Language Modeling and Text Generation

MLP LAB 임경태

Contents

01 Recap of LM

02 Statistical Language Modeling

03 Computation of LM

04 Evaluation methods for LM

05 Neural Net-based LM

06 GPT, ChatGPT and BART

01. Recap of LM

Language Modeling

- **Language Modeling** is the task of predicting what word comes next

  - **For example, given a sequence of words:**

the students opened their _____

  - More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

    where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

  - A system that does this is called a **Language Model**

참조: Stanford 224n

Language Modeling

- **Language Modeling using RNN**

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

hidden states

$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$

참조: Stanford 224n

Language Modeling

- GPT??

그래 RNN배울 때 다음 단어 예측 하는 걸 Language Modeling 이라고 했지!

해당 모델은 언어의 문법, 의미를 이해했으니까 그럴듯한 글을 생성했겠지?

어라? 언어의 문법과 의미를 이해 했어? AI가??

어라? 언어의 문법과 의미를 이해 했어? AI가??

문법과 의미를 배운 모델 (pretrained)을 NLI, 기계번역과 같은 특정 task에 튜닝 (fine-tuning)할 수만 있으면 더 좋겠다..

근데 이게 가능 하다구?

Language Modeling (LM)

- **Now we are going to deep dive into** two different types of LMs

02. Statistical Language Model

Statistical LM (SLM)

- **주어진 문서(코퍼스) 내 단어열(혹은 문장)의 등장 확률**을 기반으로 각 단어의 조합을 예측하는 전통적인 언어 모델

- 실제로 많이 사용하는 단어열(문장)의 확률 분포를 정확하게 근사하는 것이 모델의 목표

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 조건부 확률

조건부 확률 P(B|A): 사건 A가 일어났을 때 사건 B가 일어날 확률

↓

언어 모델

단어 A가 등장했을 때 바로 다음에 단어 B가 등장할 확률

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 조건부 확률과 결합확률

$$P(B|A) = \frac{P(A, B)}{P(A)}$$

$$P(A)P(B|A) = P(A, B)$$

$$P(A, B) = P(A)P(B|A)$$

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 연쇄 법칙

연쇄 법칙(Chain rule)

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

$\downarrow$ 언어 모델

$$P(\text{나는, 사과를, 먹는다}) = P(\text{나는})P(\text{사과를} \mid \text{나는})P(\text{먹는다} \mid \text{나는, 사과를})$$

Statistical LM (SLM)

- We can also think of a Language Model as a system that assigns a probability to a piece of text
- For example, if we have some text, then the probability of this text (according to the Language Model) is:

연쇄 법칙(Chain rule)

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

n개 단어(w)의 결합 확률

$$P(w_1, w_2, w_3, \cdots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_n|w_1, \cdots, w_{n-1})$$

$$= \prod_{n=1}^{n} \underbrace{P(w_n|w_1, \cdots, w_{n-1})}_{\text{This is what our LM provides}}$$

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 카운트 기반 계산: 코퍼스(corpus) 내에서 각 단어들의 조합이 나오는 횟수를 카운트(count) 한 후 이에 기반하여 확률을 계산

$$P(\text{먹는다} \mid \text{나는, 사과를}) = \frac{count(\text{나는, 사과를, 먹는다})}{count(\text{나는, 사과를})}$$

→ 모든 단어 조합의 경우의 수를 다 세어야 함

→ 계산 복잡도가 높아질 뿐만 아니라 무한한 크기의 코퍼스 필요

→ 어려움, 비현실적

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 마르코프 가정(Markov assumption)

✓ 기존 연쇄 법칙의 복잡성을 해결하고 간소화하기 위함

✓ **미래 사건에 대한 조건부가 과거에 대해서는 독립이며 현재의 사건에만 영향을 받는 다는 가정**을 전제로 연쇄법칙 설명

✓ 단어w_n이 나타날 확률은 그 앞의 단어 w_{n-1}이 나타날 확률하고만 연관이 있다고 봄

$$P(w_1, w_2, \cdots, w_n) \approx P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1})$$

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 마르코프 가정(Markov assumption)

$$P(w_1, w_2, \cdots, w_n) \approx P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1})$$

$$\downarrow$$

$$P(\text{나는, 사과를, 먹는다}) \approx P(\text{나는})P(\text{사과를}\,|\,\text{나는})P(\text{먹는다}\,|\,\text{사과를})$$

Statistical LM (SLM)

- 조건부 확률(Conditional probabilities)과 언어 모델
  - 마르코프 가정(Markov assumption)의 한계

✓ 어떤 단어의 등장 확률이 바로 앞의 단어하고만 연관이 있다는 마르코프 가정

✓ 언어 현상에 적용하기에는 **지나친 단순화**

✓ 언어의 **장기 의존성** 간과됨

✓ (예: The <u>computer</u> which I had just put into the machine room on the fifth floor <u>crashed</u>.)

✓ 예측 정확도 낮아질 수 있음

Statistical LM (SLM)

▪ N-gram 언어 모델(N-gram Language Model)

- N-gram

✓ 문장 내 단어는 **주변의 여러 단어**와 연관된다고 가정

✓ **N**: 주변 몇 개의 단어를 볼 것인지 정하는 임의의 숫자

✓ **N-gram**: N개의 단어열

✓ **1-gram(unigram):** The, boy, is, looking, at, a, pretty, girl

✓ **2-gram(bigram):** The boy, boy is, is looking, looking at, at a, a pretty, pretty girl

✓ **3-gram(trigram):** The boy is, boy is looking, is looking at, looking at a, at a pretty, a pretty girl

✓ **4-gram:** The boy is looking, boy is looking at, is looking at a, looking at a pretty, at a pretty girl

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - N-1차 마르코프 가정을 할 경우

1-gram(유니그램, unigram): $P(w_1, w_2, \cdots, w_n) \approx \prod_{i=1}^{n} P(w_i)$

2-gram(바이그램, bigram): $P(w_1, w_2, \cdots, w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-1})$

...

N-gram : $P(w_1, w_2, \cdots, w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-N}, \cdots, w_{i-1})$

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - N-1차 마르코프 가정을 할 경우

$$P(\text{먹는다} \mid \text{나는, 사과를}) \approx P(\text{먹는다} \mid \text{사과를})$$

$$\frac{count(\text{나는, 사과를, 먹는다})}{count(\text{나는, 사과를})} \approx \frac{count(\text{사과를, 먹는다})}{count(\text{사과를})}$$

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 3개 문장과 2-gram 모델로 단어열의 등장 확률 계산

코퍼스 예시

<s>I eat an apple</s>

<s>an apple I eat</s>

<s>I like cheese cake</s>

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 3개 문장과 2-gram 모델로 단어열의 등장 확률 계산

$$P(I|<s>) = \frac{count(<s>, I)}{count(<s>)} = \frac{2}{3} = 0.6667$$

$$P(an|<s>) = \frac{count(<s>, an)}{count(<s>)} = \frac{1}{3} = 0.3333$$

$$P(eat|I) = \frac{count(I, eat)}{count(I)} = \frac{2}{3} = 0.6667$$

$$P(</s>|apple) = \frac{count(apple, </s>)}{count(apple)} = \frac{1}{2} = 0.5$$

$$P(like|I) = \frac{count(I, like)}{count(I)} = \frac{1}{3} = 0.3333$$

$$P(cake|cheese) = \frac{count(cheese, cake)}{count(cheese)} = \frac{1}{2} = 0.5$$

코퍼스 예시

<s>I eat an apple</s>

<s>an apple I eat</s>

<s>I like cheese cake</s>

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 셰익스피어 작품 기반 N-gram 문장 생성

Unigram

✓ To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

✓ Hill he late speaks; or! A more to leg less first you enter

→ 1-gram 모델은 단어의 열(sequence) 고려X, 모든 단어가 독립적이라고 봄

→ 서로 무관한 단어들이 생성됨

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 셰익스피어 작품 기반 N-gram 문장 생성

Bigram

✓ Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

✓ What means, sir. I confess she? Then all sorts, he is trim, captain.

→2-gram 모델은 바로 앞 1개의 단어를 고려

→1-gram 보다 자연스럽지만 전체적으로는 여전히 매우 부자연스러움

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 셰익스피어 작품 기반 3-gram 문장 생성 계산방식

today the _____

condition on this

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

Sampling randomly from Top5

* **Try for yourself:** https://nlpforhackers.io/language-models/

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 셰익스피어 작품 기반 3-gram 문장 생성 계산방식

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 셰익스피어 작품 기반 3-gram 문장 생성 계산방식

today the price of ____

condition on this

get probability distribution

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
...	

sample

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 예제: 셰익스피어 작품 기반 3-gram 문장 생성

Trigram

✓ King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;

✓ It cannot be but so.

→ 1 - gram 및 2-gram 모델의 문장보다 자연스러움

→ 코퍼스에 존재하는 텍스트에 가깝게 생성

→ 새로운 문장이라고 보기 어려움(low diversity)

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 4-gram 계산 예제

Suppose we are learning a 4-gram Language Model.

~~*as the proctor started the clock, the*~~ *students opened their* _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 4-gram 결과 예제

Generated results of 4-gram

today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

Surprisingly grammatical!

…but **incoherent.** We need to consider more than three words at a time if we want to model language well.

But increasing *n* worsens sparsity problem, and increases model size…

Statistical LM (SLM)

- N-gram 언어 모델(N-gram Language Model)
  - 4-gram 계산 예제 → Sparsity Problems!!!

Sparsity Problem 1

Problem: What if *"students opened their w"* never occurred in data? Then *w* has probability 0!

(Partial) Solution: Add small δ to the count for every $w \in V$. This is called *smoothing*.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any w*!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Note: **Increasing *n* makes sparsity problems** *worse.* Typically, we can't have *n* bigger than 5.

[추가]

✓ 교착어인 한국어에서 희소성 문제가 크게 발생

✓ (예: 한국어에서 형태소 분석 등 전처리 진행하지 않으면 '사과가', '사과를', '사과도', '사과에'는 모두 다 른 단어로 처리되어 '사과'가 포함된 단어열의 확률값이 0에 가까워짐)

03. Computation of N-gram LM

Computation of N-gram Model

- N-gram 로그 확률(Log probabilities) 언어모델

$$P(w_1, w_2, w_3, \cdots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_n|w_1, \cdots, w_{n-1})$$

$$= \prod_{n=1}^{n} P(w_n|w_1, \cdots, w_{n-1})$$

Markov!

$$P(w_1, w_2, \cdots, w_n) \approx P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1}) \approx \prod_{i=1}^{n} P(w_i|w_{i-N}, \cdots, w_{i-1})$$

$$P(\text{나는, 사과를, 먹는다}) \approx P(\text{나는})P(\text{사과를}|\text{나는})P(\text{먹는다}|\text{사과를})$$

Log!

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

$$\log P(w_1, w_2, \cdots, w_n) \approx \sum_{i=1}^{n} \log P(w_i|w_{i-N}, \cdots, w_{i-1})$$

- ✓ **곱셈 연산**은 덧셈 연산보다 **계산 리소스가 크고** 결과값이 **0에 가까운 작은 값으로 계산될 가능성**이 있음
- ✓ 전체 확률 계산을 **로그** 공간에서 계산할 시 **곱셈을 덧셈으로 환산**할 수 있음
- ✓ **계산이 간단**해지고 원래 확률 계산보다 **언더플로를 피할 수 있음**

✓ **로그** 공간에서 n개의 단어로 이루어진 단어열의 확률을 계산하는 **N-gram 모델 식**

Computation of N-gram Model

- Implement your own 3-gram language model!

  - ✓ 전체 3-gram의 전체 수 및 빈도수를 단어사전으로 계산함
  - ✓ 확률값 계산 P(c|a, b) = (a,b,c)에 대한 3gram 빈도 / 전체 3gram 수
  - ✓ 입력 값 ”There are”에 대한 문장 생성

그래 흠… 그런데 Uni, Bi, Tri, Four-gram 언어모델 중

어떤 모델이 가장 **좋은** 모델이라고 할 수 있을까?

평가의 척도가 필요하지 않을까?

Computation of N-gram Model

- 흠… 그런데 Uni, Bi, Tri, Four-gram 언어모델 중 어떤 모델이 가장 **좋은** 모델이라고 할 수 있을까?

04. Evaluation methods for LM

Language Model Evaluation

- 언어 모델의 평가

✓두 언어모델 A,B가 있을 때 어떤 모델이 더 좋은 모델일까?
→ 직접 생성을 시켜보고 사람이 평가하면 될 듯? 그러면 평가에 걸리는 시간은? 인력은? 자동안됨?

✓퍼플렉서티(Perplexity, PPL): 언어 모델의 성능을 자체적으로 평가하는 내부 평가(intrinsic evaluation) 척도

→ 간이 실험 등 짧은 시간 안에 간단히 모델을 평가할 때, 혹은 모델 간 비교시 평가척도로 사용

Language Model Evaluation

- 퍼플렉서티(Perplexity, PPL)

✓ 주어진 확률 모델이 샘플을 얼마나 잘 예측하는가에 대한 측정 지표

✓ **'perplexity'**는 **'헷갈리는 정도'**를 뜻함

✓ 모델이 테스트 데이터셋에 대하여 확률 분포를 얼마나 확실하게 (헷갈리지 않게) 예측할 수 있는지를 나타내는 지표

✓ **PPL 점수가 낮을수록 (헷갈리는 경우가 적을수록) 좋음**

Language Model Evaluation

- 퍼플렉서티(Perplexity, PPL): 어떤 모델이 덜 헷갈려 할까?
  - 모델이 샘플의 확률을 예측하는 시점에서 얼마나 많은 후보군을 두고 고민하는가를 나타냄

정답이 gold일 경우 다음 두 모델을 비교해보자

Model A

today the price of ____

condition on this

get probability distribution

gold	0.5
the	0.3
oil	0.2
its	0.1

Model B

today the price of ____

condition on this

get probability distribution

gold	0.3
the	0.2
oil	0.1
Its	0.1
gas	0.1
nodap	0.1
hell	0.05
end	0.05

Language Model Evaluation

- 퍼플렉서티(Perplexity, PPL)

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

$$= \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$

$$= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_1 | w_1, w_2, \ldots, w_{i-1})}}$$

독립가정

전체 10개의 단어가 각각 1/10으로 출현했다면

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

$$= \left(\frac{1}{10^N}\right)^{-\frac{1}{N}}$$

$$= \left(\frac{1}{10}\right)^{-1}$$

$$= 10$$

- ✓ N-gram 모델에서는 'N-1차 마르코프 가정'에 따른 연쇄법칙이 적용됨
- ✓ 이를 반영한 2-gram 모델의 계산식

$$PPL(W) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i | w_{i-1})}}$$

Language Model Evaluation

▪ 퍼플렉서티(Perplexity, PPL)

- The standard evaluation metric for Language Models is perplexity.

$$\text{perplexity} = \underbrace{\prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)}_{\text{Inverse probability of corpus, according to Language Model}}^{1/T}$$

Normalized by number of words

- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

Lower perplexity is better!

Language Model Evaluation

- N-gram 언어 모델의 성능비교

05. Neural Net-based LM

A fixed-window neural language Model

- 5-gram with FFNN LM model

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$

A fixed-window neural language Model

- The problem of the 5-gram with FFNN LM model

Approximately: Y. Bengio, et al. (2000/2003): A Neural Probabilistic Language Model

Improvements over *n*-gram LM:
- No sparsity problem
- Don't need to store all observed *n*-grams

Remaining **problems**:
- Fixed window is too small
- Enlarging window enlarges W
- Window can never be large enough!
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W. No symmetry in how the inputs are processed.

We need a neural architecture that can process *any length input*

A fixed-window neural language Model

- The problem of the 5-gram with FFNN LM model

We need a neural architecture

that can process *any length input*

??? RNN??

A Non-fixed-window neural language Model

- RNN-based LM model

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

hidden states

$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$

A Non-fixed-window neural language Model

- RNN-based LM model

A Non-fixed-window neural language Model

- RNN-based LM model

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta)$$

- Get a big corpus of text which is a sequence of words $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\boldsymbol{y}}^{(t)}$ for *every step t*.
  - i.e., predict probability dist of *every word*, given words so far
- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$ (one-hot for $\boldsymbol{x}^{(t+1)}$):

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

- Average this to get overall loss for entire training set:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Language Model Evaluation

- N-gram VS Neural Net LM

n-gram model

Increasingly complex RNNs

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Perplexity improves (lower is better)

Source: https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/

06. GPT, ChatGPT, BART
이건 다음에…

감사합니다.