# Neural Language Model: 확률의 모델링

Language model이란 실제 우리가 사용하는 언어의 분포를 확률 모델로 모델링 한 것이다

n-gram에선 하지 못했던 generation을 할 수 있다(보지 못한 단어를 생성할 수 있다.) n-gram은 단어를 discrete symbols(이산적)취급하고, neural network는 그렇지 않기 때문이다.

- Neural Language Model: 주어진 코퍼스를 훈련하고 MLE(Maximum Likelihood Estimation)를 통해서 파라미터를 업데이트 한다. 또한 NLL을 minimize하기위해 gradient decent를 수행한다.

pytorch에서는 softmax + Cross Entropy Loss, log soft max + Negative Log Likelihood 이렇게 두개를 많이 사용한다.

Neural Network Language Model은 word embedding을 통해, unseen sequence에 대처가 가능하고, generation task에서 강점을 보인다. 하지만 연산량이 많고 해석 난이도가 증가해 설명령이 떨어진다.

# Perplexity: 언어모델의 성능을 나타내는 PPL

PPL은 쉽게 말해서 헷갈리는 단어의 개수를 뜻하며, 문장의 확률의 역수의 기하평균이다(). 확률은 낮을수록, PPL은 낮을수록 좋다. 값이 높을수록 flat하고, 낮을수록 sharp하다.

이것은 정보이론에서 불확실성을 나타내는 Entropy(정보량)와 유사하다. 자주 일어나는 사건들은 엔트로피가 낮고, 드물게 일어나는 사건은 정보량이 높다.

언어모델의 관점에서는 흔히 나올수 없는 문장은 확률이 낮은문장이고, 이것은 더 높은 엔트로피(정보량)을 가진다

수식적인 관점에서 보면,

Cross Entropy를 minimize 하는것은 Log(PPL)을 minimize하는것과 같고, (exp(CE) = PPL)

Cross Entropy를 minimize 하는것은 NLL을 minimize 하는것과 같다.

\* Entropy and Perplexity

# Auto regressive (One-to-many)

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- 현재상태가 앞/뒤 상태를 통해 정해지는 경우, Bidirectional RNN을 권장하며, None Auto regressive하다고 표현한다(non-generation). Many to one, many to many의 경우이다.

- 현재 상태가 과거에 의존해서 정해지는 경우, Bidirectional RNN을 사용할 수 없고, Autoregressive하다고 한다(Generative). One to many에 해당하는 경우다. 과거 자신의 상태를 참조하여 현재 자신의 상태를 업데이트 한다.

- Auto-regressive task에서는 보통 이전 time-step의 모델의 출력을 다음 time-step의 입력으로 넣어준다. -> 이전 time-step의 출력에 따라서 현재 모델의 sate가 바뀌게 된다. 하지만 이것은 MLE수식과 다르게 동작한다

Sequence-to-Sequence는 many to one(encoder) + one to many(decoder)형식으로 볼 수 있는데, 결국에는 Auto-regressive한 성질을 띄기 때문에 One to many로 취급한다.

# Teacher Forcing: 실제 정답을 넣어준다.

MLE수식상 정답 을 RNN의 입력으로 넣어줘야 한다. 또한 이것은 훈련의 안정성과 속도가 증가한다

Inference Mode: 이전 time step의 예측값을 입력으로 받는다. 따라서 이전 time-step 정보들이 모두 들어있다

도표, 평면도, 패턴이(가) 표시된 사진

자동 생성된 설명

Training Mode: 이전 time-step의 예측값을 무시하고 Loss에만 사용하며, 실제 정답을 입력으로 넣어준다. (Teacher Forcing) -> 학습, 추론은 잘 되지만 문제가 발생한다

도표, 텍스트, 평면도, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이전 time-step 훈련 정보들(hidden state)은 에 들어있다. 근데 와 관계없는 실제 정답인 이 입력으로 들어간다. 그렇다면 RNN은 이전 time step의 hidden state들을 무시할 수도 있다. 왜나햐면 무시하고 값을 넣어주기 때문이다.

학습과 추론을 위한 방법이 다르게 되어, 여러가지 문제가 발생된다. 흑습과 추론에 대한 코드를 따로 짜야하며, 학습과 추론방법의 괴리(discrepancy)가 발생하여, 성능이 저하될 수있다.

- Language model

실제 우리가 사용하는 언어의 분포를 확률 모델로 모델링 한 것이다.

Chain rule에 의해 문장의 확률을 모델링 하는 것은 단어들이 주어졌을 떄, 다음 단어의 확률을 모델링 하는것과 같다.

- Perplexity

매 time-step마다 모델이 동등하게 헷갈리고 있는 평균 단어 수. 헷갈리는 단어가 적을수록 좋은것. 즉, PPL은 낮을수록, 확률은 높을수록 좋은것이다.

문장의 확률의 역수에 단어 수만큼 기하 평균을 취한 것이다.

Cross Entropy에 exponential한것과 같다

- n-gram

단어를 이산적으로 인식하고, 학습 코퍼스에 word sequence가 존재하야만 확률값 추정이 가능하며 쉽고 직관적이다. 근본적인 이산적인 문제는 해결하지 못하고 generation에 약하다. 단순 문장비교를 하고자 하면 n-gram으로 충분하다. 하지만 우리가 하고자 하는 것은 generation task다.

- NNLM

단어를 연속적 벡터(continue vector)로 변환한다. Unssen word sequence에 대처 가능하며, generation에 강하다. 비싸고 느린 연산 추론 과정이 있다. 어느정도 괴리, 한계가 발생하는 것 같지만 너무나도 뛰어나다.