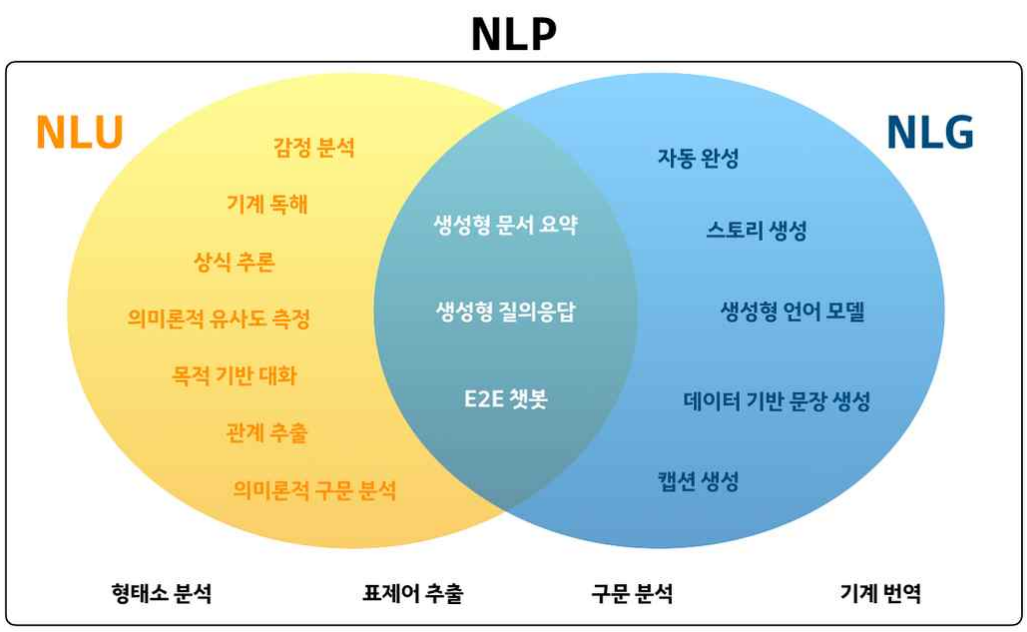
Lecture 5. Natural Language Generation

> 2023.11.05 투빅스 20기 조민영

1 Natural Language Generation란



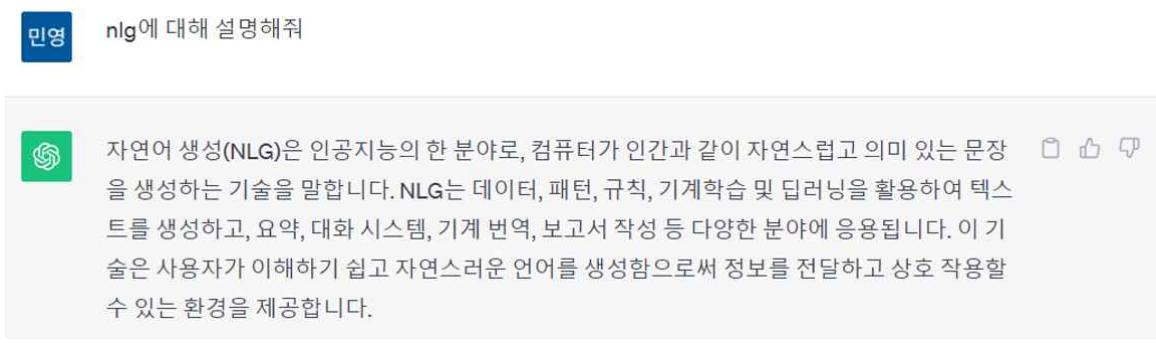
NLP = Natural Language Understanding(NLU) + Natural Language Generation(NLG)

- NLU : natural language를 기계가 이해할 수 있는 형태로 변환
- NLG : 기계가 출력한 결과를 natural language로 자동 생성 ✓

NLG는 출력으로 텍스트 시퀀스를 만드는 task를 수행한다.

- 요약
- 대화(Digital assistant)
- 창작(시나 소설)
- 이미지 캡션
- 기계 번역

이외에도 사람의 편의를 위해 text를 생성하는 모든 업무에 NLG가 사용될 수 있다.

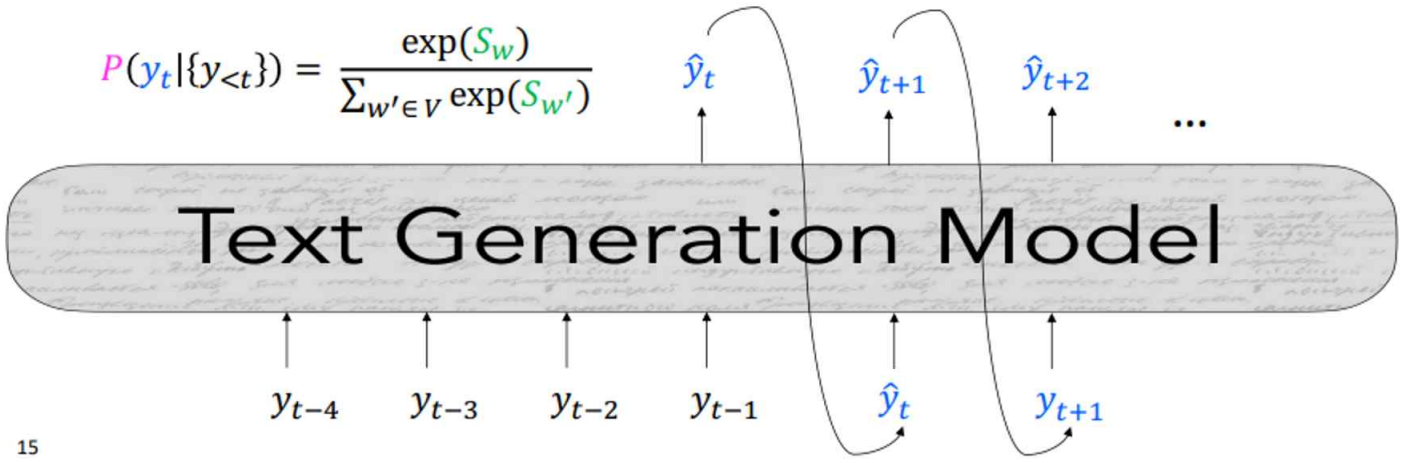


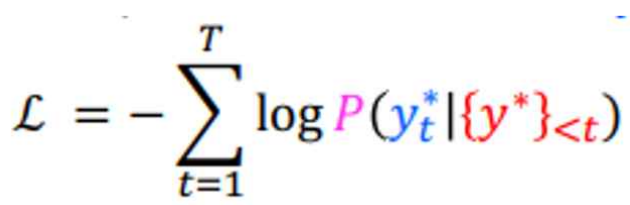
ChatGPT는 NLG의 한 예시이다.

2 Basic NLG

Basic NLG

일반적인 NLG model은 t시점의 단어를 생성하기 위해 이전 시점의 단어들을 input으로 사용한다. 이를 반복해 t 시점 이후의 단어들도 생성한다.





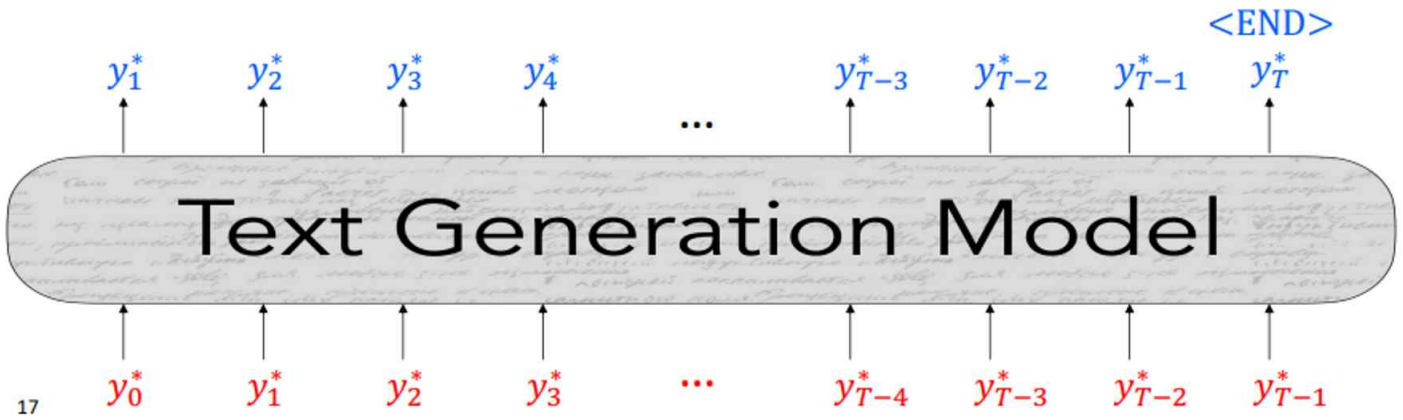
학습 시에 negative loglikelihood를 구할때 예측된 값(파란색)이 아닌 실제값(빨간색)을 이용

과거의 자신의 값을 참조하여 현재의 값을 예측하는 구조를 가졌기 때문에 Autoregressive 형태라고 하며, 학습시엔 예측된 토큰과 실제 토큰을 사용한 Negative loglikelihood를 최소화하는 방식으로 학습한다.

그러나 자신의 과거 값을 참조하기 때문에, 과거에 잘못된 예측을 하게 되면 점점 시간이 지날수록 더 큰 잘못된 예측을 할 가능성을 야기하기도 한다.

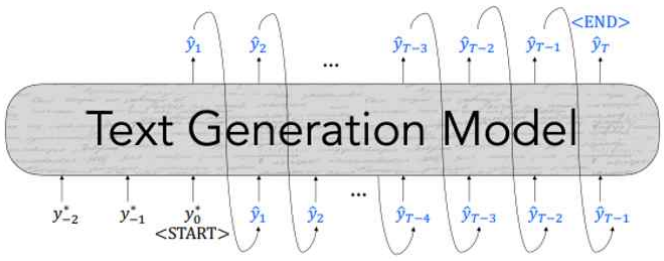
Teacher forcing 교사 강요

학습 시 모델이 예측한 값이 아닌, 실제 문장의 토큰을 단어 생성을 위한 input으로 사용할 수 있다. 초기에 잘못 생성된 단어로 인해 이후 잘못된 단어가 생성되는 것을 막아준다.



모델 학습시

정답값으로 학습된 모델을 사용하여, 평가 시에는 실제 정답값을 모르기 때문에 예측된 단어를 다음 단어 예측을 위한 입력값으로 사용한다.

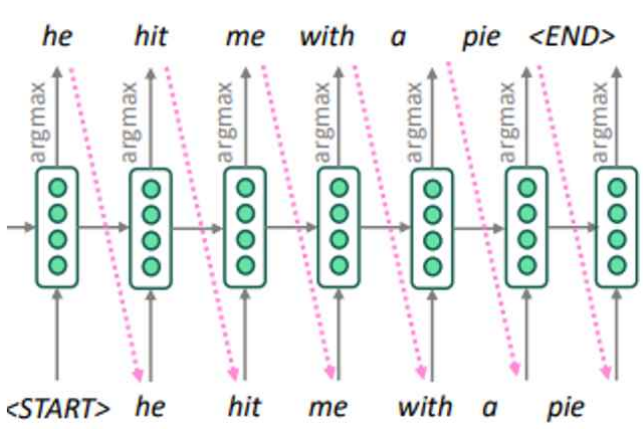


모델 평가시

3 Decoding

Greedy Methods

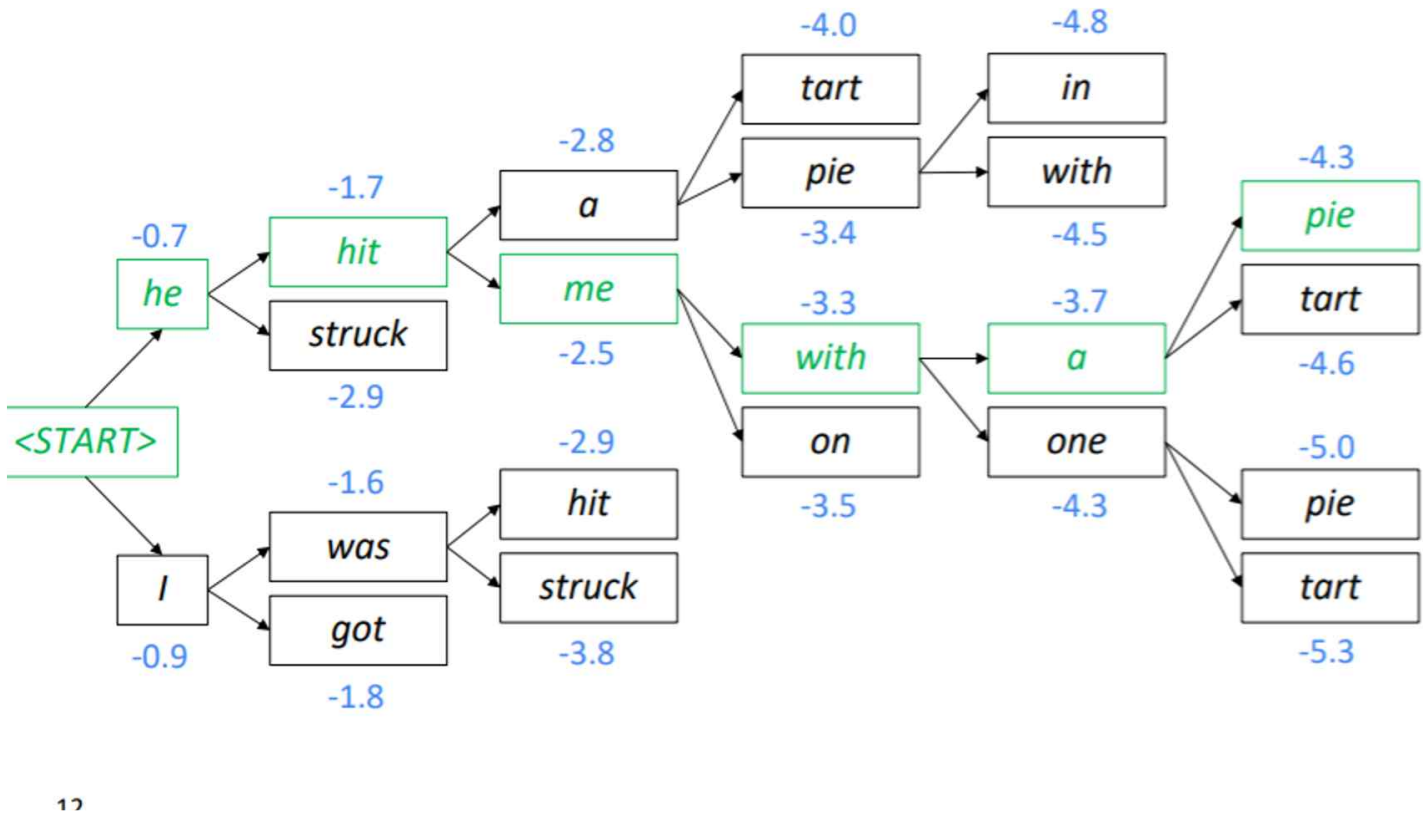
1) Greedy Decoding



seq2seq 강의자료

디코더는 t-1 시점에서 나올 수 있는 확률이 높은(argmax) 단어를 선택하여 t시점의 단어를 예측한다. <END> 토큰이 나올 때까지 반복하는데 한 시점에서 잘못된 토큰을 뽑을 경우, 그 이후 시점은 모두 잘못된 결과를 뽑을 수 밖에 없다.

2) Beam Search

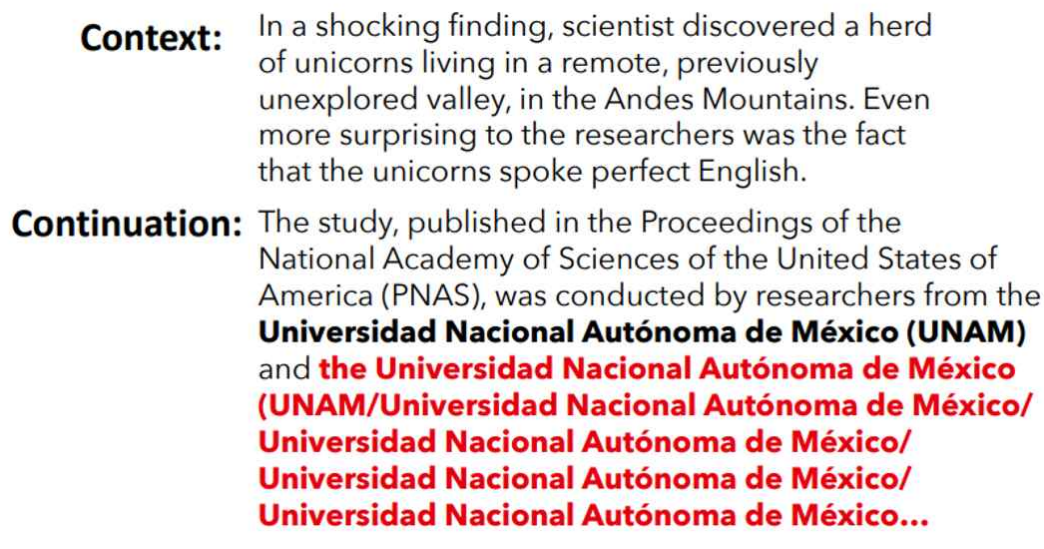


- beam size(k) = 2 인 경우. 두 번째 step에서 (hit, struck, was, got) 중 (was, hit) 만 채택된다.

beam search는 greedy decoding의 단점을 보완한 방법으로, 디코더의 각 스텝에서 가장 가능성이 높은 k개의 결과를 선택하고, 나머지는 가지치기를 한다.

가장 가능성이 높다 = score(음수)가 높다 = log probability(음수)가 높다

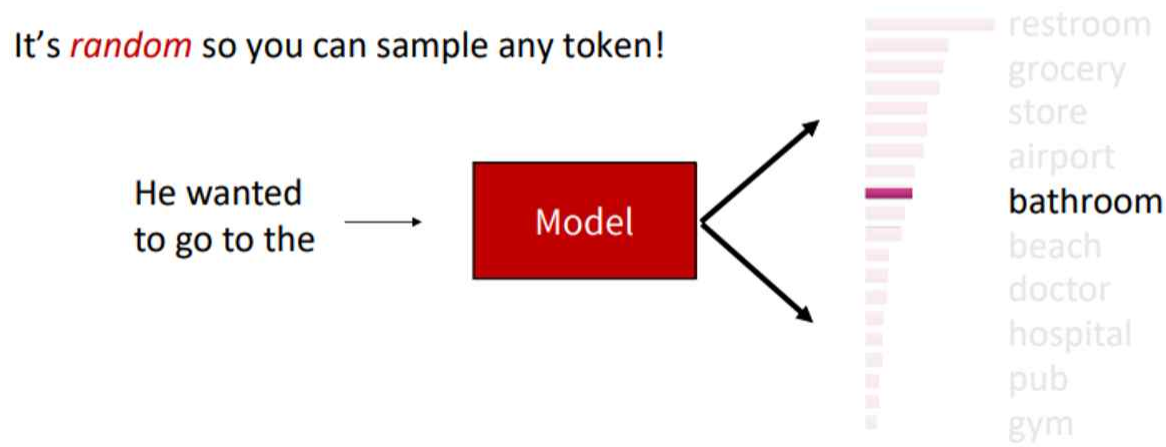
그러나 Greedy Methods 방식은 문장이 길어지면서 비슷한 단어가 반복된다는 문제를 가지고 있다.



How to Reduce Repetition

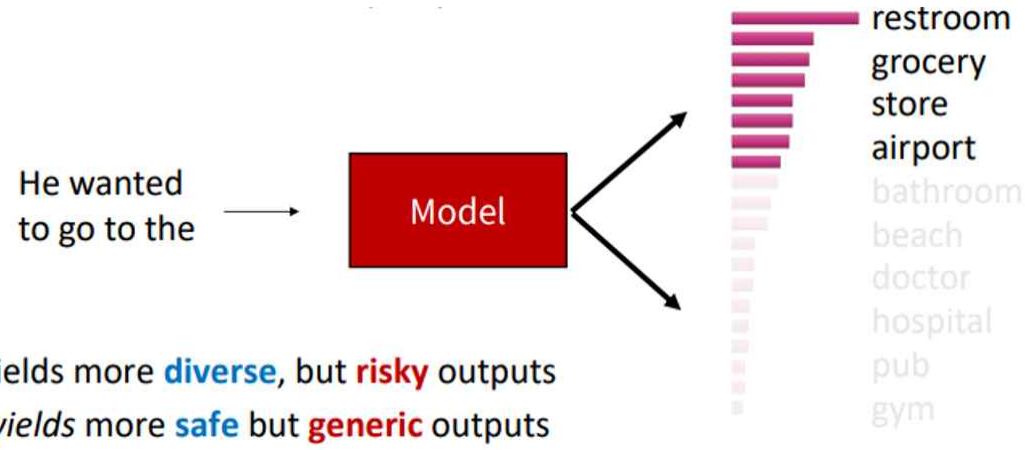
1) Random Sampling

$$\hat{y}_t \sim P(y_t = w \mid \{y\}_{<t})$$



모델이 예측한 token distribution을 토대로 랜덤으로 골라 다음 단어로 사용한다. 어떤 단어든 (prob≠0) 등장할 수 있다.

2) Top-k Sampling

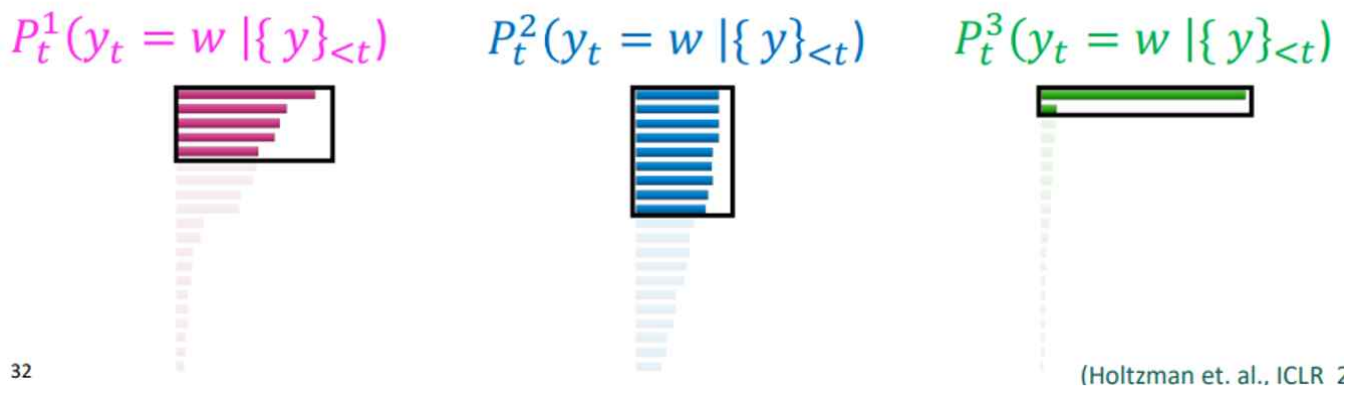


- Increase *k* yields more **diverse**, but **risky** outputs
- Decrease *k yields* more **safe** but **generic** outputs

모델이 예측한 token distribution에서 상위 k개에서 랜덤으로 샘플링한다.

k가 클수록 다양하지만 부자연스러운 문장이 생성될 수 있고, k가 작을수록 일반적이지만, 그럴듯한 문장이 만들어질 수 있다.

3) Top-p Sampling



만약 Top-k Sampling을 사용한다고 하면

- 비교적 고른 distribution에서는 높은 확률을 가지는 토큰들이 사용되지 않게 되고,
- 상위 한 두개가 높은 확률을 가지게 될 때 확률이 낮은 토큰들이 사용되게 된다.

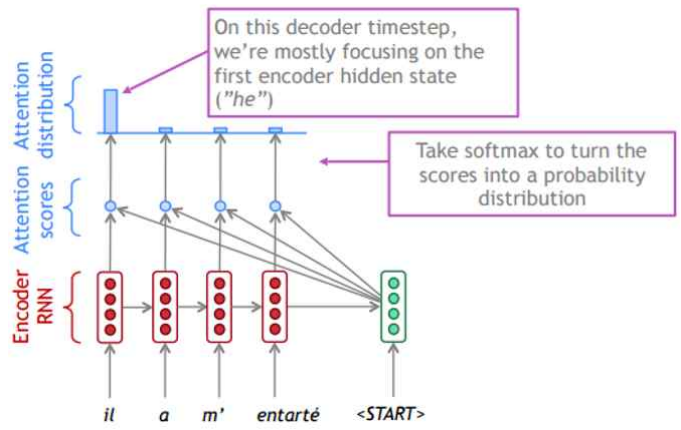
따라서 누적 확률 값이 p 보다 큰 상위 토큰들만 샘플링에 사용한다.

p 값이 작을수록 더 많은 토큰들이 샘플링에 사용되므로 다양하지만 부자연스러운 문장이 생성되고, p가 클수록 일반적이고 안정적인 문장을 생성할 수 있다.

4) Softmax Temperature

t 시점에서 score에 대한 distribution을 구하기 위해 softmax 함수를 이용하여 확률값을 구한다.

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

> 💡 Recap) Attention
>
> 
>

softmax temperature는 score에 τ값을 나눠준 항을 이용하여 구한다. 여기서 τ가 온도에 해당한다.

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

소프트맥스 함수는 지수함수를 포함하므로 입력값이 작을수록 작은 값을 가진다. 그리고, 지수함수 값이 클수록 변화가 더 크다.

이를 이용해서 τ, 온도로 나눠주면 소프트맥스를 통과한 분포가 조금 더 평평해진다.

- 온도가 오를수록, τ가 클수록 분포가 녹아서 더 균일해지므로 다양한 표현들이 나올 수 있고,
- 온도가 내려갈수록, τ가 작을수록 분포는 얼어서 더 뾰족해진다. 뾰족한 분포에선 더 일반적인 표현들이 나오게 된다.

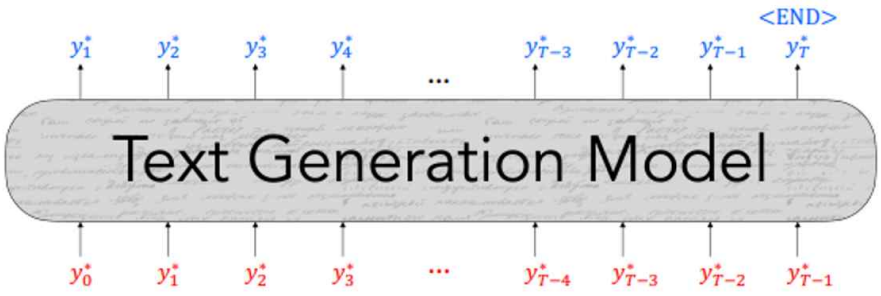
4 Training

Repetition 문제가 모델을 학습시키는 과정에서 발생된게 아닐까?

Exposure Bias

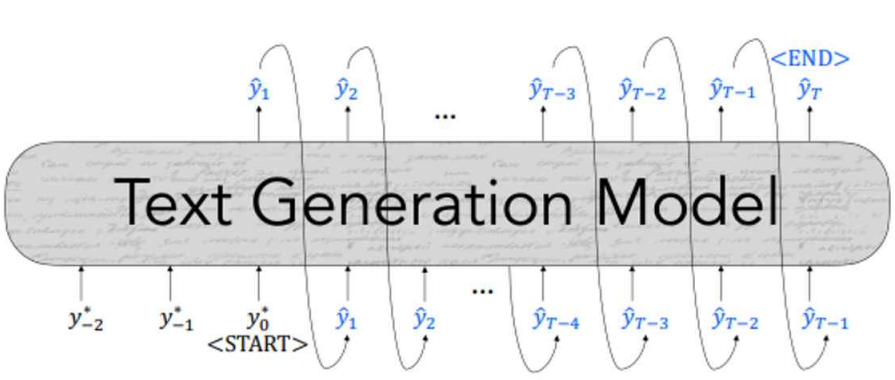
언어 모델 학습 시 잘못 생성된 단어가 다음 토큰 생성에 반복해서 사용되는 것을 막기 위해 실제 문장의 단어를 그대로 사용하는 Teacher forcing을 일반적으로 사용한다. 그러나 teacher forcing은 test 시에 모델로 생성된 단어가 input으로 들어가기 때문에 불필요한 bias를 야기한다고 볼 수 있다.

- 모델 학습 : 실제값을 input으로 다음에 올 단어를 예측한다.



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

- text generation(test) : 이전에 디코딩된 토큰이 input으로 들어간다.



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

Exposure Bias solutions

- Scheduled sampling
- Dataset Aggregation
- Retrieval Augmentation
- Reinforcement Learning

1) Scheduled sampling

- teacher forcing에서 training과 generation의 차이는 input으로 실제값(gold token)과 예측값(docode a token) 중 어느 것을 사용하는가이다.
- scheduled sampling은 두 값을 랜덤하게 선택하자는 방식이다.
- t시점에서 teacher forcing을 사용할 확률을 p라고 했을때, p=1이면 실제값(gold token)을 사용하겠다는 것이고, p=0이면 예측값(docode a token)을 사용하겠다는 뜻이다. (Figure 1)
- 학습 초기에는 모델의 학습 수렴이 잘 안되기 때문에 true token을 많이 사용하고, 나중에는 inference와 비슷하게 되어야 하기 때문에 model prediction token을 많이 사용한다. (Figure 2)

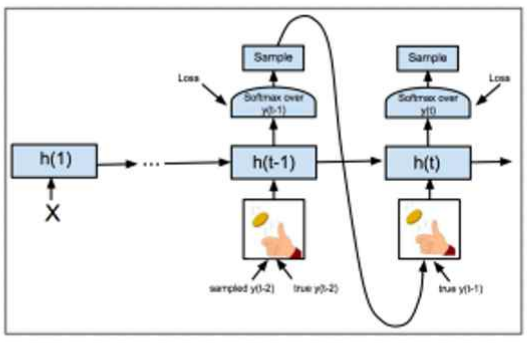


Figure 1: Illustration of the Scheduled Sampling approach, where one flips a coin at every time step to decide to use the true previous token or one sampled from the model itself.

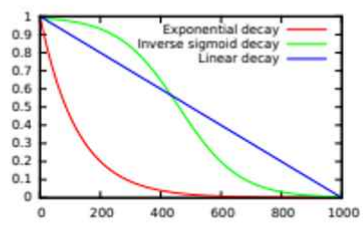


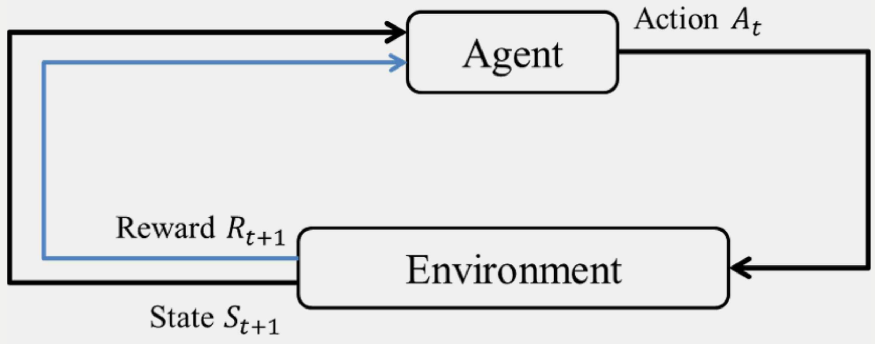
Figure 2: Examples of decay schedules.

https://arxiv.org/pdf/1506.03099.pdf

2) Reinforcement Learning

- 언어모델의 성능평가지표에는 주로 BLEU, ROUGE score 등이 사용되고, 이 평가지표들은 학습에 직접적으로 사용될 수 없다.
- 그러나 Text 생성 모델을 Markov decision process로 구성해 강화학습 알고리즘을 사용하면 이 평가지표들을 reward로 직접 사용할 수 있게 된다.

> 💡 Markov decision process(MDP)
>
> <S,A,P,R,γ>라는 tuple로 구성된다.
>
> - S, State : 이전 context의 representation
> - A, Actions : 현재 step에서 생성될 수 있는 단어
> - P, Policy : Decoder, 학습을 통해 업데이트될 대상
> - R, Rewards : score 함수로부터 받게 될 보상(BLEU, ROUGE 등)
> - γ : 즉각적으로 얻는 reward와 미래의 얻을 수 있는 reward 간의 중요도를 조절하는 변수



MDP의 동작

- 의도하지 않은 Shorcut을 모델이 학습하지 않도록 reward function을 잘 정의해야 된다는 문제가 남아있다.

5 Evaluating

N-gram overlap metrics

1) **BLEU** : n-gram precision 기반

$$BLEU = BP \times exp(\sum_{n=1}^{N} w_n \log p_n)$$

- pn : 보정된 n-gram precision
- N : n-gram에서 n의 최대 숫자
- wn : 각 gram의 보정된 정밀도에 서로 다른 가중치 부여 가능

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

c : Candidate의 길이
r : Candidate와 가장 길이 차이가 작은 Reference의 길이

- 실제문장(reference) 보다 짧은 문장을 생성(candidate)할 경우에 패널티를 부여하는 방식이다.
- 일반적으로 많이 사용되는 metric이다.

2) **ROUGH** : n-gram recall 기반

$$ROUGE-N = \frac{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

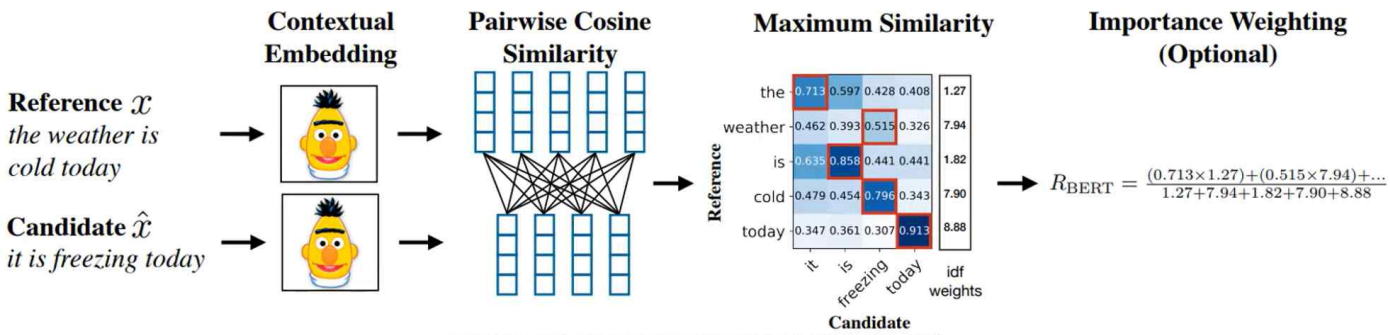
https://supkoon.tistory.com/26

- Brevity penalty가 따로 없다.
- BLEU와는 달리 n-gram 별로 따로 비교한다. (ROUGE-1, ROUGE-2, ROUGE-L)

그러나 N-gram overlap metrics은 open-ended MT(summarization output text가 길 때, dialogue task 등)에서 적절한 지표가 될 수 없다. 단어의 문맥적인 의미를 반영할 수 없고, 사람의 평가와 상관성이 낮기 때문이다.

Model based metrics

1) **BERT SCORE**(word distance function)



BERTScore: Evaluating Text Generation with BERT (Zhang et.al. 2020)

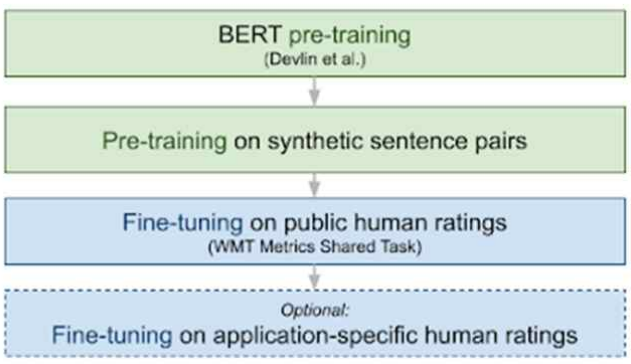
<Recall metric>

- reference와 candidate에 BERT를 적용하여 contextual embedding 계산
- 모든 token-pair에 대해 cosine similarity를 계산
- greedy matching 후 weighted average를 적용하여 BERT score를 구한다.

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

inverse document frequency score

2) **BLEURT**(beyond word matching)



$$\boldsymbol{v}_{[\text{CLS}]}, \boldsymbol{v}_{x_1}, ..., \boldsymbol{v}_{x_r}, \boldsymbol{v}_1, ..., \boldsymbol{v}_{\tilde{x}_p} = \text{BERT}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$$

$$\hat{y} = \boldsymbol{f}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \boldsymbol{W}\tilde{\boldsymbol{v}}_{[\text{CLS}]} + \boldsymbol{b}$$

- reference 문장과 candidate 문장의 유사도를 예측하는 BERT 기반의 regression 모델을 직접 학습한다.
- Pre-training : 위키피디아 텍스트 데이터를 사용하여 transformer 모델을 미리 학습시킨다.

- Fine-tuning : WMT와 사람의 평가를 구축해 모델을 재훈련시킨다.

정리

- nlg는 기계가 출력한 결과를 자연어로 자동생성하는 기술분야이다.
- 일반적인 nlg 모델은 teacher forcing으로, 모델 훈련시에는 실제 토큰을 사용하고, 예측시에는 전 시점에 모델이 예측한 토큰을 input으로 사용한다.
- 그러나 nlg 모델은 output 문장이 길어질수록 단어가 반복된다는 문제가 있다.
- 디코딩에서 해결하는 방법들
  - random sampling, top-k sampling, top-p sampling, softmax temperature
- 단어 반복 문제가 모델을 학습시킬때 발생한다는 관점도 있었다.
- 모델 학습시 exposure bias를 해결하는 방법들
  - Scheduled sampling, Dataset Aggregation, Retrieval Augmentation, Reinforcement Learning
- nlg 모델 평가 지표
  - n-gram overlap metrics, model-based overlap metrics

references

투빅스 15&16기 텍스트 세미나
https://velog.io/@tobigs1516text/CS224n-Lecture-15-Natural-Language-Generation

[DSBA] CS224n 2021 Study
https://www.youtube.com/watch?v=RkCbFQ1W6_Q

BERT score
https://velog.io/@tobigs-nlp/BERTScore-Evaluating-Text-Generation-with-BERT