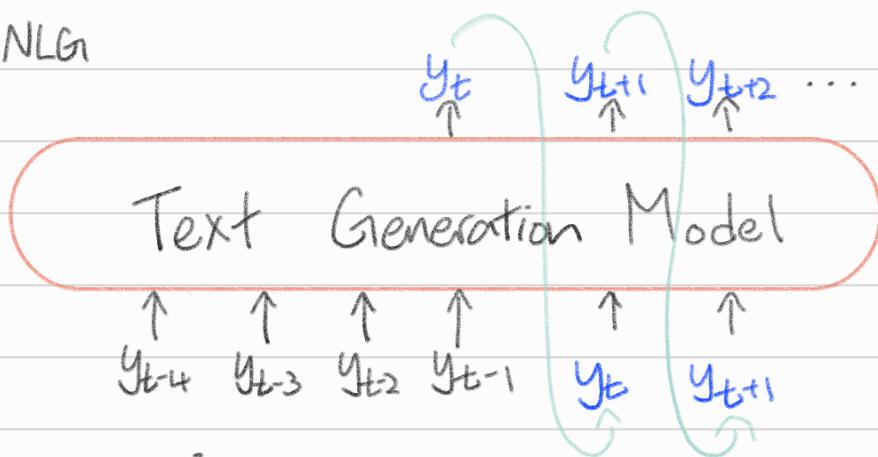


Natural Language Generation

Basic NLG



- Autoregressive 형태
- 생성하고자 하는 step의 단어 생성 위해 t 이전 단어들을 입력으로 사용
- 학습시 예측된 token과 실제 token 사용한 negative loglikelihood를 최소화

Teacher forcing

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

- 학습시 모델이 예측한 token이 아닌 실제 문장의 token을 다음 단어 생성 위한 입력으로 사용
- 초기에 잘못 생성된 단어로 인해 이후 계속 잘못된 단어 생성 방지
- 학습된 모델을 사용해 decoding 시엔 예측된 단어를 다음 단어 예측 위한 입력으로 사용

NLG decoding (Greedy Method)

• Argmax Decoding

- Step마다 생성될 수 있는 words의 likelihood 계산, 그 중 가장 큰 값의 token을 다음 step의 입력으로

• Beam Search

- Argmax 보다 좀 더 많은 후보들을 고려 argmax decoding 보다 선호

Greedy method 사용 시 문장이 길어질 때 단어가 반복되는 문제 발생



Decoding algorithm 대안:

random sampling, top-k sampling, top-p sampling, ...

연속되는 문장의 hidden representation similarity 낮추기

별도 loss 사용, unlikelihood objective 등

Random Sampling

(prob ≠ 0)

- 모델이 예측한 token distribution을 Sampling에 사용 → 어떤 단어든 등장 가능

Top-K Sampling

- Random Sampling에서 낮은 확률로 생성된 부자연스러운 문장을 방지하고자 사용
- 모델이 예측한 token distribution에서 상위 K개에서만 sampling
- K가 큼수록 다양하지만 부자연스러운 risk

주로 $K=5, 10, 15$

작을수록 일반적이고 안정적

Top-p Sampling

- 누적 확률 값이 p보다 작은 상위 token sampling에 사용
- 개수 상관 없이 확률에 의존

Scaling randomness

- Logits 값을 상수로 나누고 softmax의 입력으로 사용
- decoding algorithm은 아님, distribution 조정 목적으로 사용

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 결과 보정

Re-balancing : PPLM guide 제공

- 추가적인 model을 사용해 LM의 distribution 조정 (sentiment, perplexity)
- Attribute 모델로부터 gradient 전달 받아 latent 업데이트 → Model dist. update

Unlikelihood Training

- 이미 생성된 토큰이 생성될 확률을 낮추는 penalty를 기준 loss function 사용

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

$$L_{UL} = - \sum_{y_{neg} \in C} \log(1 - P(y_{neg} | \{y^t\} < t))$$

$$L_{ULE} = L_{MLE} + \alpha L_{UL}$$

- 반복된 단어, 구의 생성을 줄일 수 있음
- 생성되는 text의 다양성을 증가시킴

Scheduled Sampling : teacher forcing 편향하지 사용

Sequence re-writing : 학습 data로 구축한 prototype set을 text에 넣어 사용

Reinforcement Learning

N-gram overlap metrics - BLEU, ROUGE

Word Mover's Distance

- 두 text와 reference text의 단어 or 문장의 semantic similarity 계산

1. 모든 단어의 word2vec embedding 계산

2. Doc 1의 모든 단어들의 Doc 2 까지 최단거리합 계산

BERTSCORE

사전학습된 BERT 사용

- reference x 와 candidate \hat{x} 의 contextual embedding 계산

- 모든 pair의 cosine 유사도 계산

- greedy matching 후 weighted average 구함

- Inverse document frequency score 사용

자주 사용되는 the, is 등은 가중치 낮기 부여

BLEURT

- 문장 유사도 예측하는 BERT 기반 regression model