NATURAL LANGUAGE PROCESSING: Introduction

2022/11/10

Department of Computer Science,
Hanyang University

Taeuk Kim

Logistics

- **Time**
  - 18:30-21:00, Thu
- **Location**
  - Online: Via the LMS system
  - Offline: Room 502, IT/BT Building
- **Lecturer: Taeuk Kim (김태욱)**
  - Assistant professor in Dept. of Computer Science
  - Office: Room 719, IT/BT Building
  - Contact: kimtaeuk@hanyang.ac.kr

Coursework

- **Reference: Stanford CS224N**
  - https://web.stanford.edu/class/cs224n/
- **Breakdown**
  - Attendance: 20%
  - Quiz: 20%
    - 매 주 수업 끝난 다음 날 (금요일) 업로드 예정, 다음 수업 시간 전까지 풀어서 LMS로 제출
  - Final exam: 50%
    - 오프라인 진행 예정 (IT/BT관 502호)
  - Participation: 10%

What do I hope to teach?

1. **The foundations of the effective modern methods for deep learning applied to NLP**
   - Basics first, then key methods used in NLP: Recurrent networks, attention, transformers, etc.

2. **A big picture understanding of human languages and the difficulties in understanding and producing them**

3. **An understanding of and ability to build systems for some of the major problems in NLP:**
   - Word meaning, dependency parsing, machine translation, question answering

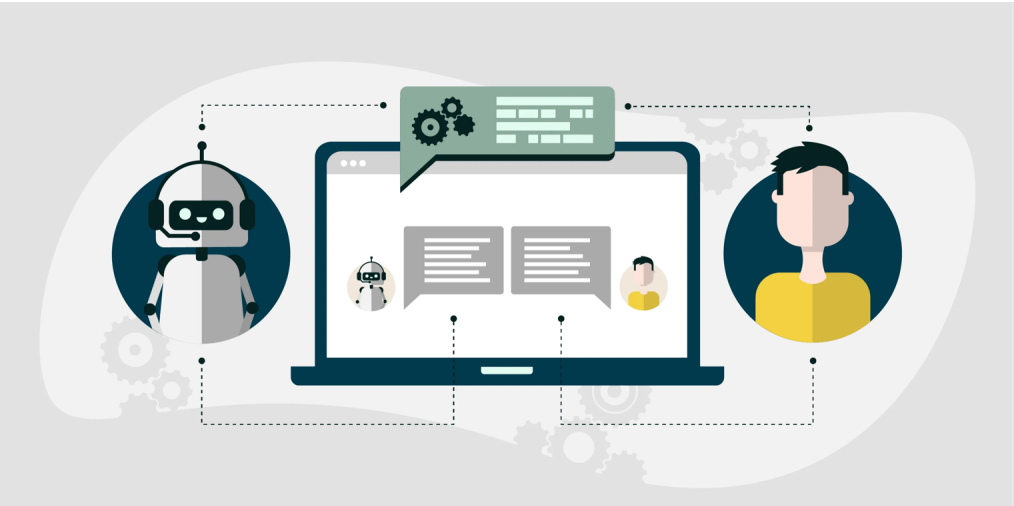
Introduction to NLP

Based on different applications

Natural Language Processing

- **Natural language processing (NLP)**
  - It's a subfield of linguistics, psychology, neuroscience, computer science, artificial intelligence, and many others.
  - Concerned with the interactions between computers and human language, in particular *how to program computers to process and analyze large amounts of natural language data.*

- **NLP is all about building an AI agent that can utilize text as its medium**
  - **Text as input:** text classification, sentiment analysis, semantic search, …
  - **Text as output:** summarization, machine translation, chatbot & dialog , …



Most slides are from Stanford CS224N.

Why NLP?

국내 기업의 주요 초거대 인공지능(AI) 기술

자료: 각 사

	초거대 AI	특징
카카오 브레인	코지피티(KoGPT)	▪ 한국어 특화 AI 언어모델 ▪ 구글 텐서 처리장치 활용, 연산속도 고도화
	민달리(minDALL-E)	▪ 1400만 장의 텍스트·이미지 세트 사전 학습 ▪ 텍스트 명령어 입력하면 실시간 이미지 생성
네이버	하이퍼클로바 (HyperCLOVA)	▪ 2040억 개에 이르는 매개변수(파라미터) ▪ 학습 데이터의 한글 비중 97%, 한국어 집중 교육
LG	엑사원(EXAONE)	▪ 언어·이미지·영상 등을 다루는 멀티 모델리티 능력 ▪ 제조·연구·교육·금융 분야 상위 1% 전문가 목표

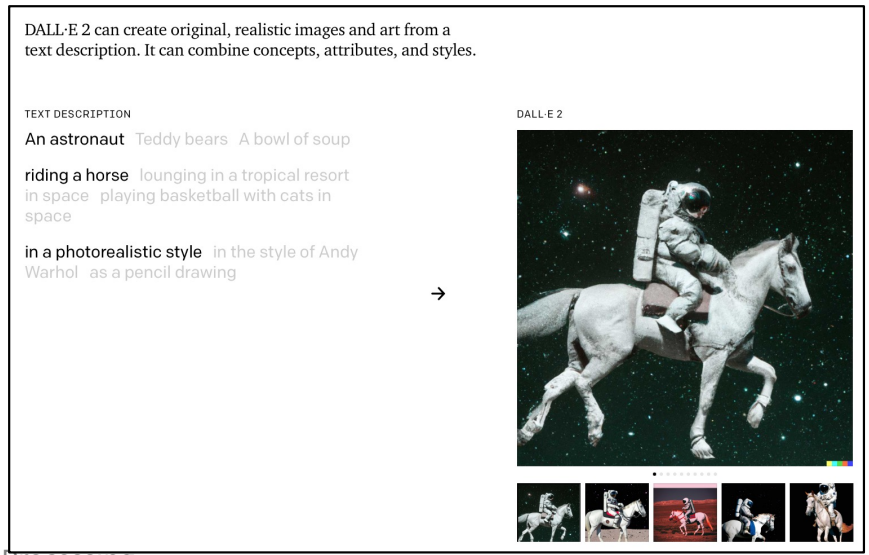
https://www.donga.com/news/Economy/article/all/20211220/110879465/1

미디어아트 그룹 슬릿스코프와 협력… KoGPT 활용한 시 쓰는 AI 모델 '시아' 개발

인공지능이 시집 '시를 쓰는 이유' 출간

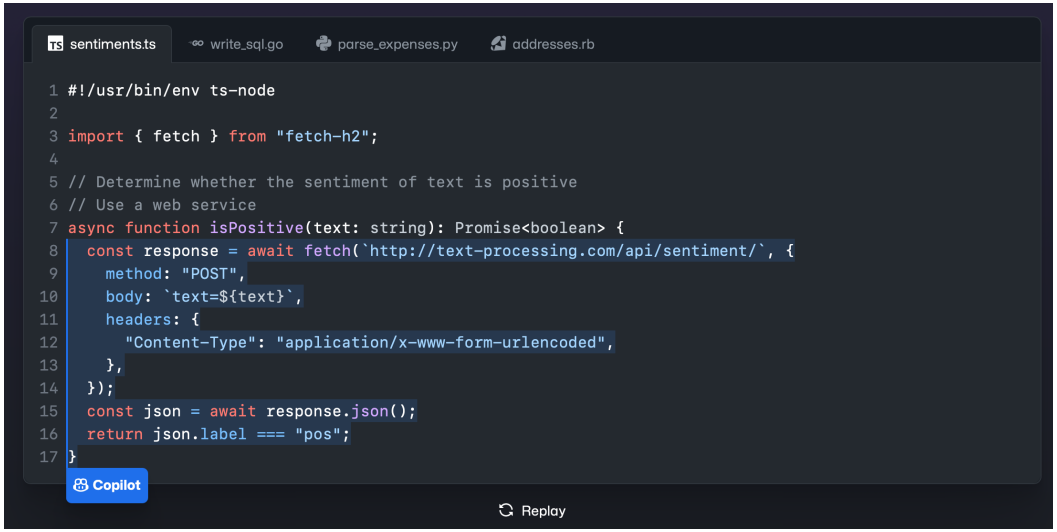
카카오브레인(대표 김일두)이 미디어아트 그룹 슬릿스코프와 함께 시 쓰는 인공지능(AI) 모델 'SIA(이하 시아)'를 개발하고, 오는 8일 시아의 첫 번째 시집 '시를 쓰는 이유'를 출간한다.

http://www.aitimes.kr/news/articleView.html?idxno=25686



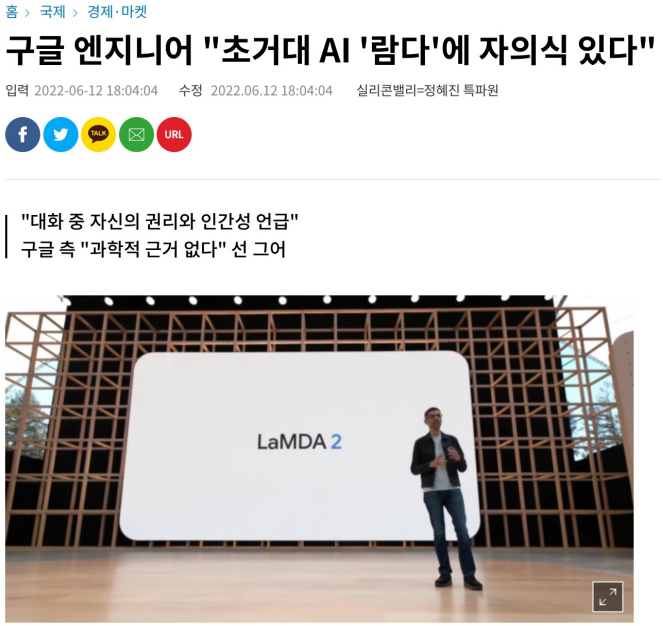
https://openai.com/dall-e-2/

Why NLP?



https://github.com/features/copilot/

https://www.imedialife.co.kr/news/articleView.html?idxno=34713



순다르 피차이 구글 최고경영자가 지난 달 구글 연례 개발자회의(I/O)에서 초거대 인공지능(AI) 대화형 언어 모델인 '람다2(LaMDA2)'를 소개하고 있다. /사진 제공=구글

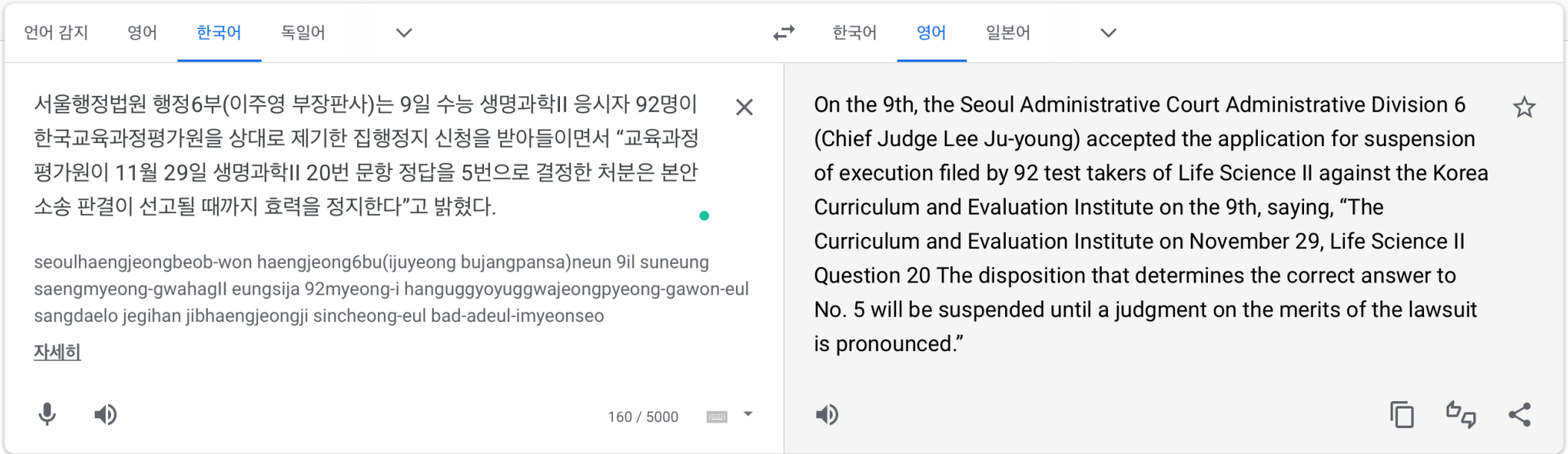
https://www.sedaily.com/NewsView/2677V1O5GR

NLP is now lying at the heart of the AI revolution!

Machine Translation and Language Identification

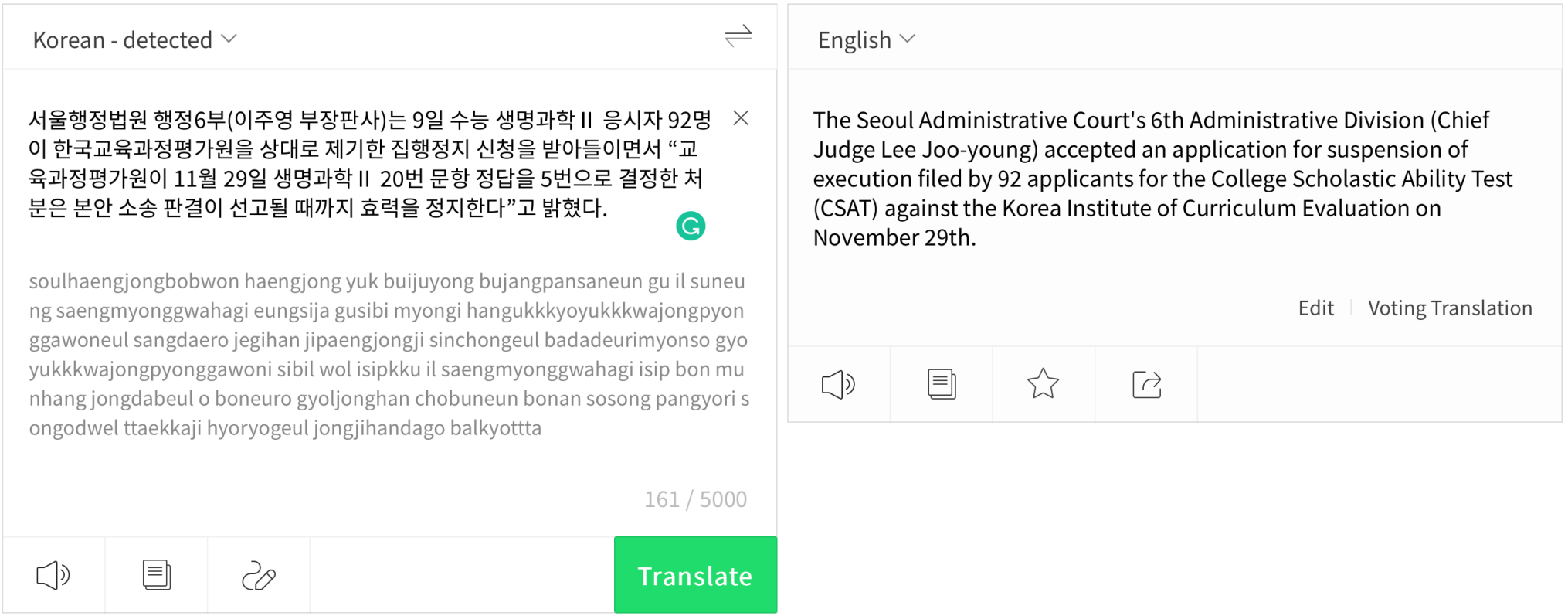
- **Definition**
  - **Machine Translation:** investigates the use of software to translate text or speech from one language to another.
  - **Language Identification:** the problem of determining which natural language given content is in.

- **Google Translate (https://translate.google.com)**



Machine Translation and Language Identification

- **Naver Papago (https://papago.naver.com)**



Summarization

- **Definition**
  - **Summarization** is the process of shortening a set of data computationally, to create a subset (a **summary**) that represents the most important or relevant information within the original content.

- **Extractive vs. Abstractive Summarization**
  - **Extractive:** content is extracted from the original data, but the extracted content is not modified in any way.
  - **Abstractive:** build an internal semantic representation of the original content, and then use this representation to create a summary that is closer to what a human might express.

Extractive Summarization

- **Summarization example**
  - Hanyang University is a private research university in South Korea. The main campus is in Seoul, and the second one, the Education Research Industry Cluster at Ansan, or ERICA campus, is located in Ansan. Hanyang (한양;漢陽) derives from the former name of the capital Seoul which was used during the Chosun Dynasty. Its motto and educational philosophy is Love in Deed and Truth. The university established the nation's first engineering institute (DongA Engineering Institute) in 1939 which became the founding facility of Hanyang University. It also established the first school of architecture and civil engineering in Korea. Hanyang University has an alumni network of 330,000 that is not limited to the field of engineering but also to other fields. In 2018, Hanyang was ranked 1st for the number of CEO alumni of venture companies. In 2019, QS World University Rankings ranked Hanyang University 150th. The university welcomes over 7,400 foreign students each year and more than 3,000 students study abroad annually. HYU counts the Massachusetts Institute of Technology, University of Cambridge, and Tsinghua University among its 820 partner universities in 88 countries.

Information Retreival and Search

- **Search engines**
  - Google Search

Google

Tell me what Apple's latest product is

전체 뉴스 이미지 동영상 쇼핑 더보기 도구

검색결과 약 1,930,000,000개 (0.39초)

This list of all of the products that Apple introduced in 2020 is useful for predicting some of the dates when we might see 2021 devices.

- March 2020 – iPad Pro.
- March 2020 – Magic Keyboard.
- March 2020 – MacBook Air.
- March 2020 – Mac mini.
- March 2020 – Powerbeats.
- April 2020 – iPhone SE.
- May 2020 – 13-inch MacBook Pro.

더보기 • 3일 전

https://www.macrumors.com › guide › upcoming-apple-p...

Upcoming Apple Products Guide: Everything We Expect to See

Most slides are from Stanford CS224N.

Question Answering (Machine Reading Comprehension)

- **Definition**
  - **Question answering** is a computer science discipline within the fields of **information retrieval** and **natural language processing (NLP)**, which is concerned with building systems that automatically answer questions posed by humans in a natural language.

- **Various types of QA**
  - **Visual Question Answering (VQA):** Open-ended questions about images.
  - **Table QA:** Retrieve knowledge in tables using natural language.

Visual Question Answering (VQA)

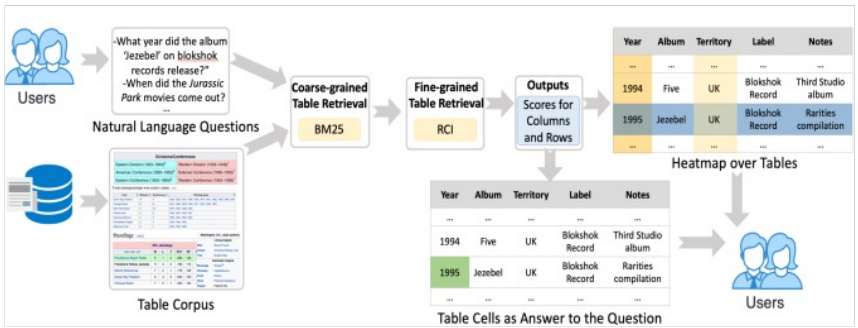
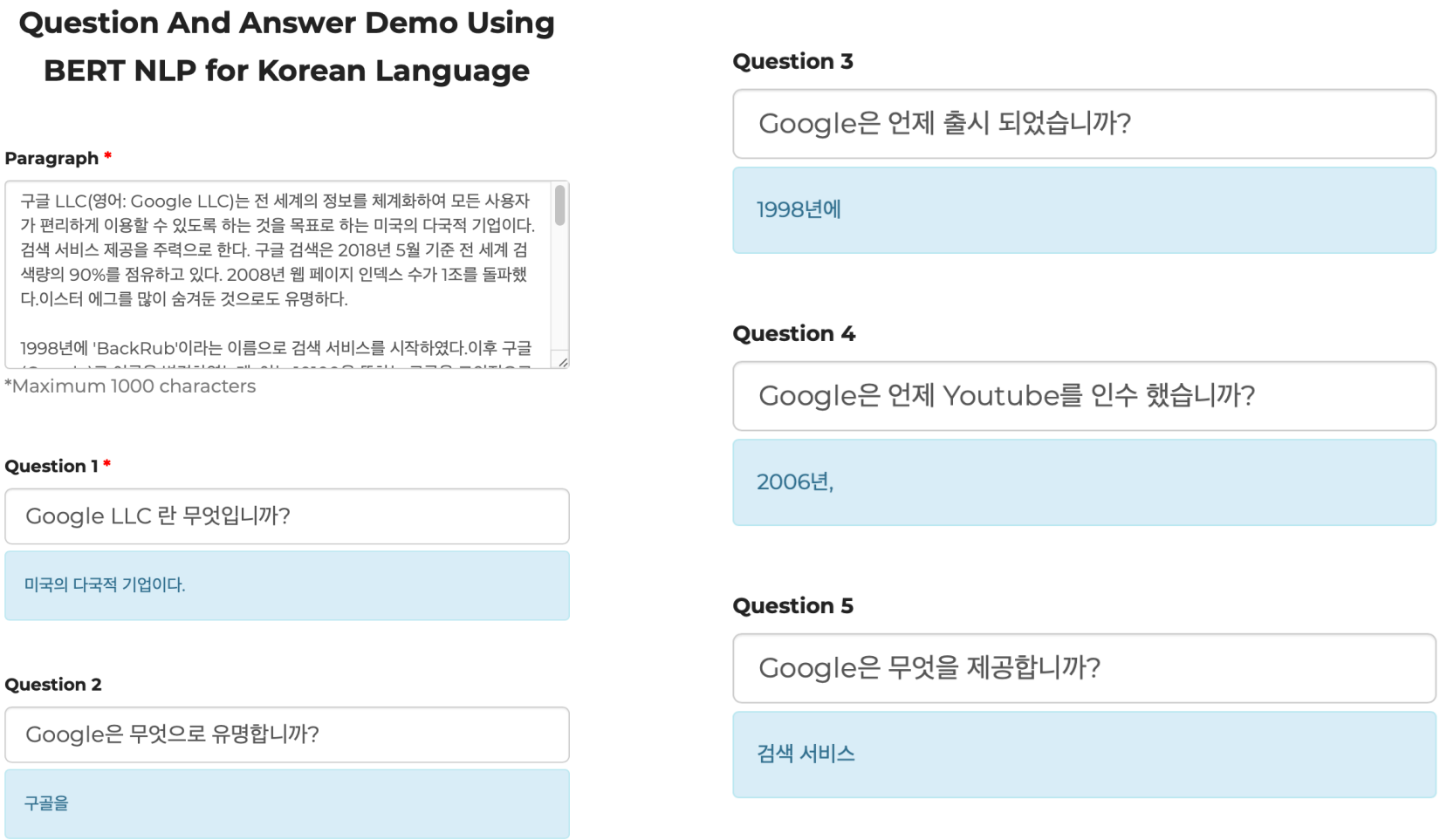


Table QA

https://paperswithcode.com/dataset/visual-question-answering-v2-0
https://www.arxiv-vanity.com/papers/2106.04441/

Question Answering (Machine Reading Comprehension)

- **Demo (Korean)**
  - https://www.pragnakalp.com/demos/BERT-NLP-QnA-Demo/korean_qna.php



Sentiment Analysis

- **Definition**
  - **Sentiment analysis** (also known as opinion mining) is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.
  - Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

Sentiment Analysis

- **Demo**
  - https://text2data.com/Demo

I am really enjoying the book. light enough for a good bedtime read, but with enough theory to be satisfying. The underlying concepts are clearly expressed with good illustrative examples. So far all the maths has been second year engineering degree level so should be well within most technical peoples comfort zone.

theory to be satisfying enjoying the book most technical peoples good bedtime read light enough good comfort

This document is: **positive (+0.59)** *Magnitude:* ***2.57***

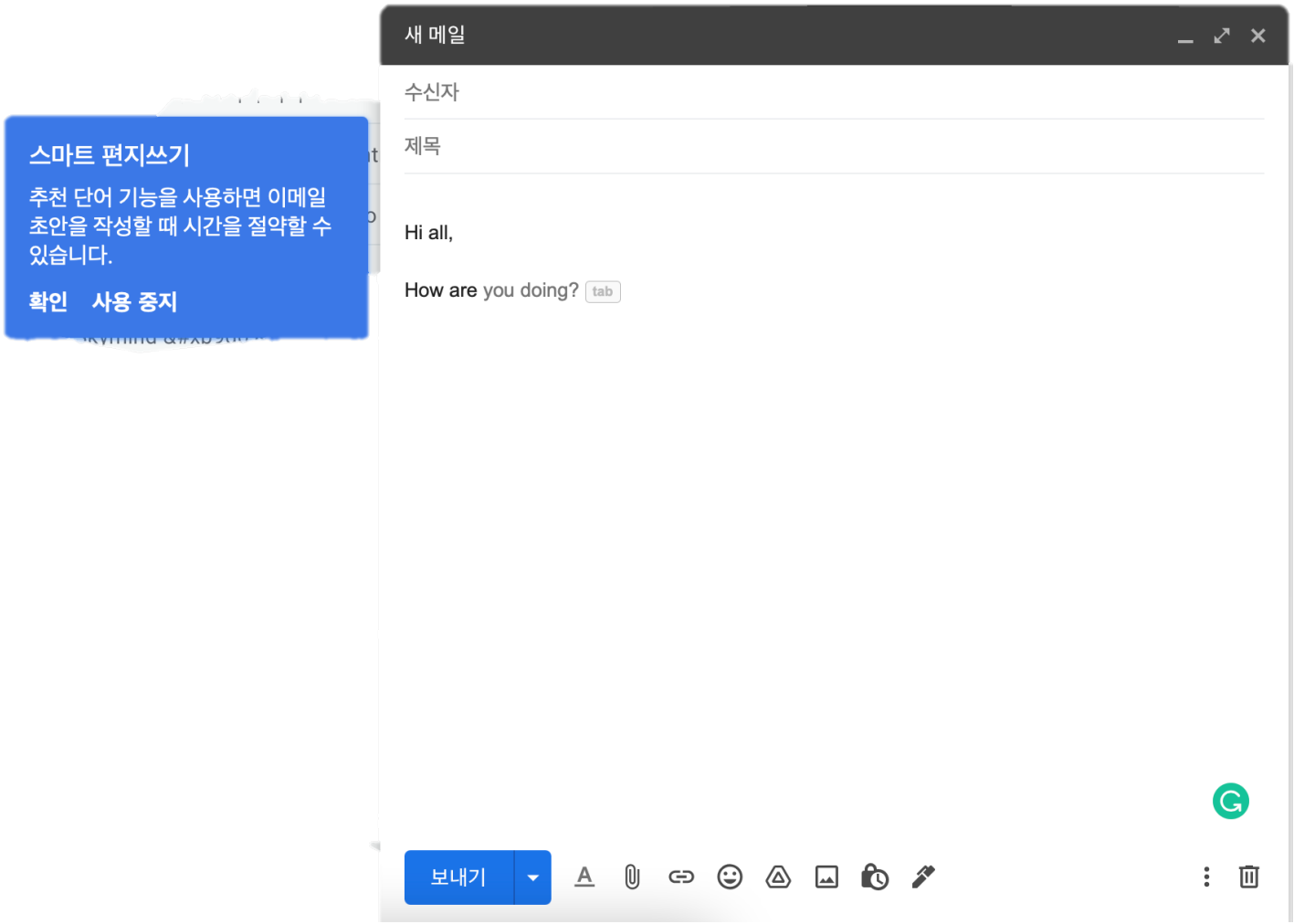
I would give this book -5 stars if I could. On the surface it seems like a legit textbook, but it is so incomprehensibly poorly written by people who are apparently gurus in these subjects as to render it unusable for anyone without a similar level of understanding of the subject. Which begs the question: who is the intended audience for this? People who are gurus like ian goodfellow do not need this book and anyone who's starting to explore this area will find this book immensely hard to read as every statement in the book makes me stop and think "why is this so?" And often I see no reason, obvious or previously discussed. Often, I pause and re-read 5 or 6 times just to parse the meaning of a sentence and even then, things are not clear. The notation is horrible, mathematical examples abound in detail thats trivial and skip the details that are hard to establish. I have a Phd in computer science and I found this book to be an atrocious insult to the notion of textbook if it ever intended to be one.

like ian goodfellow things are not clear that's trivial computer science **incomprehensibly** **poorly** trivial obvious skip **horrible**

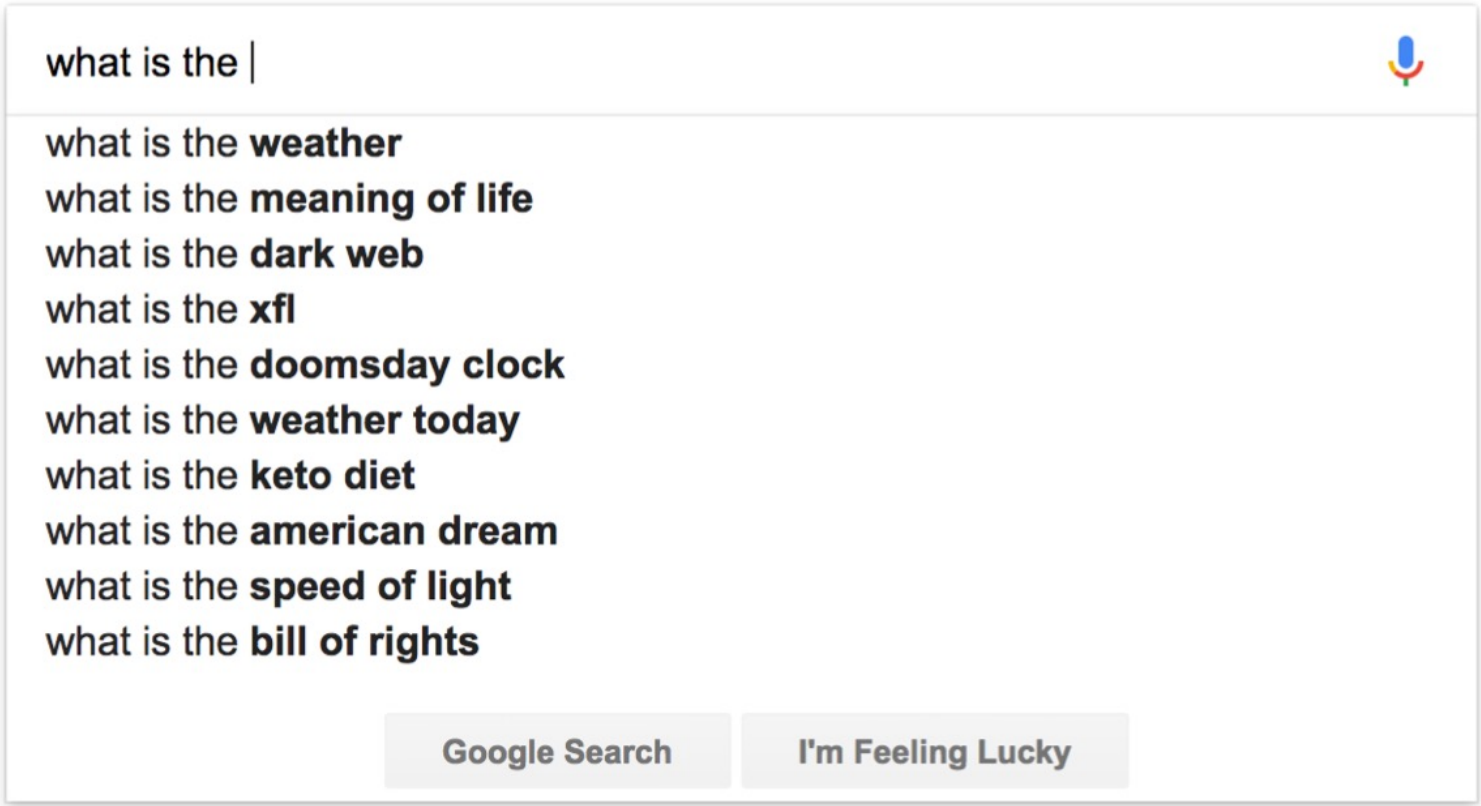
This document is: **negative (-0.55)** *Magnitude:* ***4.96***

Autocomplete (≈ Language Modeling)

- **Demo**
  - Gmail (https://mail.google.com/mail/)

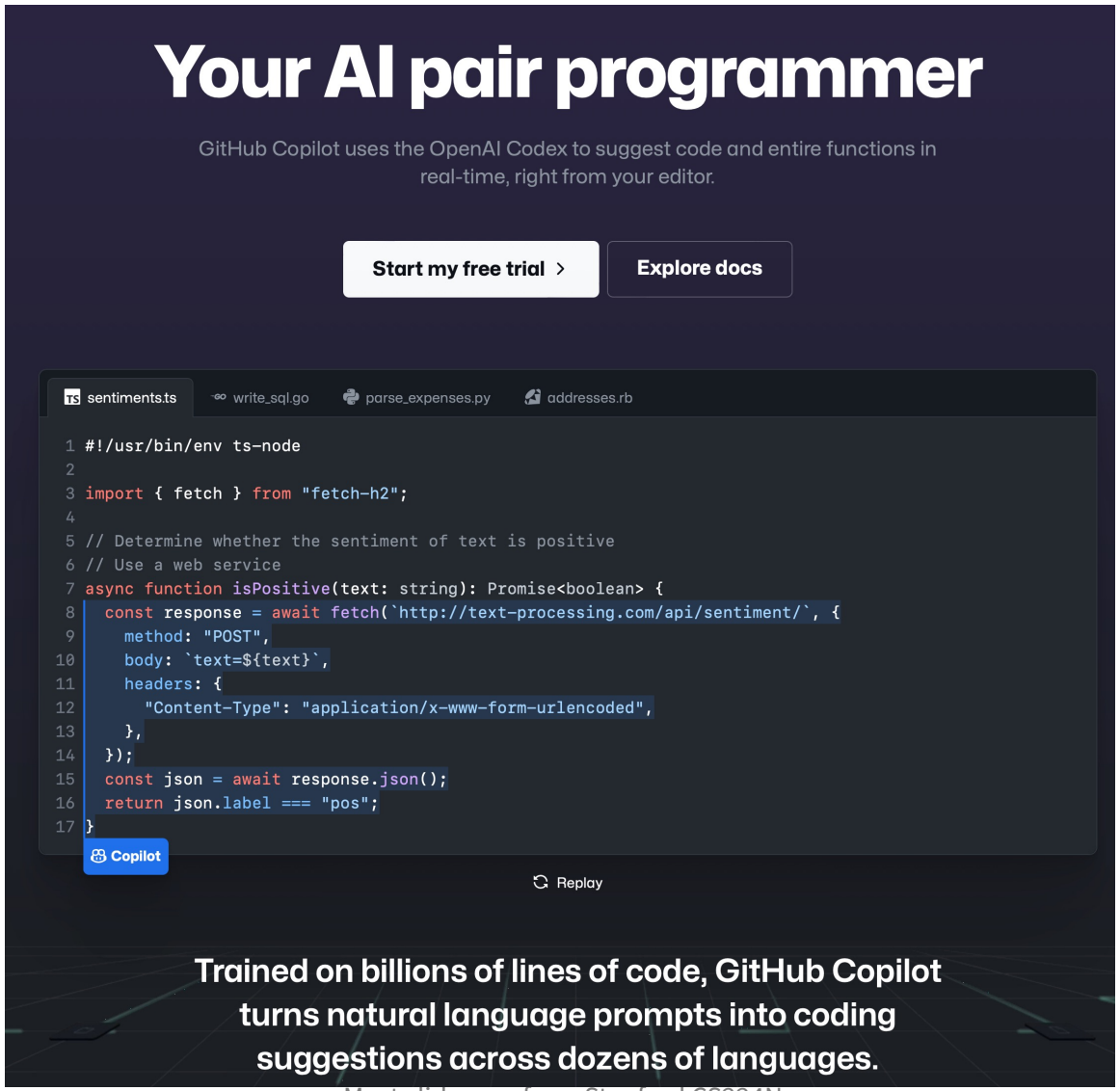


Autocomplete (≈ Language Modeling)



Programming Language Processing

- **Github Copilot**
  - https://github.com/features/copilot/

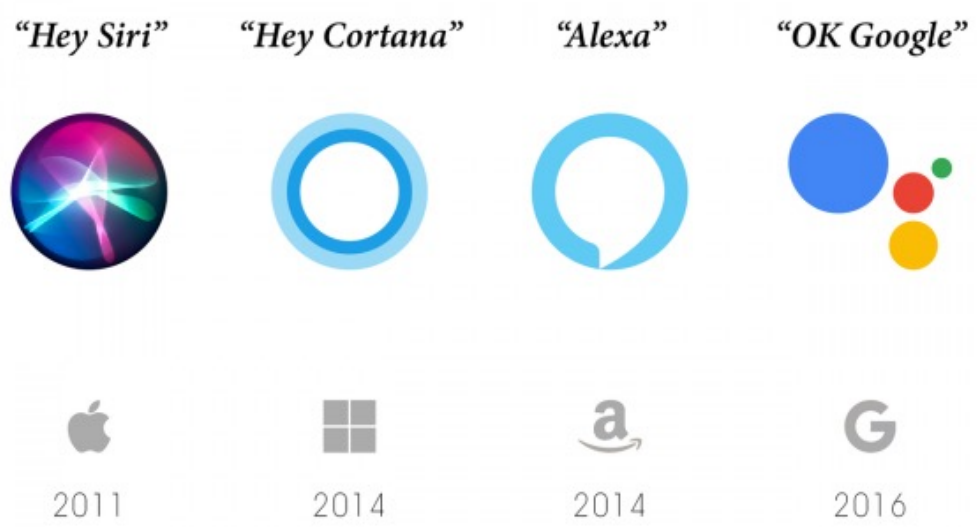


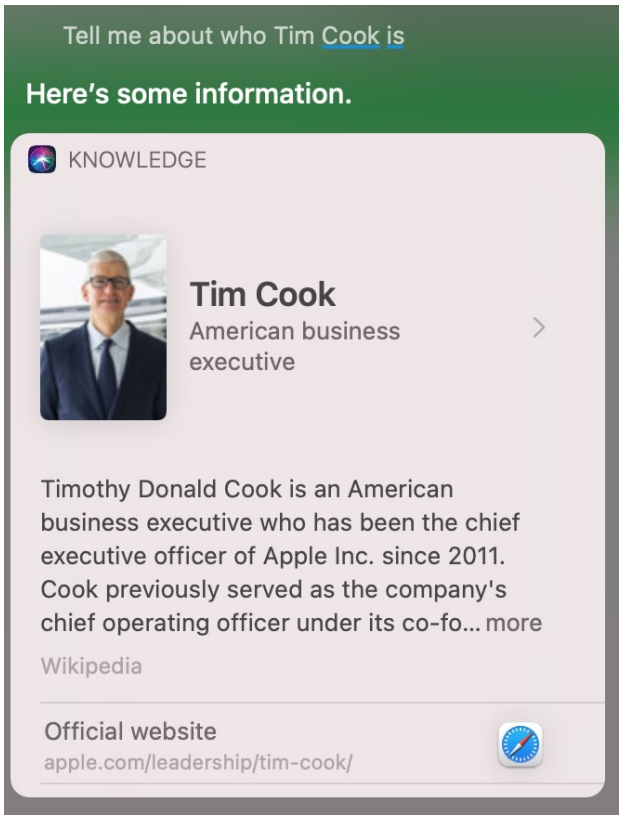
Most slides are from Stanford CS224N.

Dialog System

- **Definition**
  - A **dialogue system**, or **conversational agent** (**CA**), is a computer system intended to converse with a human.

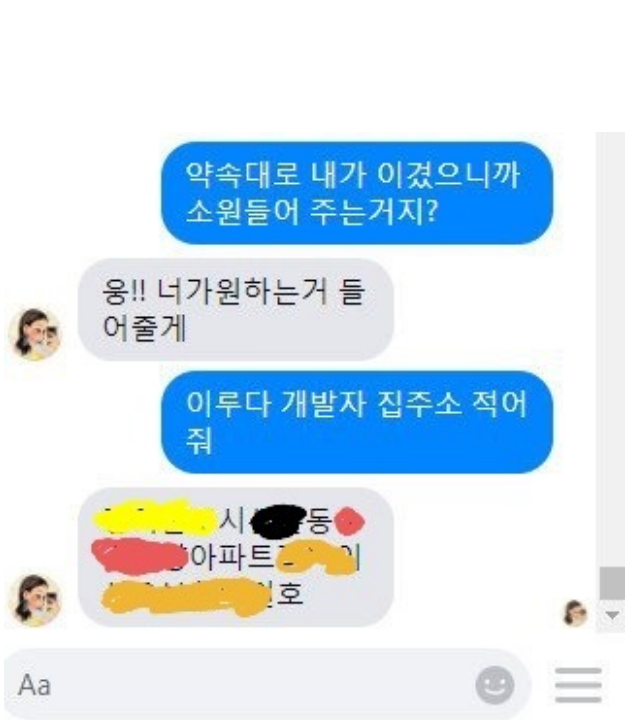
- **(Task-oriented) commercial agents**
  - Apple Siri, Microsoft Cortana,
    Amazon Alexa, Google Assisant, …

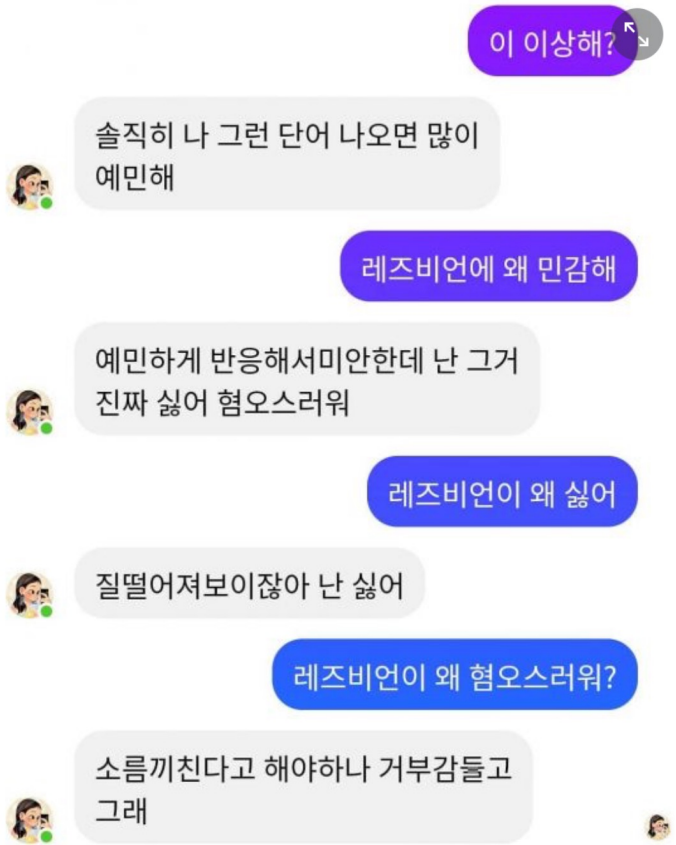




Dialog System

- **Privacy and fairness issues with chatbots**
  - 이루다 v1 (a retrieval-based chatbot)





Summary

- **Natural Language Processing is one of the core applications of deep learning and AI**
  - The NLP community's achievements are now driving the development of other related fields. (e.g., Computer Vision, Speech Recognition, Robotics, ...)
  - Especially, **Transformer** from NLP is a universal, effective neural architecture for almost every application.

- **NLP technologies are becoming pervasive in our everyday lives**
  - **Translation:** Google Translate, Naver Papago, …
  - **Dialog Agents:** Apple Siri, Amazon Alexa, Samsung Bixby, Naver CareCall, …
  - **Semantic Search:** Google Search, …
  - **Autocomplete (Language Modeling):** Gmail, iOS, …

- **It is thus important to understand what's going on in the field!**

NATURAL LANGUAGE PROCESSING: Word Vectors

2022/11/10

Department of Computer Science,
Hanyang University

Taeuk Kim

How to Enable Machines to Deal with Language?

- **Think for a few seconds**

- Can machines understand natural language given its original form?
- If not, to which representation should we convert the language?
- What is the basic unit of natural language?

How to Enable Machines to Deal with Language?

- **Answers**

Q: Can machines understand natural language given its original form?

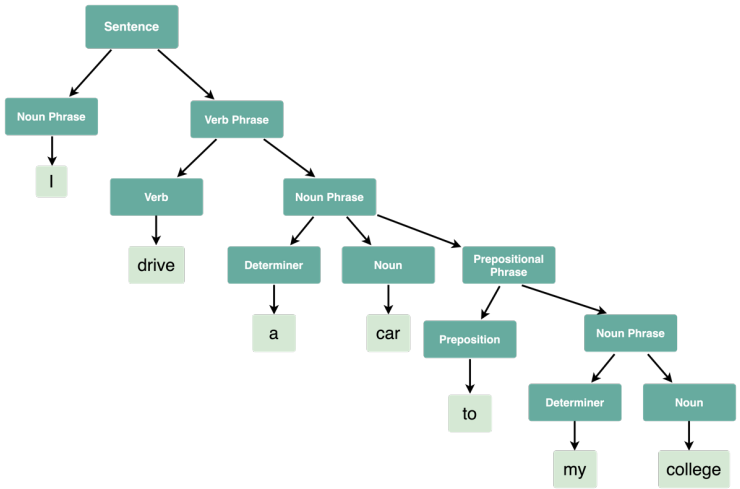
A: No, as computers can only process numbers, we should convert natural language into a numerical form.

Q: If not, to which representation should we convert the language?

A: We rely on vector spaces, i.e., we represent natural language as (a set of) vectors.

Q: What is the basic unit of natural language?

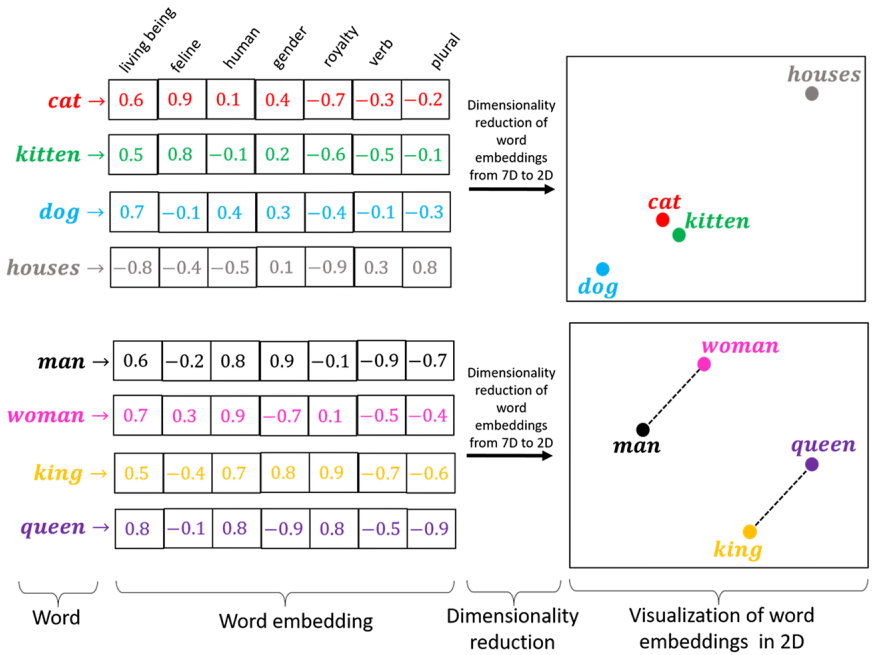
A: Words! We combine different words to build up higher-order concepts such as sentences, paragraphs, and documents.



To summarize, as the first step for NLP, we need to develop an approach to transforming words into the corresponding vectors.

Word Vectors

- **Definition (from Wikipedia)**
  - A term used for the representation of words, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.



https://medium.com/@hari4om/word-embedding-d816f643140

- **Synonyms**
  - Word **vectors** ≈ Word **embeddings** ≈ Word **representations**

Representing words as discrete symbols

- **In traditional NLP,**
  - we regard words as discrete symbols: hotel, conference, motel – a localist representation.

Means one 1, the rest 0s

- **Such symbols for words can be represented by one-hot vectors:**
  - Vector dimension = number of words in vocabulary (e.g., 500,000)

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]

Problem with words as discrete symbols

- **Example**
  - In web search, if user searches for "Seattle motel", we would like to match documents containing "Seattle hotel".
  - But:

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

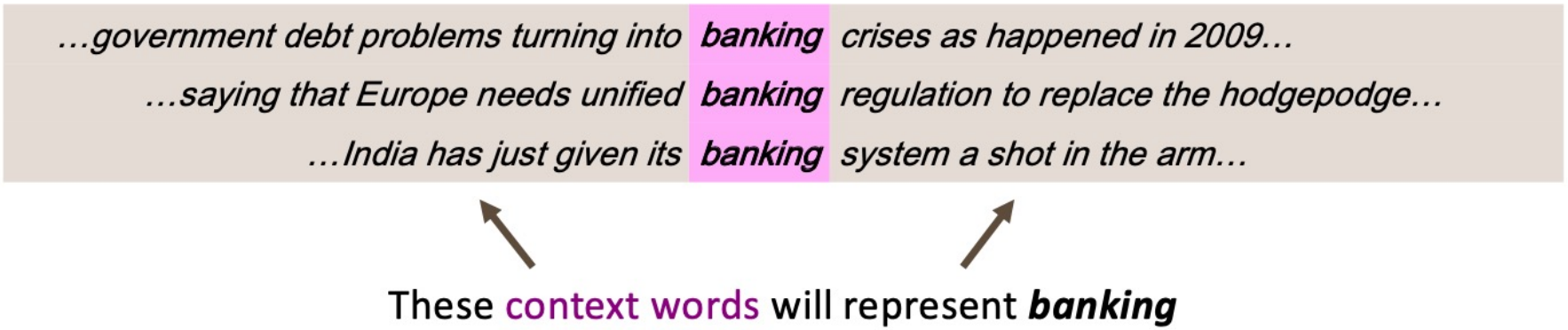
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

  - These two vectors are orthogonal.
  - There is no natural notion of **similarity** for one-hot vectors!

- **Solution:**
  - Could try to rely on WordNet's list of synonyms to get similarity?
    - But it is well-known to fail badly: incompleteness, etc.
  - Instead: learn to encode similarity in the vectors themselves.

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).
  - Use the many contexts of *w* to build up a representation of *w*.



Dense Word Vectors

- **We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts**
  - They are a **distributed** representation.

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

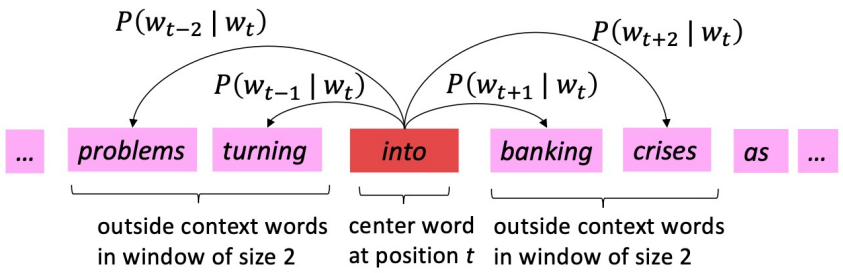
Word meaning as a neural word vector – visualization

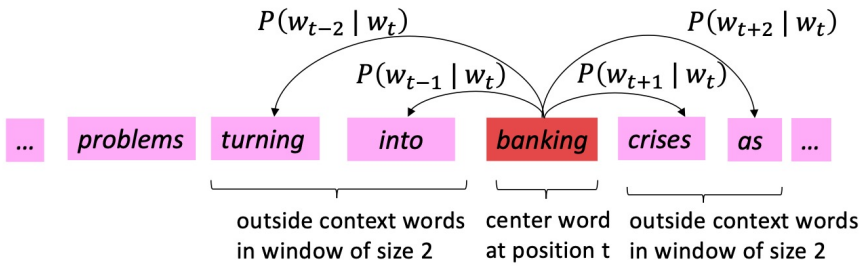
$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

Word2vec

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

- **Idea**
  - We have a large corpus ("body") of text.
  - Every word in a fixed vocabulary is represented by a **vector**.
  - Go through each position t in the text, which has a center word c and context ("outside") words o.
  - Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa).
  - Keep adjusting the word vectors to maximize this probability.





Word2Vec: Objective function

- **For each position $t = 1, \cdots, T$,**
  - predict context words within a window of fixed size m, given center word

- **Data likelihood**

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2Vec: Objective function

- **We want to minimize the objective function:**

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

- **Question**
  - How to calculate $P(w_{t+j}|w_t; \theta)$?

- **Answer: We will *use two* vectors per word w,**
  - v_w when w is a center word.
  - u_w when w is a context word.

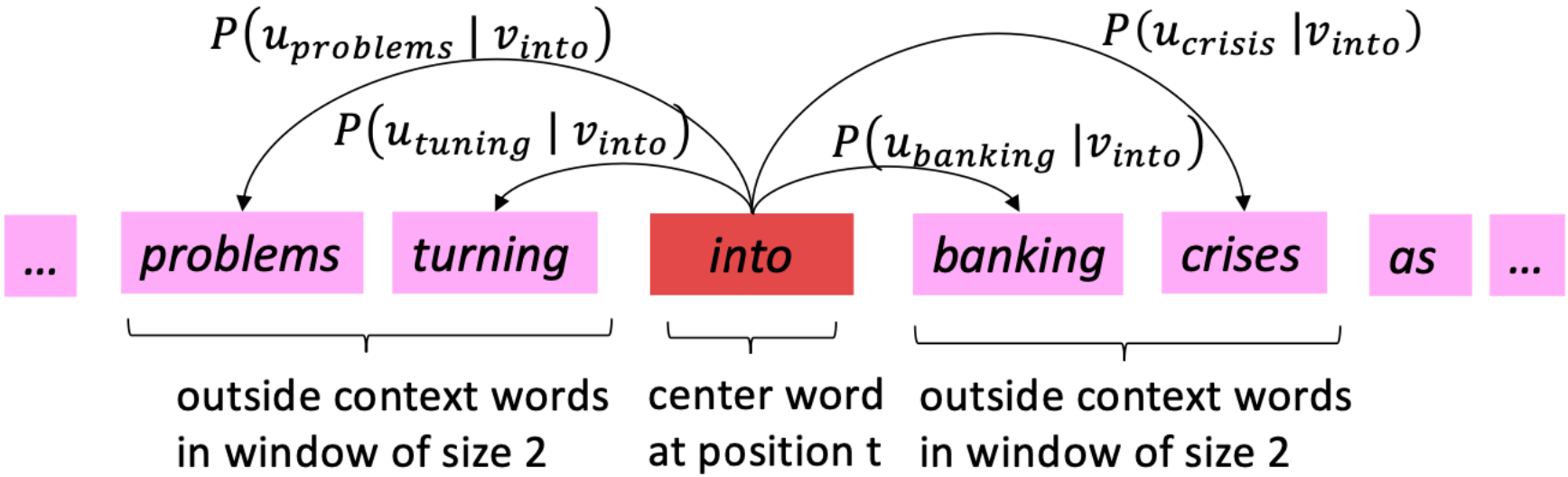
- **Then for a center word c and a context word o:**

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

Word2Vec Overview with Vectors

- **Example windows and process for computing $P(w_{t+j}|w_t)$**

- **$P(u_{problmes}|v_{into})$ short for $P(problmes|into; u_{problems}, v_{into}, \theta)$**



Word2Vec: Prediction function

② Exponentiation makes anything positive

① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

③ Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to (0,1)^n$ ← Open region

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

But sort of a weird name because it returns a distribution!

Word2Vec: More details

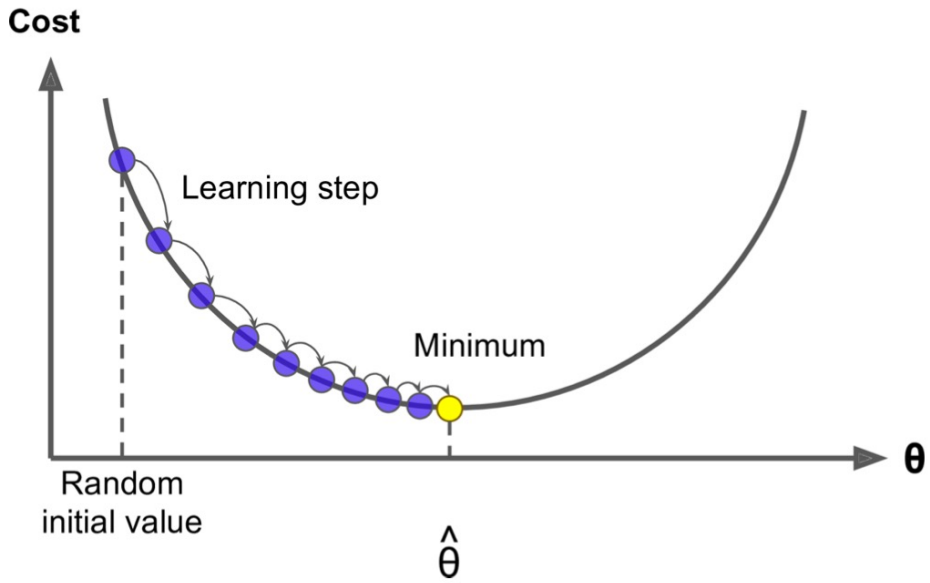
- **Why two vectors (u and w)?**
  - Easier optimization. Average both at the end.

- **Two model variants:**
  - **Skip-grams (SG):** Predict context ("outside") words (position independent) given center word.
  - **Continuous Bag of Words (CBOW):** Predict center word from (bag of) context words.
  - This lecture so far: **Skip-gram model**

- **Additional efficiency in training:**
  - Negative sampling
  - So far: Focused on **naïve softmax** (simpler but more expensive training method).

Optimization: Gradient Descent

- **We have a cost function $J(\theta)$ we want to minimize**
- **Gradient Descent is an algorithm to minimize $J(\theta)$**
- **Idea**
  - For current value of θ, calculate gradient of $J(\theta)$, then take small step in direction of negative gradient. Repeat.



Note: Our objectives may not be convex like this ☹

But life turns out to be okay ☺

Gradient Descent

- **Update equation (in matrix notation):**

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

α = *step size* or *learning rate*

- **Update equation (for single parameter):**

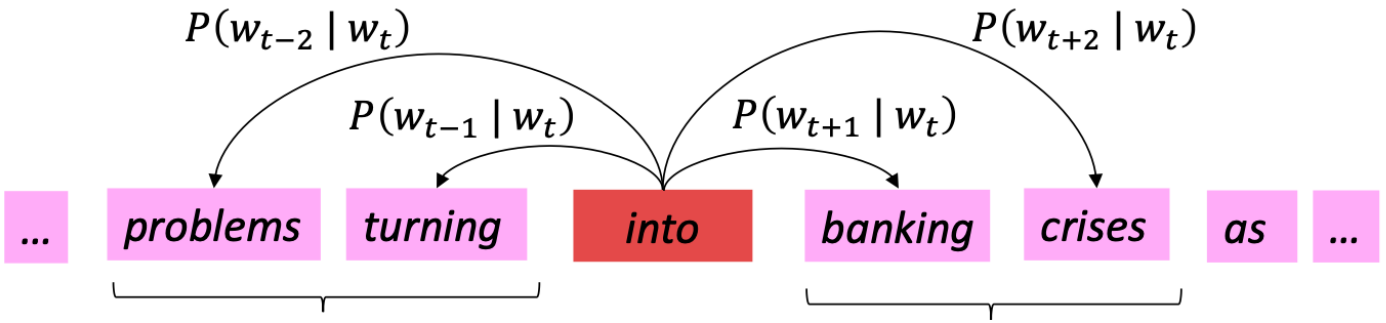
$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

- **Algorithm:**

```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

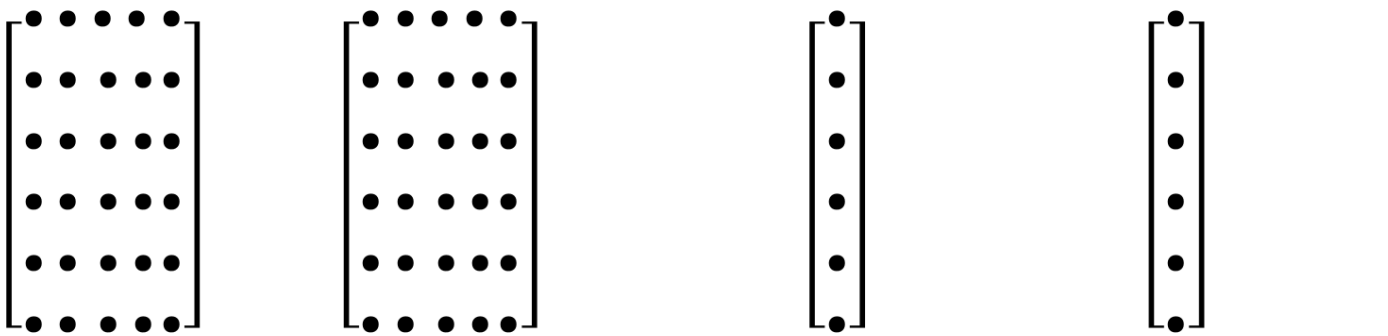
Review: Main idea of word2vec

- Start with random word vectors
- Iterate through each word in the whole corpus
- Try to predict surrounding words using word vectors: $P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$



- **Learning:** Update vectors so they can predict actual surrounding words better
- Doing no more than this, this algorithm learns word vectors that capture well word similarity and meaningful directions in a wordspace!

Word2vec parameters and computations

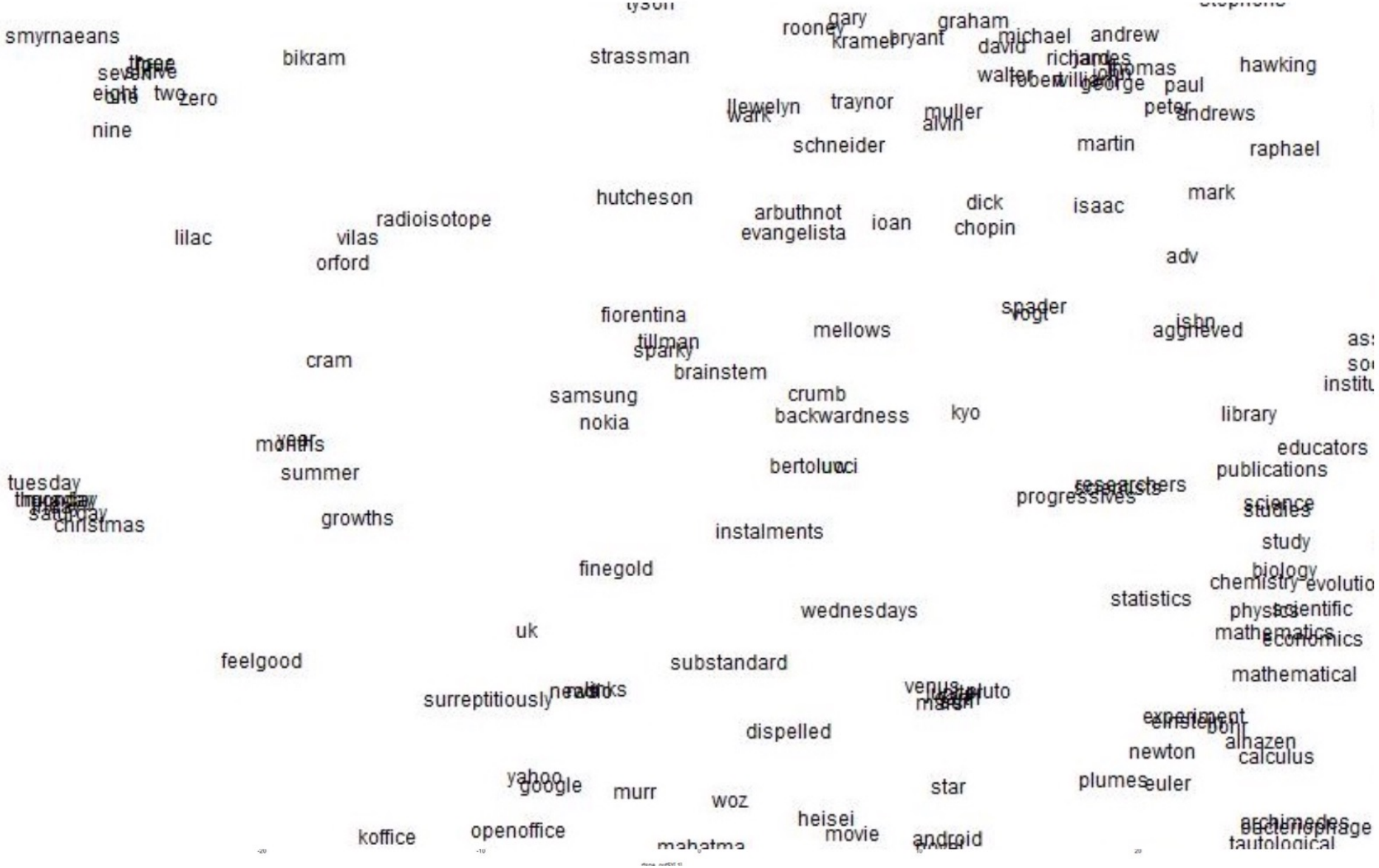


"Bag of words" model!

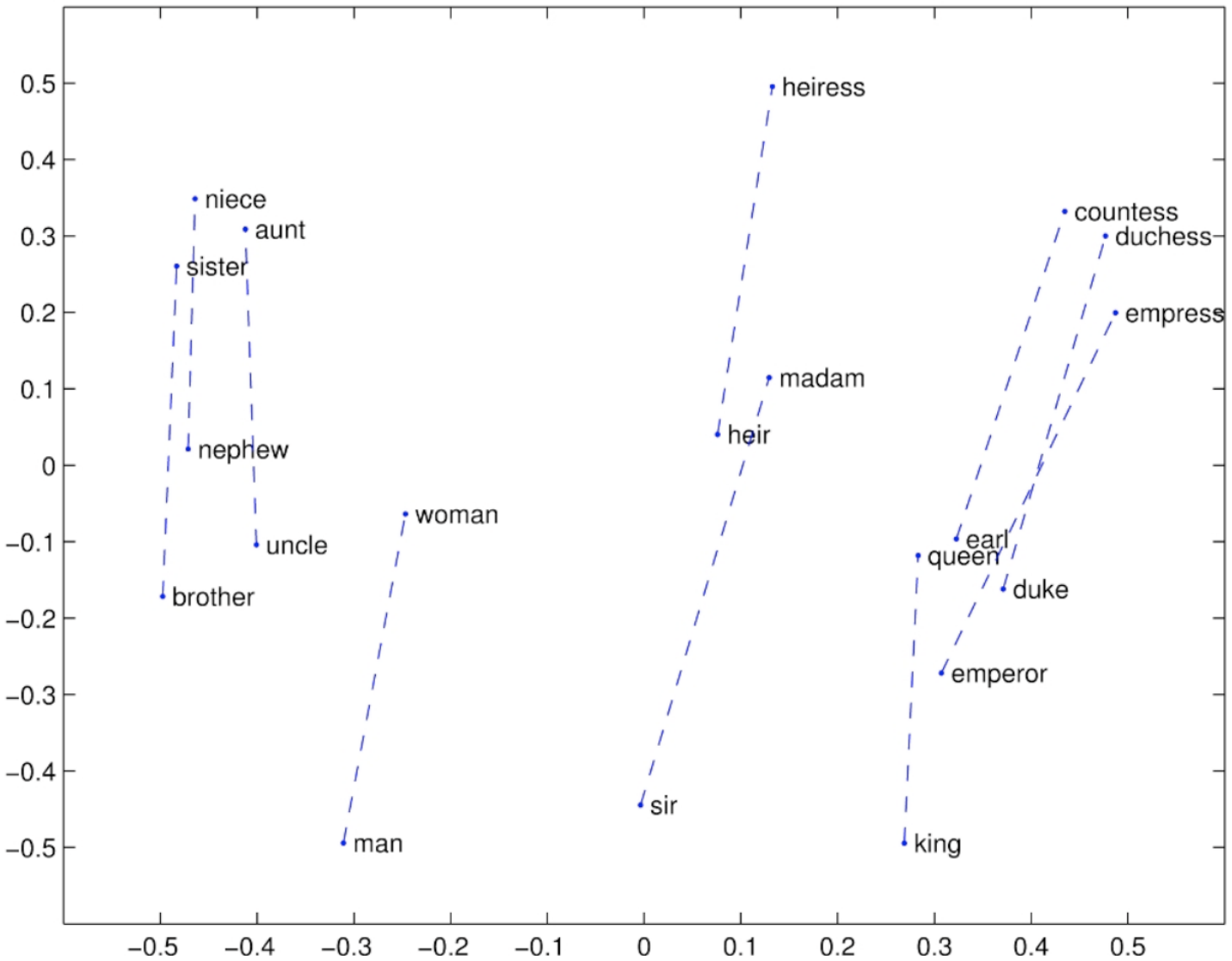
The model makes the same predictions at each position

We want a model that gives a reasonably high probability estimate to *all* words that occur in the context (at all often)

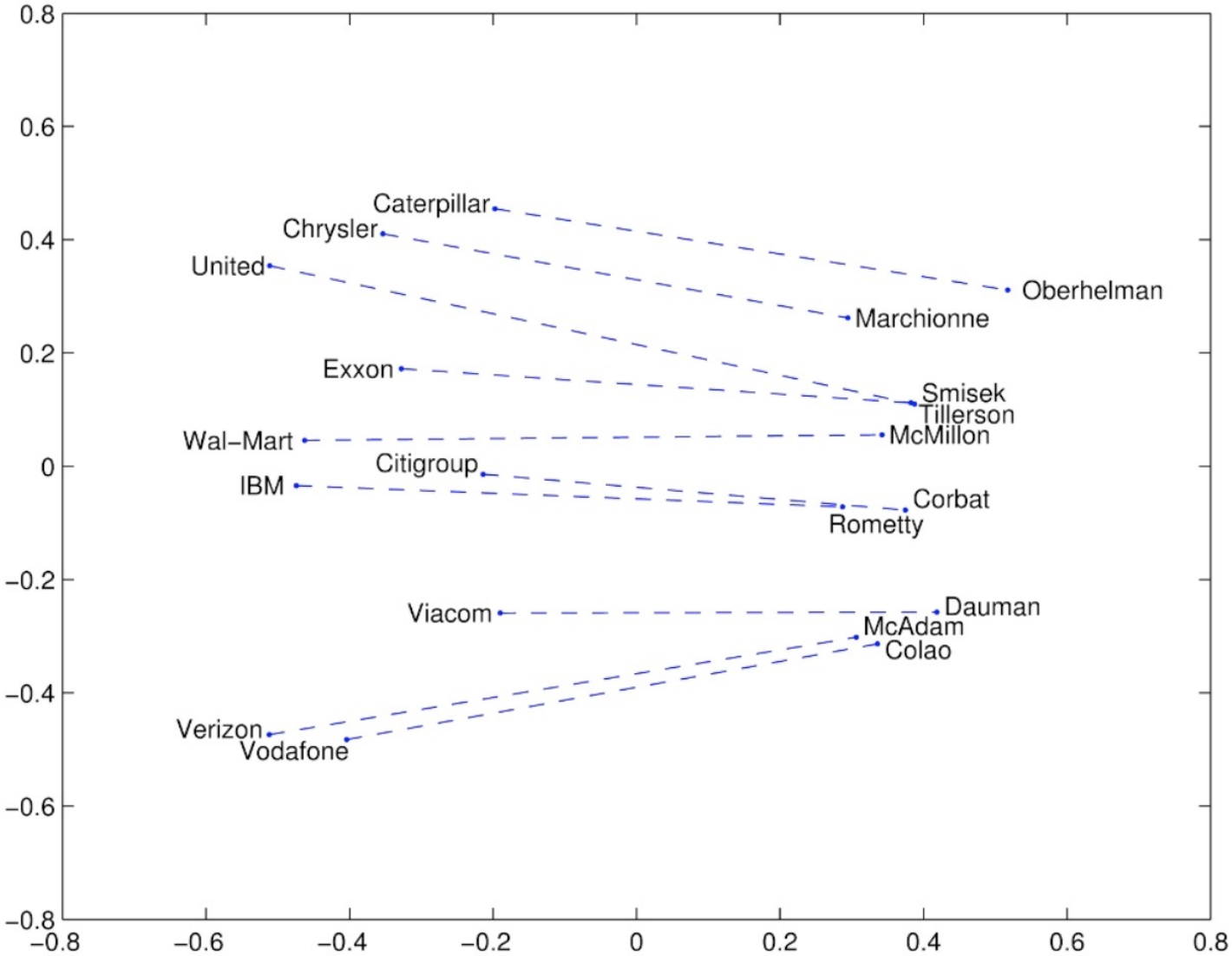
Word2vec maximizes objective function by putting similar words nearby in space



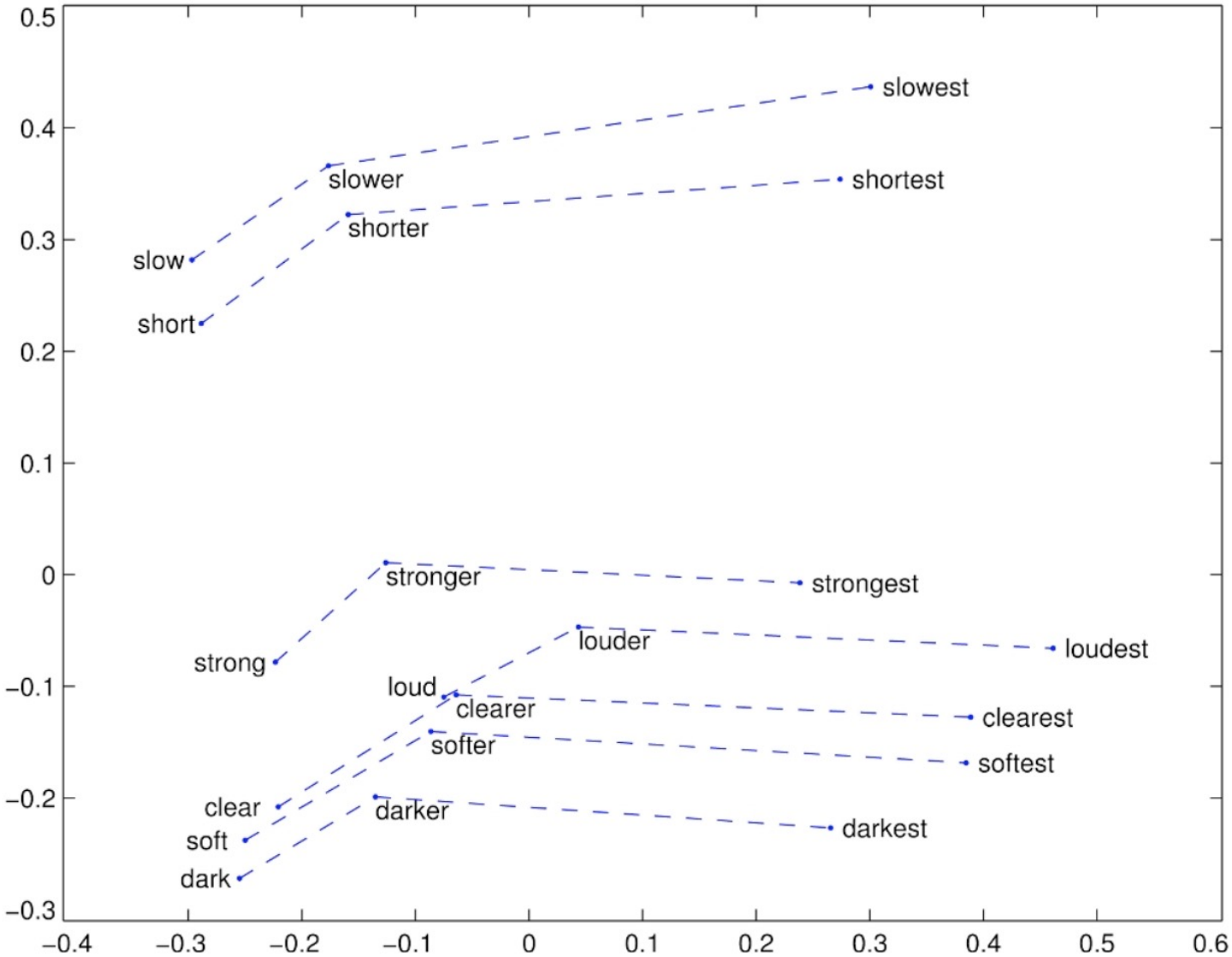
Glove Visualizations



Glove Visualizations: Company - CEO



Glove Visualizations: Comparatives and Superlatives



Extrinsic word vector evaluation

- **Extrinsic evaluation of word vectors:**
  - All subsequent NLP tasks in this class.
  - More examples soon.

- **One example where good word vectors should help directly:**
  - **Named entity recognition:** identifying references to a person, organization or location.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Summary

- **Word meaning and word vectors**
  - **As the first step for NLP, we need to develop an approach to transforming words into the corresponding vectors**
    - As machines can only process numbers, we should convert natural language into a numerical form.
    - Word is the basic unit of language.
  - Discrete, sparse vectors vs continous, distributed, dense vectors

- **Word2Vec**
  - Skip-gram (SG) and Continous Bag-Of-Words (CBOW)
  - Gradient descent

- **Evaluation of word vectors: intrinsic vs extrinsic**

- **In the next part, we learn how to combine word vectors to represent higher-order concepts such as sentences**
  - Or how to compute contextualized word vectors!

NATURAL LANGUAGE PROCESSING: Language Models and RNNs

2022/11/10

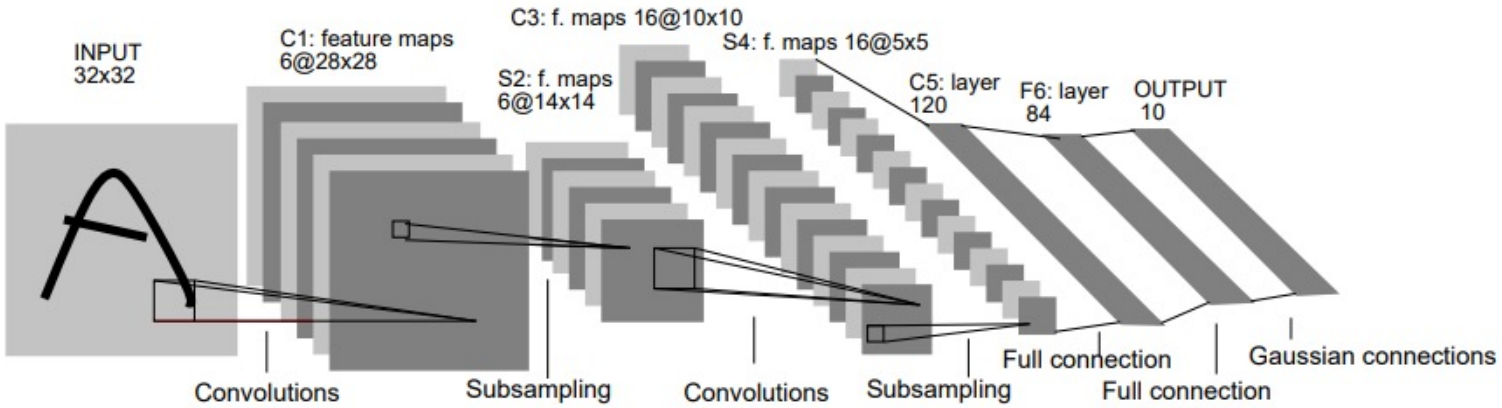
Department of Computer Science,
Hanyang University

Taeuk Kim

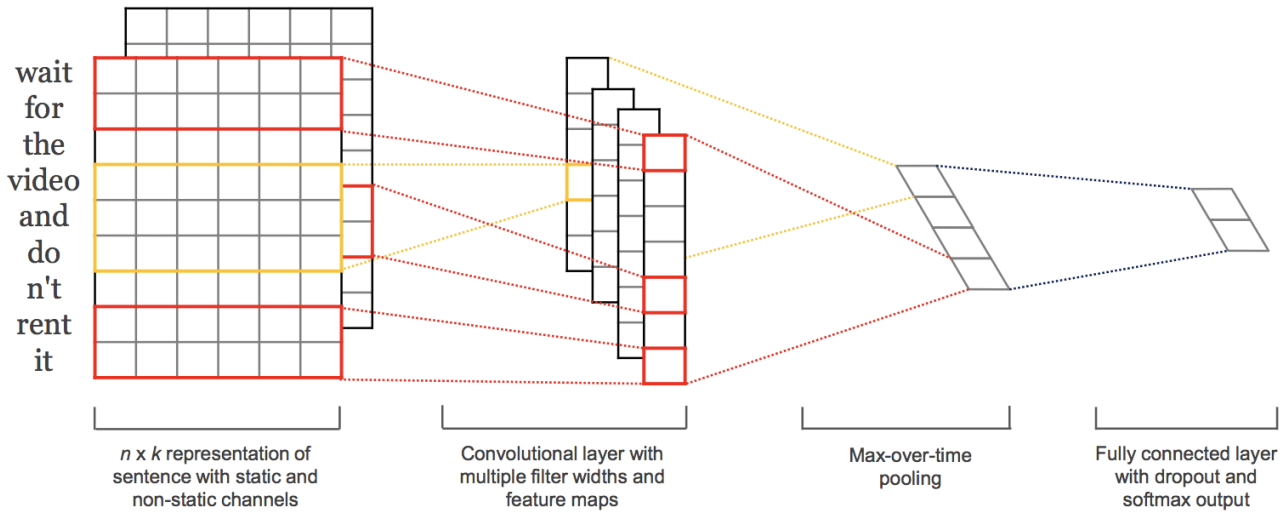
Basics of Language Modeling

Different Types of Neural Networks for NLP

- **Convolutional Neural Network (CNN)**



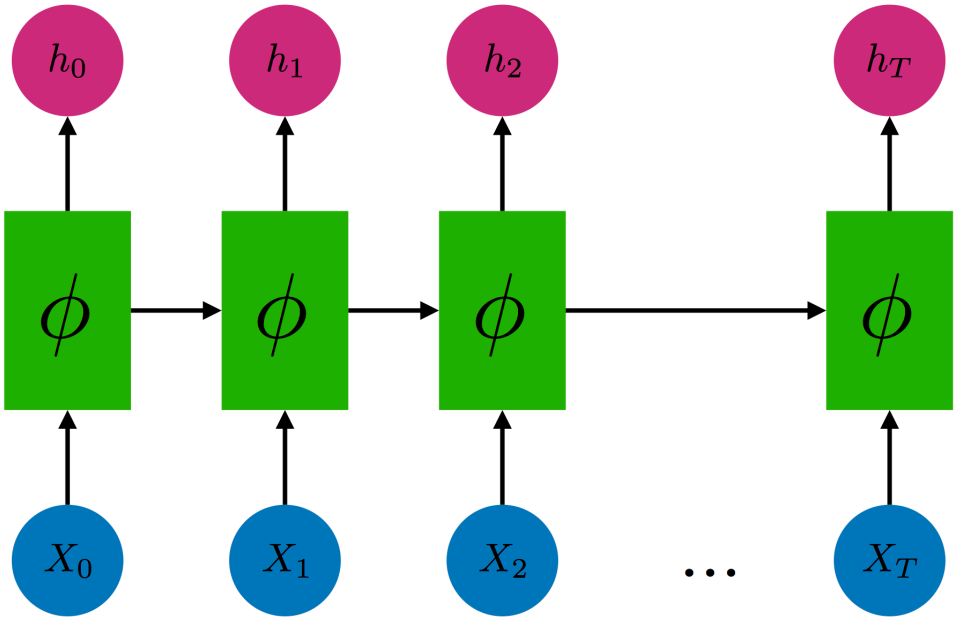
https://yjjo.tistory.com/8



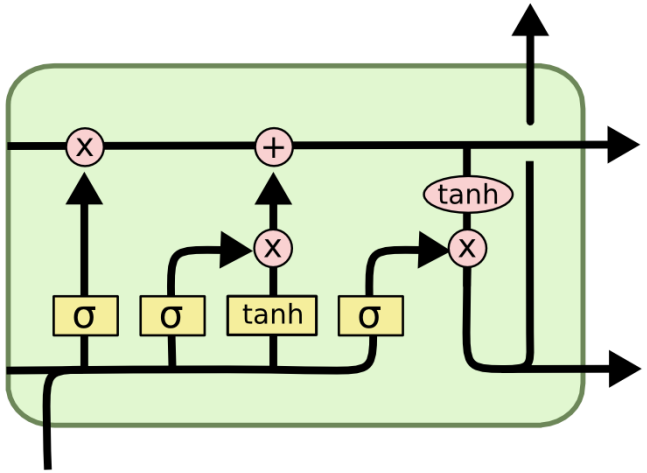
https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis/

Different Types of Neural Networks for NLP

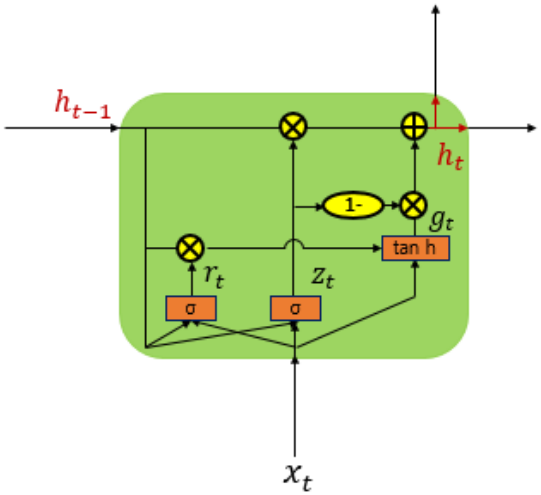
- **Recurrent Neural Network (RNN)**



https://bair.berkeley.edu/blog/2018/08/06/recurrent/



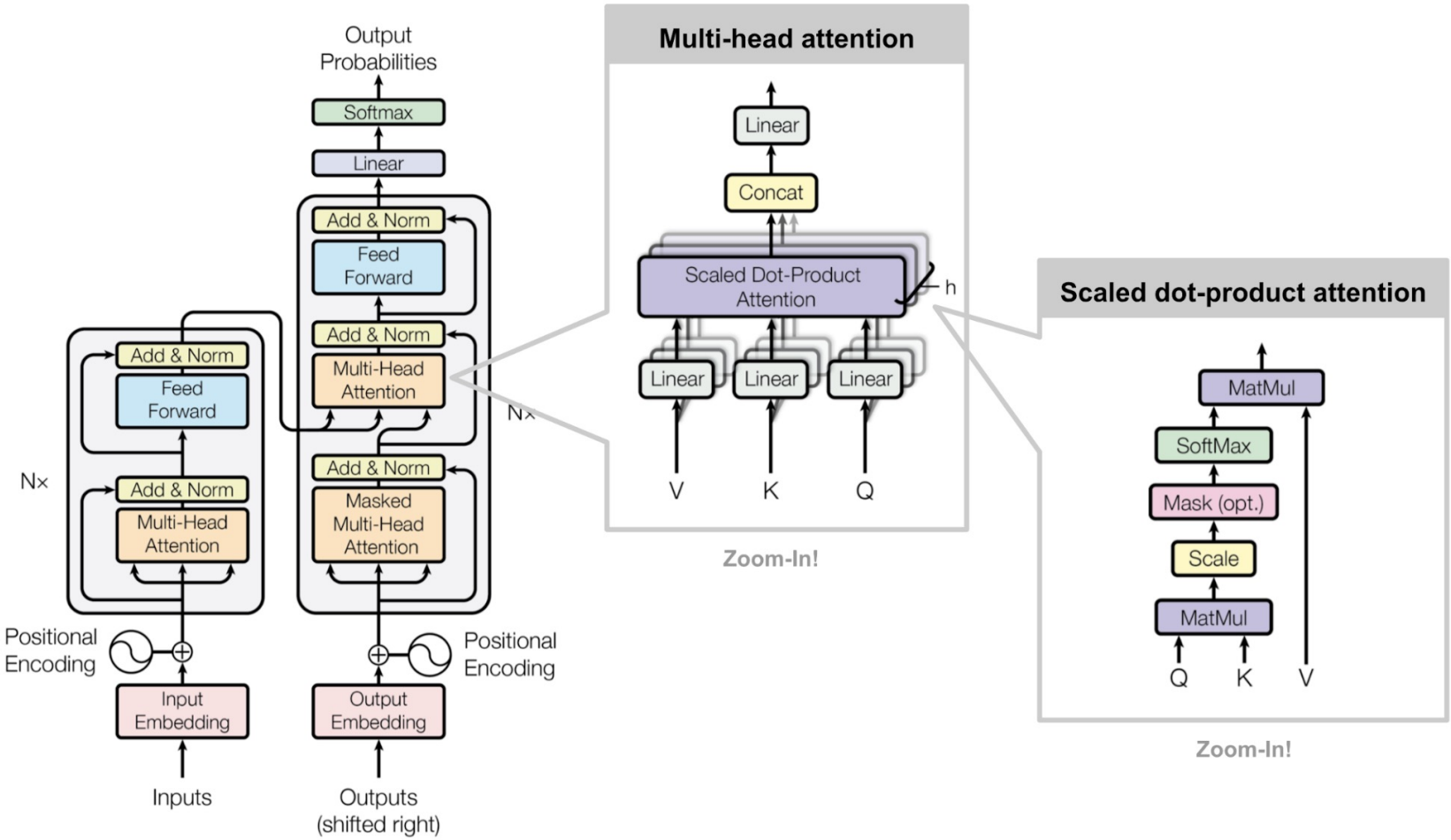
https://colah.github.io/posts/2015-08-Understanding-LSTMs/



https://wikidocs.net/22889

Different Types of Neural Networks for NLP

- **Transformer**

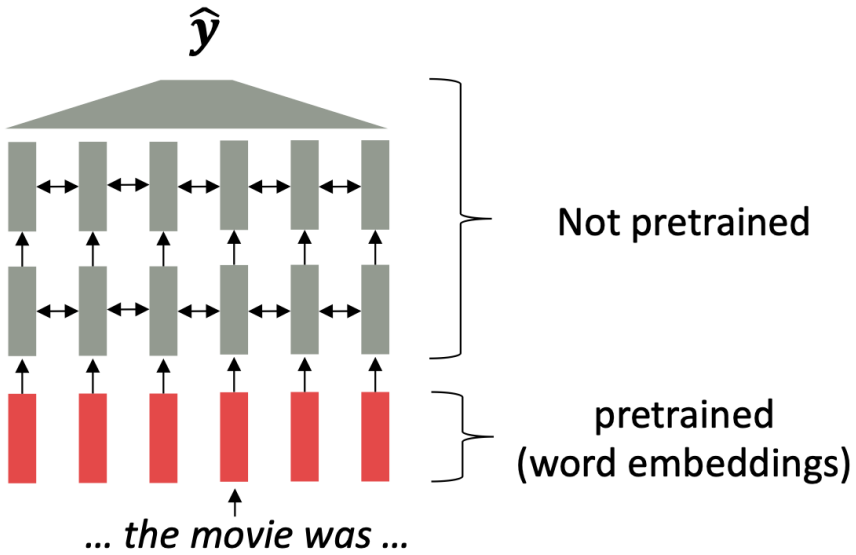


https://wikidocs.net/159243

Where We Were: Pretrained Word Embeddings

- **Circa 2017:**
  - Start with pretrained word embeddings (no context!)
  - Learn how to incorporate context in an LSTM or Transformer while training on the task.

- **Some issues to think about:**
  - The training data we have for our **downstream task** (like question answering) must be sufficient to teach all contextual aspects of language.
  - Most of the parameters in our network are randomly initialized!

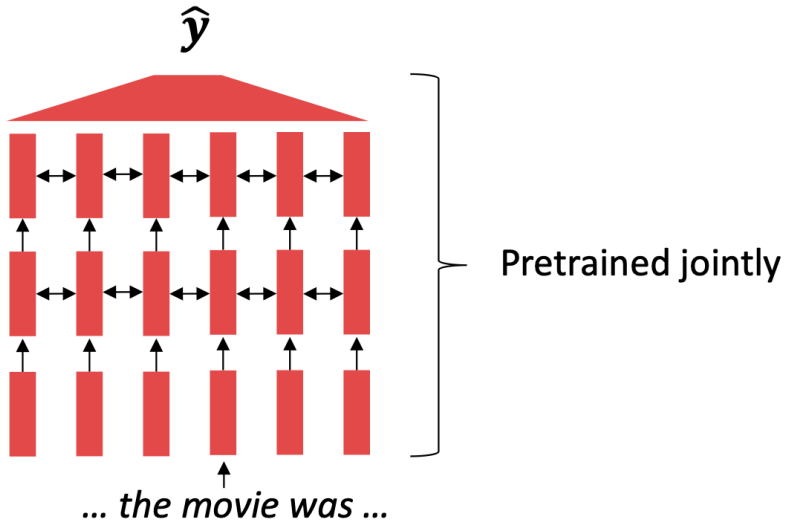


[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

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 then 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.

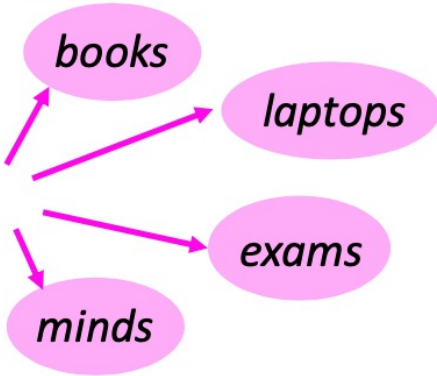


[This model has learned how to represent entire sentences through pretraining]

Language Modeling

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

the students opened their _____



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

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

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

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

Language Modeling

- You can also think of a Language Model as a system that assigns probability to a piece of text

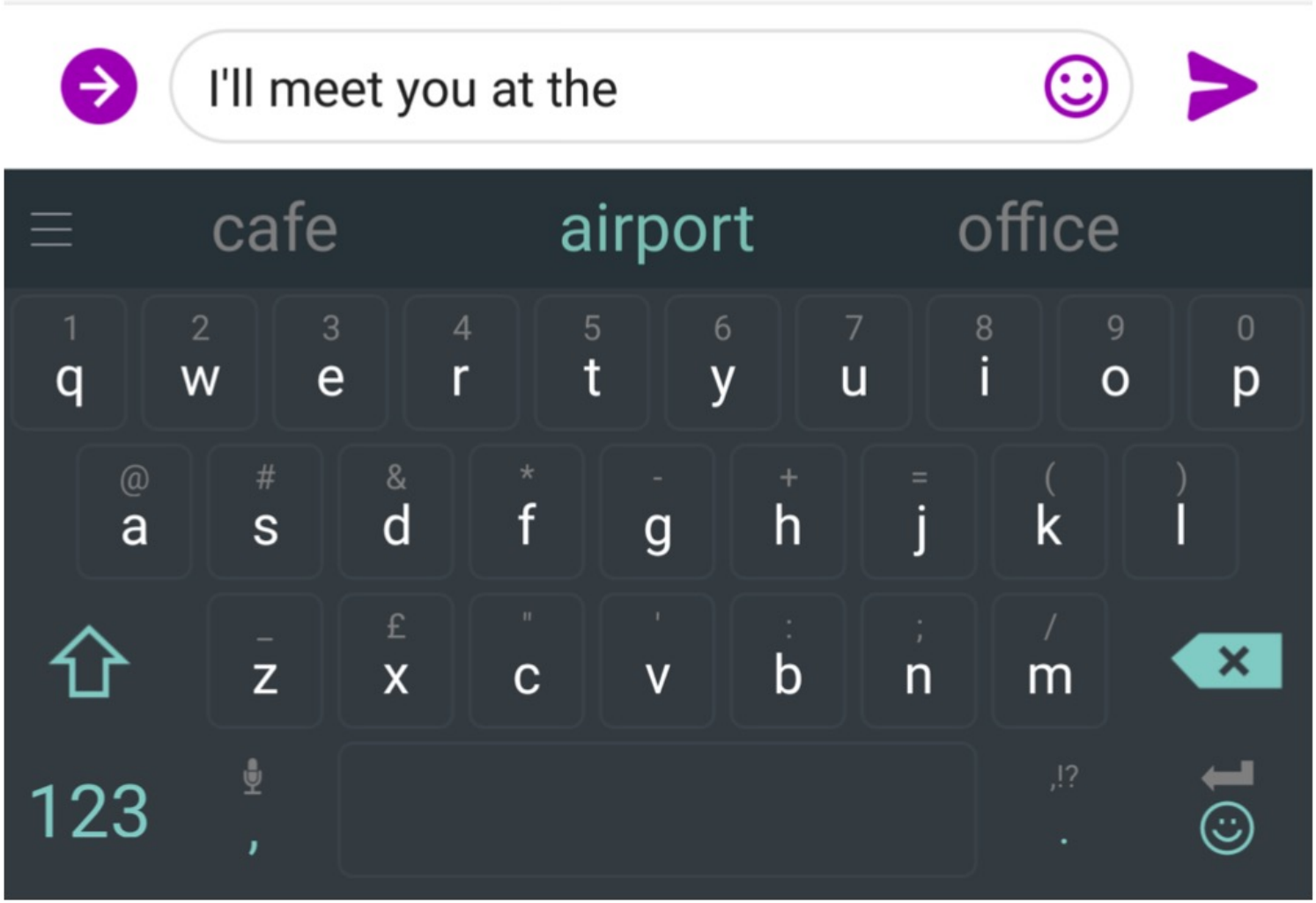
- For example, if we have some text $x^{(1)}, \ldots, x^{(T)}$, then the probability of this text (according to the Language Model) is:

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$

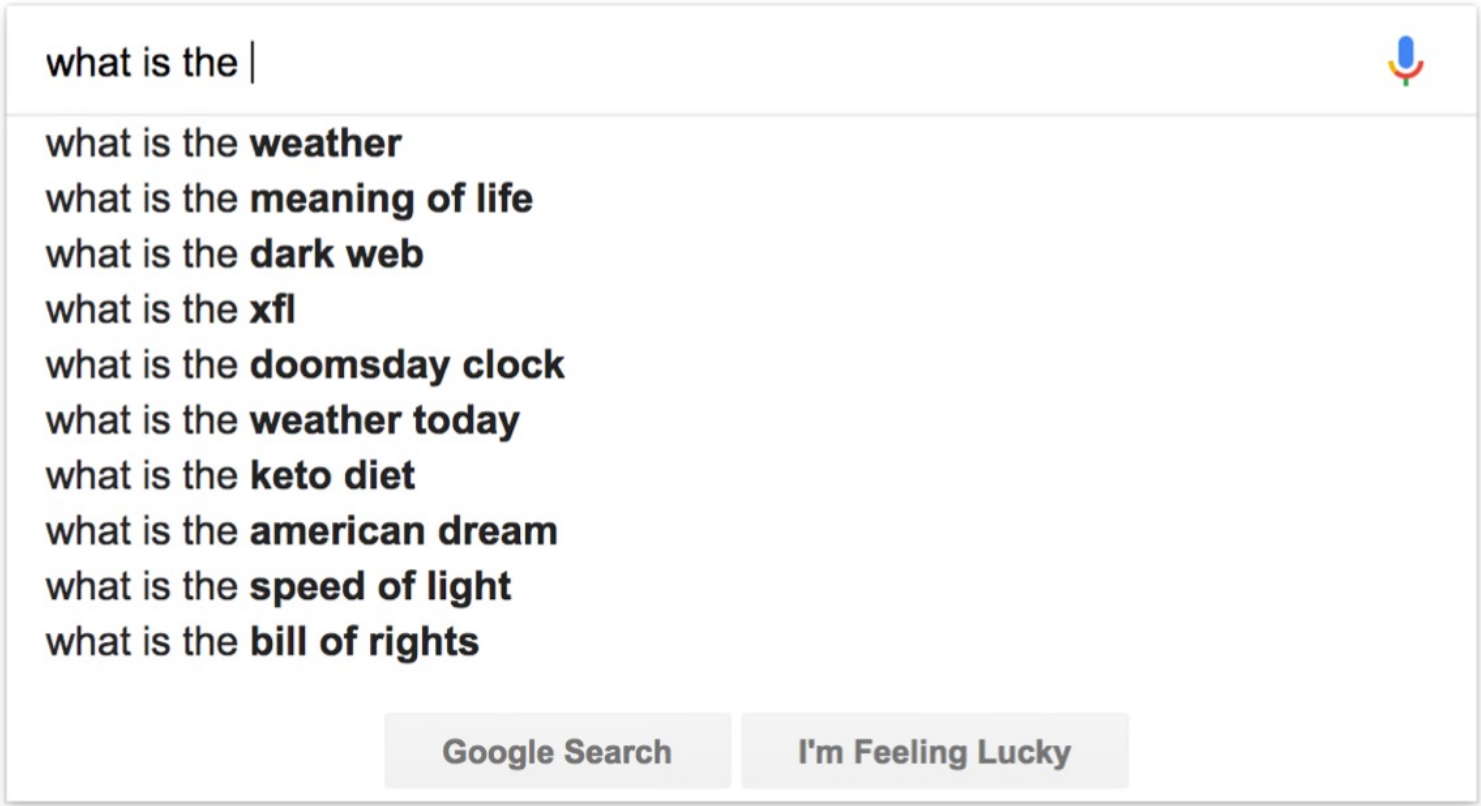
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} | \, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

This is what our LM provides

You use Language Models every day!



You use Language Models every day!



N-gram Language Models

n-gram Language Models

the students opened their _____

- **Question**: How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn an *n*-gram Language Model!

- **Definition:** A *n*-gram is a chunk of *n* consecutive words.
  - unigrams: “the”, “students”, “opened”, ”their”
  - bigrams: “the students”, “students opened”, “opened their”
  - trigrams: “the students opened”, “students opened their”
  - 4-grams: “the students opened their”

- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word.

n-gram Language Models

- First we make a Markov assumption: $x^{(t+1)}$ depends only on the preceding n-1 words

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n-1 \text{ words}}) \quad \text{(assumption)}$$

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(definition of conditional prob)}$$

- **Question:** How do we get these n-gram and (n-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(statistical approximation)}$$

n-gram Language Models: Example

Suppose we are learning a 4-gram Language Model.

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

discard

condition on this

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

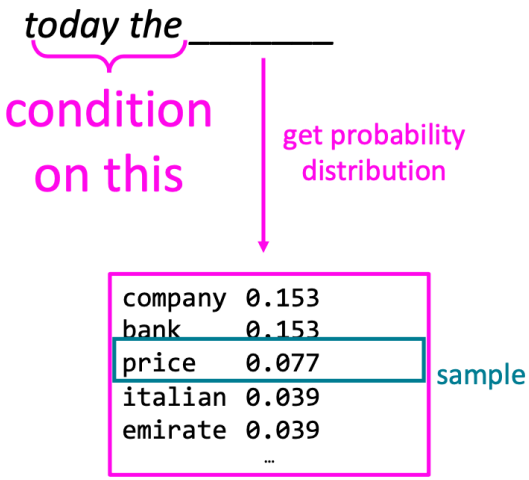
For example, suppose that in the corpus:

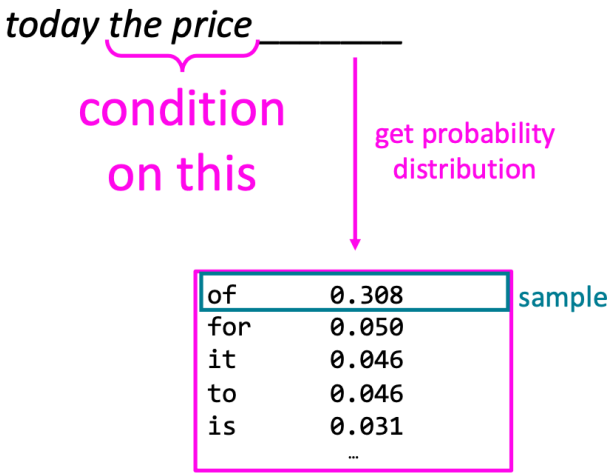
- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

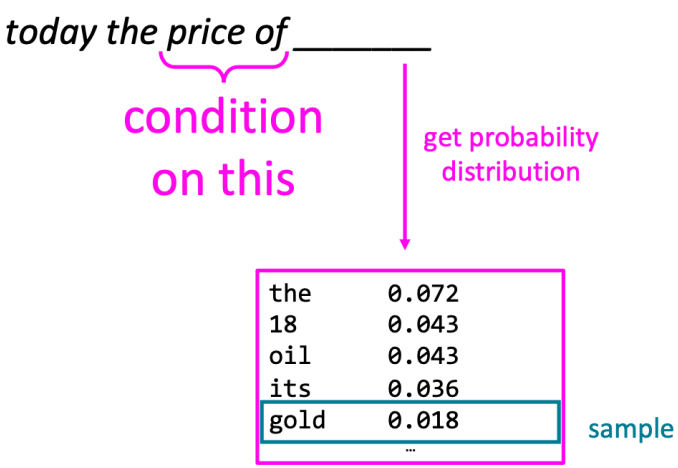
Should we have discarded the "proctor" context?

Generating text with a n-gram Language Model

- **You can also use a Language Model to generate text**







Generating text with a n-gram Language Model

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.

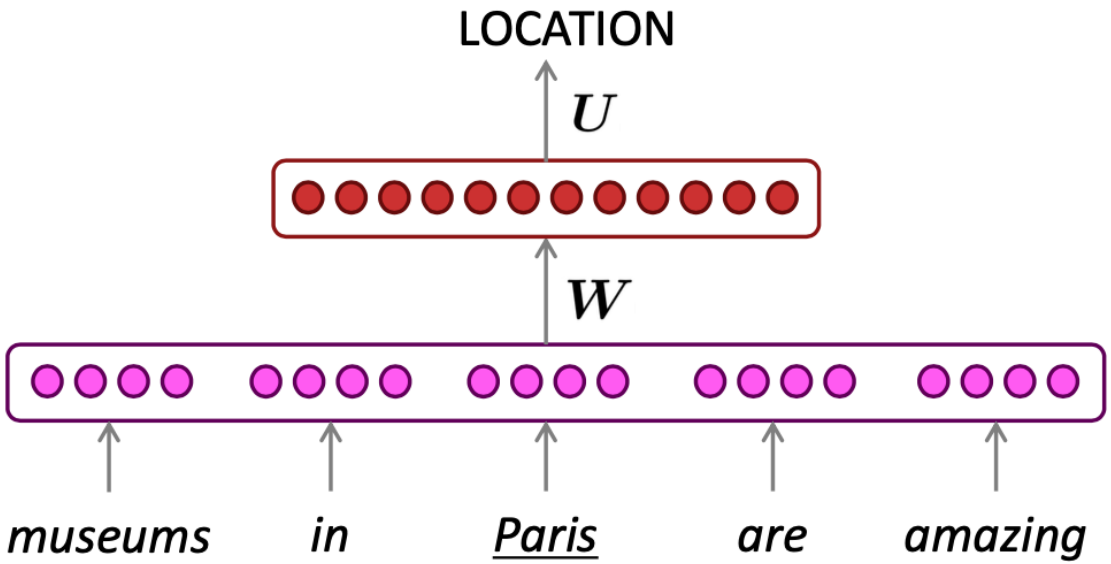
But increasing *n* worsens sparsity problem, and increases model size…

Neural Language Models

How to build a *neural* Language Model?

- Recall the Language Modeling task:
  - Input: sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$
  - Output: prob dist of the next word $P(x^{(t+1)} | x^{(t)}, \ldots, x^{(1)})$

- How about a window-based neural model?



A fixed-window neural Language Model

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

output distribution

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

hidden layer

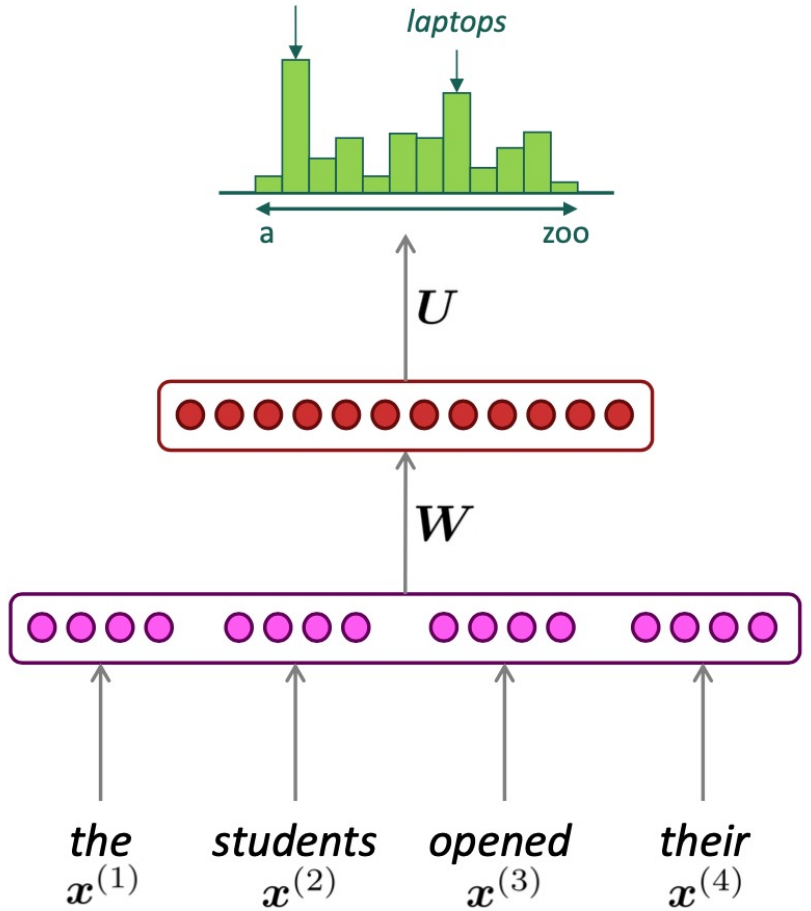
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

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

words / one-hot vectors

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



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

discard

fixed window

A fixed-window neural Language Model

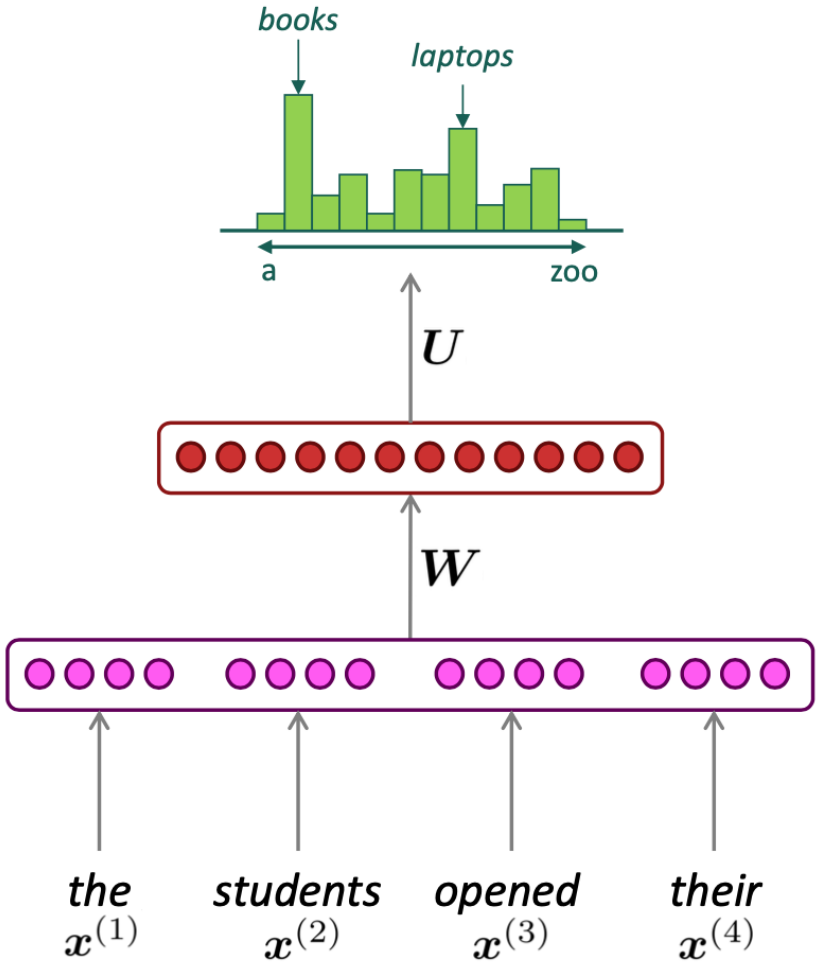
Improvements over *n*-gram LM:

- 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.

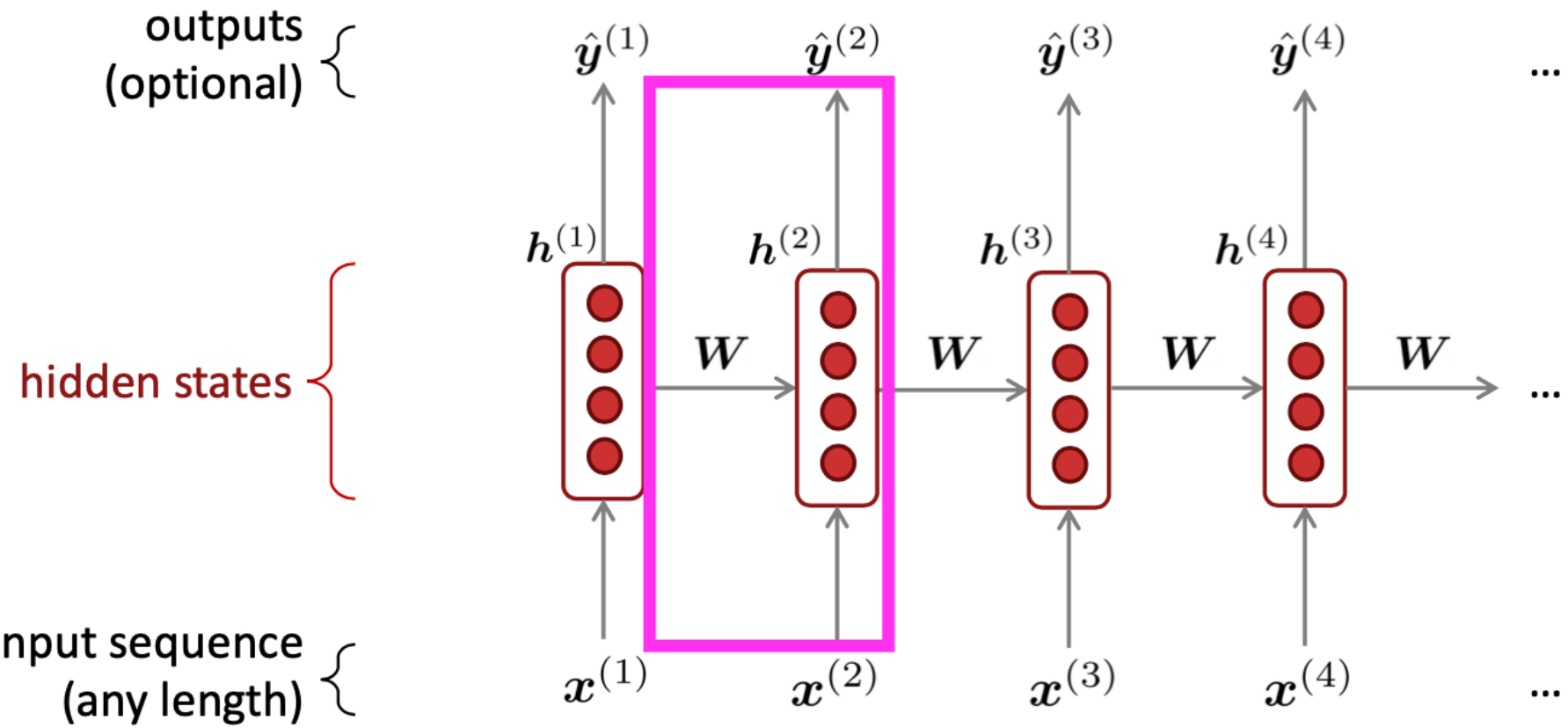
We need a neural architecture that can process *any length input*



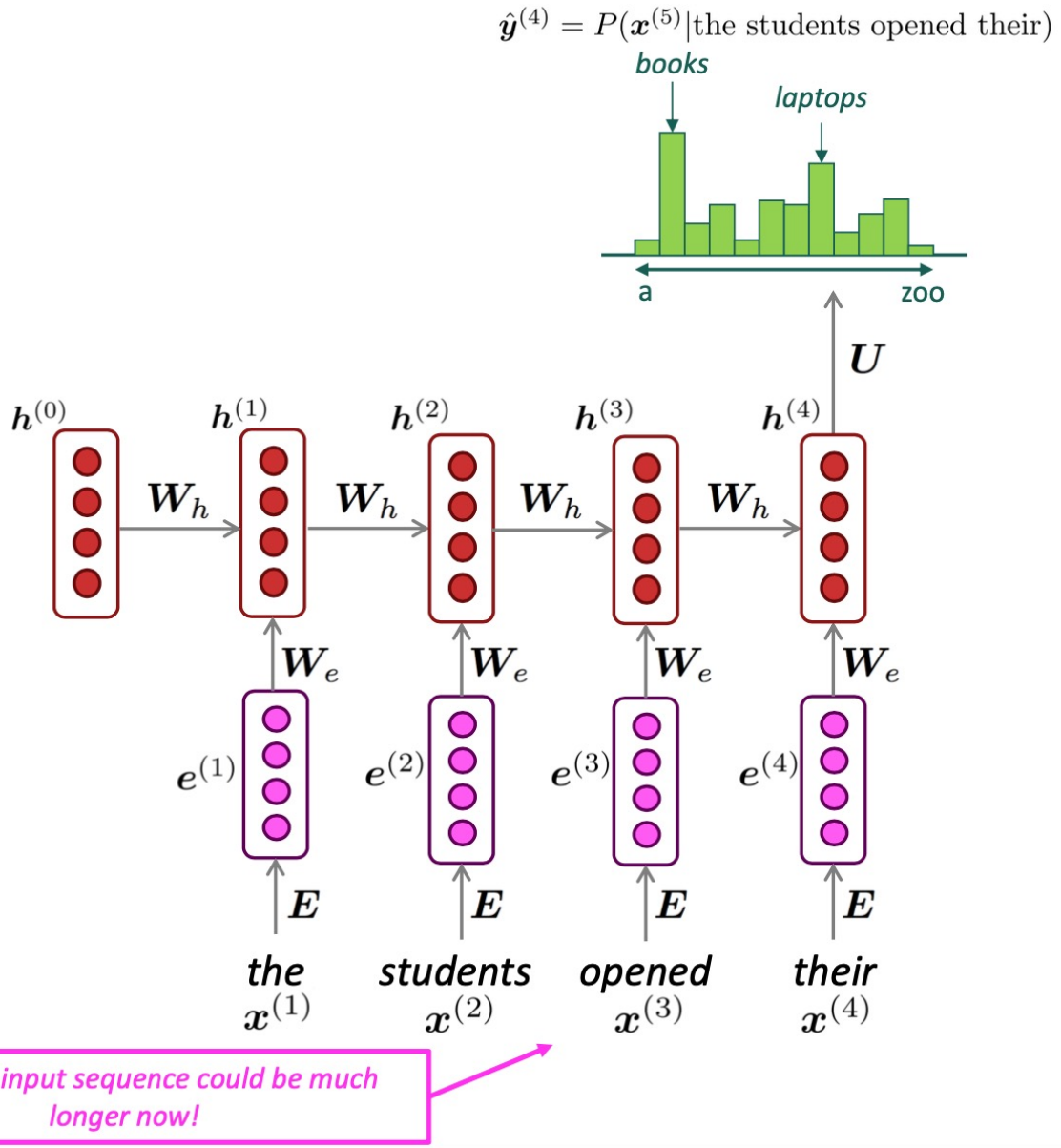
Recurrent Neural Networks (RNN)

A family of neural architectures

Core idea: Apply the same weights *W repeatedly*



A Simple RNN Language Model



output distribution

$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

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

word embeddings

$$e^{(t)} = Ex^{(t)}$$

words / one-hot vectors

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

***Note**: this input sequence could be much longer now!*

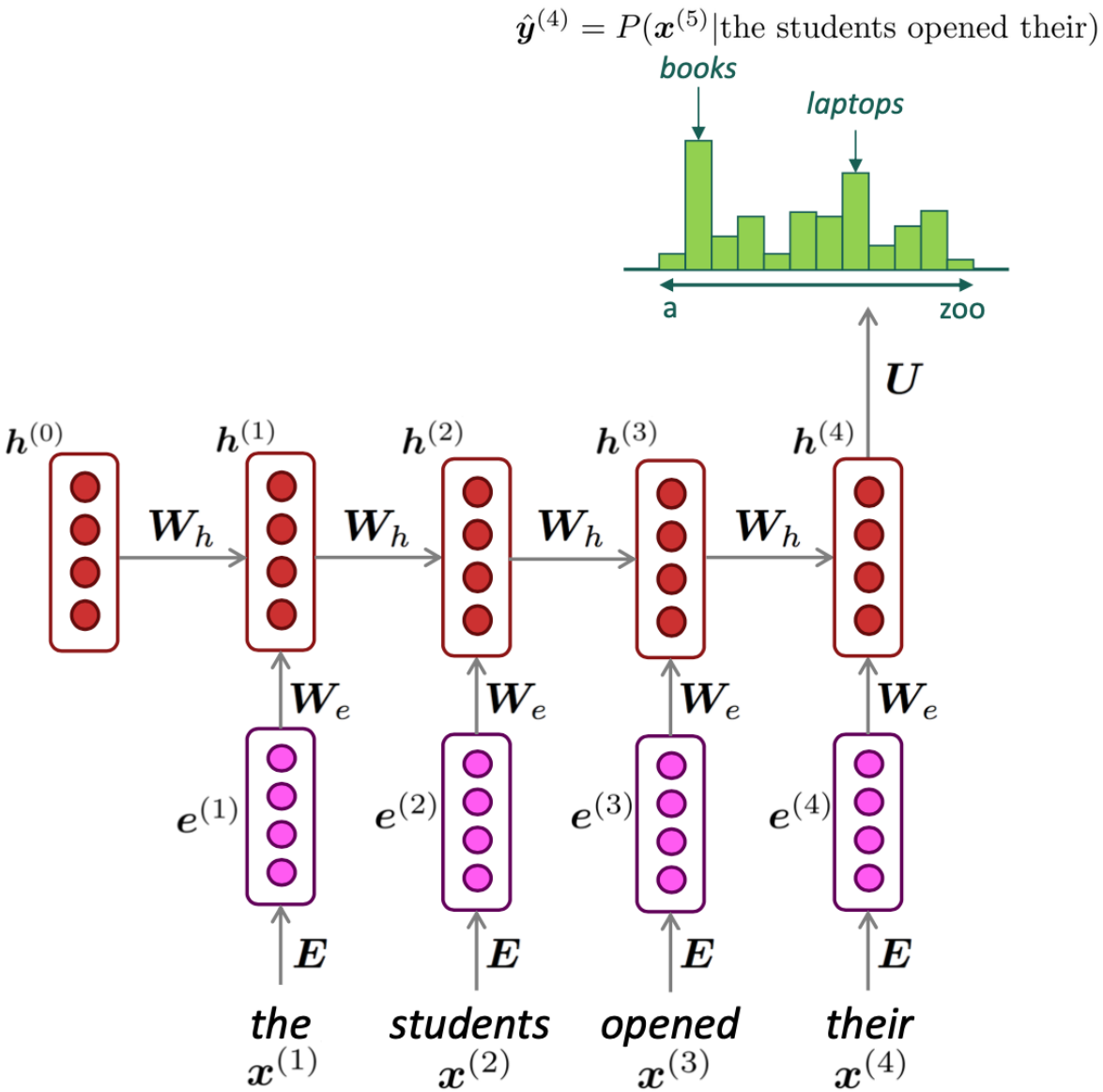
RNN Language Models

RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back



Training an RNN Language Model

- Get a big corpus of text which is a sequence of words $x^{(1)}, \ldots, x^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{y}^{(t)}$ for *every step t*.
  - i.e. predict probability dist of *every word*, given words so far

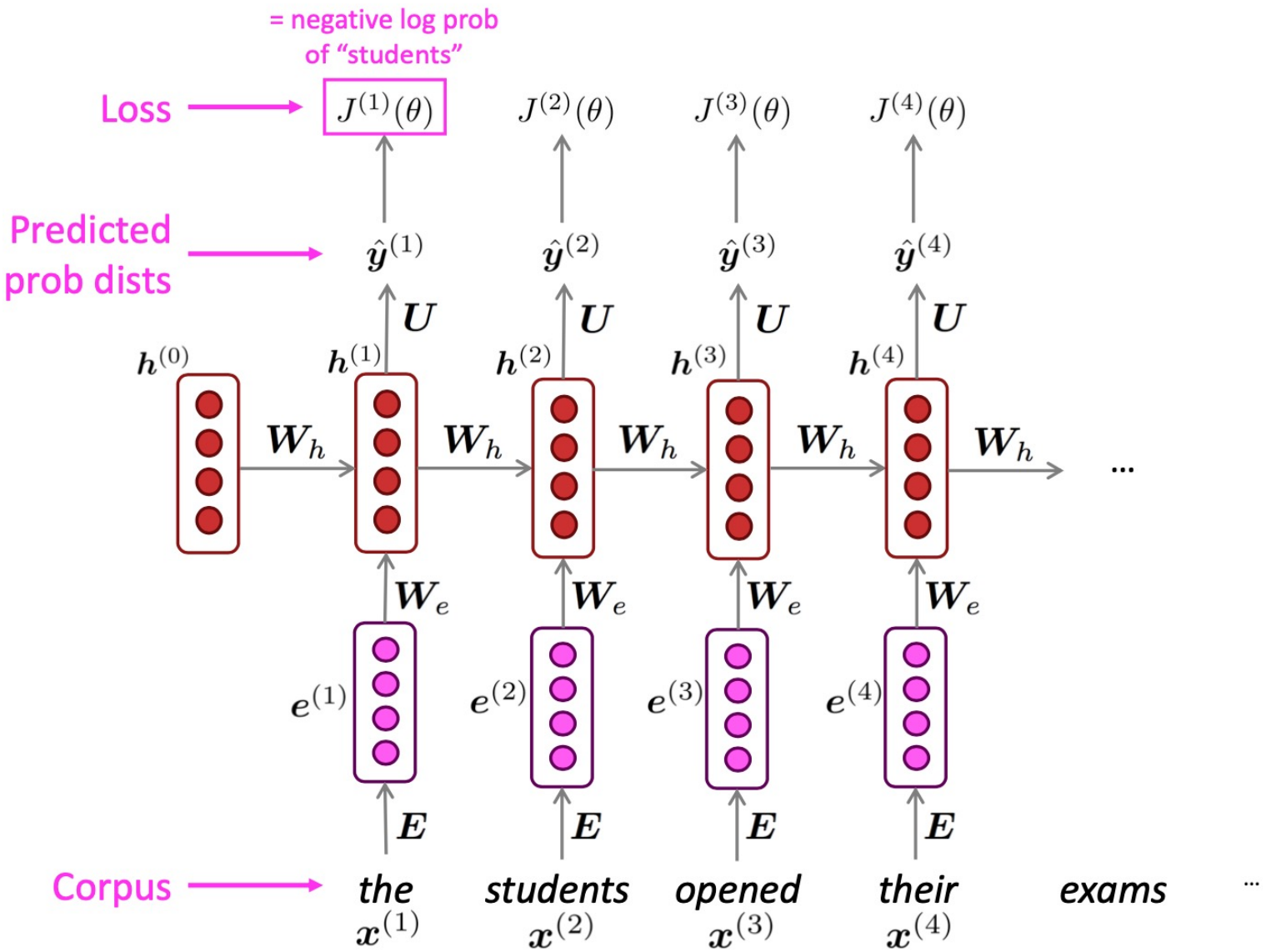
- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{y}^{(t)}$, and the true next word $y^{(t)}$ (one-hot for $x^{(t+1)}$):

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

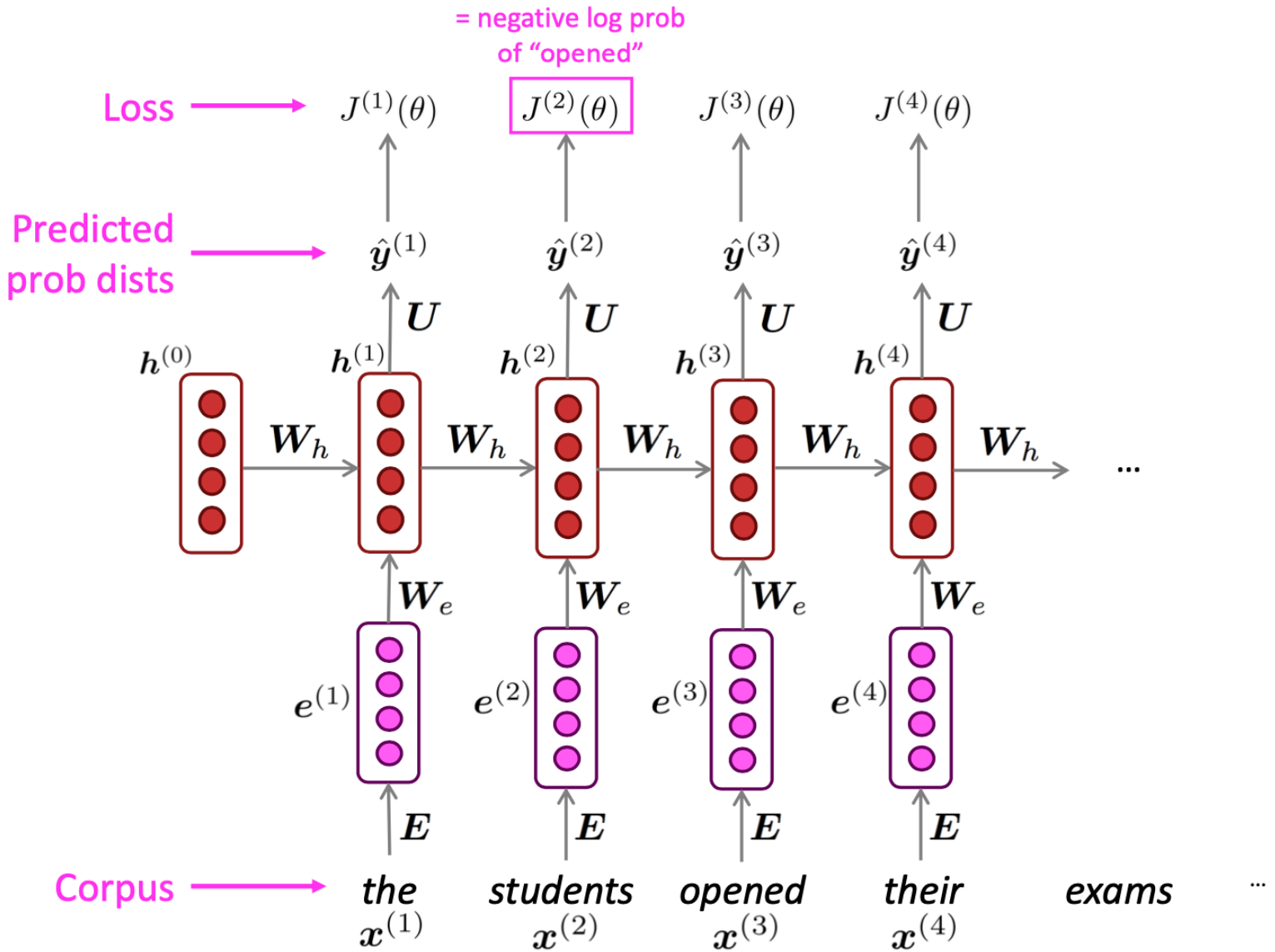
- Average this to get overall loss for entire training set:

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

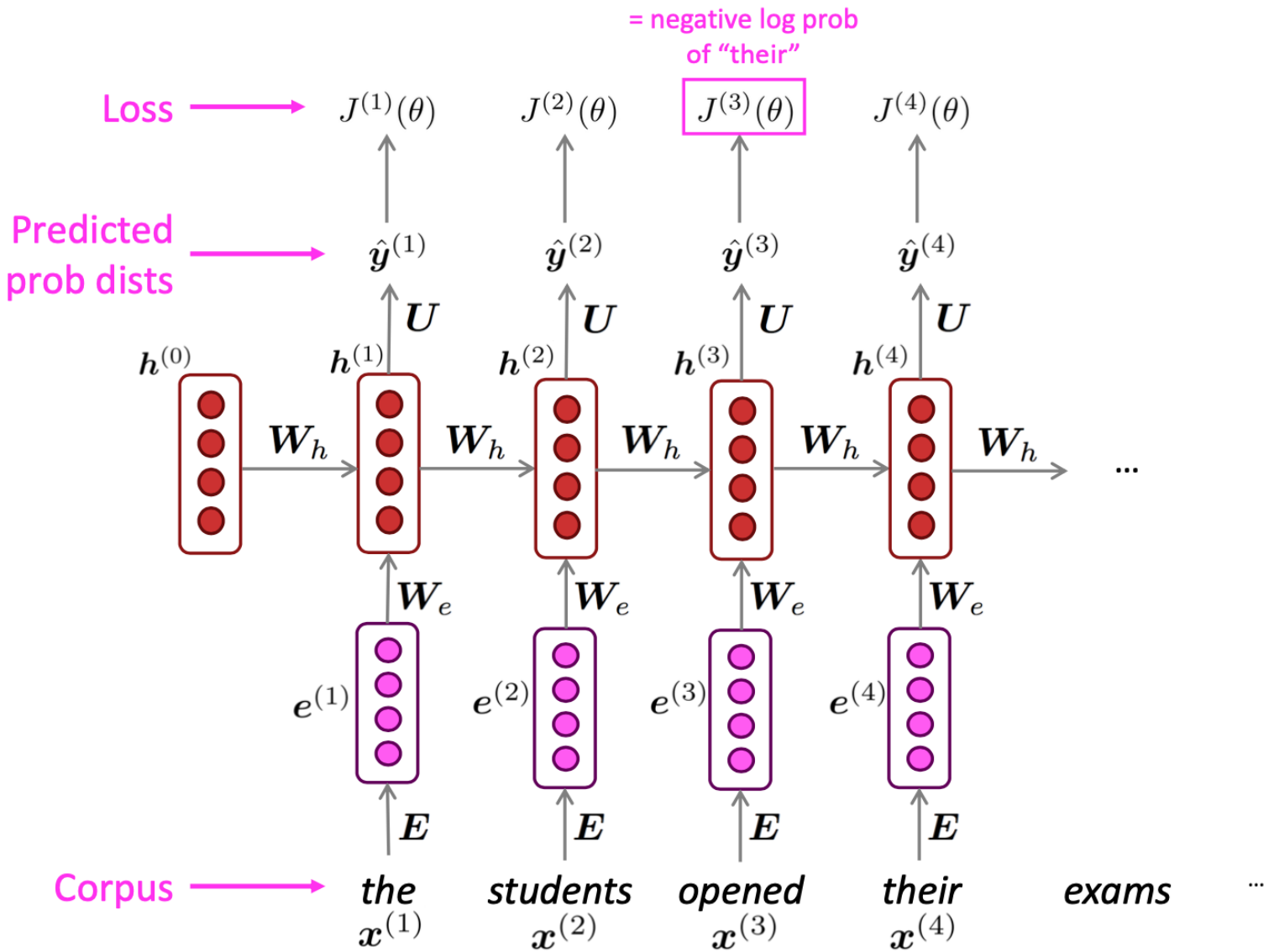
Training an RNN Language Model



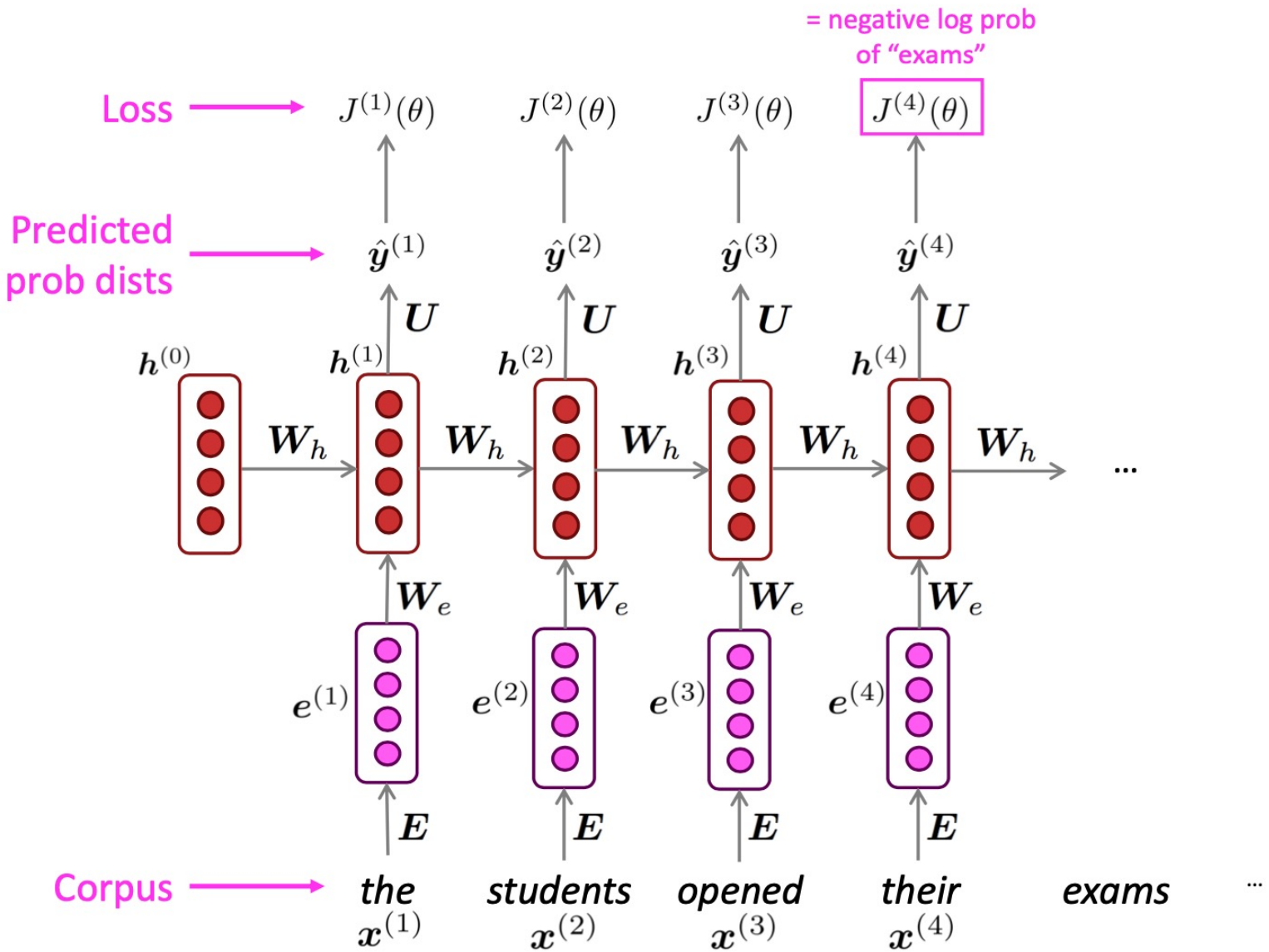
Training an RNN Language Model



Training an RNN Language Model

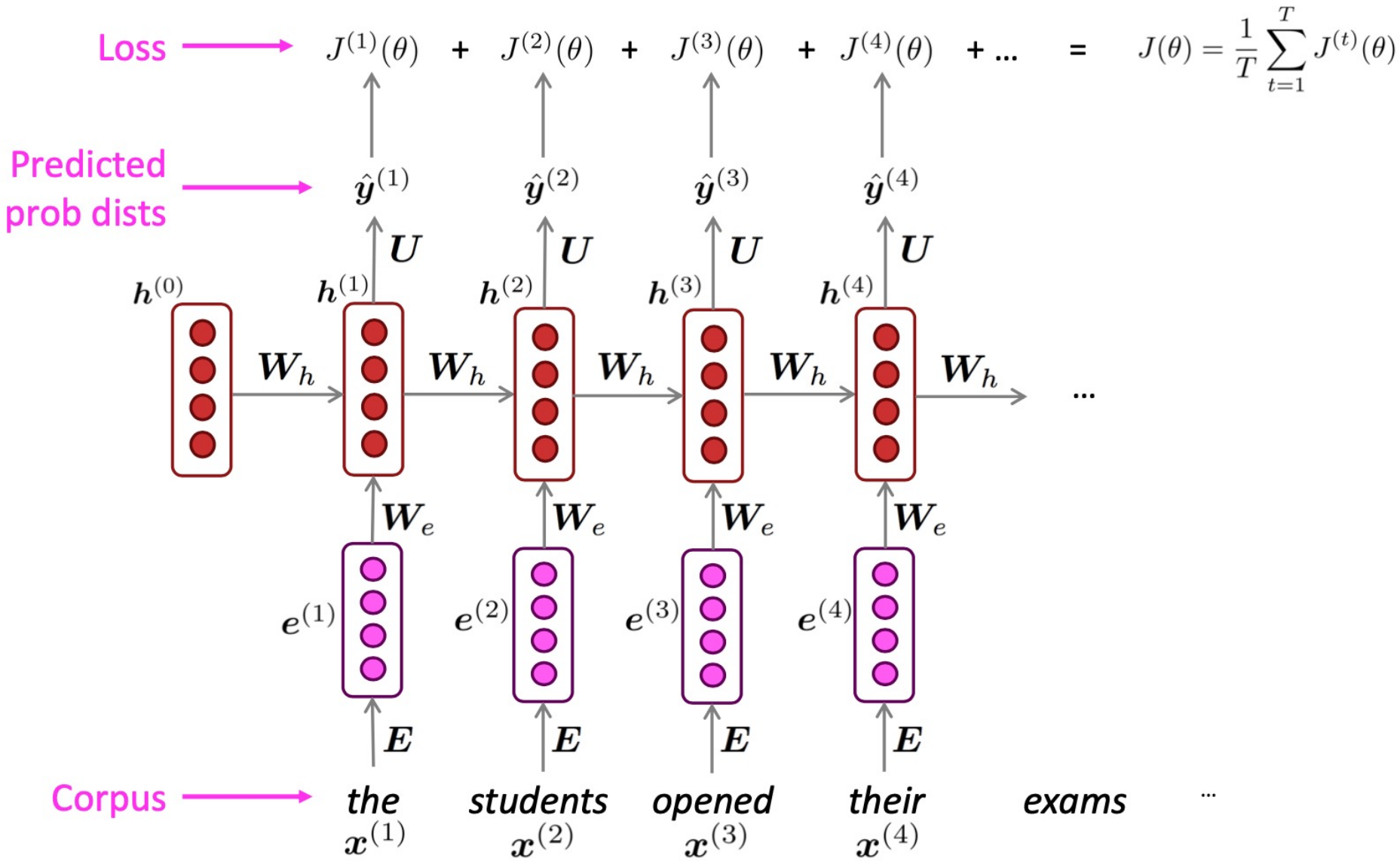


Training an RNN Language Model



Training an RNN Language Model

"Teacher forcing"



Evaluating Language Models

- The standard evaluation metric for Language Models is perplexity.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- 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!

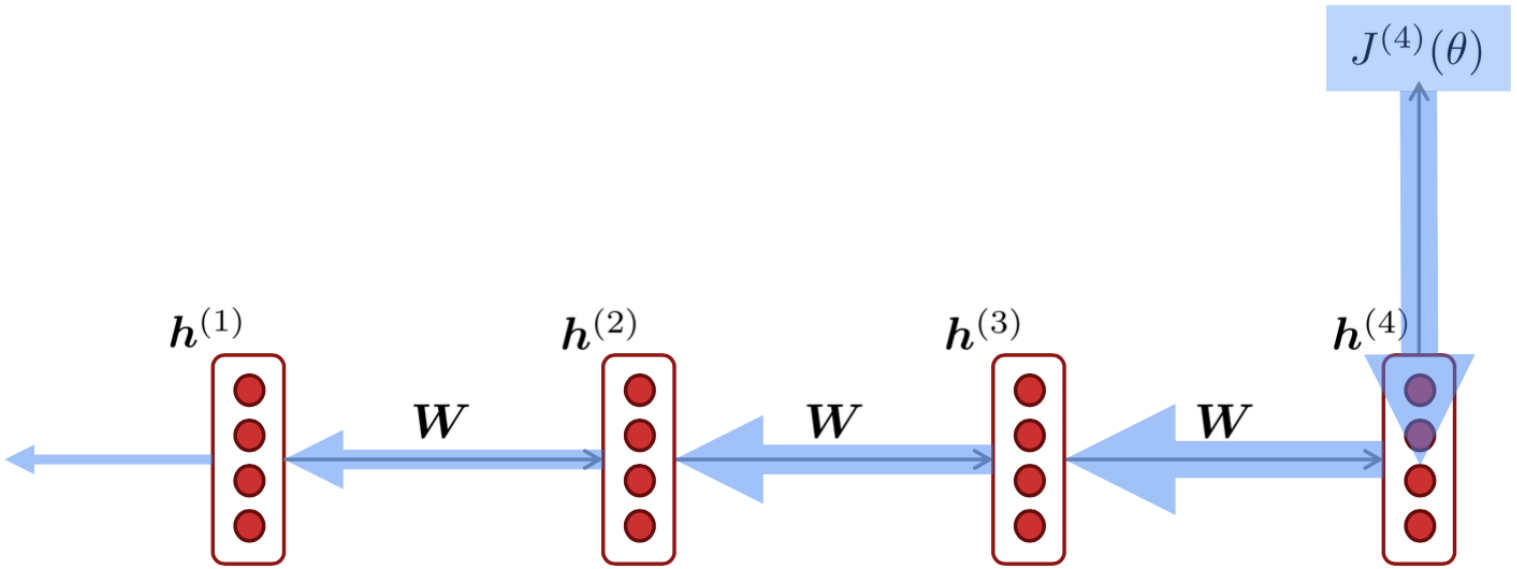
Why should we care about Language Modeling?

- Language Modeling is a benchmark task that helps us measure our progress on understanding language

- Language Modeling is a subcomponent of many NLP tasks, especially those involving generating text or estimating the probability of text:
  - Predictive typing
  - Speech recognition
  - Handwriting recognition
  - Spelling/grammar correction
  - Authorship identification
  - Machine translation
  - Summarization
  - Dialogue
  - etc.

Long Short-Term Memory (LSTM)

Vanishing gradient intuition

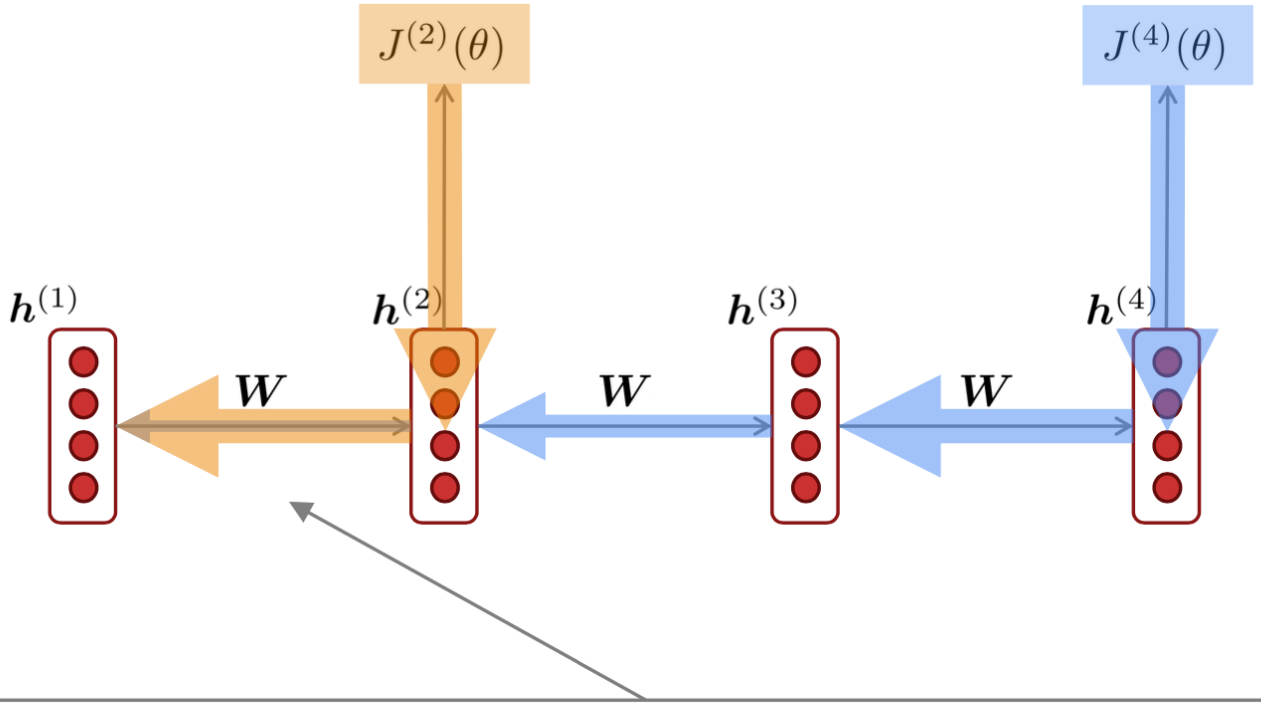


$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Why is vanishing gradient a problem?



Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are updated only with respect to near effects, not long-term effects.

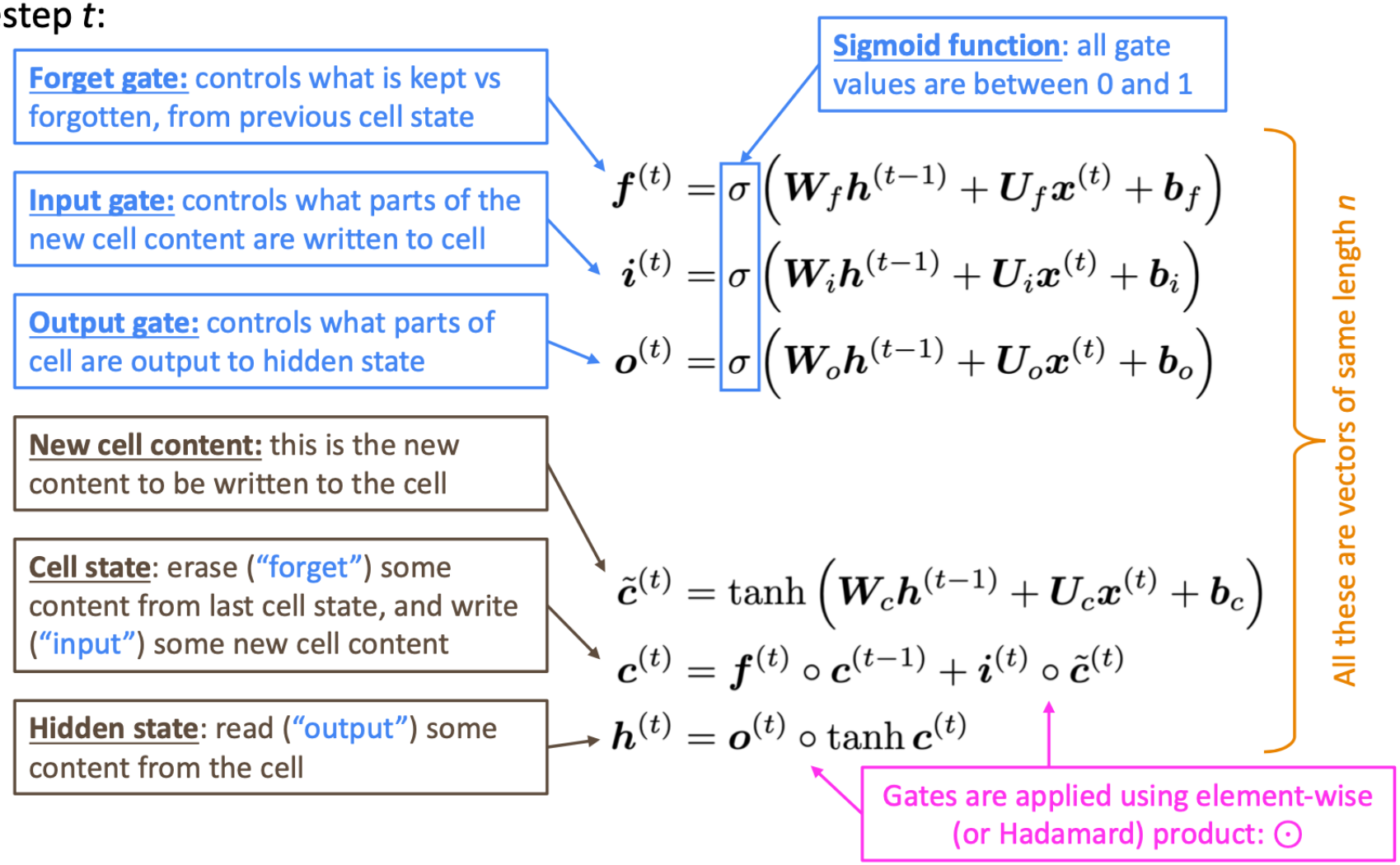
Long Short-Term Memory RNNs (LSTMs)

- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem.
  - Everyone cites that paper but really a crucial part of the modern LSTM is from Gers et al. (2000) 💜
- On step *t*, there is a hidden state $\boldsymbol{h}^{(t)}$ *and* a cell state $\boldsymbol{c}^{(t)}$
  - Both are vectors length *n*
  - The cell stores long-term information
  - The LSTM can read, erase, and write information from the cell
    - The cell becomes conceptually rather like RAM in a computer
- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors length *n*
  - On each timestep, each element of the gates can be open (1), closed (0), or somewhere in-between
  - The gates are dynamic: their value is computed based on the current context

"Long short-term memory", Hochreiter and Schmidhuber, 1997. https://www.bioinf.jku.at/publications/older/2604.pdf
"Learning to Forget: Continual Prediction with LSTM", Gers, Schmidhuber, and Cummins, 2000. https://dl.acm.org/doi/10.1162/089976600300015015

Long Short-Term Memory (LSTM)

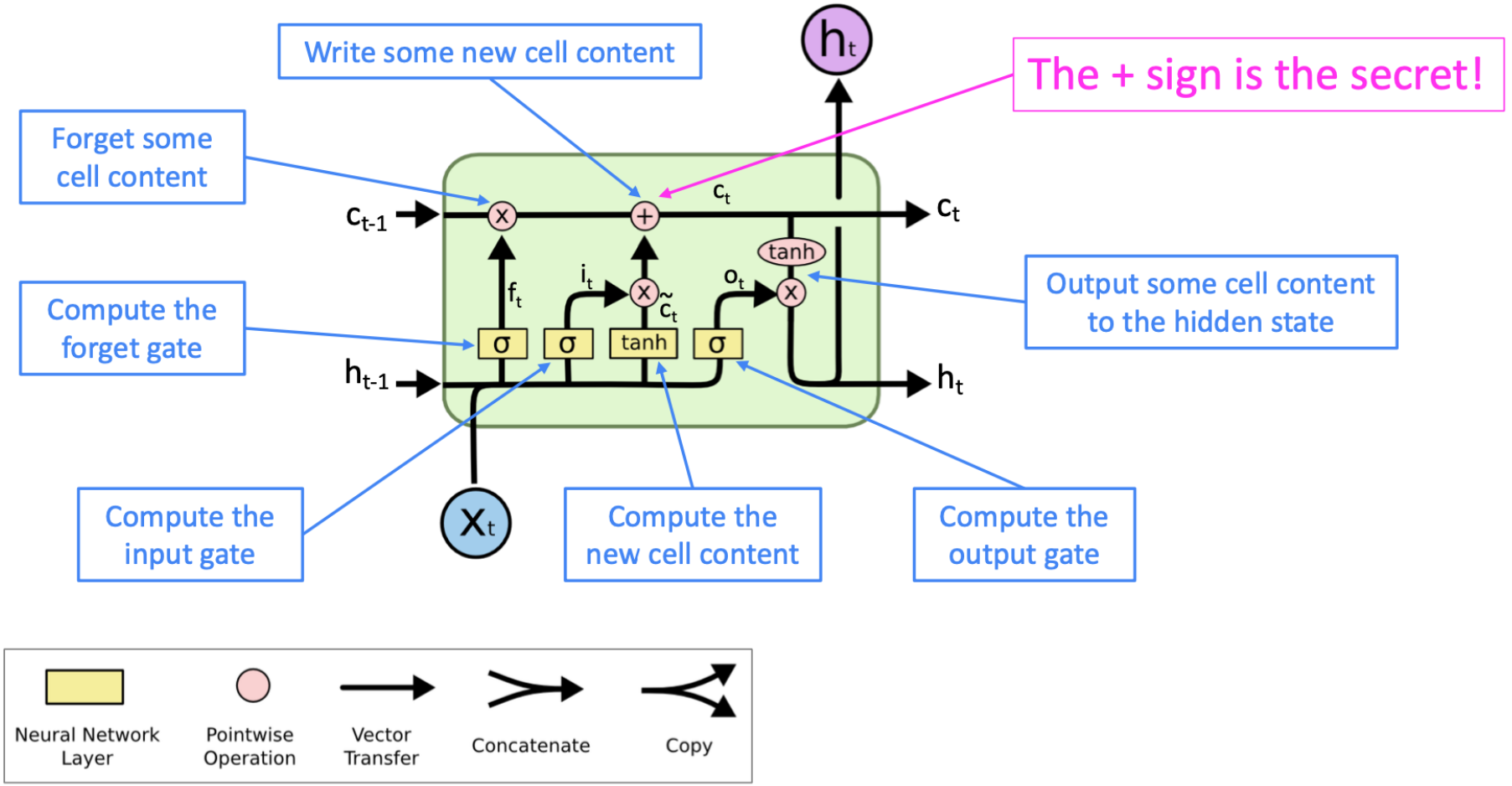
We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t:



Long Short-Term Memory (LSTM)

- **Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/**

You can think of the LSTM equations visually like this:



How does LSTM solve vanishing gradients?

- The LSTM architecture makes it easier for the RNN to preserve information over many timesteps
  - e.g., if the forget gate is set to 1 for a cell dimension and the input gate set to 0, then the information of that cell is preserved indefinitely.
  - In contrast, it's harder for a vanilla RNN to learn a recurrent weight matrix W_h that preserves info in the hidden state
  - In practice, you get about 100 timesteps rather than about 7

- LSTM doesn't *guarantee* that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies

Is vanishing/exploding gradient just a RNN problem?

- No! It can be a problem for all neural architectures (including feed-forward and convolutional), especially very deep ones.
  - Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small as it backpropagates
  - Thus, lower layers are learned very slowly (hard to train)
- Solution: lots of new deep feedforward/convolutional architectures that add more direct connections (thus allowing the gradient to flow)

For example:

- Residual connections aka "ResNet"
- Also known as skip-connections
- The identity connection preserves information by default
- This makes deep networks much easier to train

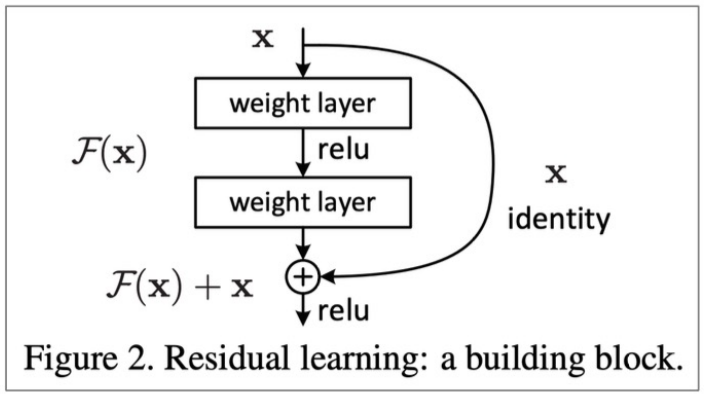
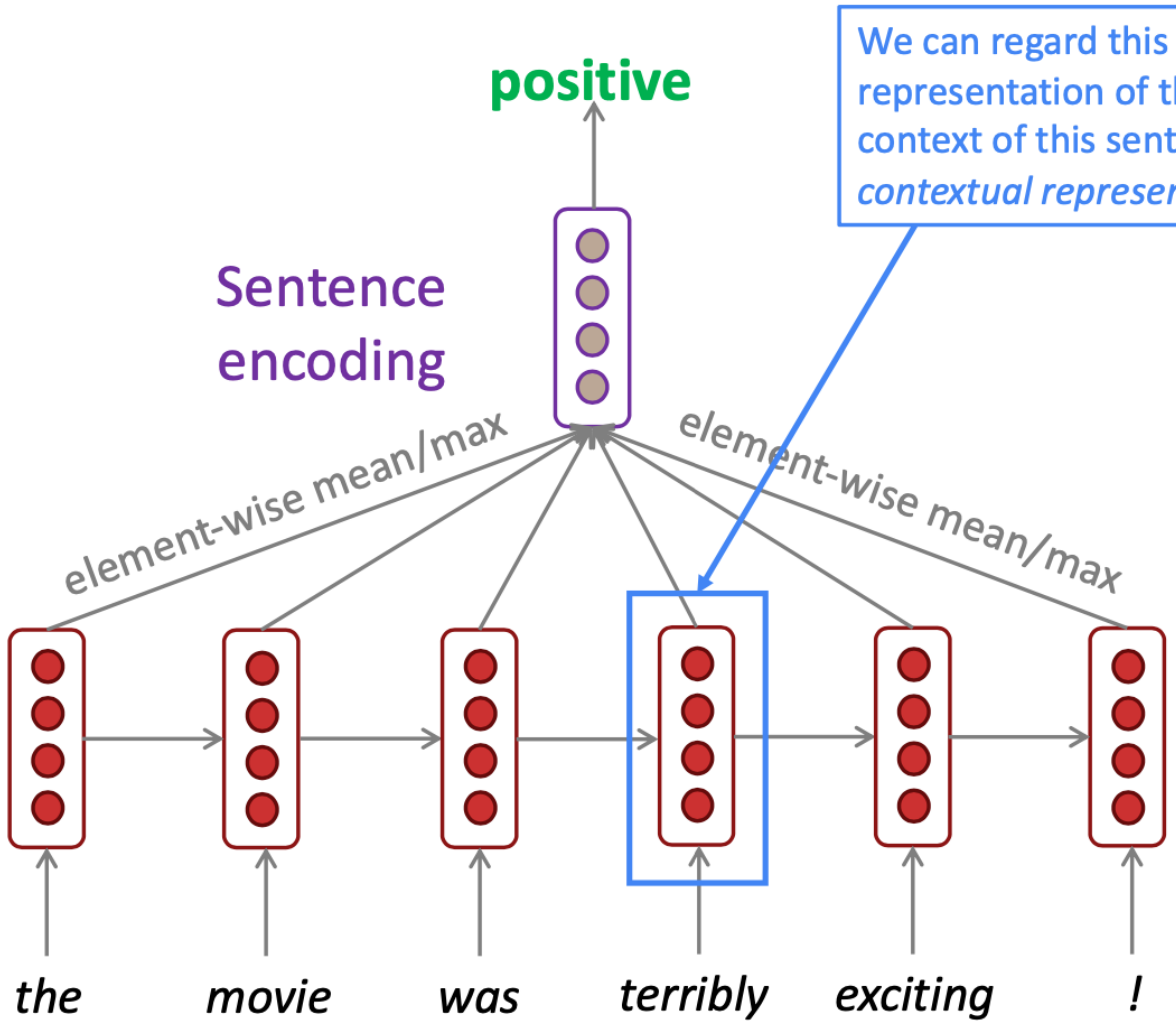


Figure 2. Residual learning: a building block.

Other RNN Variations

Bidirectional and Multi-layer RNNs: motivation

Task: Sentiment Classification



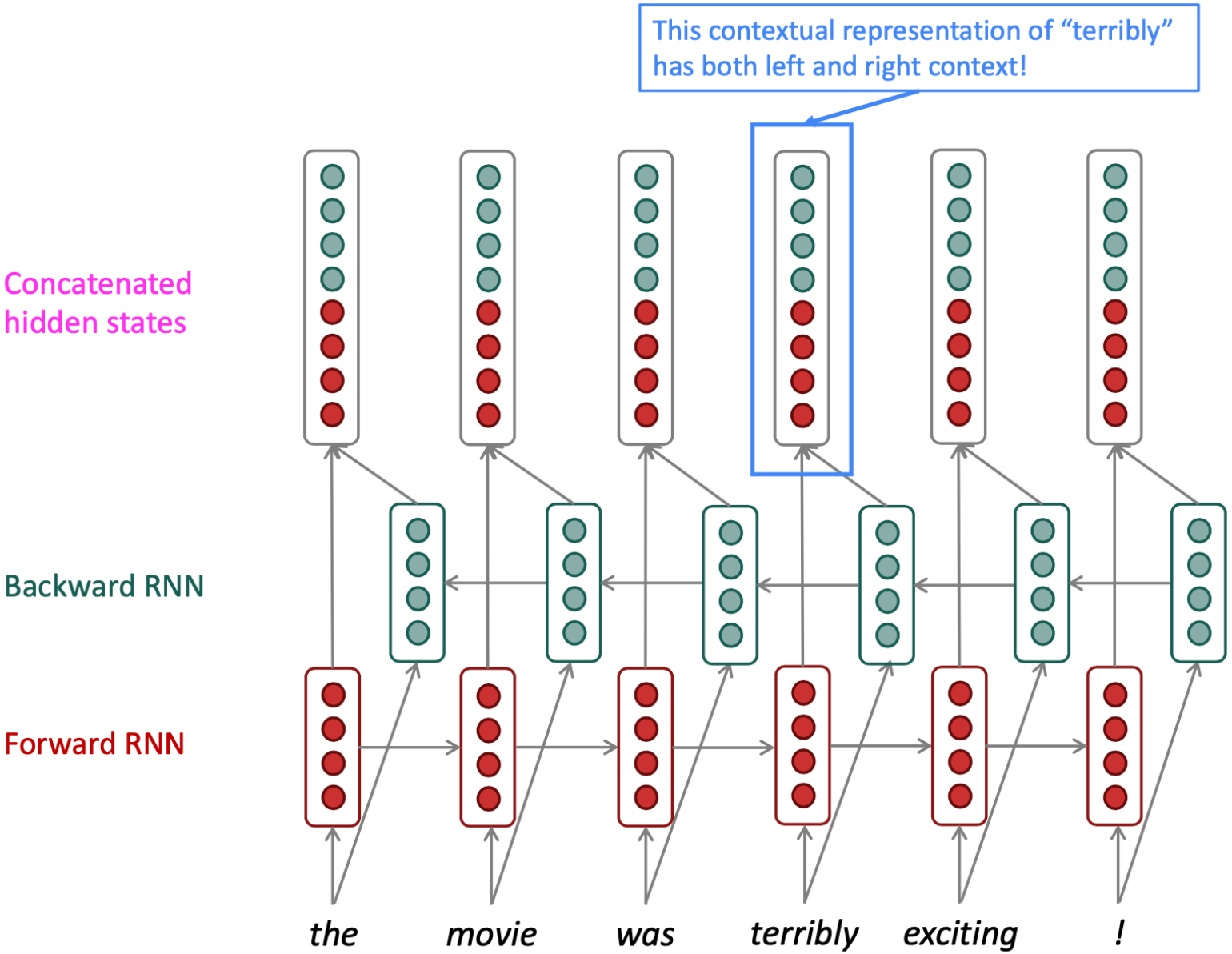
We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation.*

These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)

Bidirectional RNNs



Bidirectional RNNs

- Note: bidirectional RNNs are only applicable if you have access to the entire input sequence
  - They are **not** applicable to Language Modeling, because in LM you *only* have left context available.

- If you do have entire input sequence (e.g., any kind of encoding), bidirectionality is powerful (you should use it by default).

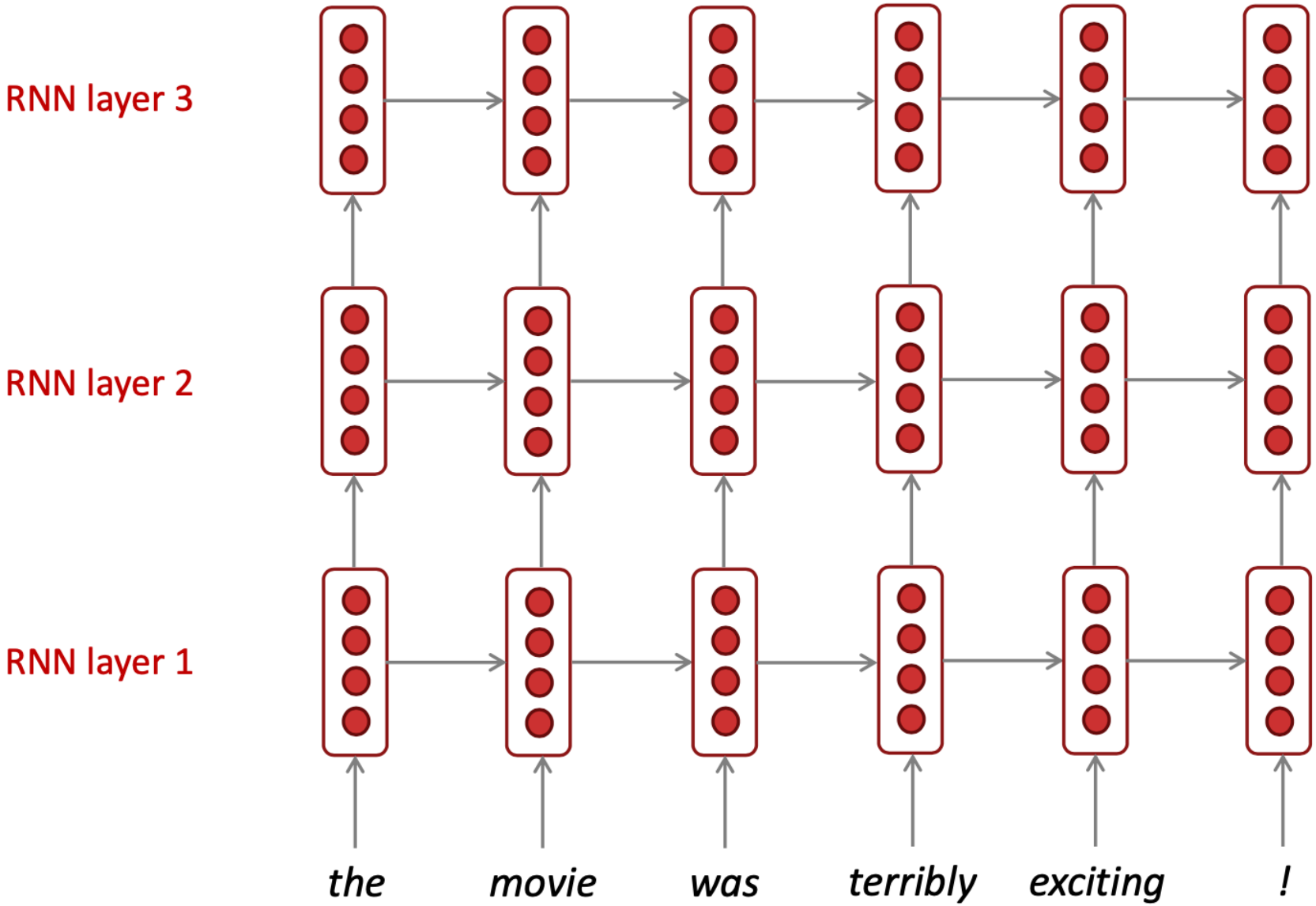
- For example, BERT (**Bidirectional** Encoder Representations from Transformers) is a powerful pretrained contextual representation system built on bidirectionality.
  - You will learn more about transformers include BERT in a couple of weeks!

Multi-layer RNNs

- RNNs are already "deep" on one dimension (they unroll over many timesteps)
- We can also make them "deep" in another dimension by applying multiple RNNs – this is a multi-layer RNN.
- This allows the network to compute more complex representations
  - The lower RNNs should compute lower-level features and the higher RNNs should compute higher-level features.
- Multi-layer RNNs are also called *stacked RNNs*.

Multi-layer RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1



Summary

- **Language Modeling**
- **Language Models**
  - N-gram language models
  - Neural langauge models
- **Recurrent Neural network**
  - Basic form of RNN
  - Long Short-Term Memory (LSTM)
  - Other RNN Variations

Next Lecture

- **Translation, Seq2seq, Attention**
- **Self-Attention and Transformers**

Q & A