Pretrained Language Models

MLP LAB 임경태

Contents

01. Motivating word meaning and context

Word Representation

- Symbolic한 문자를 연산 가능한 숫자로 표현하는 방법
- 단순히 숫자로만 표현하면 될까? 사람은 어떨까? “Tree”라는 단어를 떠올려보자

• signifier (symbol) ⟺ signified (idea or thing)

= denotational semantics

tree ⟺ { 🌳 , 🌲 , 🌴 ,…}

- 단어를 표현할 때 단어의 **의미(semantics)**를 내포해야 하겠네?

Ref: Starford cs224n

해결책

- Dimensionality: Sparse한 표현을 Dense한 표현으로 변경해서 차원을 줄인다
- 의미의 내포: 유사한 단어는 유사한 벡터 값으로 표현하게 한다

Example: in web search, if a user searches for "Seattle motel", we would like to match documents containing "Seattle hotel"

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

These two vectors are orthogonal

There is no natural notion of **similarity** for one-hot vectors!

Solution:

motel = [0.34 -1.22 3.31]
hotel = [0.30 -1.01 3.31]

now "motel" and "hotel" is similar

Ref: Starford cs224n

의미의 내포: 유사한 단어는 유사한 벡터 값으로 표현하게 한다

이거 어떻게함?

주변 Context를 고려한 word embedding

Representing words by their context

- Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!
- When a word *w* appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- We use the many contexts of *w* to build up a representation of *w*

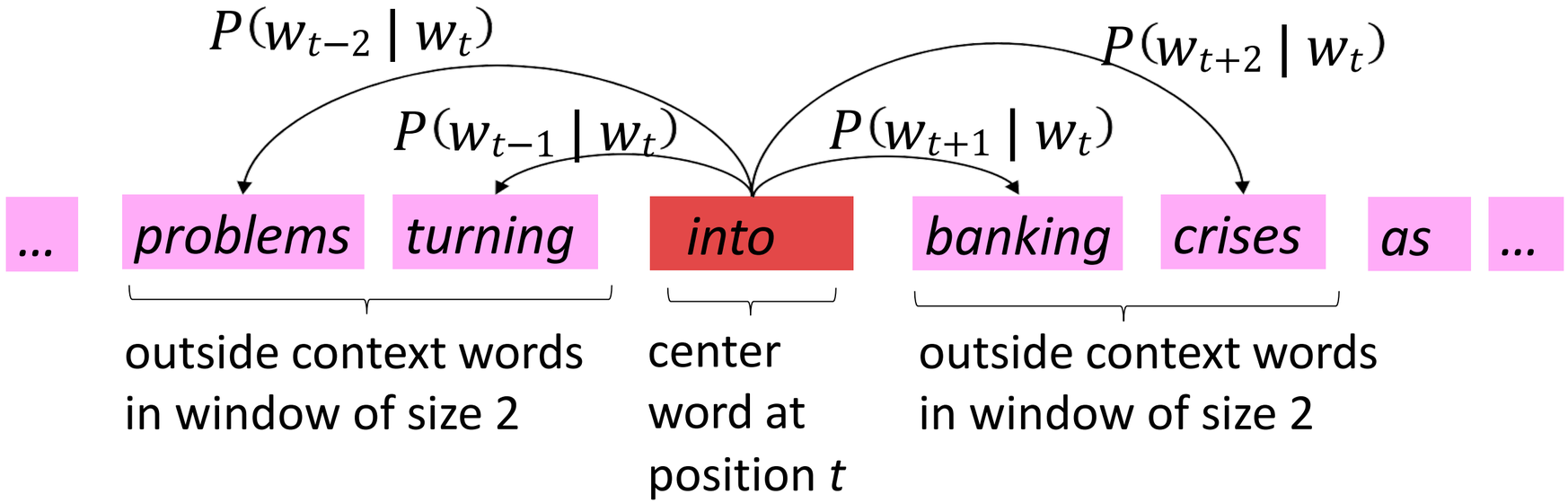
…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

Word2Vec Overview (Skipgram)

- Word2vec (Mikolov et al. 2013) is a framework for learning word vectors

Example windows and process for computing $P(w_{t+j} \mid w_t)$



Ref: Starford cs224n

흠.. 그런데 다음 문장의 두 개의 “record” 어떻게 구분하지??

*I **record** the **record***

The two instances of ***record*** mean different things.

흠.. 그런데 다음 문장의 두 개의 “record” 어떻게 구분하지??

*I **record** the **record***

The two instances of ***record*** mean different things.

아하! RNN을 통과시키면 주변의 문맥을 고려하면서 두 Record의 표현이 달라지겠구먼!

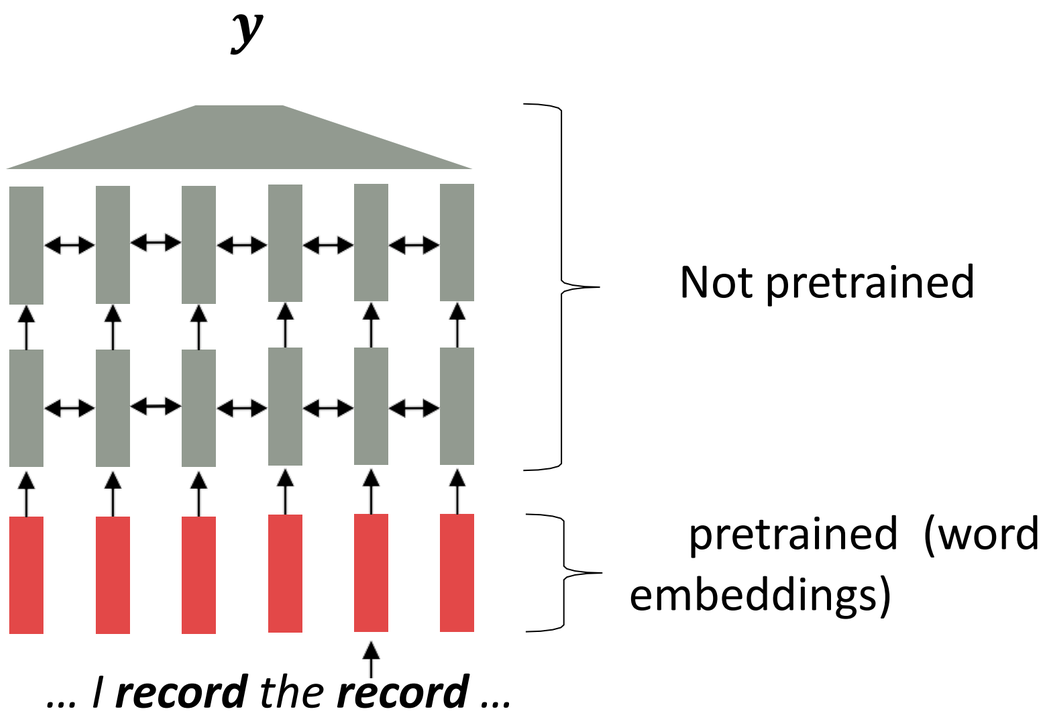
Where we were: pretrained word embeddings

Circa 2017:

- Start with pretrained word embeddings (no context!)
- Learn how to incorporate context in an RNN or Transformer while training on the task.

Some issues to think about:

- The training data we have for our **downstream task** (like Sentiment ANA) must be sufficient to teach all contextual aspects of language.
- Most of the parameters in our network are randomly initialized!



[Recall, *record* gets the same word embedding, no matter what sentence it shows up in]

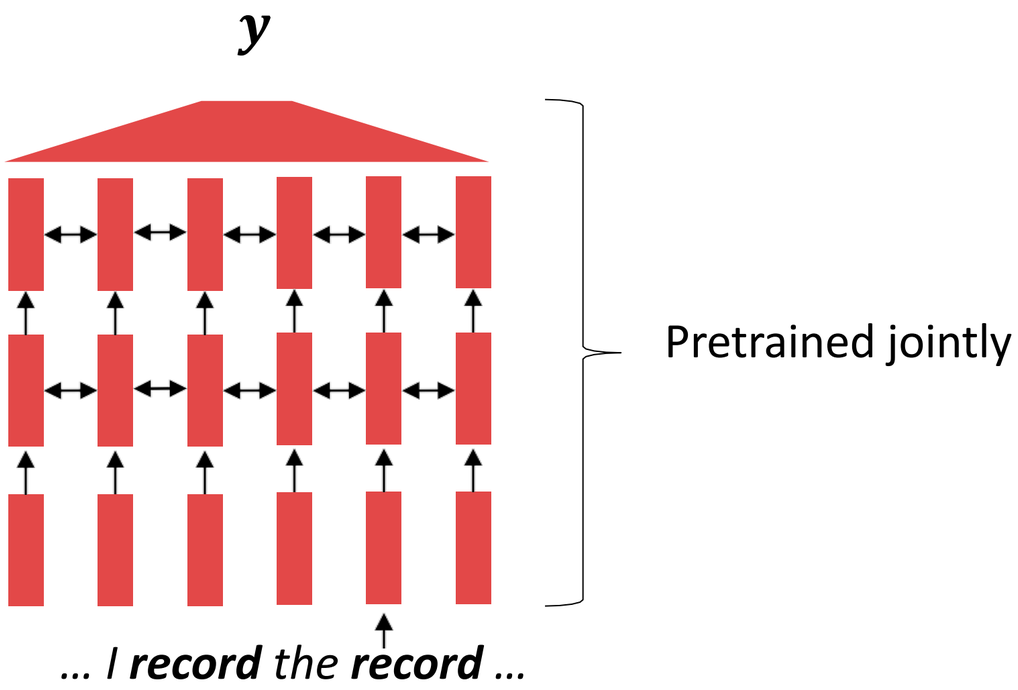
Ref: Starford cs224n

Where we're going: pretraining whole models

In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via **pretraining**.
- Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.

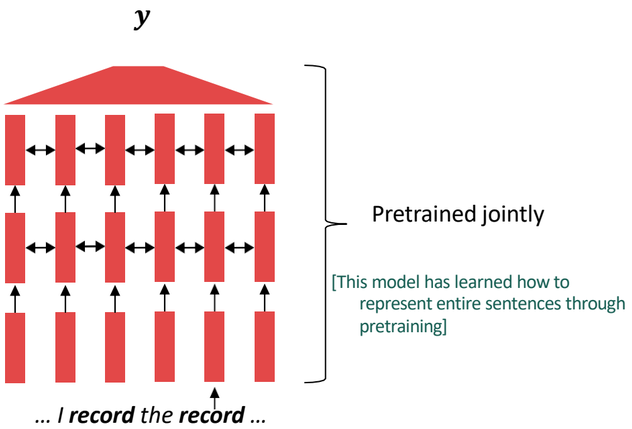
- This has been exceptionally effective at building strong:
  - **representations of language**
  - **parameter initializations** for strong NLP models.
  - **Probability distributions** over language that we can sample from



[This model has learned how to represent entire sentences through pretraining]

Ref: Starford cs224n

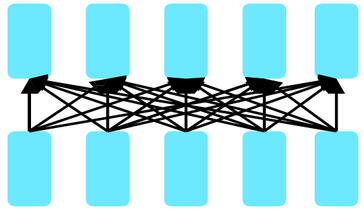
02. Pretraining Methods



그렇다면 word embedding + RNN을 어떤 구조로 사전학습 할 수 있을까??

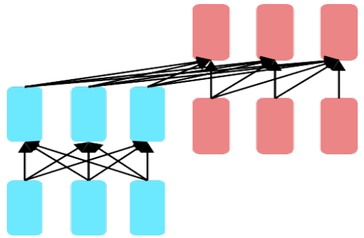
Pretraining for Three types of architectures

- 대표적인 Neural Net 기반 pretraining 방법 3가지



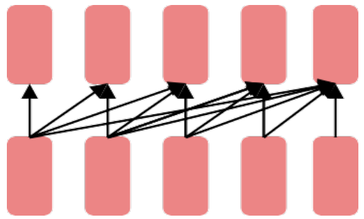
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

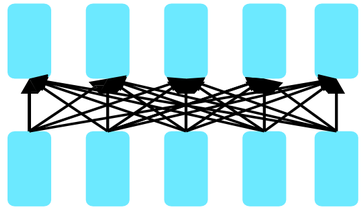


Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pretraining for Three types of architectures

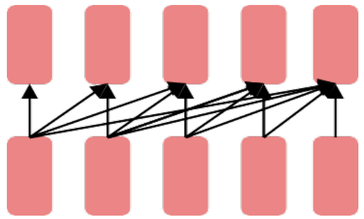
- The structure of Encoder VS Decoder



Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?

어라 Encoder랑 Decoder의 생김새가 다른데?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pretraining for Three types of architectures

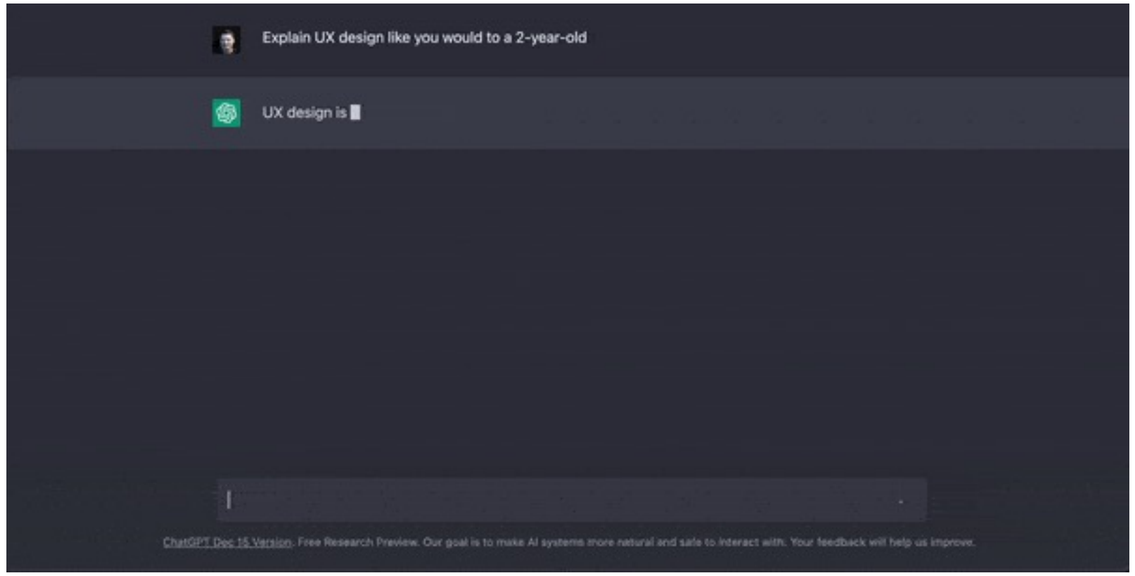
- The structure of Encoder VS Decoder

Encoder부분: Explain UX design like you would to a 2-year-old

입력 시 확정됨

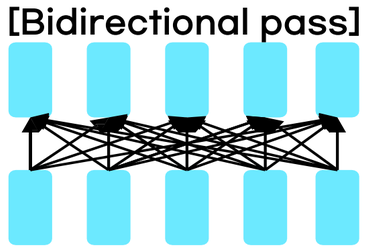
Decoder부분: UX design is …

출력 시 확정됨



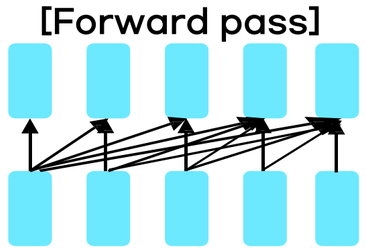
Pretraining for Three types of architectures

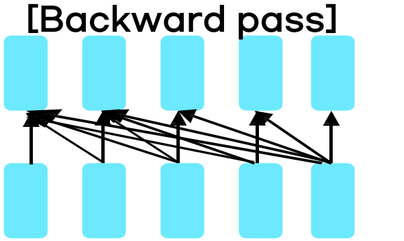
- The structure of Encoder VS Decoder



Encoder부분: Explain UX design like you would to a 2-year-old

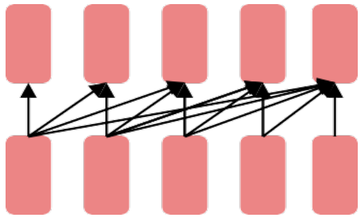
입력 시 확정됨 == 양방향 연산가능 == Auto Encoding





Decoder부분: UX design is …

출력 시 확정됨 == 단방향 연산만 가능 == Auto Regressive



Auto Regressive and Auto Encoding

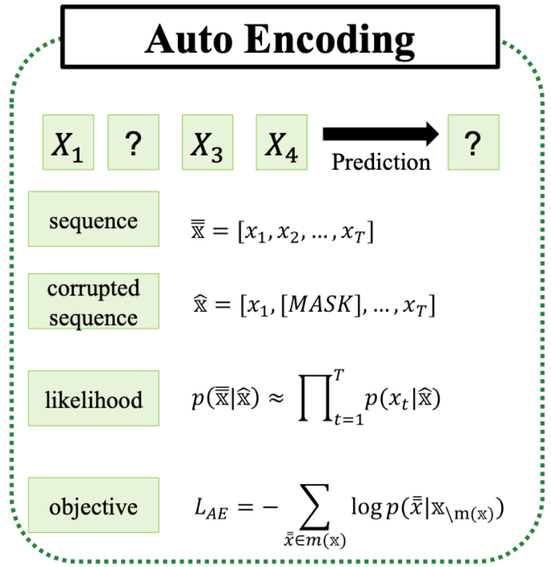
- The structure of Encoder VS Decoder

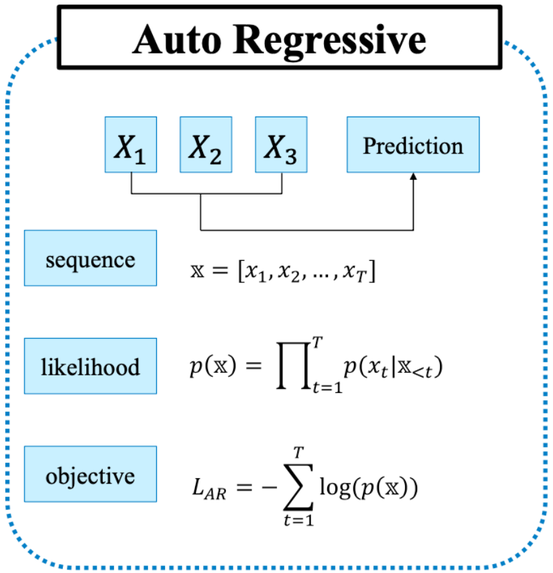
Encoder부분: Explain UX design like you would to a 2-year-old

입력 시 확정됨 == 양방향 연산가능 == Auto Encoding

Decoder부분: UX design is …

출력 시 확정됨 == 단방향 연산만 가능 == Auto Regressive



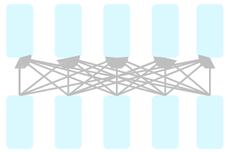


$m(\mathbb{x})$ and $\mathbb{x}_{\backslash m(\mathbb{x})}$ denote the masked words from $\mathbb{x}$ and the rest words respectively (MLM은 AE의 한 종류에 속함)

03. Pretraining on Decoder

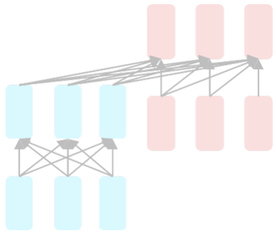
Pretraining on Decoder

- 자연어 생성에 초점을 맞춘 Decoder 모델 기반의 사전학습



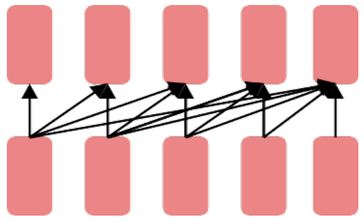
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- All the biggest pretrained models are Decoders.

Pretraining on Decoder

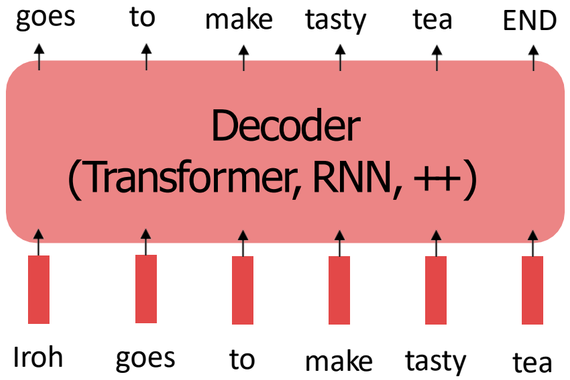
- Decoder의 사전학습 방법: Language Modeling 학습 후 Parameter 저장

Recall the **language modeling** task:

- Model $p_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a **large amount of text**.
- **Save the network parameters.**

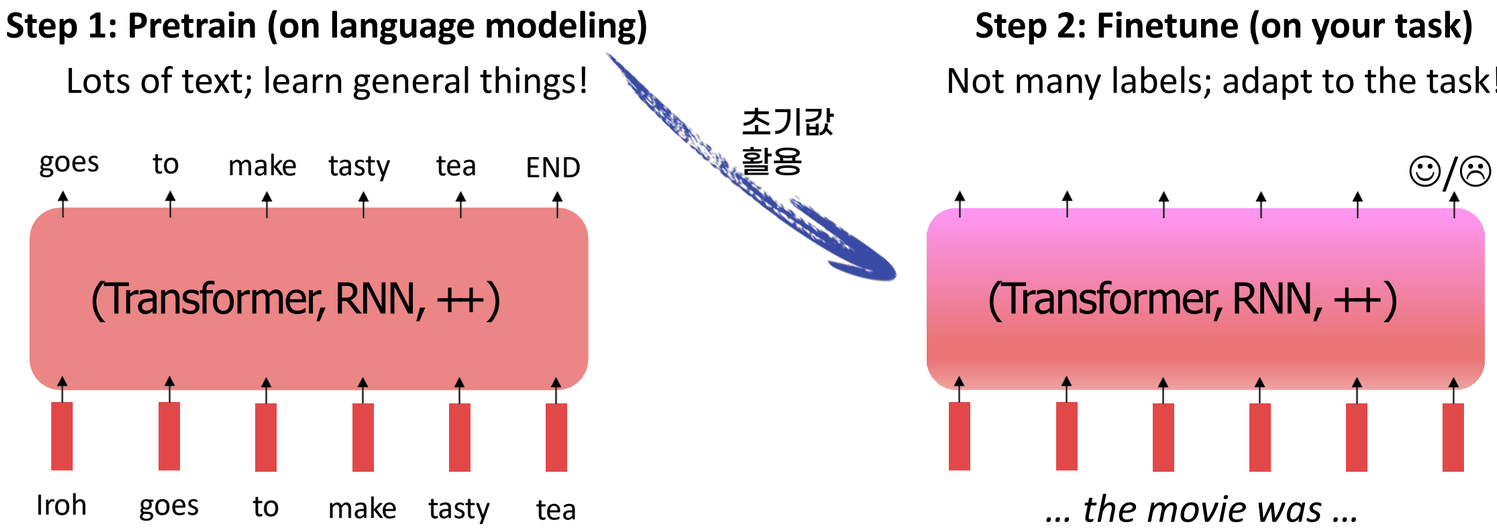


Pretraining on Decoder

- Pretraining모델 활용 방법: Finetuning

  - Language Modeling으로 사전 학습된 parameter를 특정 task를 학습할 때 초기값으로 활용

Pretraining can improve NLP applications by serving as **parameter initialization**.



Pretraining on Decoder

- Pretraining모델 활용 방법: Finetuning

  - Language Modeling으로 사전 학습된 parameter를 특정 task를 학습할 때 초기값으로 활용

  - 이거슨 마치 롤러스케이팅을 10년 탄 사람이 피겨 스케이팅을 7일만에 배우는 놀라운 마법

Step 1: Pretrain (on language modeling)

Lots of text; learn **general** things!

초기값
(경험)활용

Step 2: Finetune (on your task)

Not many labels; adapt to the task!

Pretraining on Decoder

- Pretraining모델 활용 방법: Finetuning for Classification (e.g., Sentiment ana)

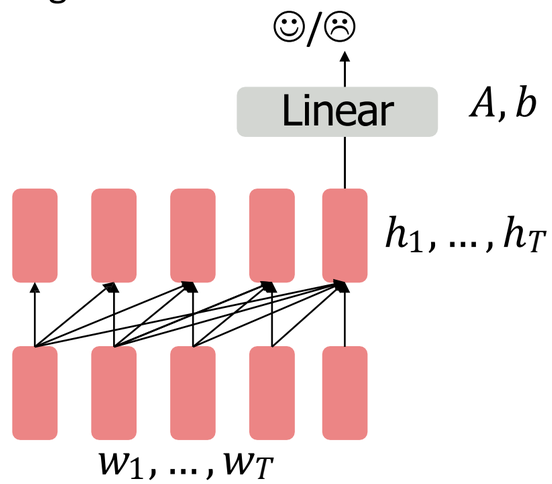
When using language model pretrained decoders, we can ignore that they were trained to model $p(w_t|w_{1:t-1})$.

We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$y \sim Ah_T + b$$

Where A and b are randomly initialized and specified by the downstream task.

Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

Pretraining on Decoder

- Pretraining모델 활용 방법: Finetuning for Generation (e.g., Chatbot)

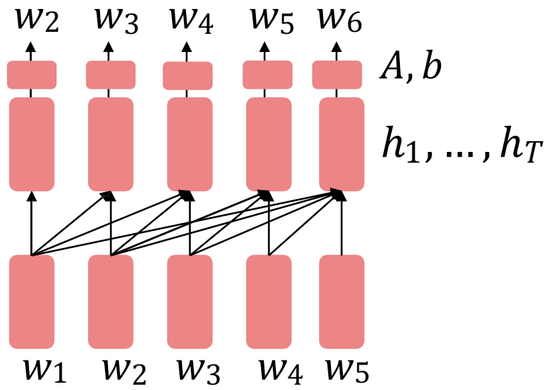
It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!

This is helpful in tasks **where the output is a sequence** with a vocabulary like that at pretraining time!

- Dialogue (context=dialogue history)
- Summarization (context=document)

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

Where A, b were pretrained in the language model!



[Note how the linear layer has been pretrained.]

Pretraining on Decoder 대표모델

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

2018's GPT was a big success in pretraining a decoder!

- Transformer decoder with 12 layers, 117M parameters.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
  - Contains long spans of contiguous text, for learning long-distance dependencies.
- The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"

Pretraining on Decoder 대표모델

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

How do we format inputs to our decoder for **finetuning tasks?**

Natural Language Inference: Label pairs of sentences as *entailing/contradictory/neutral*

Premise: *The man is in the doorway*
Hypothesis: *The person is near the door* } **entailment**

Radford et al., 2018 evaluate on natural language inference.
Here's roughly how the input was formatted, as a sequence of tokens for the decoder.

[START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

The linear classifier is applied to the representation of the [EXTRACT] token.

Pretraining on Decoder 대표모델

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

- GPT + finetuning 방법을 통해 NLI 문제에 대해서 기존 제안된 방법들을 압도!

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

Pretraining on Decoder 대표모델

Increasingly convincing generations (GPT2) [Radford et al., 2018]

- GPT2는 문서요약, 번역, 문장생성 등에서 사람과 거의 흡사한 문장을 생성.

We mentioned how pretrained decoders can be used **in their capacities as language models. GPT-2,** a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pretraining on Decoder 대표모델

GPT-3, In-context learning, and very large models

- GPT3는 Finetuning 과정 없이 몇가지 예제 명령을 토대로 답변을 생성

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

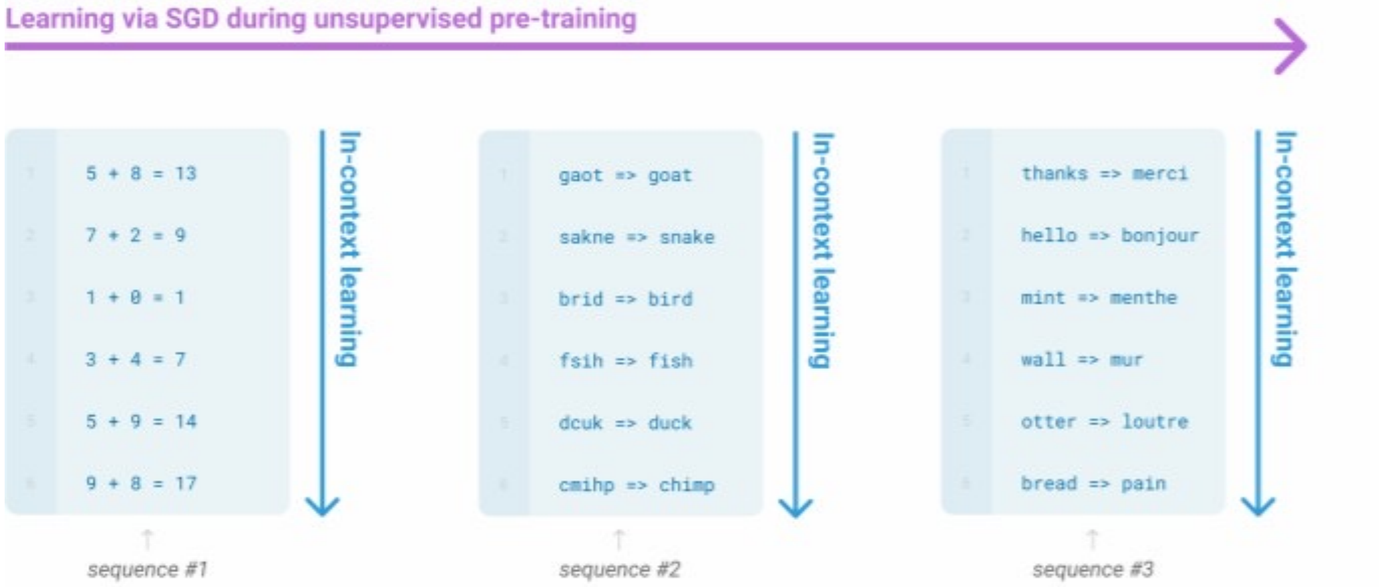
GPT-3 has 175 billion parameters.

Pretraining on Decoder 대표모델

GPT-3, In-context learning, and very large models

- GPT3는 Finetuning 과정 없이 몇가지 예제 명령을 토대로 답변을 생성

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.



Pretraining on Decoder 대표모델

- ChatGPT, GPT4, BARD, ChatLLaMA, Alpaca 등등 전국시대 개막

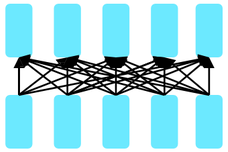
Stanford Alpaca

04. Pretraining on Encoder

Pretraining on Encoder

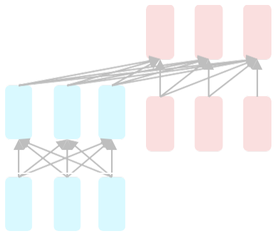
- 자연어의 이해의 관점에 초점을 맞춘 Encoder 모델 기반의 사전학습

The neural architecture influences the type of pretraining, and natural use cases.



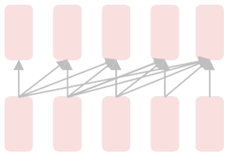
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

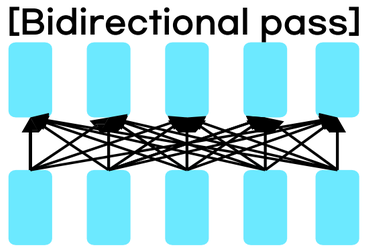


Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

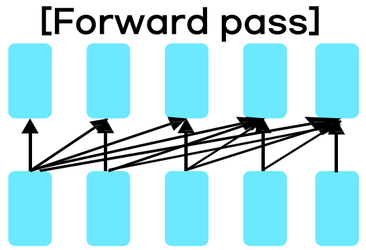
Pretraining for Three types of architectures

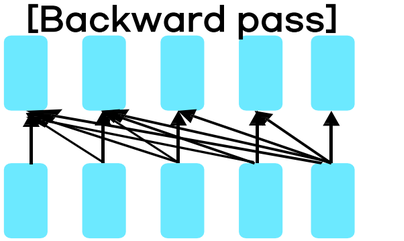
- The structure of Encoder VS Decoder



Encoder부분: Explain UX design like you would to a 2-year-old

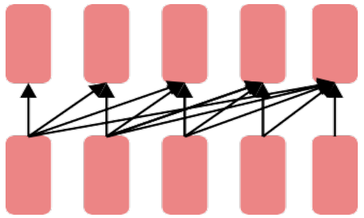
입력 시 확정됨 == 양방향 연산가능 == Auto Encoding





Decoder부분: UX design is …

출력 시 확정됨 == 단방향 연산만 가능 == Auto Regressive



Pretraining on Encoder

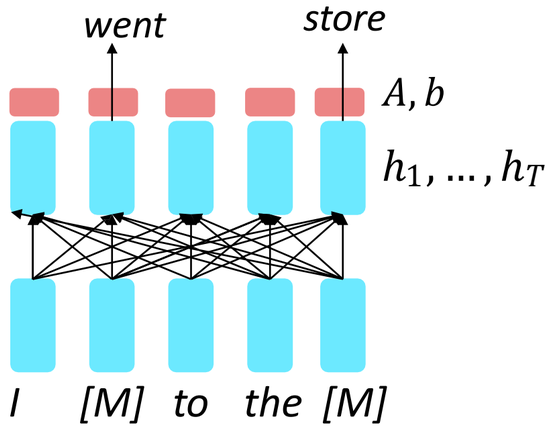
- 자연어의 이해의 관점에 초점을 맞춘 Encoder 모델 기반의 사전학습

So far, we've looked at language model pretraining. But **encoders get bidirectional context,** so we can't do language modeling!

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$$
$$y_i \sim Aw_i + b$$

Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$ Called **Masked LM**.



[Devlin et al., 2018]

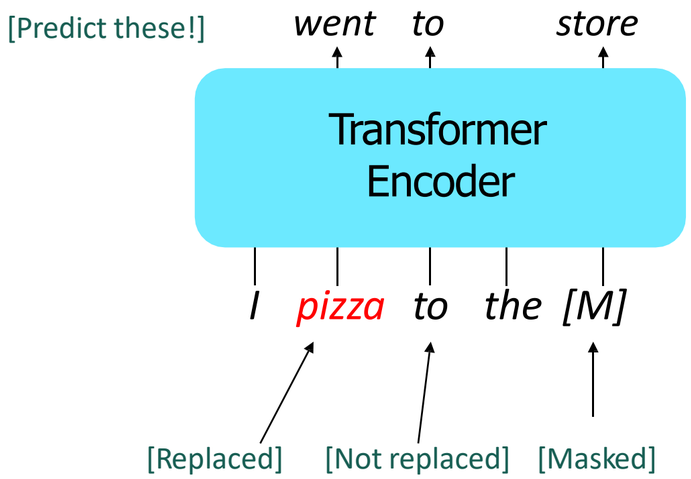
Pretraining on Encoder 대표모델

- BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the “Masked LM” objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

Some more details about Masked LM for BERT:

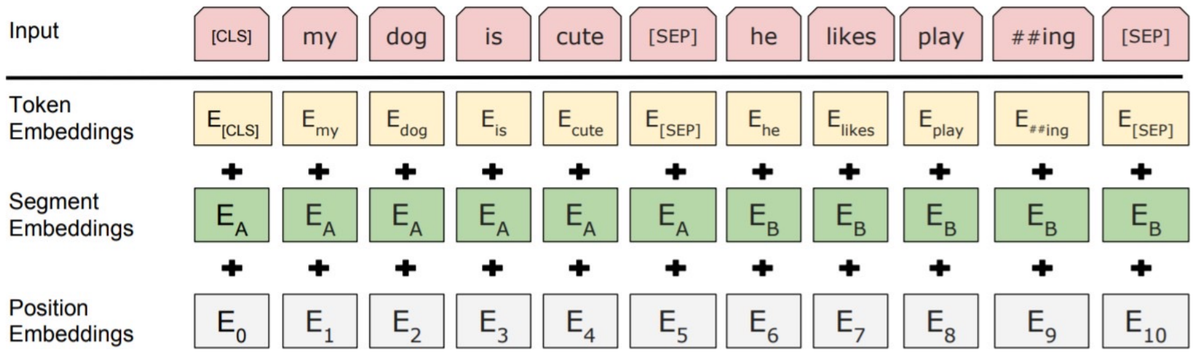
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn’t let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



[Devlin et al., 2018]

Pretraining on Encoder 대표모델

- BERT: Bidirectional Encoder Representations from Transformers
  - The pretraining input to BERT was two separate contiguous chunks of text:



  - BERT was trained to predict whether one chunk follows the other or is randomly sampled.
    - Later work has argued this “next sentence prediction” is not necessary.

Pretraining on Encoder 대표모델

- BERT: Bidirectional Encoder Representations from Transformers

BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

05. Applying BERT and GPT

이건 다음에…

감사합니다.