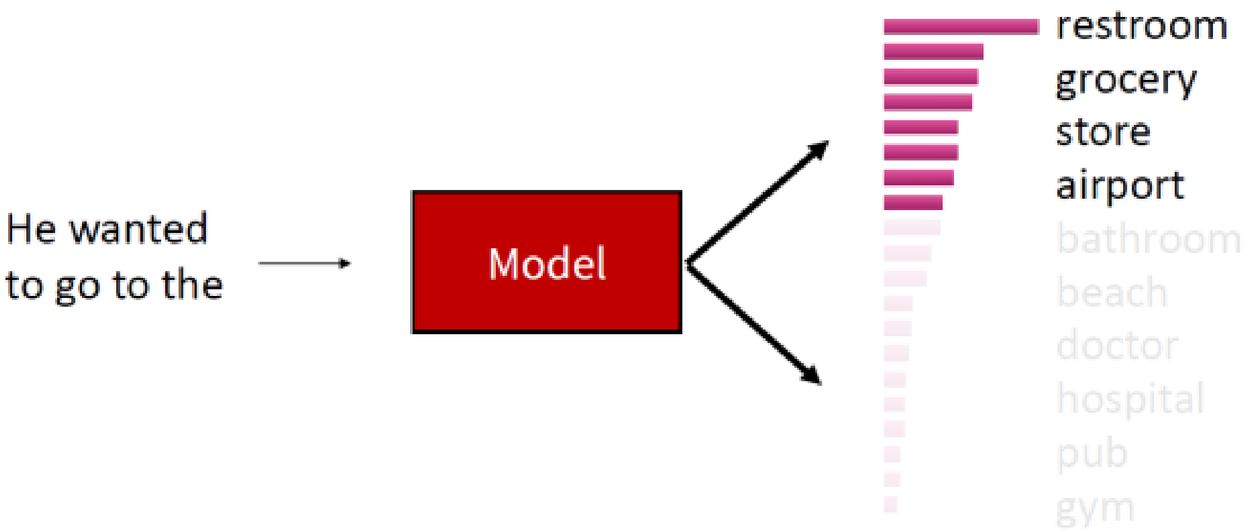
- Greedy Decoding과 비슷하지만, argmax 대신 sampling 사용
- 모델이 예측한 token distribution을 sampling에 사용 -> prob가 0이 아닌 어떤 단어도 등장 가능.

#02 Decoding

#3 Other methods

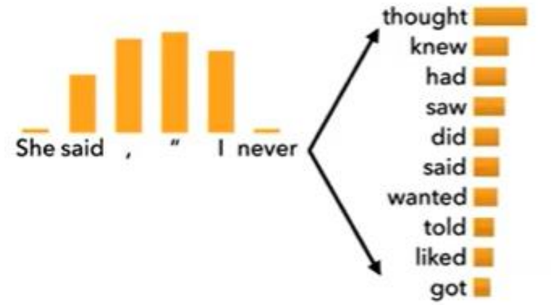
#3.2 Top-k sampling

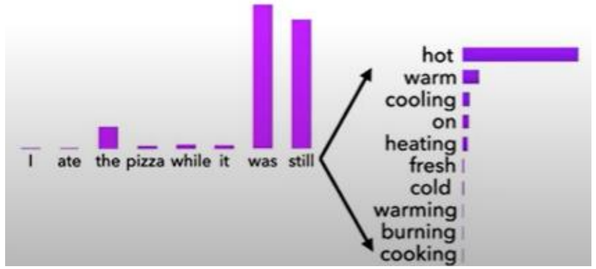




Issues:

- Top-k sampling can cut off too quickly!





- Top-k sampling can also cut off too slowly!

- 모델이 예측한 token distribution에서 상위 k개에서만 sampling
- k의 개수: ↑: more diverse/risky outputs vs. ↓: more generic/safe outputs

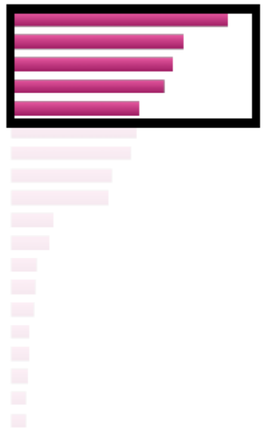
#02 Decoding

#3 Other methods

Varies k depending on the uniformity of P

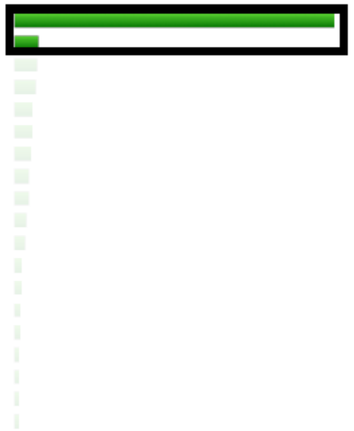
#3.3 Top-p sampling-> 누적확률을 사용(∵분포가 flatter/peaker 여부에 따라 k 선택에 영향)

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



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

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

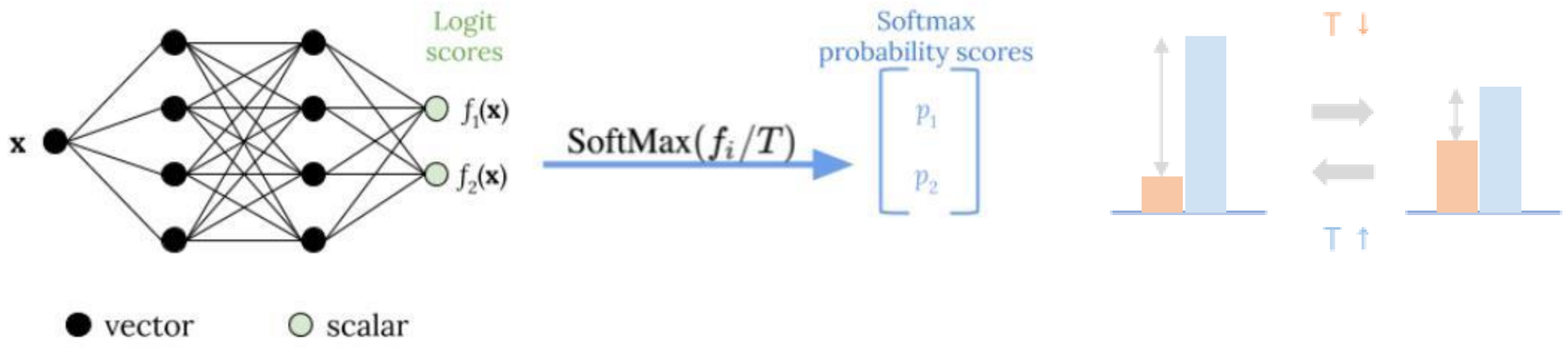


- 누적 확률 값이 p보다 작은 상위 토큰들만 샘플링에 사용
- P: ↑: diverse/risky outputs vs. ↓: generic/safe outputs

#02 Decoding

#3 Other methods

#3.4 Scaling randomness - 디코딩 알고리즘은 아니지만 다양한 디코딩 알고리즘과 함께 사용되는 방법



- Logits 값을 상수 T로 나누고, softmax의 입력 값으로 사용
- T: ↑ : diverse/risky outputs vs. ↓ : generic/safe outputs

#02 Decoding

#3 Other methods

#3.4 Scaling randomness - 디코딩 알고리즘은 아니지만 다양한 디코딩 알고리즘과 함께 사용되는 방법

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

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

- You can apply a *temperature hyperparameter* τ to the softmax to rebalance P_t:

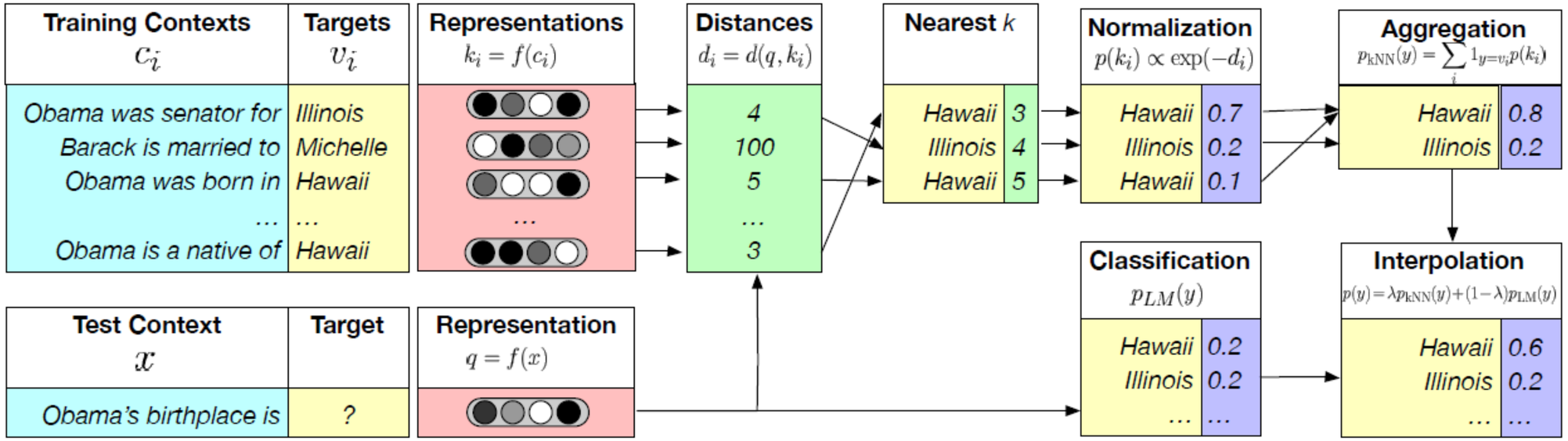
$$P_t(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

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

#02 Decoding

#3 Other methods (What if I don't trust how well my model's distributions are calibrated?)

#3.5 Re-balancing : KNN-LM

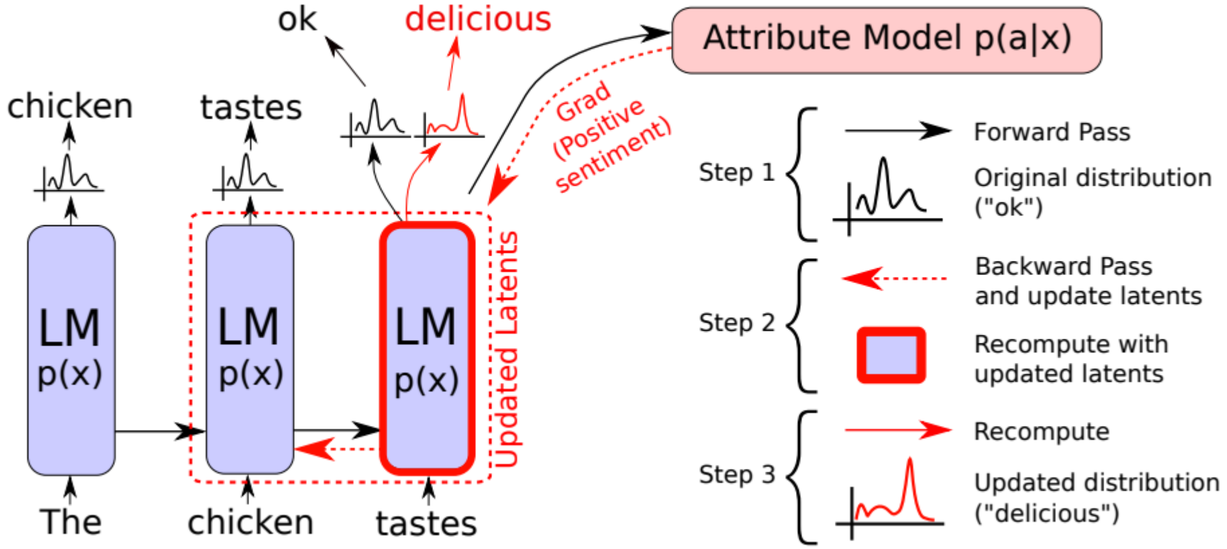


- Inference 시 평가 문장을 학습된 문장들의 representation과 비교하여 모델의 토큰 distribution을 보정해주는 방식
- K개의 인접한 representation 문장의 target 값을 사용해 모델 평가 결과를 보정(Re-balance P using induced distribution P_phrase over words that follow these phrases)

#02 Decoding

#3 Other methods (What if I don't trust how well my model's distributions are calibrated?)

#3.5 Re-balancing : PPLM

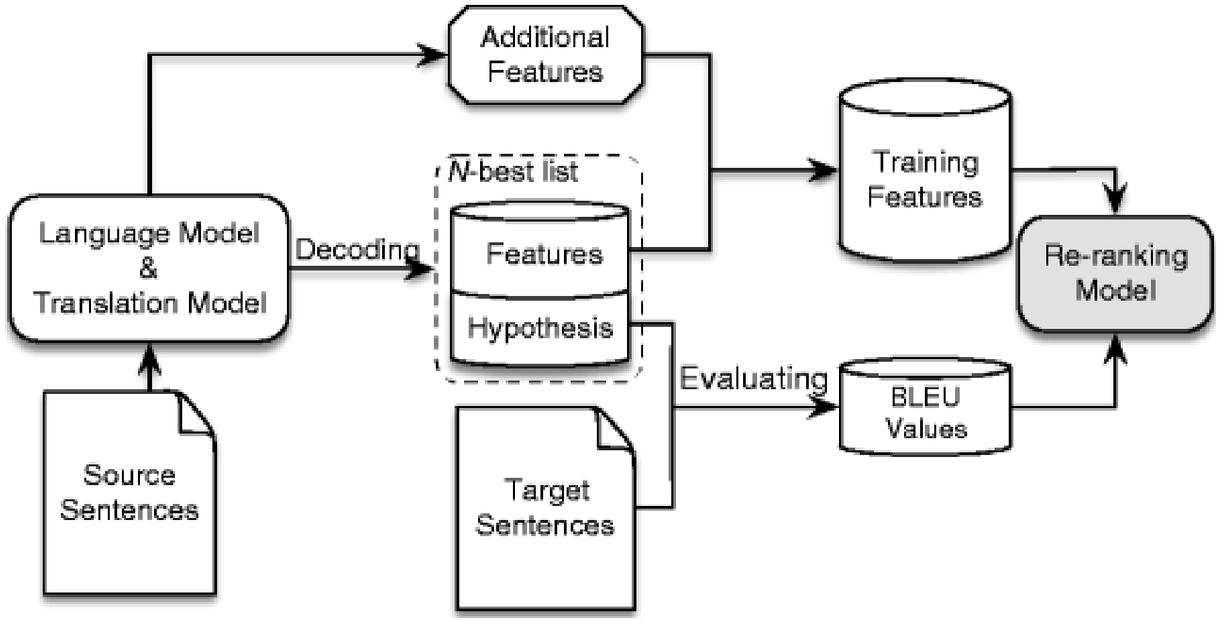


- 추가적 모델 사용해(e.g., Gumbel Softmax) 언어모델의 분포 조정 (ex. Sentiment, perplexity)
- Attribute model로부터 gradient를 전달받아 latent 업데이트 -> 모델 분포 업데이트

#02 Decoding

#3 other methods(What if I decode a bad sequence from my model?)

#3.6 Re-ranking



Decode a bunch of sequences (common number: 10)

Define a score to approximate quality of sequences and re-rank by this score

Re-rankers can score a variety of properties:

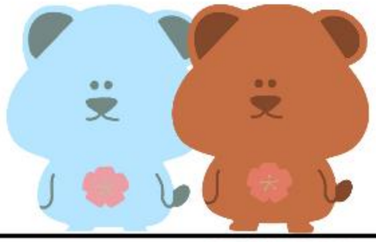
→ style, discourse, entailment/factuality, logical consistency, and many more..

-Can use multiple re-rankers in parallel

목차

training
-diversity issues
-exposure bias

#evaluating
- n-gram overlap metrics
- model based metrics
- human evaluations



EWHA EURON

Training – diversity issues

Unlikelihood training

$$C = \{y^*\}_{<t}$$

$$\mathcal{L}_{UL}^t = -\sum_{y_{neg} \in C} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

$$\mathcal{L}_{ULE}^t = \mathcal{L}_{MLE}^t + \alpha \mathcal{L}_{UL}^t$$

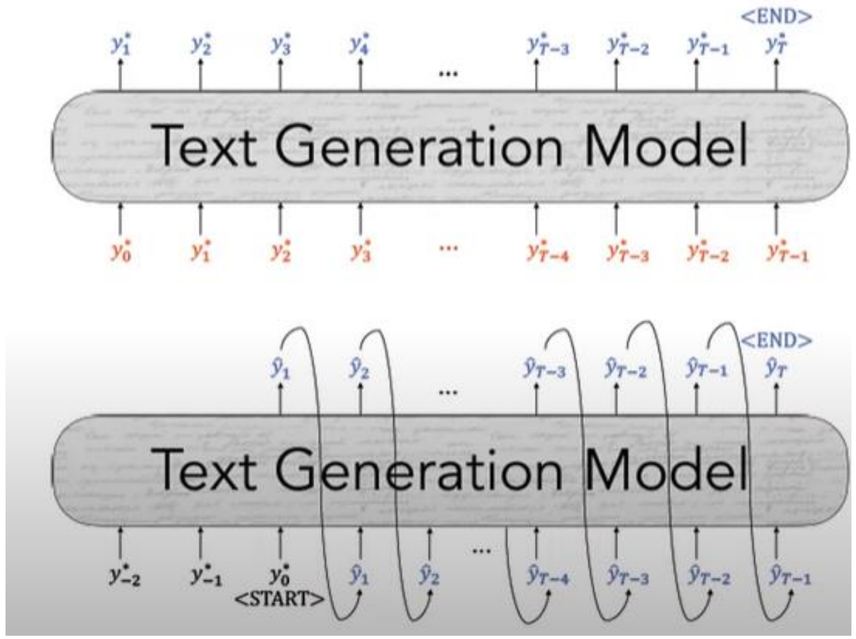
Neural Text Generation with Unlikelihood Training

Q. 학습된 언어 모델이 디코딩 과정에서 비슷한 단어 or 구로 반복해서 생성됨

A. 학습 과정에서 이미 생성된 토큰의 probability를 낮추는 penalty loss를 추가

Training – exposure bias

Teacher forcing으로 인한 bias



Q. Overfitting ; teacher forcing으로 인해 모델이 불필요한 bias를 학습하게 됨.

; teacher forcing을 부자연스러운 문장을 생성하는 원인으로 봄.

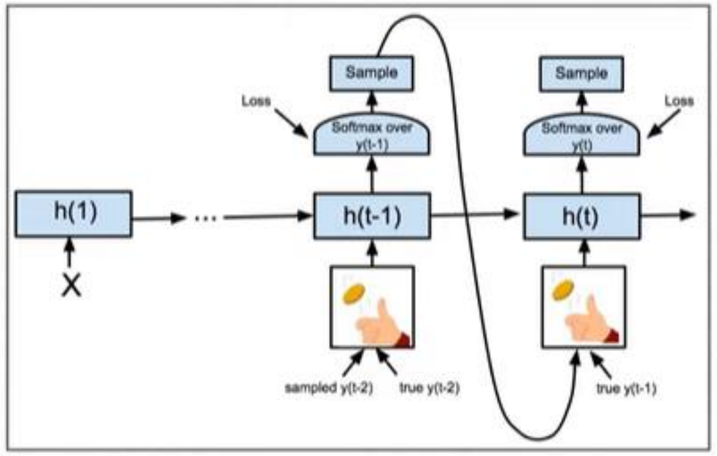
-> 언어 모델 학습 시 잘못 생성된 단어가 다음 토큰 사용을 위해 반복되어 상용됨을 막기위해 원문장을 그대로 사용하는 teacher forcing을 사용함으로써 학습 시 exposure bias가 야기 된다.

A. dataset aggregation
scheduled sampling
sequence re-writing
reinforcement learning

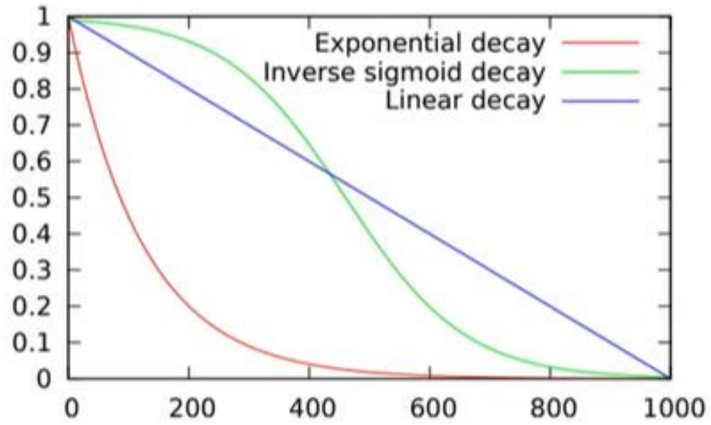
Training – exposure bias

Exposure bias solutions – scheduled sampling

<Overall framework>



<Decay scheduels>

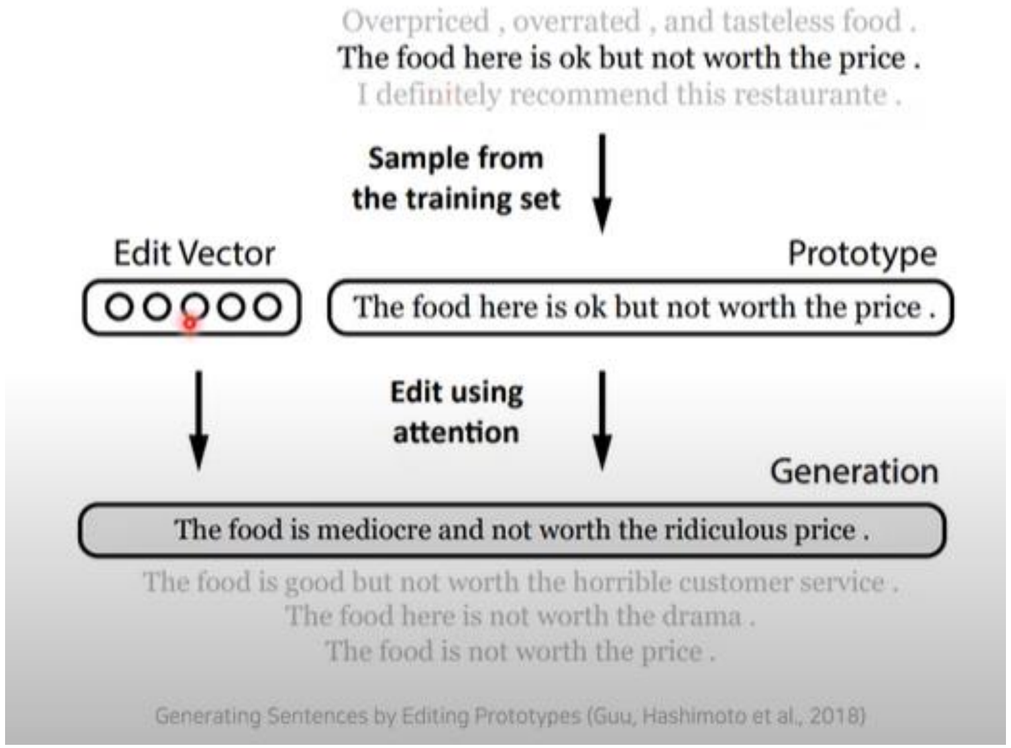


Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks (Bengio et al., 2015)

; teacher forcing을 랜덤하게 사용하는 방식
;모델 예측 토큰을 다음 스텝의 입력으로 사용
;샘플링에 사용되는 확률 p(t시점에 teacher forcing울 사용할 확률)는 학습이 진행될수록 더 작은 값을 사용하게 됨.

Training – exposure bias

Exposure bias solutions – sequence re-writing



;좀 더 자연스러운 문장을 생성하기 위해 prototype set 을 사용

;학습데이터에서 prototype을 샘플링하고 edit vector는 이 prototype을 변형하여 targe sentence를 생성

;edit vector는 adding, removing, modifying tokens 3방식을 이용해 sampling

Training – exposure bias

Exposure bias solutions – reinforcement bias solution

$$\mathcal{L}_{RL} = -\sum_{t=1}^{T} r(\hat{y}_t) \log P(\hat{y}_t | \{y^*\}; \{\hat{y}_t\}_{<t})$$

; text생성 모델을 Markov decision process로 구성하고 강화학습 알고리즘을 사용하면 평가지표들을 reward 로 직접 사용할 수 있게 됨.

State : 이전 context의 representation
Actions: 현재 step에서 생성될 수 있는 단어
Policy: decoder
Rewards: score함수로부터 받게 될 보상
(ex. BLEU, ROUGE ,CIDEr, SPIDEr등이 score로 사용)

Reward estimation
;의도하지 않은 shortcut을 모델이 학습하지 않도록 reward function을 잘 정의해야 함.

Evaluating – n-gram overlap metrics

n-gram overlap metrics – BLEU, ROUGE

;summarization task에서 요약문장이 길 때, open-ended MT에서 적절한 지표가 되지 못함.

;단어의 문맥적 의미를 반영할 수 없고 사람이 평가했을 때와 차이가 큼.

> BLEU

$$BLEU = min(1, \frac{output\ length(예측\ 문장)}{reference\ length(실제\ 문장)})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

짧은 문장에 대한 penalty

N-gram precision 의 기하평균

;실제 문장 대비 짧은 문장을 산출할 경우 penalty부여
;윈도우 사이즈에 따라 계산되는 precision을 모두 사용하여 하나의 지표로 산출

> ROUGE

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

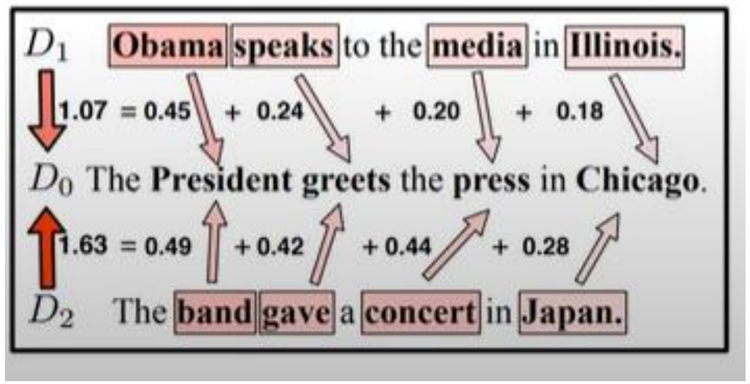
;n-gram recall을 사용하여 짧은 생성 문장에 penalty를 부여하는 brevity penalty가 따로 없다.

;rouge1,2와 같이 n-gram별로 다른 지표를 사용하여 비교

Evaluating – model based metrics

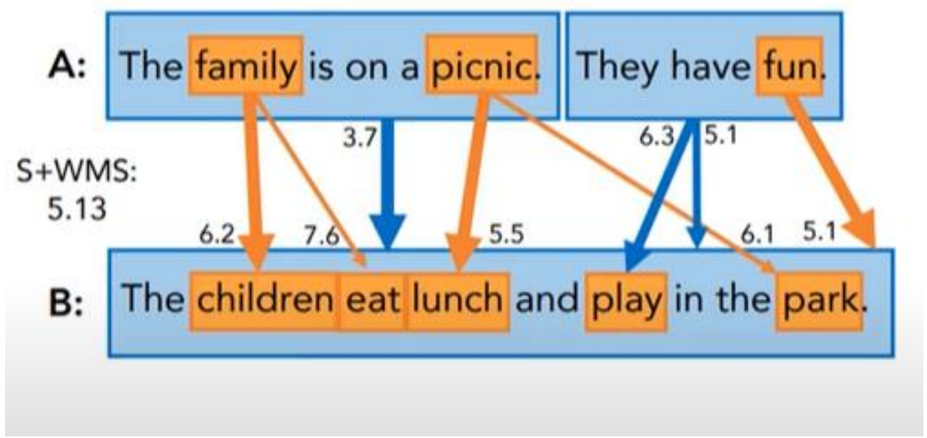
model based metrics– word mover's distance, sentence mover's similarity

> word mover's distance

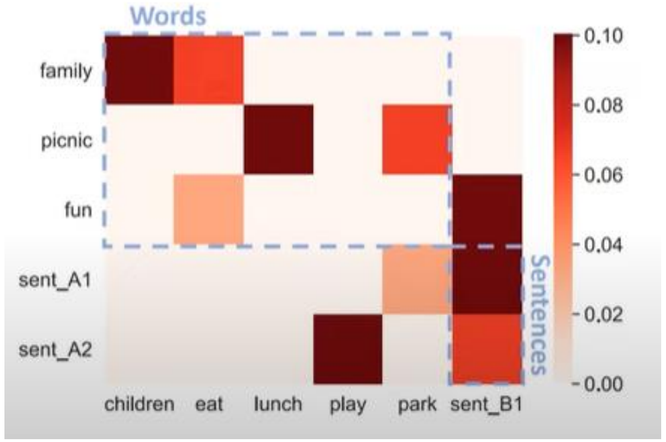


;document 단어들의 거리를 계산하여 상대적으로 유사한 의미를 가진 문장을 판단할 수 있음.

> sentence mover's distance

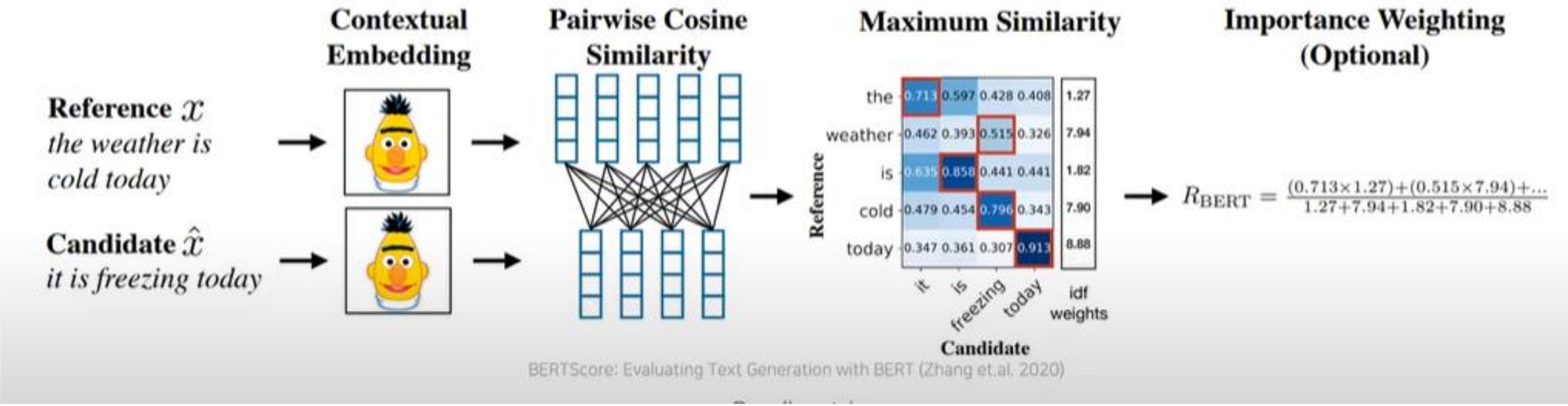


;word level, sentence level 모두에서 similarity 계산



Evaluating – model based metrics

model based metrics– BERT SCORE



; reference문장과 candidate 문장의 contextual embedding을 사전 학습된 bert를 사용하여 계산하고 bedding 간의 cos similarity를 계산하는 방식

;모든 pair에 대해 cos similarity를 계산하고 greedy matching 후 weighted average를 계산하여 bert score를 계산.

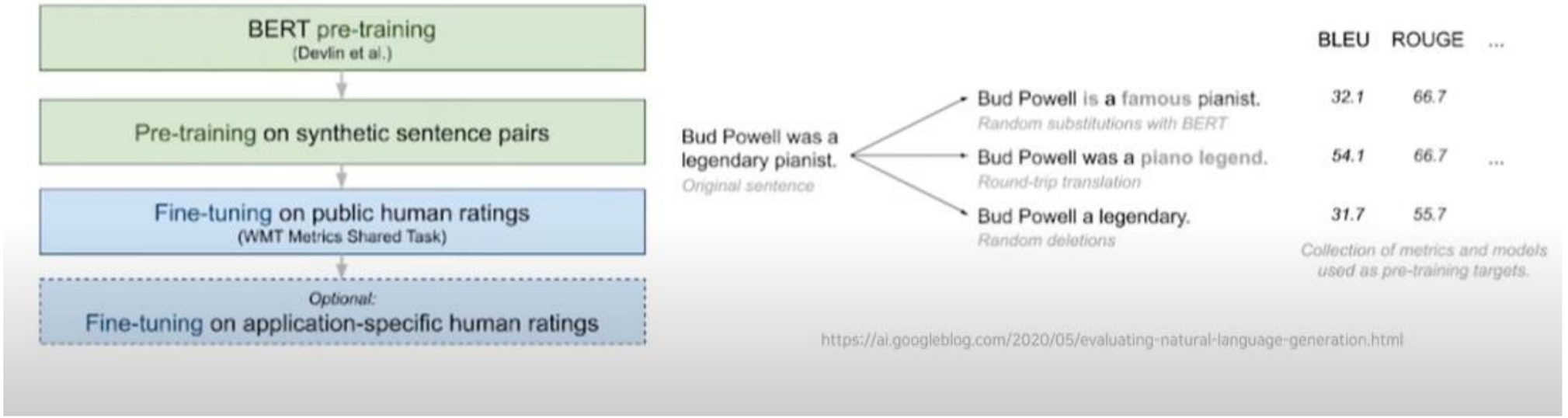
; df : 특정 키워드가 출현한 문서 빈도수 , idf : df의 역수 -> idf를 가중치로 사용

$$\text{idf}(w) = -\log \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}[w \in x^{(i)}]$$

; x(i)는 i시점 이전까지 사용된 단어들의 집합으로,
w가 거기에 속할 때 그만큼의 가중치를 주는 방식

Evaluating – model based metrics

model based metrics– BLEUART



;reference문장과 candidate 문장의 유사도를 출력하는 bert기반의 regression model을 직접 학습하는 방식을 사용.

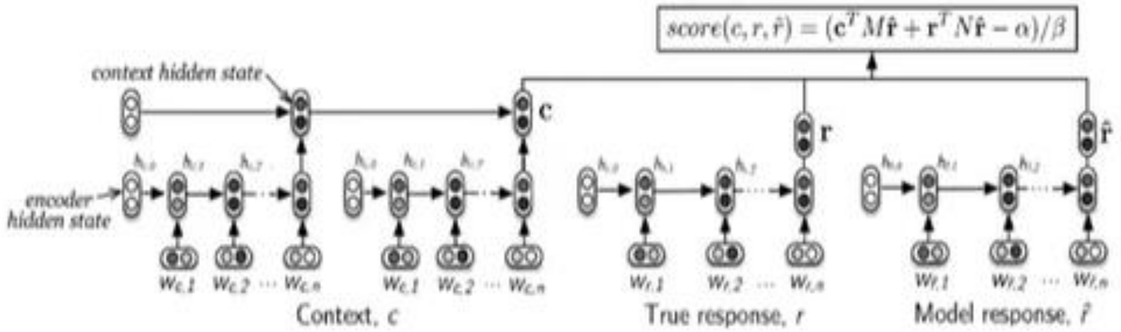
;평가 score를 학습을 통해 계산

; pre-train과정에서 위키피티아 셋을 사용.

;추가적으로 평가하고자 하는 data와 task에 대해 human rating data를 구축하고 application specific fine tuning을 할 경우 추가적으로 성능향상이 가능함.

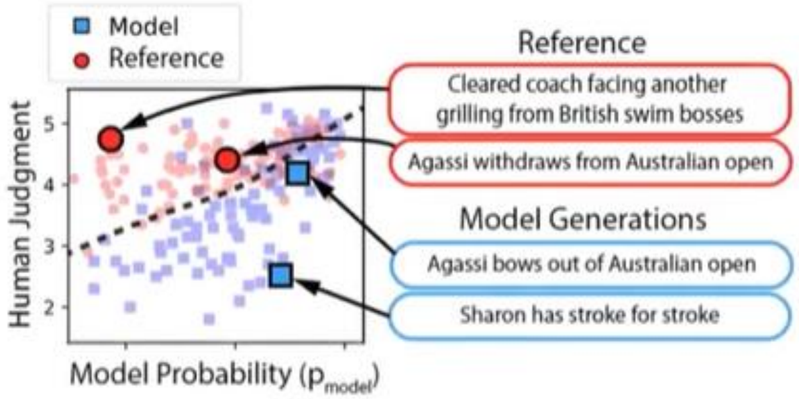
Evaluating – human evaluation

human evaluation



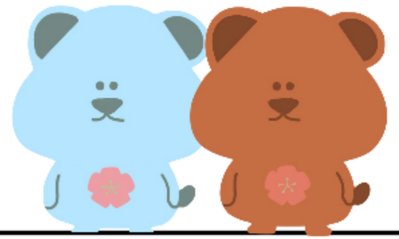
; 생성된 문장의 퀄리티를 평가하기 위해 fluency, grammaticality, typicality 등의 구체적인 척도를 가지고 사람이 평가하는 방식

; 사람이 직접 평가하므로 상대적으로 주관적인 평가를 하게 됨.



THANK YOU

출처: Stanford CS224N NLP with Deep Learning | Winter 2021 | Lecture 12 - Natural Language Generation – YouTube
[DSBA] CS224n 2021 Study | #12 Natural Language Generation - YouTube



EWHA EURON