Deep Learning for IR

- SI650 / EECS549 Information Retrieval
- October 15, 2025

Slides adapted from David Juergen

2023 © *University of Michigan*

Announcements

- HW3 (focused on deep learning) is released
- CRLT results were shared with me last night. I will share insights and our plans next week
- Proposals will be graded soon

2023 © *University of Michigan*

Today's goals

- Exposure to deep learning methods
  - Text-based
  - Their connection to IR

2023 © *University of Michigan*

Query: “natural landscape”

- doc1: “a sandy desert scene with a blue sky”
- doc2: “neural network training loss landscape”

Problem: Our query-document comparison doesn’t recognize similarities between word meanings

2023 © *University of Michigan*

Let's recall Pivoted Normalization…

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{1 + \ln(1 + \ln(c(w,d)))}{1 - b + b\frac{|d|}{avg\_dl}} \ln \frac{N+1}{df(w)}, b \in [0,1]$$

Why do we only care about terms in both Q and D?

What if we modified Pivoted Normalization so that it took word similarity into account?

$$f(q,d) = \sum_{w_1 \in q} \sum_{w_2 \in d}$$

If similarity(w_1,w_2)=1 only when w_1 and w_2 are the same word (and 0 otherwise) this is the same as Pivoted Normalization

2023 © University of Michigan

How might we estimate the similarity of words?

$$f(q,d) = \sum_{w_1 \in q} \sum_{w_2 \in d} similarity(w_1, w_2) \left[c(w_1, q) \frac{1 + \ln(1 + \ln(c(w_2, d)))}{1 - b + b \frac{|d|}{avg\_dl}} \ln \frac{N+1}{df(w_2)}, b \in [0,1] \right]$$

- We could use a thesaurus!
  - Problem: Limited coverage
- Maybe look at word-occurrences, e.g., PMI?
  - Problem: Huge space requirements
- What if learned representations for each word that are comparable?

2023 © University of Michigan

Word Vectors

2023 © *University of Michigan*

Word Representations

- Classical representations
  - Bag of Words representation

- Problem: data sparsity

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

2023 © *University of Michigan*

Neural word embeddings

- Neural word embedding (Bengio et al. 2003)
  - All the words are represented in a low-dimensional space
  - Similar words are located closely to each other

2023 © *University of Michigan*

Background: The distributional hypothesis

- "You shall know a word by its company"- J.R. Firth 1957:11

government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

2023 © *University of Michigan*

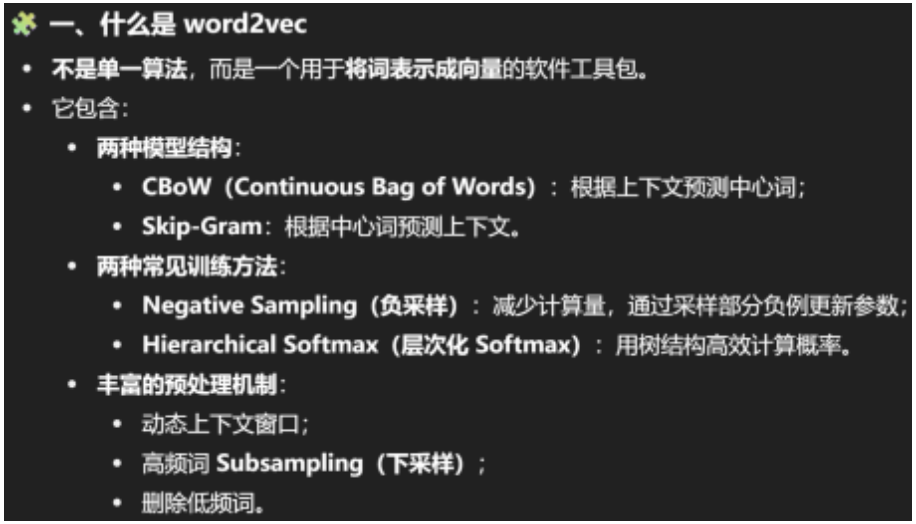
word2vec: a library for learning dense word vectors

- Breakthrough in early 2010s
- Fast approach to learn dense vector representations of words where vector similarity ≈ semantic similarity
  - dense in contrast to sparse bag of words
- Many downstream applications used (and still use) these vectors

2023 © *University of Michigan*

What is word2vec? （一种特征生成工具）

- word2vec is not a single algorithm
- It is a software package for representing words as vectors, containing:
  - Two distinct models
    - CBoW
    - Skip-Gram
  - Various training methods
    - Negative Sampling
    - Hierarchical Softmax
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words



2023 © University of Michigan

General idea behind word2vec

- Model this probability (sort of):
  - p(context-word | target-word)
  - what words are more likely to show up nearby any given word?
- Use a simple neural network to do the prediction task
  - Each word has its own weights in the network
  - The weights become the dense vector
- **Intuition**: if words show up in very similar contexts, they'll need similar parameters to predict the same context words

2023 © *University of Michigan*

🧠 二、word2vec 的基本思想

- 它要建模一个概率：

$$p(\text{context word} \mid \text{target word})$$

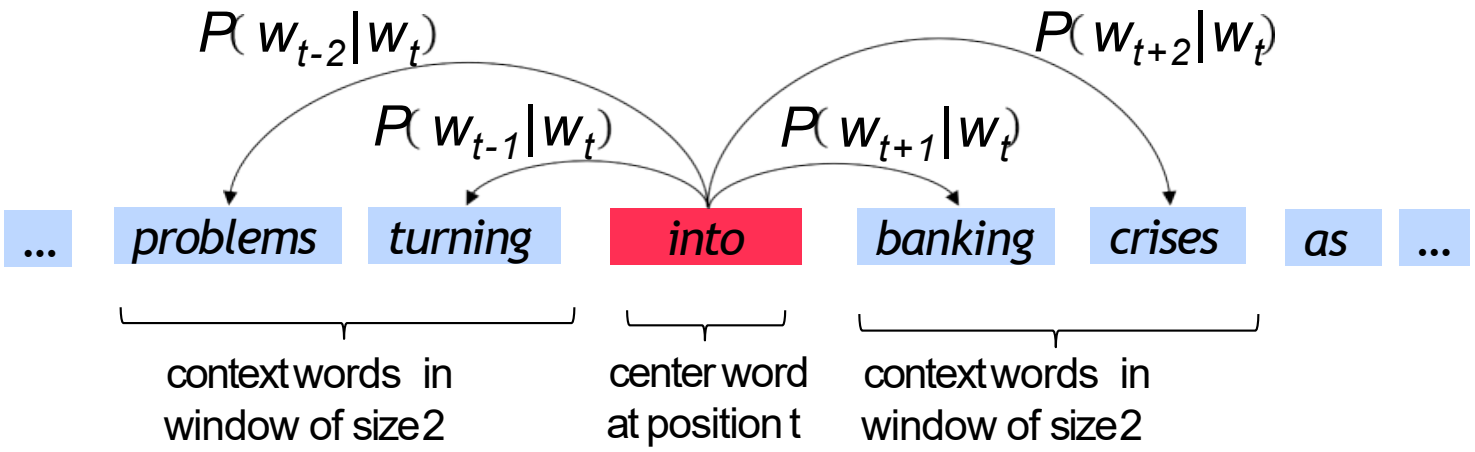
  即：给定一个词，**它周围哪些词更可能出现?**

- 实现方式：
  - 用一个**简单的神经网络**预测上下文；
  - 每个词在网络中都有自己的一组权重；
  - 这些权重经过训练后就成为**词向量（dense vector）**。

2023 © *University of Michigan*

word2vec overview

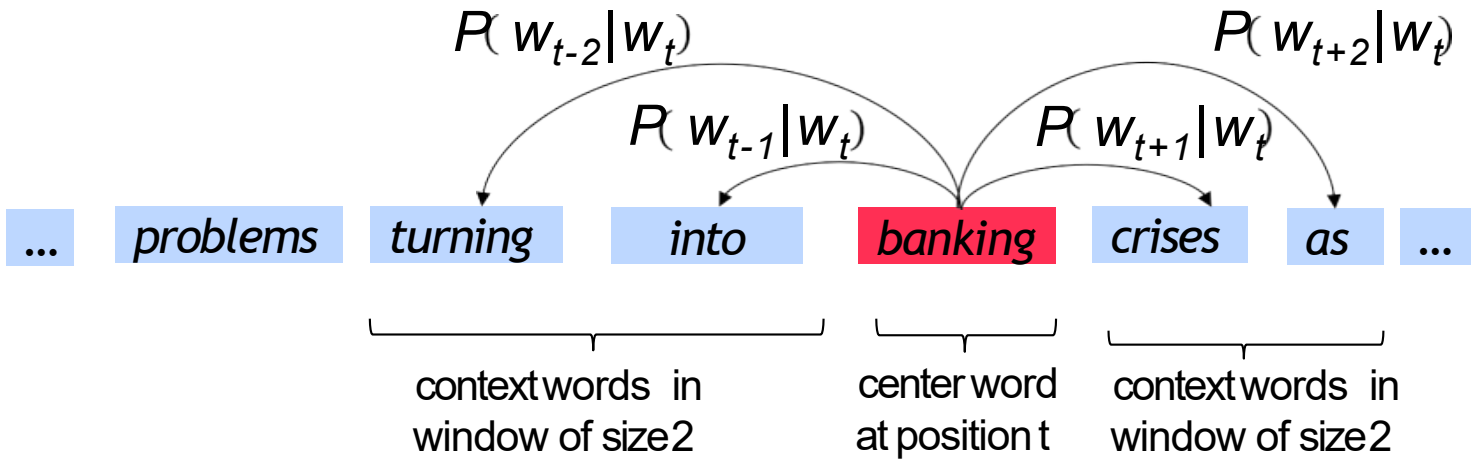
- Example windows and process for computing $P(w_{t+j}|w_t)$



2023 © University of Michigan

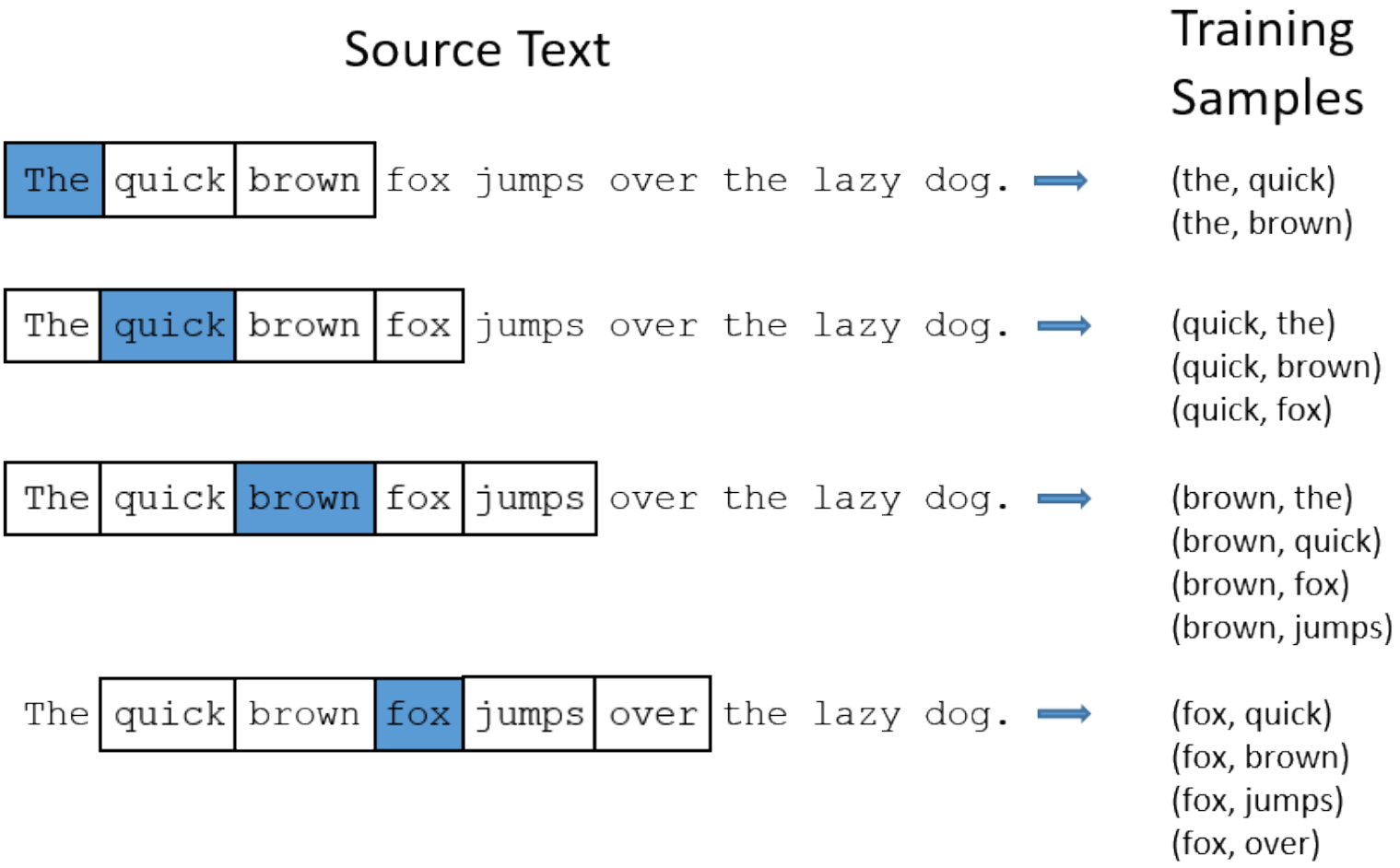
word2vec overview

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



2023 © University of Michigan

Generating positive training examples



2023 © *University of Michigan*

image credit: http://mccormickml.com/

Learning Word Embeddings

- Word2vec is popular because it is *very efficient* at learning from large amounts of data
- You can learn meaningful vectors from small amounts of text ~1M words—but only for the most common words
- Vector similarities depend on what data you train on. Ideas for what we can use?
  - Wikipedia?
  - News Articles?
  - Social Media?

2023 © *University of Michigan*

Word vectors-nearest neighbors

Word	Neighbors
heart	hart, heart's, iheart, hearth, hearted, art
please	pleased, pleas, pleases, pleaser, plea
plug	plugs, plugged, slug, pug, pluck
chareety	charity, sharee, cheri, tyree, charice, charities

2023 © *University of Michigan*

Simple idea: Bag of Vectors for IR

- Queries and documents are the *average* word vector
- Relevance is cosine similarity of query and document vectors
- Good: Fast, deals with synonyms
- Bad? …
  - Ignores word order
  - Ignores which words are important
- Issue: How to adapt?

2023 © *University of Michigan*

K-NRM:
Using word vectors for IR

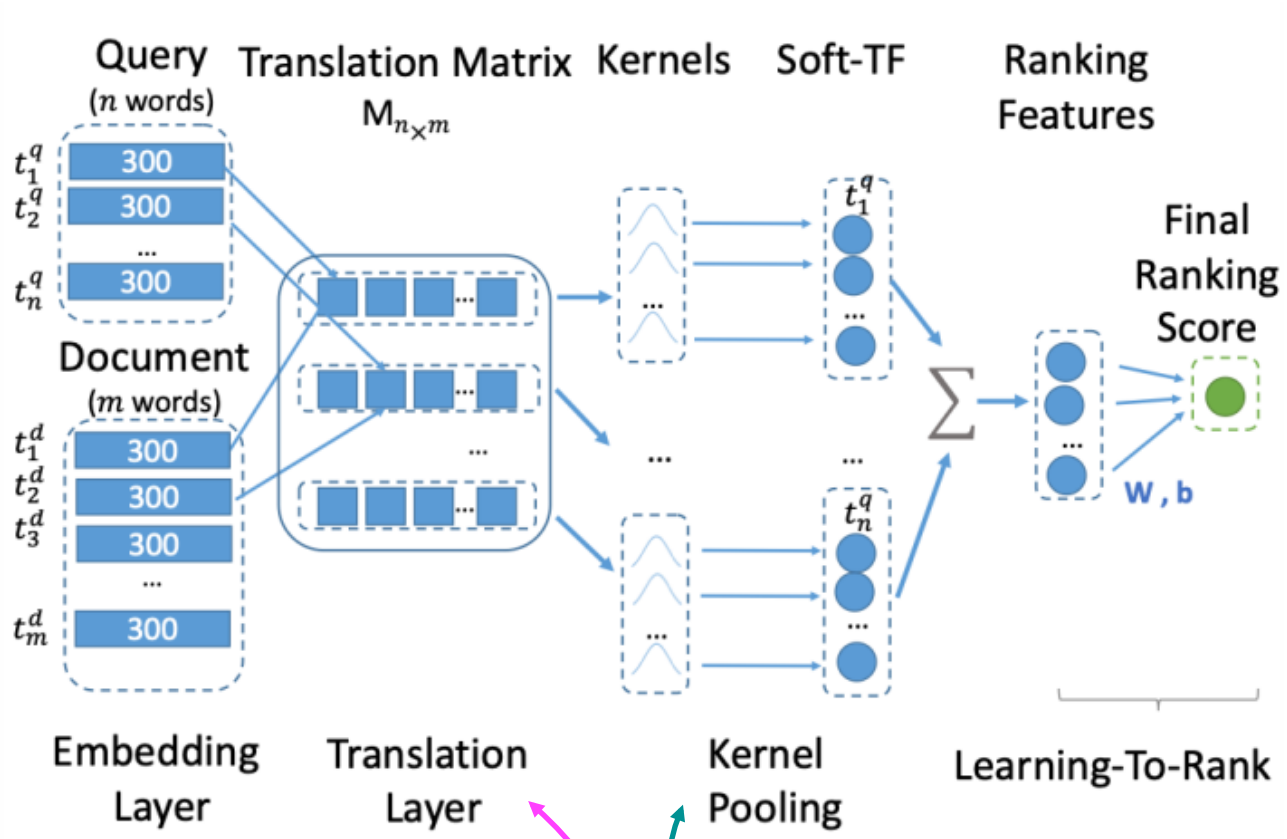
2023 © *University of Michigan*

Building dense representations of documents

- The BoW term-document representations may miss potential relevance matches
  - E.g., synonyms, related terminology
- Idea: Use dense vector representations for documents and queries
- Recall our fuzzy Pivoted Normalization… $f(q,d) = \sum_{w_1 \in q} \sum_{w_2 \in d} similarity(w_1, w_2) \left[c(w_1, q) \frac{1+\ln(1+\ln(c(w_2,d)))}{1-b+b\frac{|d|}{avg\_dl}} \ln \frac{N+1}{df(w_2)}, b \in [0,1] \right]$
  - Q: Should we be weighting the similarity somehow?
- Idea: Use a neural network to ***learn*** how to determine relevance from word-pair similarity

2023 © University of Michigan

K-NRM: kernel-based neural ranking model



The translation layer computes pair-wise similarities between document words and query words

Learns to score how doc-query relevance from the soft-similarity scores

2023 © University of Michigan

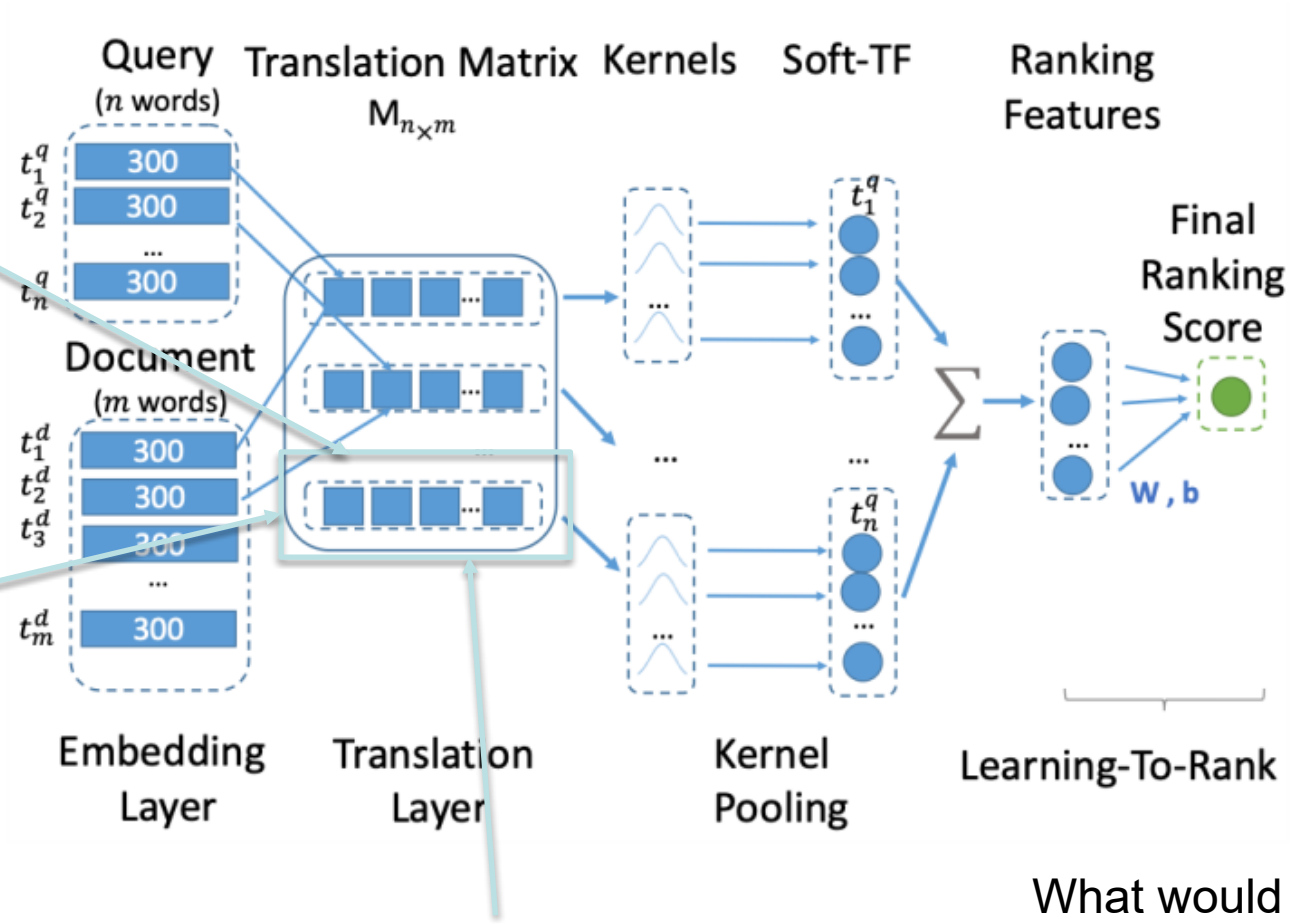
🧩 模型结构分解（从左到右）

1. **Embedding Layer（嵌入层）**
   - 把 query 和 document 的每个词都映射为一个 300 维的词向量（可以用 word2vec、GloVe 等）。
   - 所以每个文档和查询变成一个矩阵。
2. **Translation Layer（翻译层）**
   - 计算每个 query 词和每个 document 词之间的相似度（一般用余弦相似度）。
   - 得到一个“相似度矩阵” $M_{n\times m}$，其中每个元素 $M_{ij} = sim(t_i^q, t_j^d)$。

     👉 图中左侧粉色框解释的就是这一步：“computes pair-wise similarities between document words and query words”。
3. **Kernel Pooling（核池化层）**
   - 不是简单取最大/平均相似度，而是用多个 **Gaussian kernels（高斯核）** 来对不同相似度区间进行软聚合（Soft-TF）。
   - 比如一个核专门统计相似度≈1（完全匹配）的词对，另一个核统计相似度≈0.5（语义相关）的词对。
   - 这样可以捕捉到“软匹配”——不完全相同但语义相近的词。
4. **Learning-to-Rank（学习排序层）**
   - 把每个 query 词的 kernel pooling 结果汇总成一个特征向量；
   - 然后送入一个神经网络，学习输出最终的**相关性分数（Final Ranking Score）**。

     👉 图中绿色框解释的就是：“Learns to score doc-query relevance from the soft-similarity scores”。

K-NRM: kernel-based neural ranking model

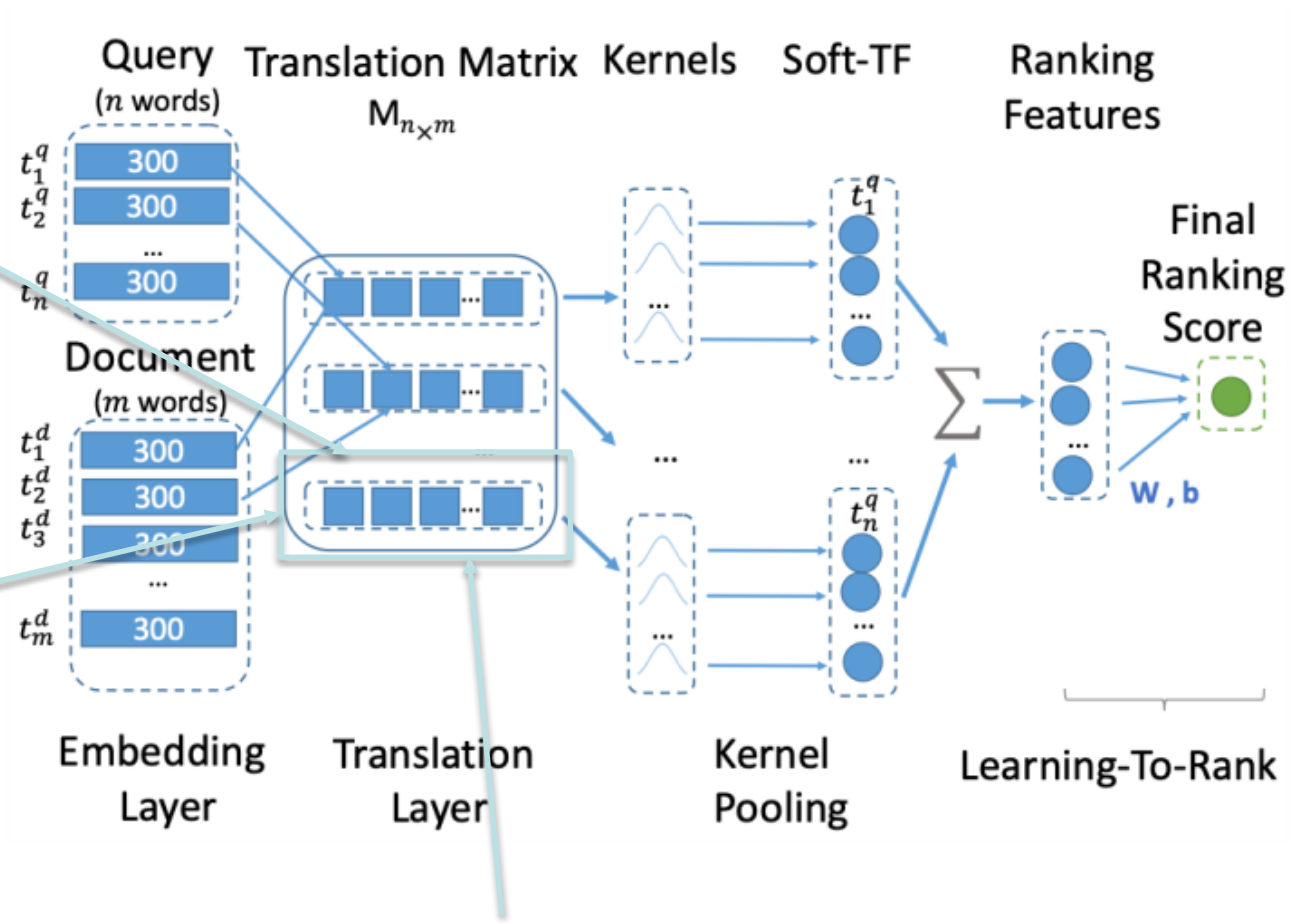


Similarity of landscape to each word in doc (a sandy desert scene with a blue sky)

What would happen if the similarity is 1 if identical and zero otherwise?

2023 © University of Michigan

K-NRM: kernel-based neural ranking model



natural landscape

neural network training loss landscape

Similarity of landscape to each word in doc (neural network training loss landscape)

2023 © University of Michigan

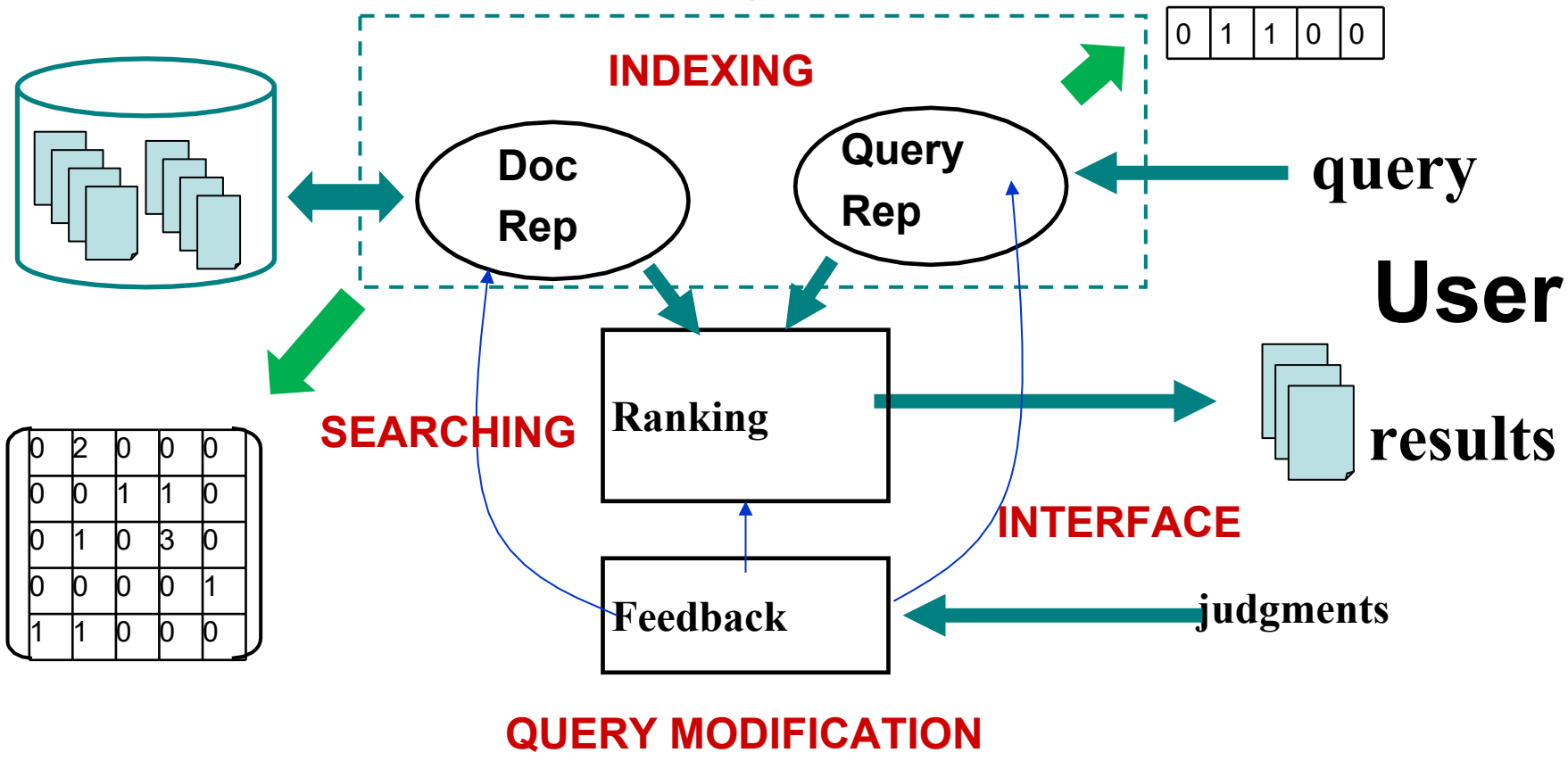
K-NRM Performance

(a) Testing-SAME. Testing labels are inferred by the same click model (DCTR) as the training labels used by neural models.

Method	NDCG@1		NDCG@3		NDCG@10		W/T/L
Lm	0.1261	−20.89%	0.1648	−26.46%	0.2821	−20.45%	293/116/498
BM25	0.1422	−10.79%	0.1757	−21.60%	0.2868	−10.14%	299/125/483
RankSVM	0.1457	−8.59%	0.1905	−14.99%	0.3087	−12.97%	371/151/385
Coor-Ascent	$0.1594^{\ddagger\S\P}$	–	$0.2241^{\ddagger\S\P}$	–	$0.3547^{\ddagger\S\P}$	–	–/–/–

2023 © *University of Michigan*

What else can Deep Learning help with?

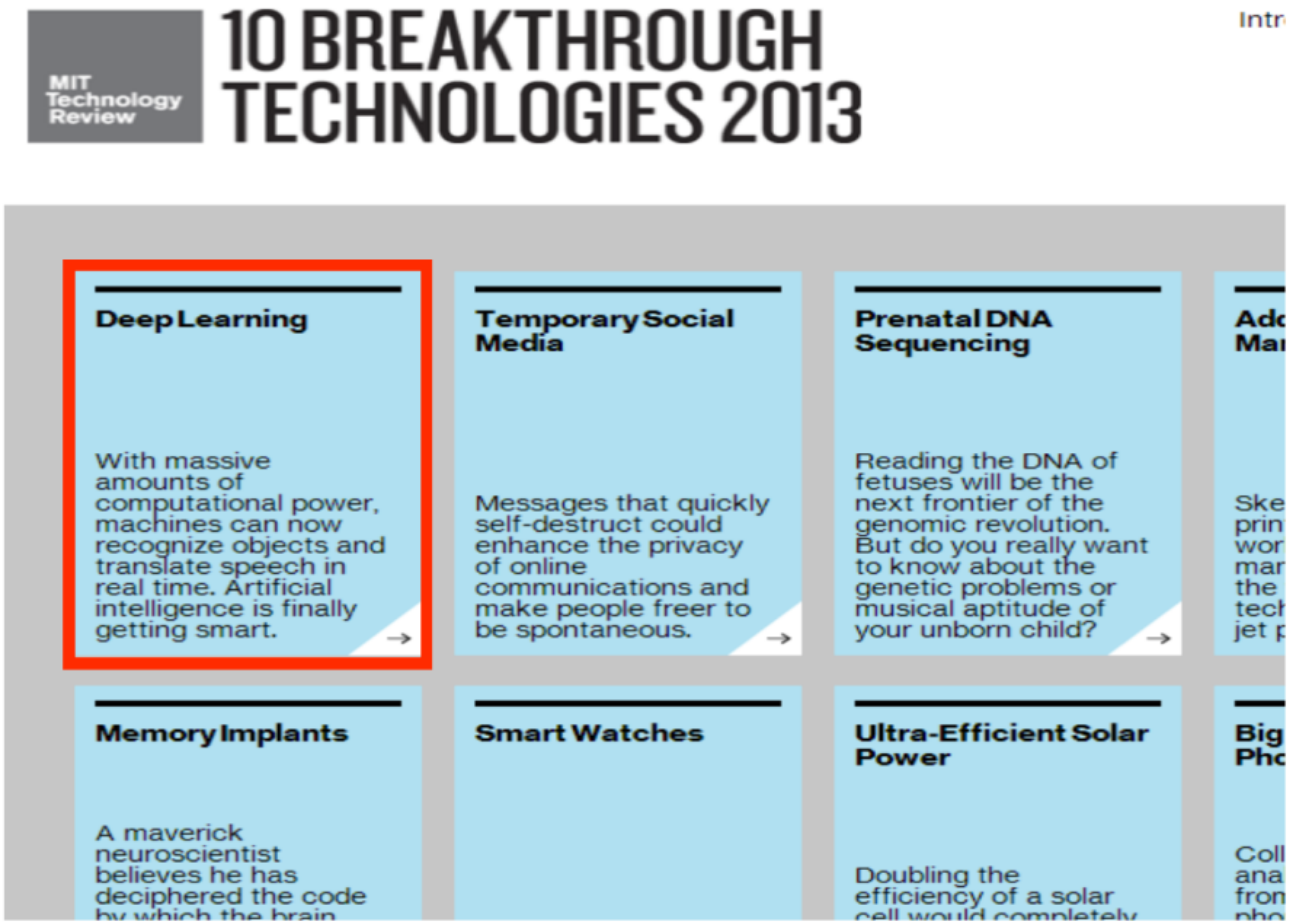


- can we learn a better document representation?
- … or a model that learns IR-specific representations directly?
- … or how to augment our documents or queries?

2023 © University of Michigan

What is Deep Learning in General?

2023 © *University of Michigan*

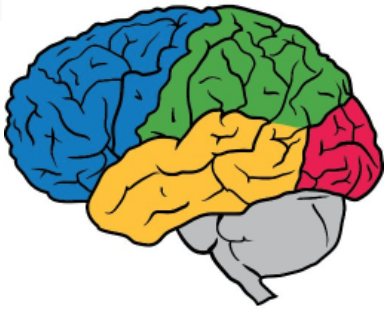


2023 © *University of Michigan*

Industry Research Labs of Deep Learning

ANTHROP\C

DeepMind



OpenAI

Facebook AI Research

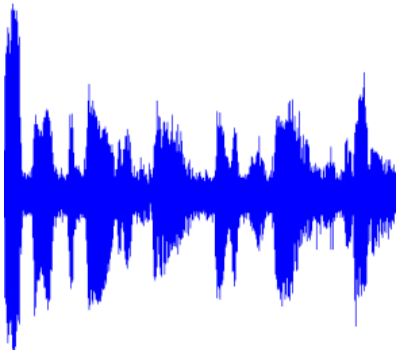
Baidu Research

Microsoft Research

Tencent AI Lab

2023 © *University of Michigan*

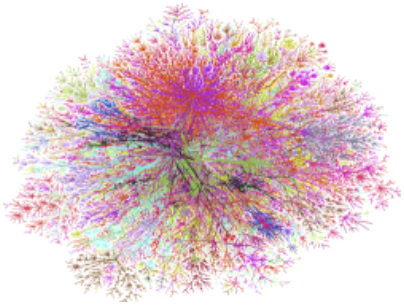
Applications in Multiple Domains



speech

image

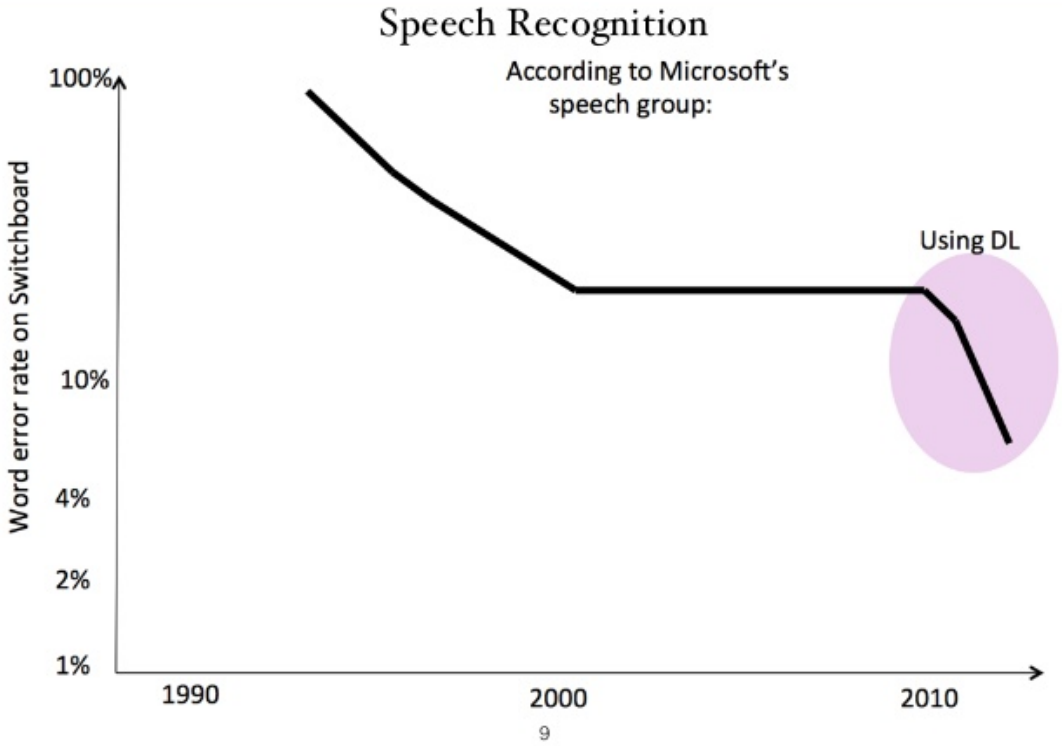
text



network

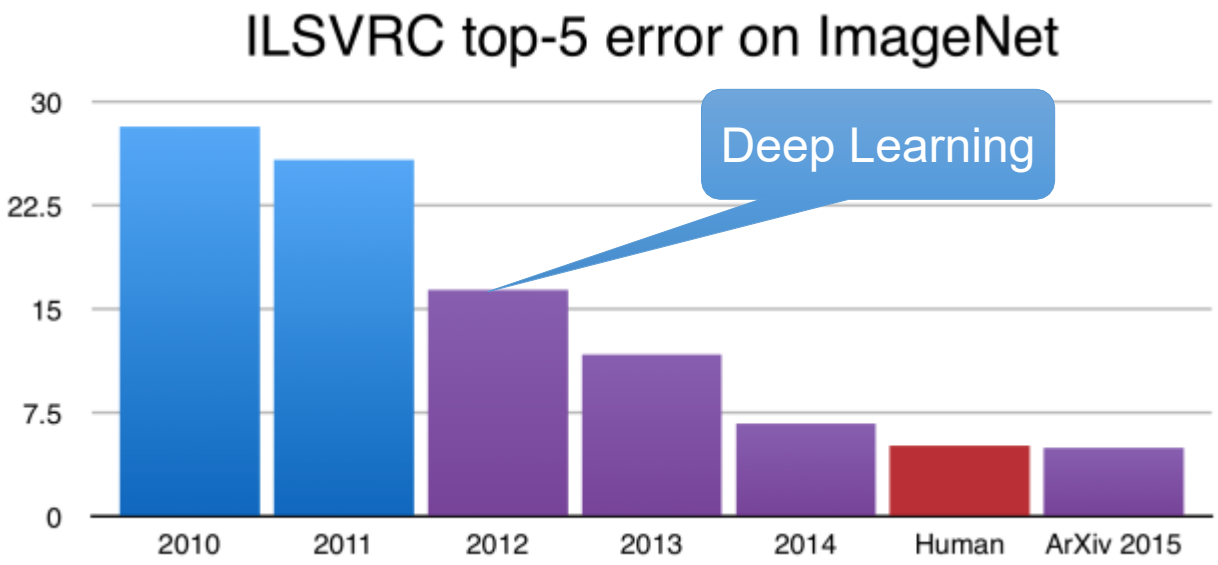
2023 © *University of Michigan*

Speech Recognition Results



2023 © University of Michigan

Image Recognition Results



2023 © *University of Michigan*

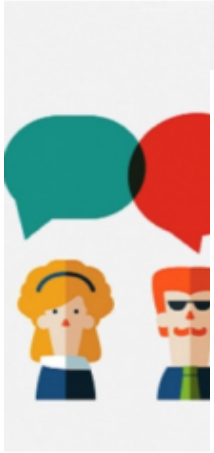
Fast evolvement of Deep Neural Nets

Best NLP Model Ever? Google BERT Sets New Standards in 11 Language Tasks

Synced Follow
Oct 16, 2018 · 3 min read ★

CMU & Google XLNet Tops BERT; Achieves SOTA Results on 18 NLP Tasks

Synced Follow



EDITORIAL ▸ Front Artif Intell. 2024 Jan 12;6:1350306. doi: 10.3389/frai.2023.1350306

Natural language processing in the era of large language models

Arkaitz Zubiaga [1,*]

The new Goog
Transformers f
the machine le
bidirectional T
natural langua
the chall
paper:

▸ Author information ▸ Article notes ▸ Copyright and License information

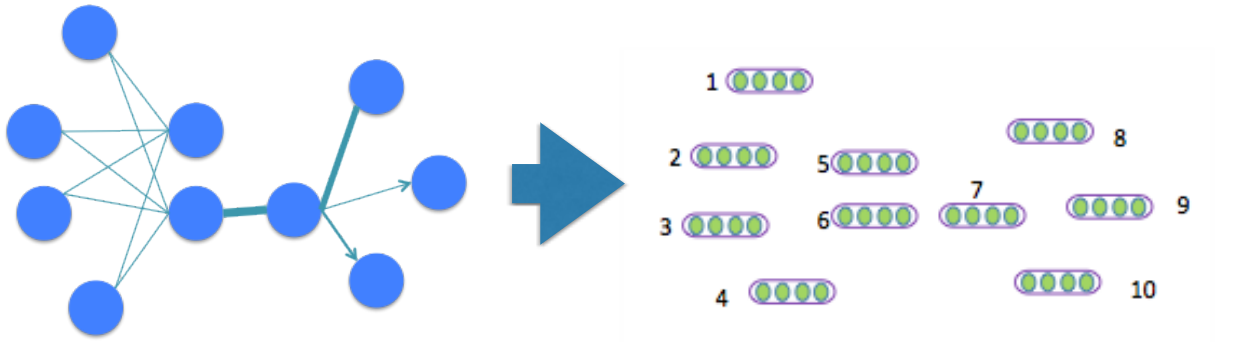
PMCID: PMC10820986 PMID: 38282904

https://medium.com/syncedreview/best-nlp-model-ever-google-bert-sets-new-standards-in-11-language-tasks-4a2a189bc155

2023 © University of Michigan

Deep Learning for Networks

- State-of-the-art results on many applications such as node classification, link prediction, cascade prediction
  - DeepWalk, LINE, Node2Vec, GCN, GAT, …



Network **Node representations**

2023 © University of Michigan

How Deep is Deep?

- Like “Big” data, “deep” learning is also not well defined.
  - Originally referred to Neural Nets of more than 2 layers (input, output, hidden..), but that dates back to the 80s
  - Other machine learning models can also be “deep” (or non-linear): Random forests, SVM with kernel tricks
- So, what is deep learning?

2023 © *University of Michigan*

Traditional Machine Learning

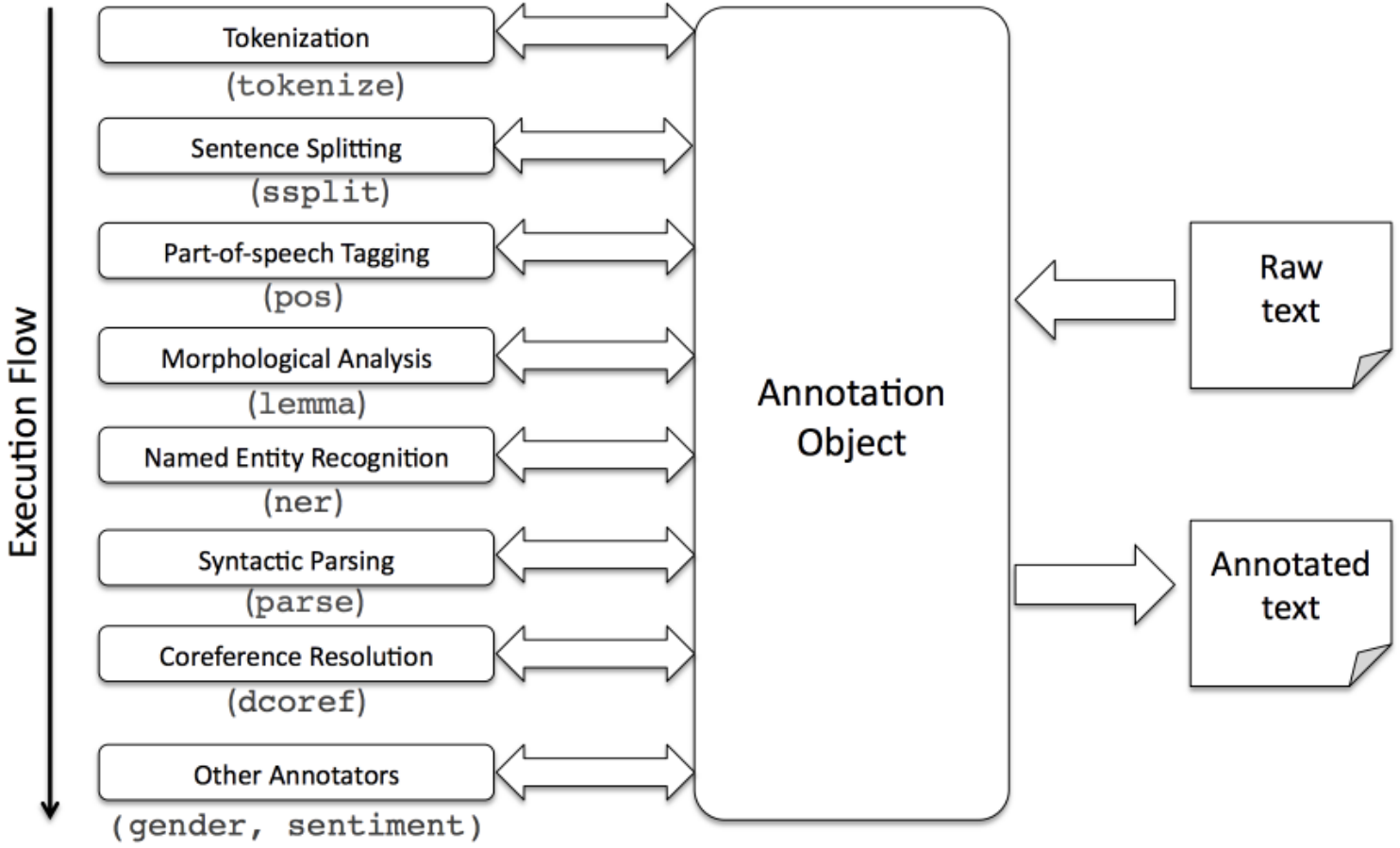
Hand-crafted
Features

Trainable Classifie
Usually Linear

Domain experts

2023 © *University of Michigan*

Traditional Approach: NLP Pipelines



https://stanfordnlp.github.io/CoreNLP/pipelines.html

2023 © *University of Michigan*

Problems

- Every level of feature extraction results in a loss of accuracy, and the errors accumulate
- No mechanism to backpropagate the errors made in later stages to improve the previous stages.

2023 © *University of Michigan*

Deep Learning = End-to-end Learning/Feature Learning

Trainable
Feature Extractor

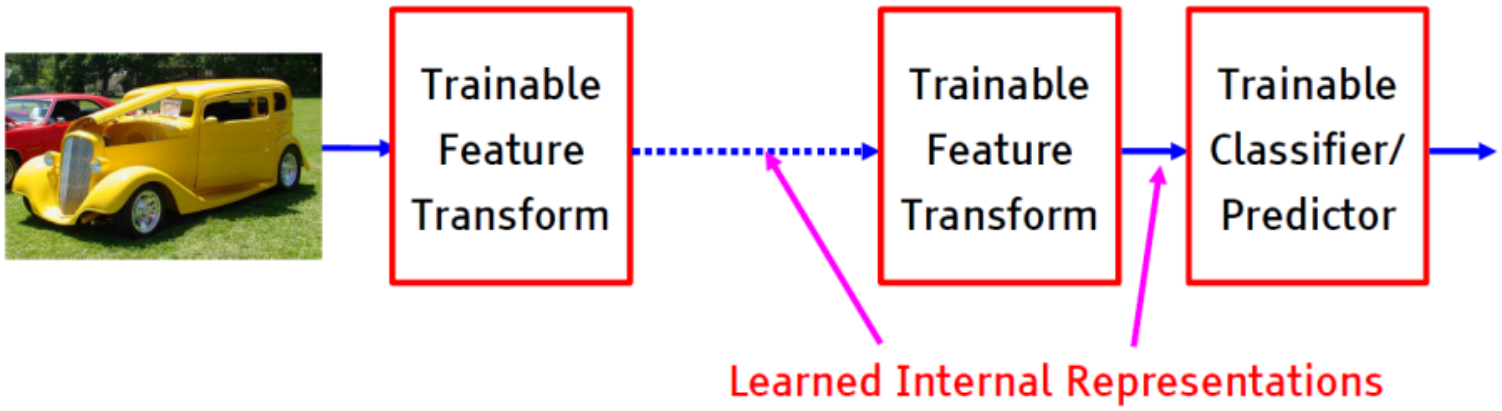
Trainable Classifier
Usually DNN

Domain experts

2023 © University of Michigan

Trainable feature Hierarchies

- A hierarchy of trainable feature transforms
  - Each module transforms its input representations into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



2023 © University of Michigan

How do neural nets work?

2023 © *University of Michigan*

Neural networks

- Tremendous flexibility on design choices (exchange feature engineering for model engineering)
- Articulate model structure and use the chain rule to derive parameter updates.
- Discrete, high-dimensional representation of inputs (one-hot vectors) -> low-dimensional “distributed” representations.
- Non-linear interactions of input features
- Multiple “layers” to capture hierarchical structure

2023 © University of Michigan

Neural network libraries

theano

(Keras)

(Dynet)

2023 © *University of Michigan*

You already know how neural networks* work!

2023 © *University of Michigan*

Logistic regression

- perceptron (one-layer NN)

$$\hat{y} = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)}$$

	x	β
not	1	-0.5
bad	1	-1.7
movie	0	0.3

2023 © *University of Michigan*

SGD

Algorithm 2 Logistic regression stochastic gradient descent

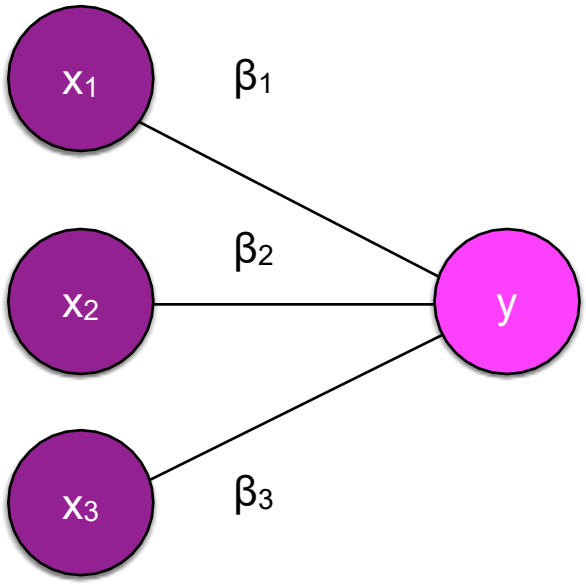
1: Data: training data $x \in \mathbb{R}^F, y \in \{0, 1\}$
2: $\beta = 0^F$
3: **while** not converged **do**
4: **for** $i = 1$ to N **do**
5: $\beta_{t+1} = \beta_t + \alpha \left(y_i - \hat{p}(x_i) \right) x_i$
6: **end for**
7: **end while**

Calculate the derivative of some loss function with respect to parameters we can change, update accordingly to make predictions on training data a little less wrong next time.

2023 © University of Michigan

Logistic regression

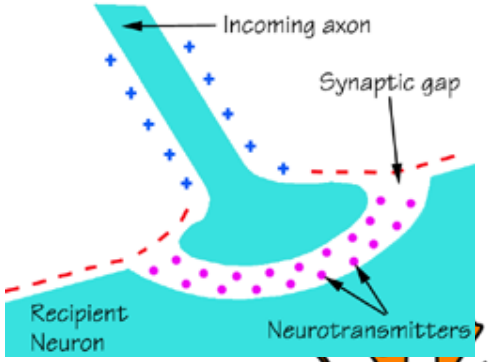
$$\hat{y} = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)}$$



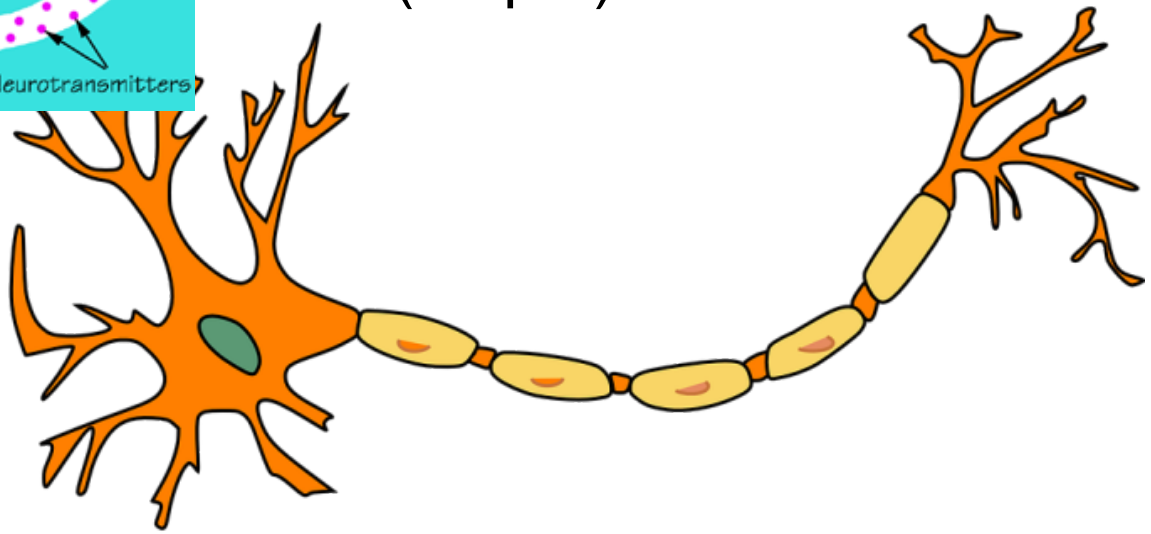
	x	β
not	1	-0.5
bad	1	-1.7
movie	0	0.3

2023 © *University of Michigan*

Biological analog of the neuron

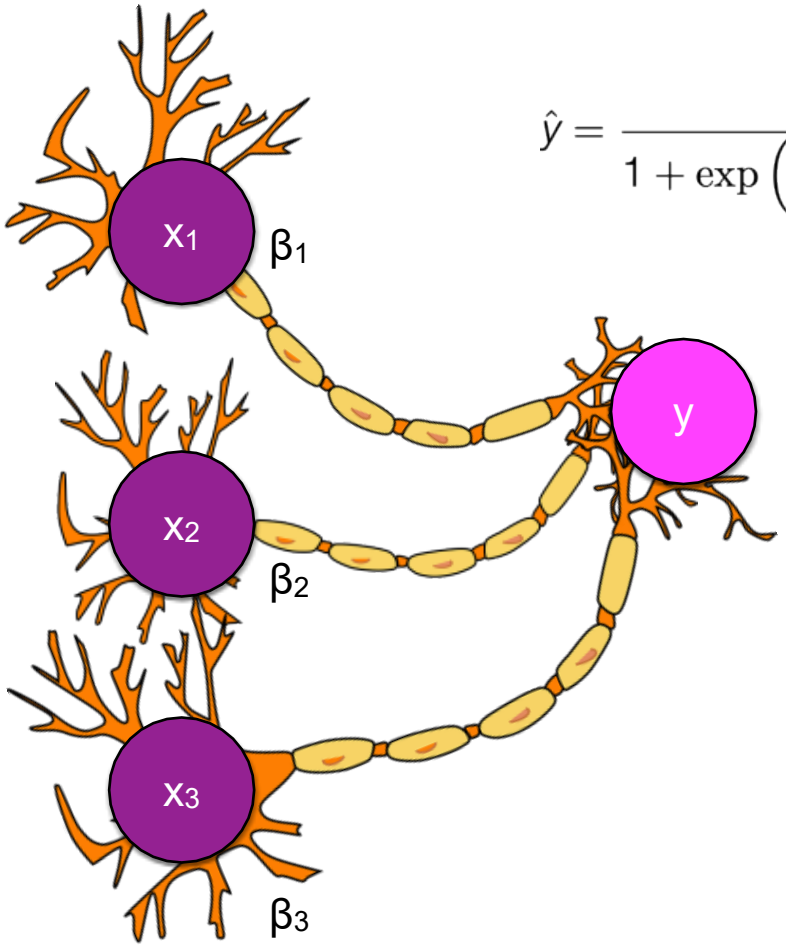


How much activation (input) does it take for this neuron to fire (output)?



2023 © University of Michigan
Image credit: https://commons.wikimedia.org/wiki/File:Synapse_diagram.png

Logistic regression



$$\hat{y} = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)}$$

	x	β
not	1	-0.5
bad	1	-1.7
movie	0	0.3

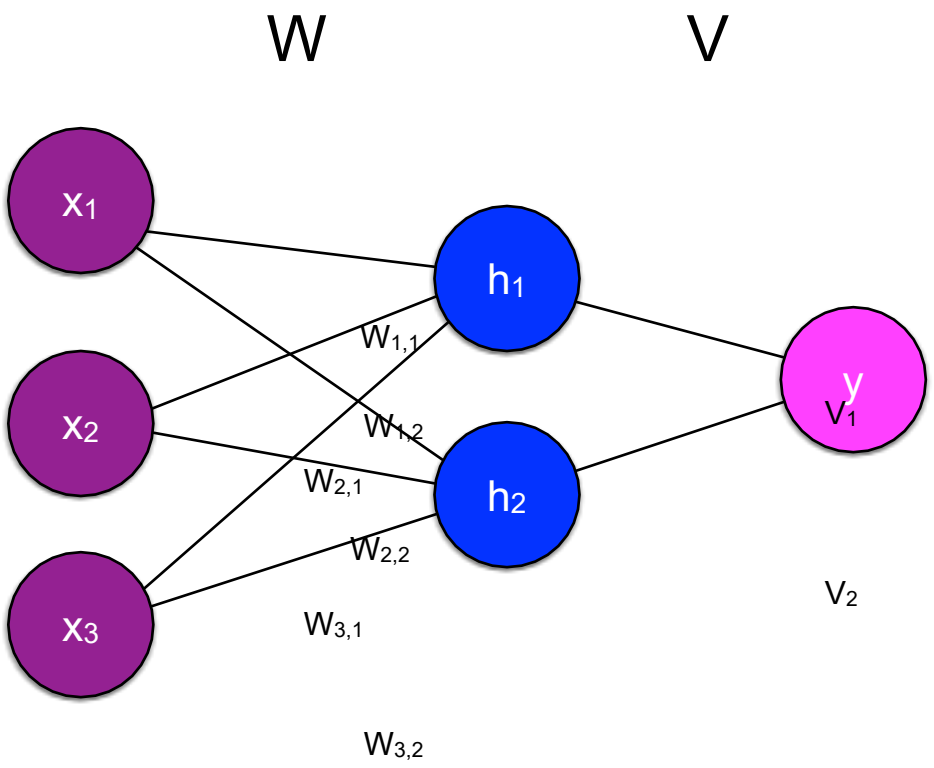
2023 © *University of Michigan*

Deep Neural networks

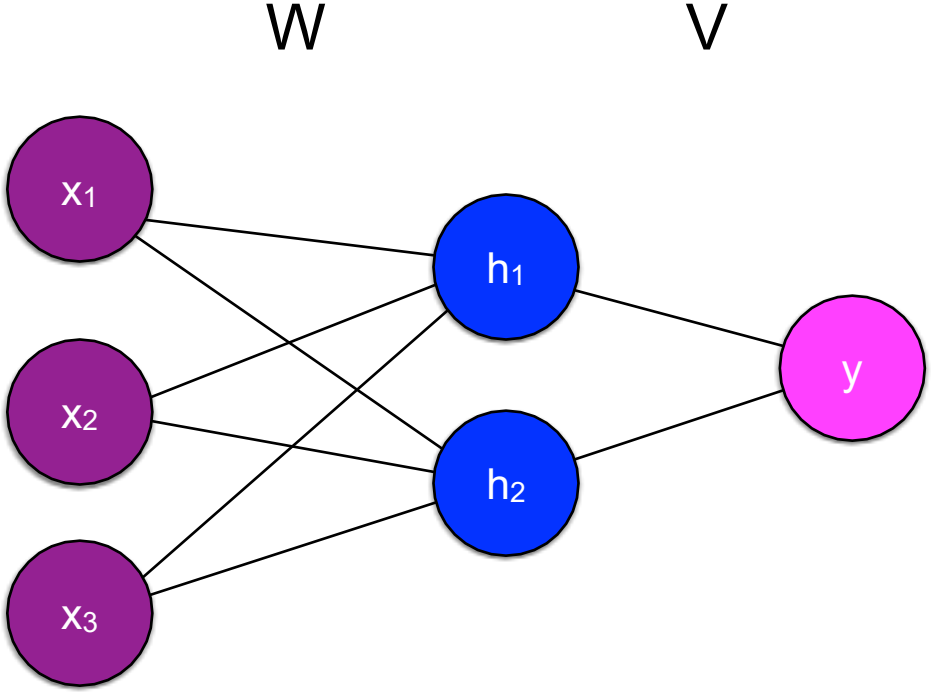
- Two core ideas:
  - Non-linear activation functions
  - Multiple layers
- These have been around for decades
  - Challenge was how to train them efficiently

2023 © *University of Michigan*

*For simplicity, we're leaving out the bias term, but assume most layers have them as well.

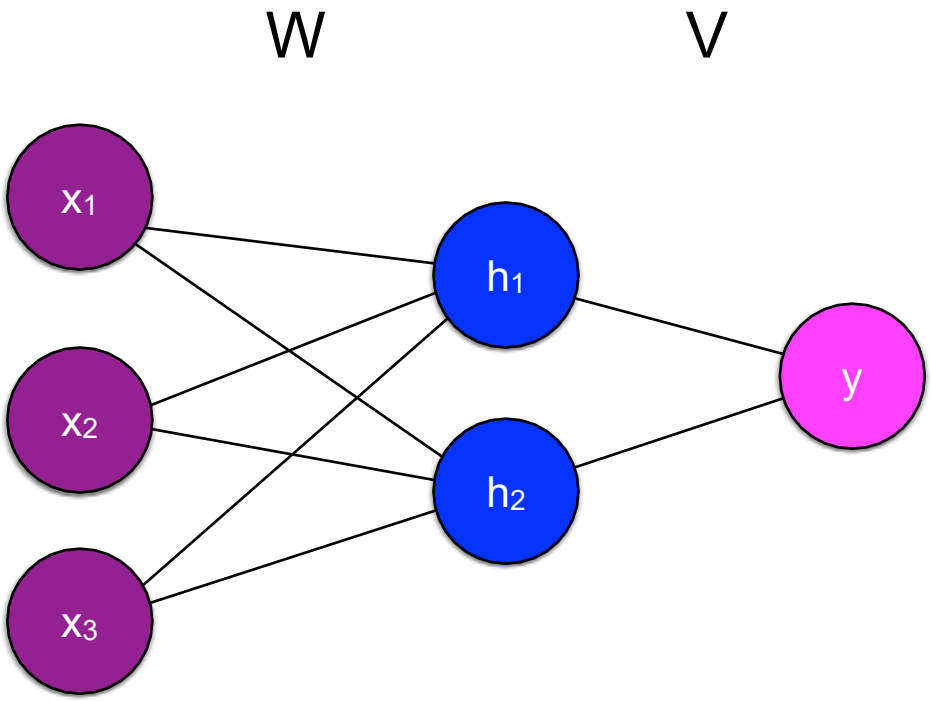


2023 © University of Michigan



	x	W		V	y
not	1	-0.5	1.3	4.1	1
bad	1	0.4	0.08	-0.9	
movie	0	1.7	3.1		

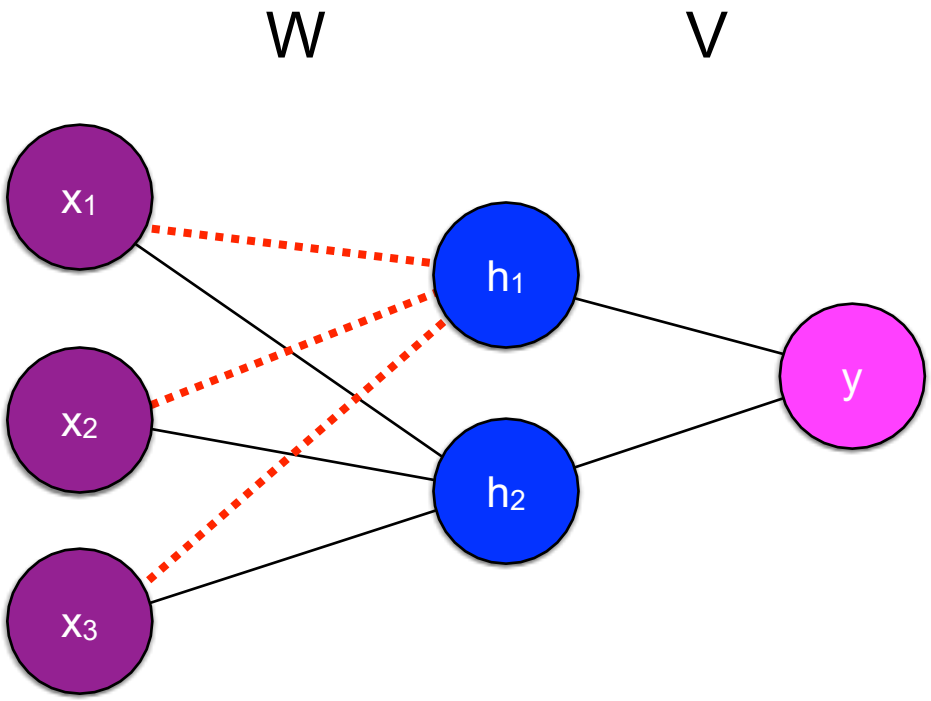
2023 © University of Michigan



$$h_j = f\left(\sum_{i=1}^{F} x_i W_{i,j}\right)$$

the hidden nodes are completely determined by the input and weights

2023 © University of Michigan

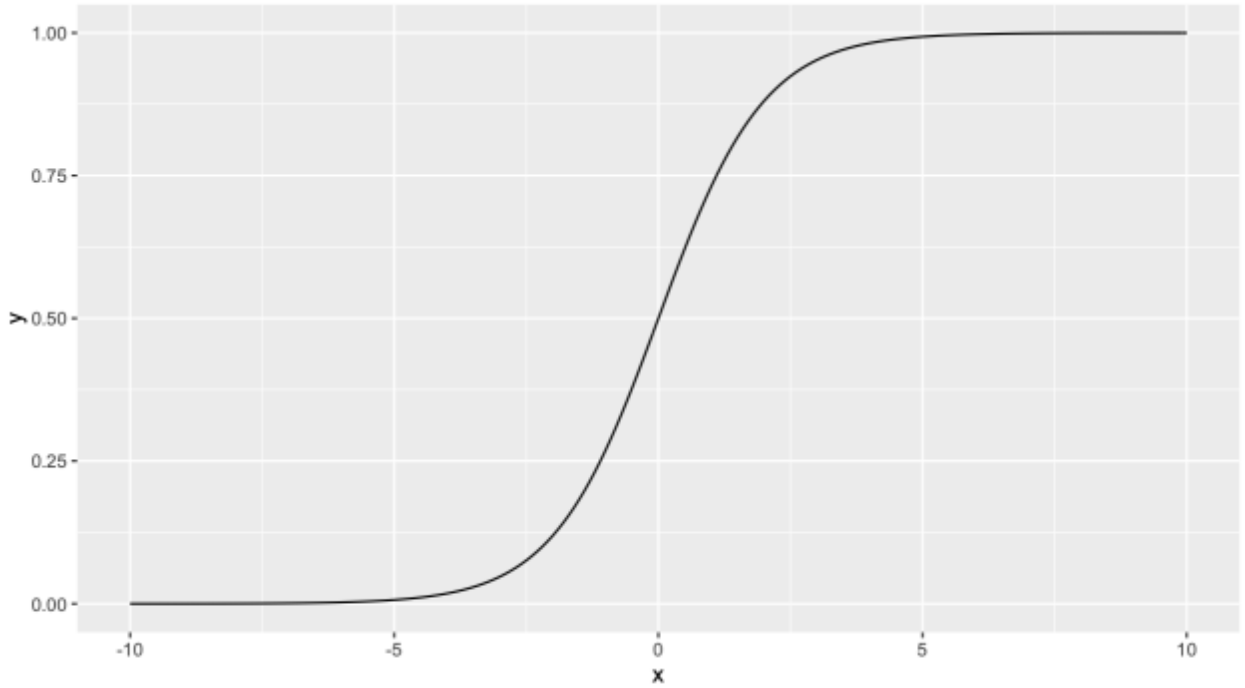


$$h_1 = f\left(\sum_{i=1}^{F} x_i W_{i,1}\right)$$

2023 © University of Michigan

Activation functions

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



2023 © University of Michigan

Three Factors to the Success of Deep Learning

- End-to-end learning
- Learning Non-linear functions
  - Theoretically, a neural network with two layers could approximate any function
  - In practice, usually need specialized architecture
- Efficiently learn from large-scale labeled data
  - Through backpropagation
  - GPUs, optimization tricks, distributed computing, etc.

2023 © *University of Michigan*

What do we need for IR?

- Core task: How relevant is a document for a query?
- Needs: some comparable representations for queries and documents
- Bag of Words:
  - good: fast
  - bad: misses synonyms, different kinds of queries…
- Bag of Vectors: (e.g., average word2vec vector)
  - good: fast-ish, capture synonyms
  - bad: ignores word order
- Big Question: Can we get better representations of text for IR?

2023 © *University of Michigan*

Deep Learning for Text

2023 © *University of Michigan*

Deep Learning for Text

- Learn vector representations of words
- Learn representations of sentences/documents
- Natural language generation
- Sequence to sequence modeling

Let's start with how we can learn word vector representations in a new way (other than word2vec)

2023 © University of Michigan

Recall what a Language Model is

- A language model computes the probability of a sequence of words $p(x_1, \ldots, x_n)$

- The joint probability can be factorized as follows:

$$p(x_1, \ldots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i | x_{i-1}, \ldots, x_1)$$

- Previous approaches: unigram, bigram, trigram,….
  - Sparse
  - Ignore the similarity between the words

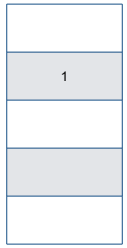
2023 © University of Michigan

Can we learn a language model with a Feed-Forward Neural Network?

"I forgot the toppings for the pizza so I went to the store to get _____"

Simple feed-forward multilayer perceptron (e.g., one hidden layer)

input x = vector concatenation of a conditioning context of fixed size k



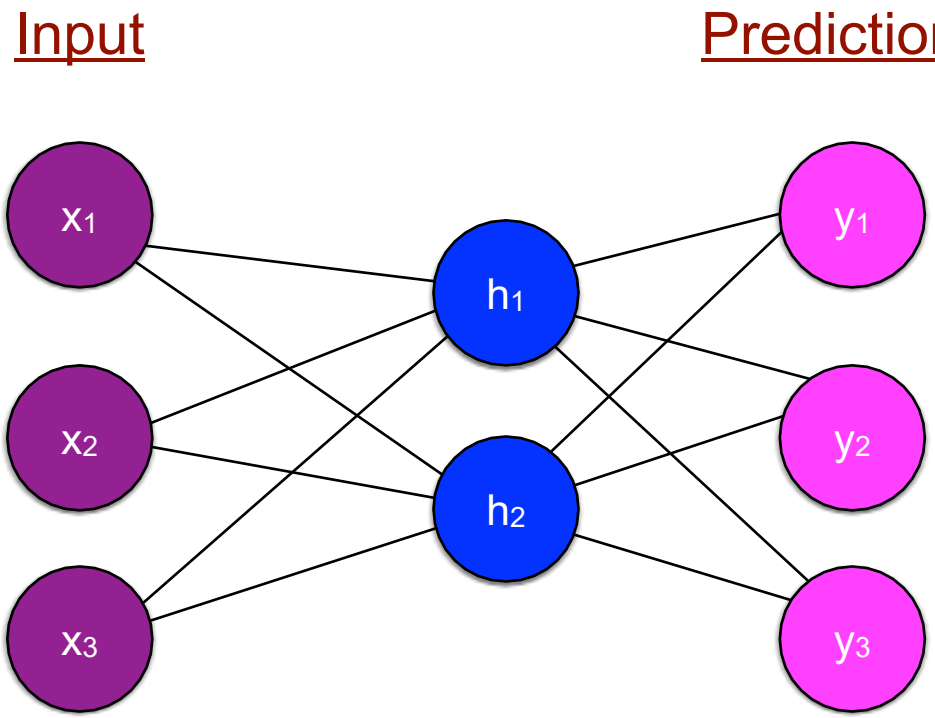
$$x = [v(w_1); \ldots; v(w_k)]$$

Bengio et al. 2003

2023 © University of Michigan

A bigram language model

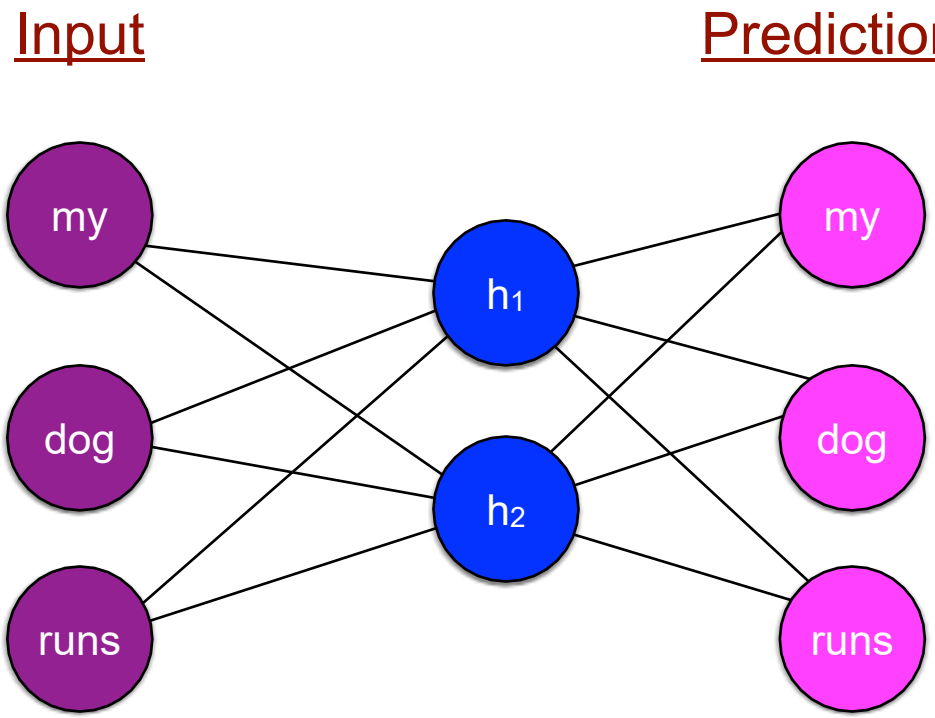
$\mathcal{V}$ = {dog, my, runs}



2023 © University of Michigan

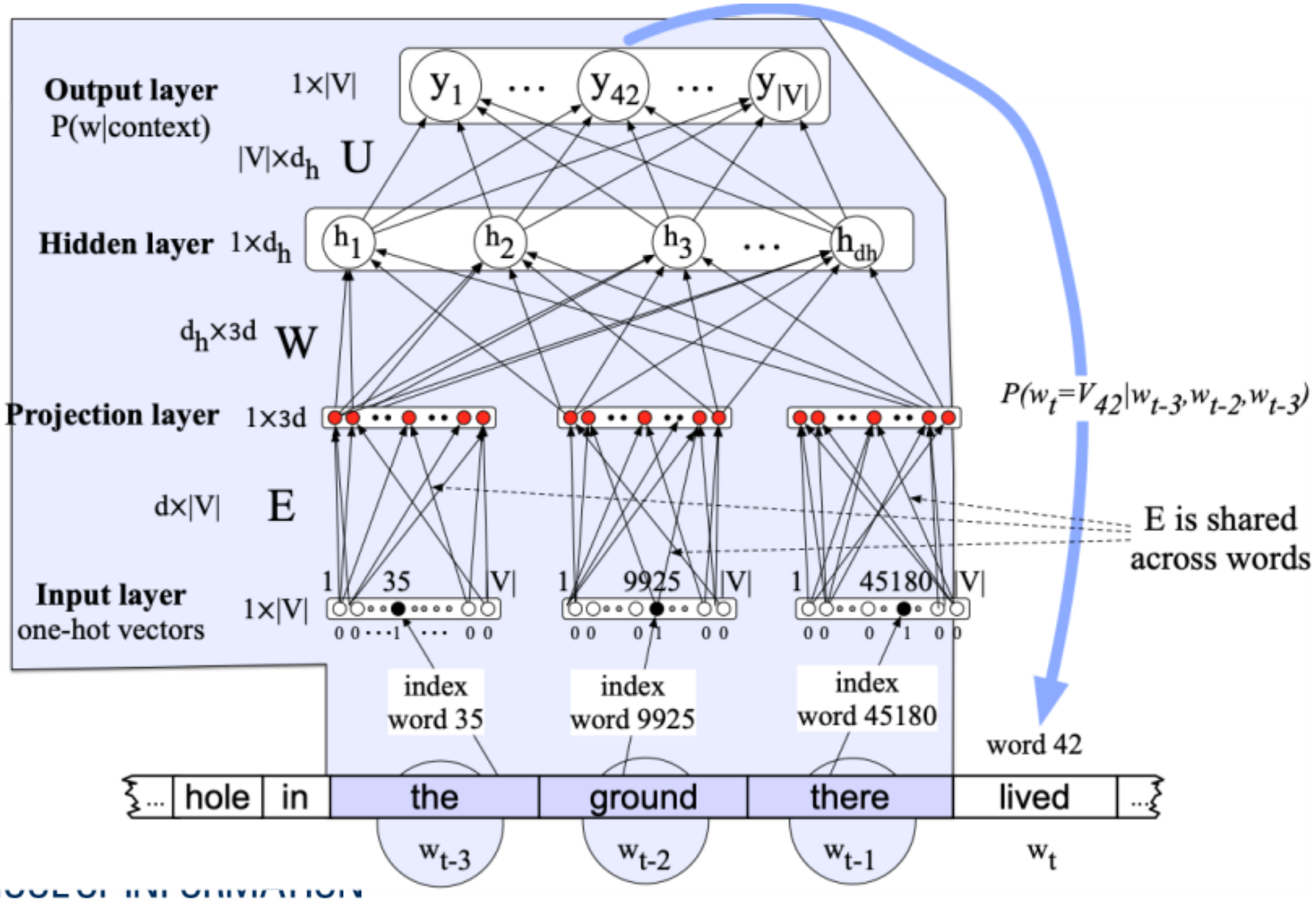
A bigram language model

$\mathcal{V}$ = {dog, my, runs}



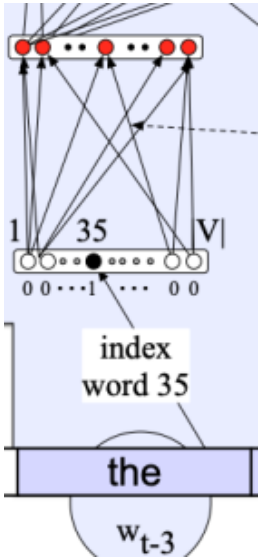
2023 © University of Michigan

A slightly more complicated neural network language model

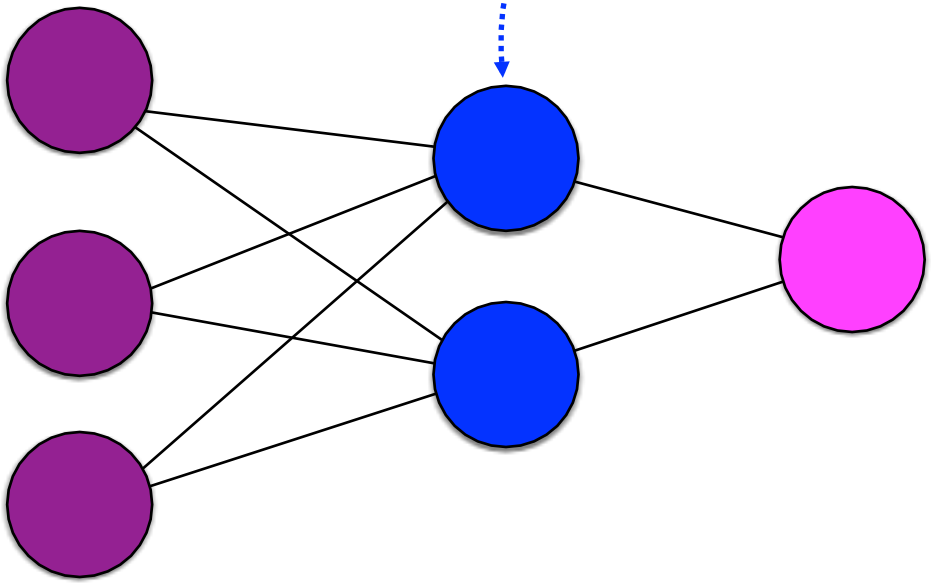


2023 © University of Michigan

A closer look at that network

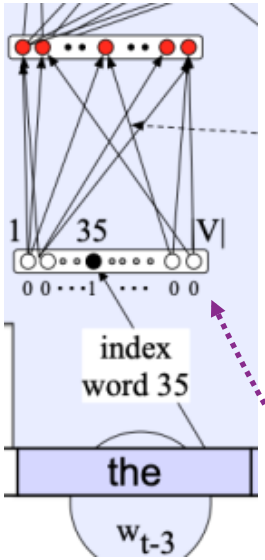


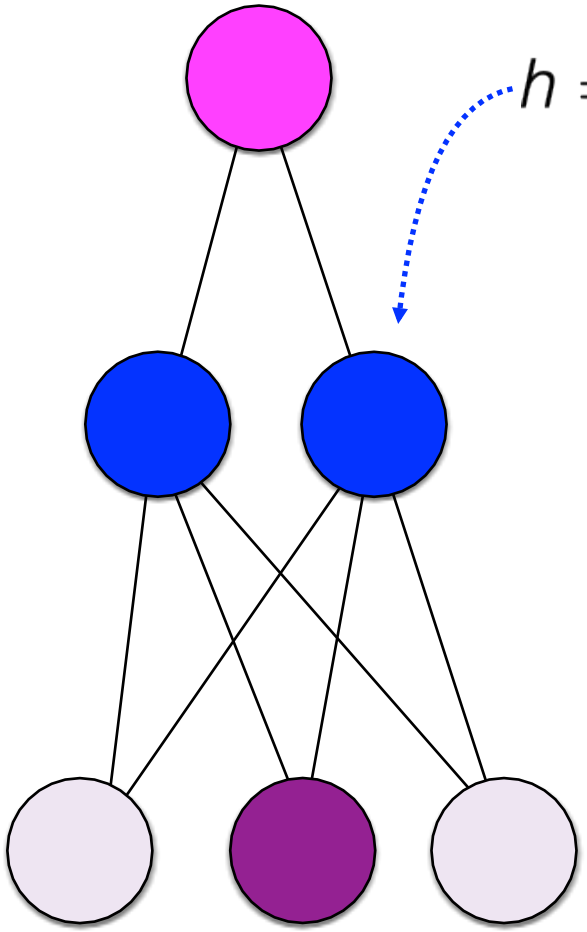
$$h = g(xW_1 + b_1)$$



2023 © University of Michigan

A closer look at that network





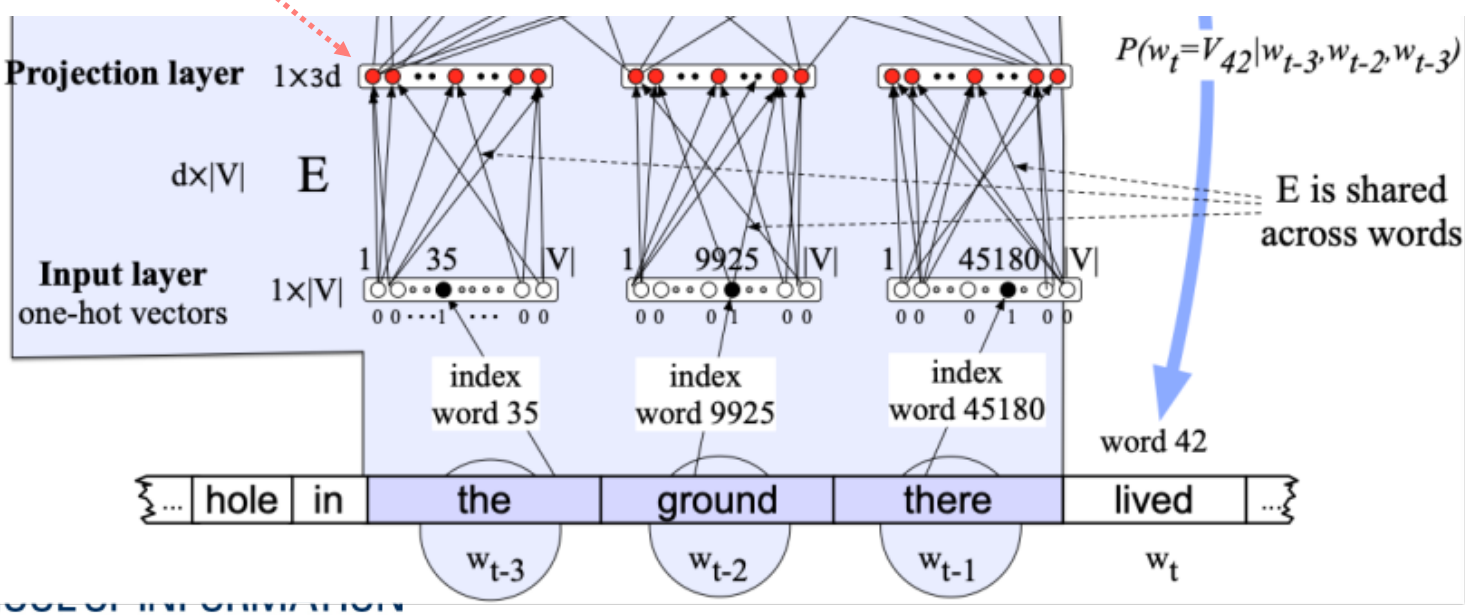
$$h = g(xW_1 + b_1)$$

"the"

2023 © University of Michigan

A closer look at that network

The projection layer is also known as an **embedding layer**.
It's a distributed vector representation of the word



2023 © University of Michigan

Sidenote: Softmax

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

The softmax function is the general case of the sigmoid function for multiple classes.

$$\hat{y} = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)} = \sigma\left(\sum_{i=1}^{F} x_i \beta_i\right)$$

2023 © University of Michigan

Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

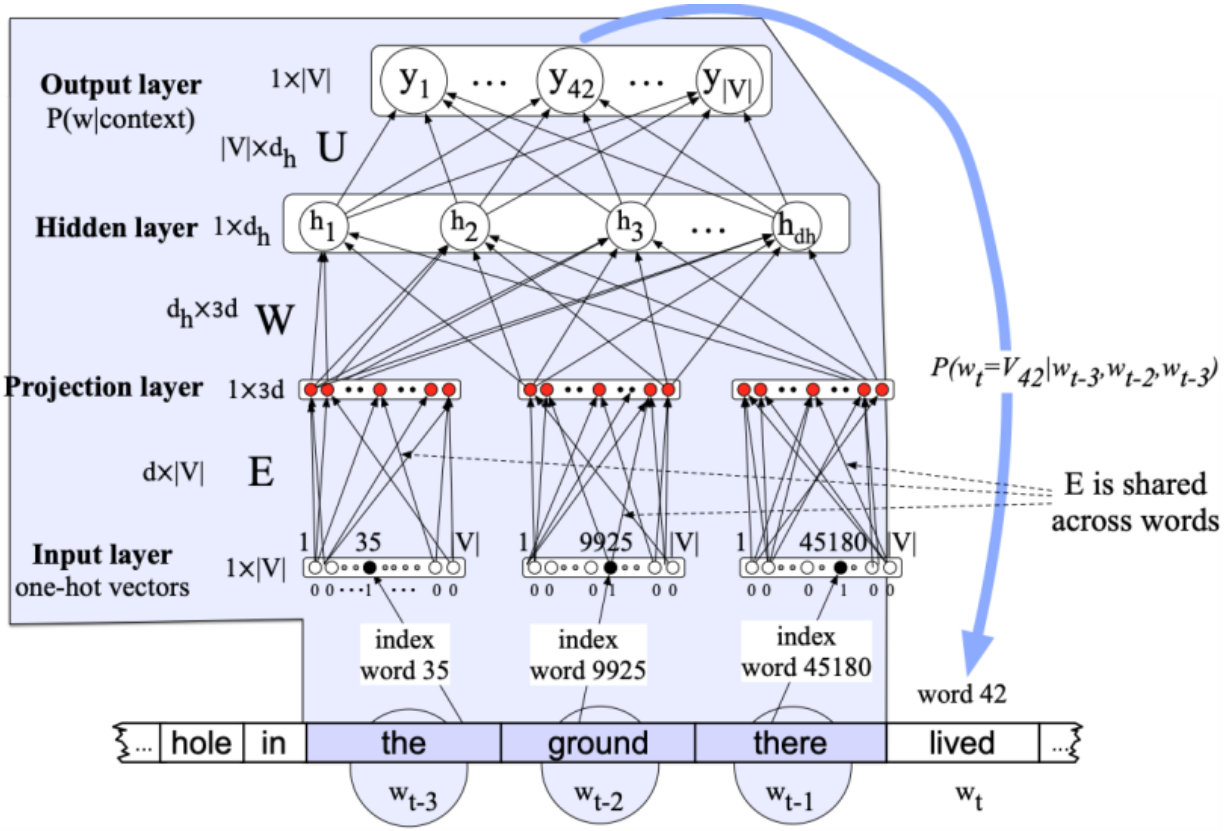
y

2023 © *University of Michigan*

Feed-forward networks can be limited

- No ability to "look back" at distant prior inputs
  - Can only see what's in the current window

Attend discussion sections for more models (eg models that give longer look back)



2023 © University of Michigan

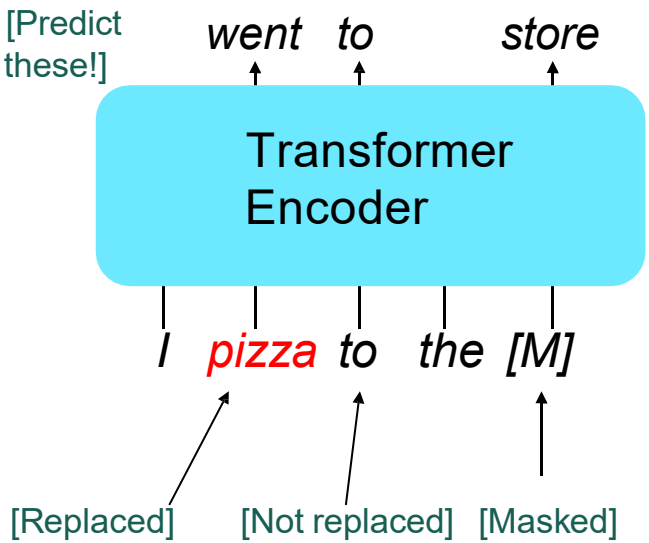
More Models – Discussion Section

- RNNs
- LSTMs
- Transformers

2023 © *University of Michigan*

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.
- 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]

BERT: Bidirectional Encoder Representations from Transformers

- Mask out k% of the input words, and then predict the masked words
  - They always use k = 15%

store gallon

↑ ↑

the man went to the [MASK] to buy a [MASK] of milk

- Too little masking: Too expensive to train
- Too much masking: Not enough context

2023 © University of Michigan

🔹 背景

Devlin 等人在 2018 年提出 BERT，它基于 Transformer 编码器（Encoder）结构，并通过 **“Masked LM” 目标函数** 进行预训练。

- “Bidirectional” 表示 BERT 同时从**左右两个方向**理解上下文，而不像传统语言模型那样只看左边或右边的词。

🔹 Masked LM（掩码语言模型）机制

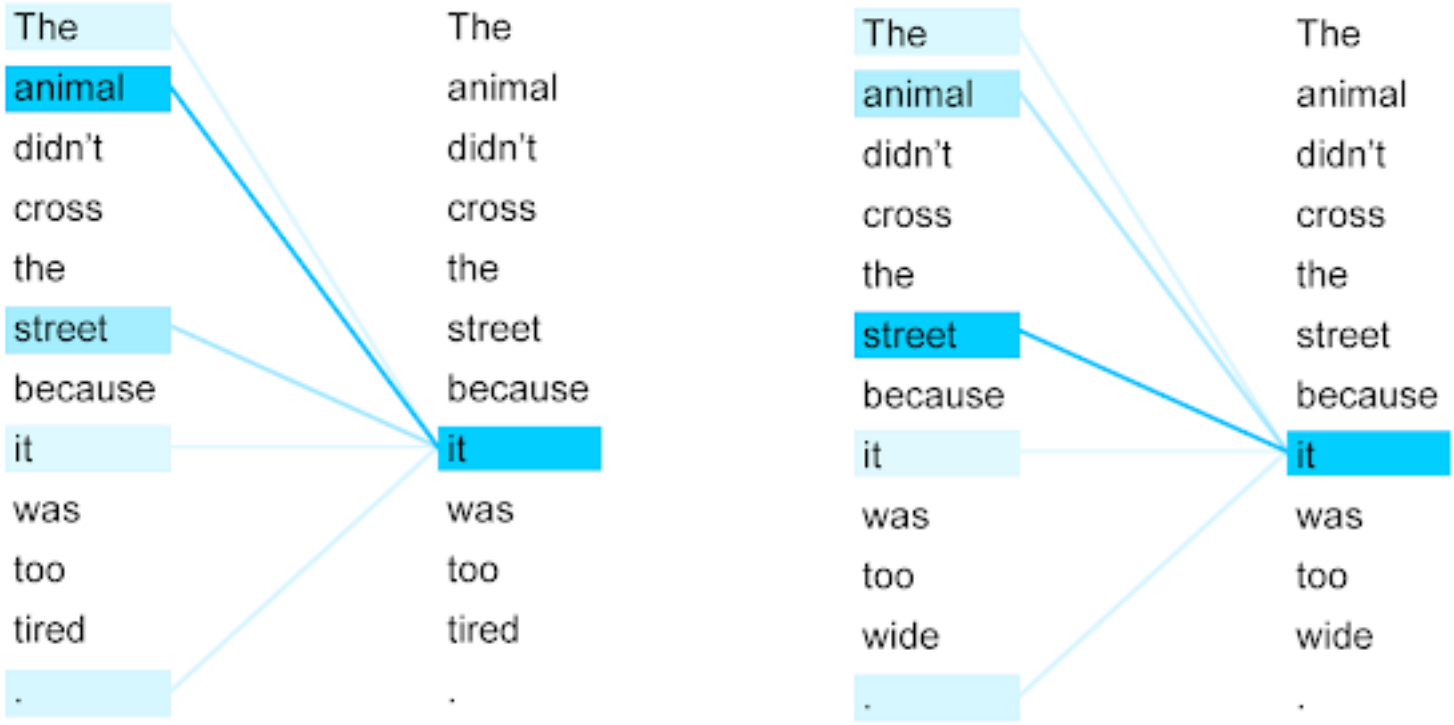
训练时，BERT 随机选择 15% 的词（或子词 token）进行掩码处理，让模型去预测这些被遮盖的词。

BERT 在掩码时采用以下策略：

1. **80% 的概率** → 把词替换成 `[MASK]`
   例如：
   “I went to the store” → “I went to the [MASK]”
2. **10% 的概率** → 把词随机换成别的词
   “I went to the store” → “I went to the pizza”
3. **10% 的概率** → 不替换，保留原词
   “I went to the store” → “I went to the store”
   但模型仍然要预测这个词。

👉 这样设计的原因是：
如果全部用 `[MASK]`，模型会依赖掩码标志；若全保留原词，模型又无法学到遮蔽词预测能力。混合方式让模型既能**学习语义上下文**，又能**泛化到没有 [MASK] 的 实任务**。

BERT's attention can be a way of model interpretation



https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

2023 © *University of Michigan*

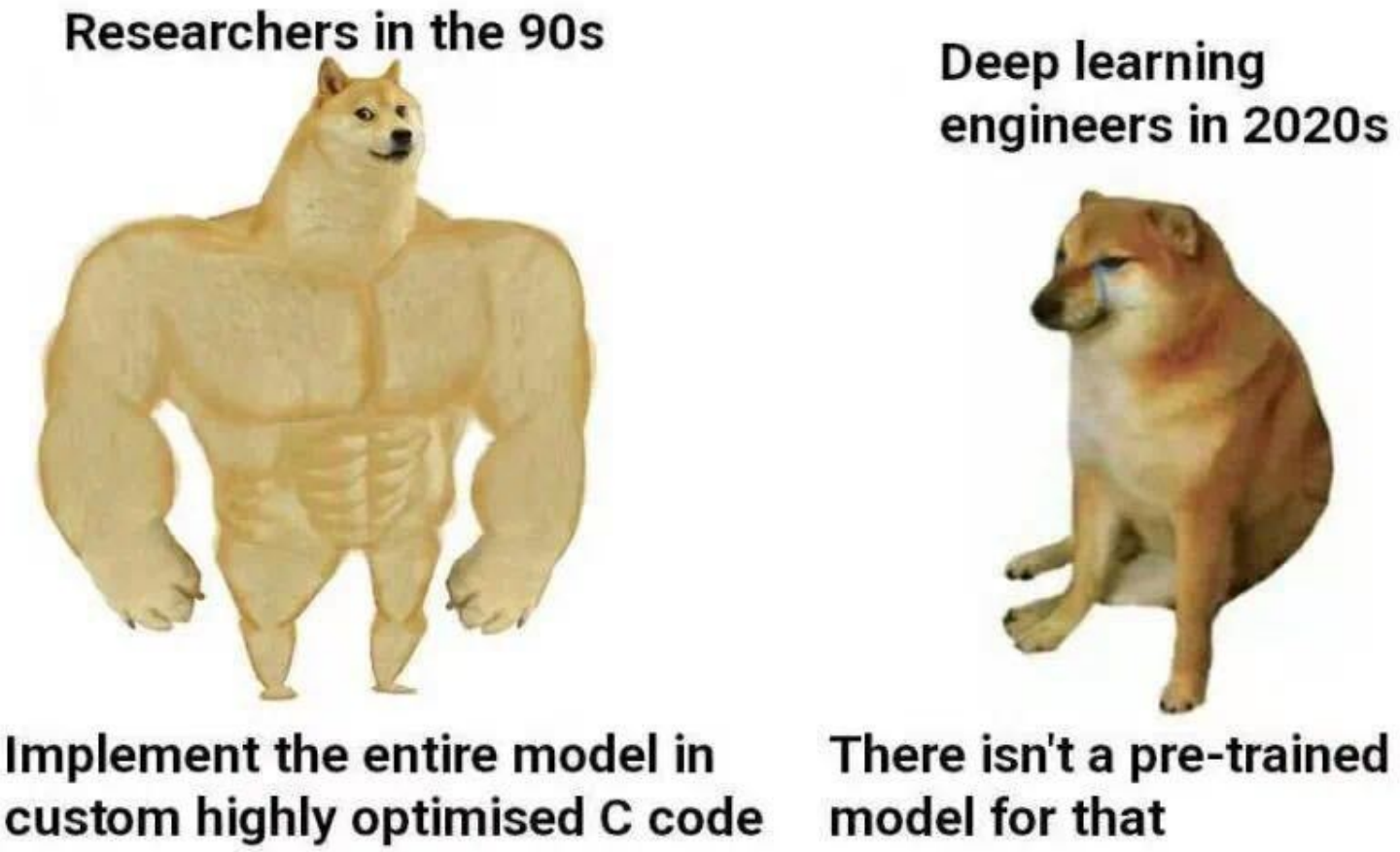
BERT: Bidirectional Encoder Representations from Transformers

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
    - TPUs are special tensor operation acceleration hardware
- Finetuning is practical and common on a single GPU
  - "Pretrain once, finetune many times."

BERT is 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**: small natural language inference corpus

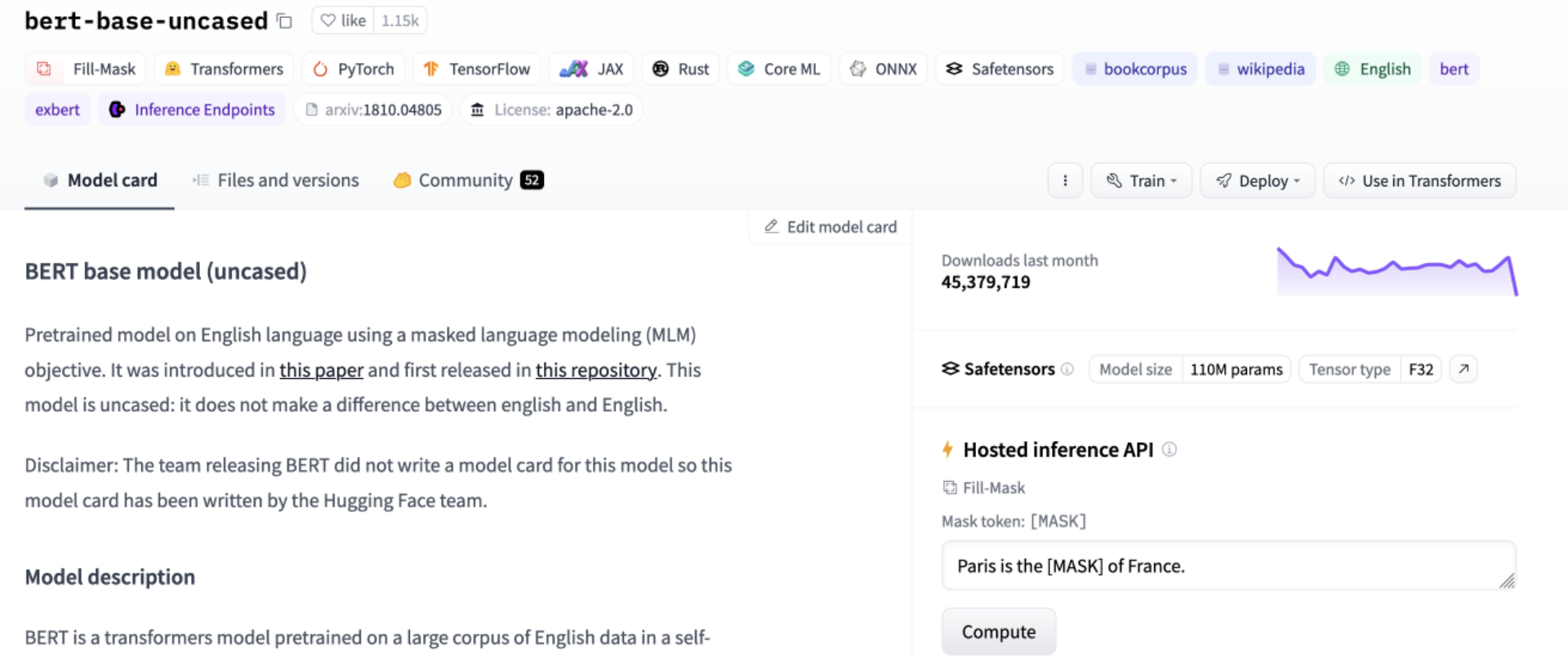
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	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**



2023 © University of Michigan

Huggingface makes models easy

- Example: https://huggingface.co/bert-base-



2023 © University of Michigan

Encoded inputs become vectors

- Can get the [CLS] token or even the average word embedding

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained("bert-base-uncased")
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
```


2023 © University of Michigan

[CLS] 是 BERT 等 Transformer 编码器里的一种**特殊标记（special token）**：

- **位置**：放在输入序列最开头（句首）。
- **作用**：它对应的隐藏向量作为**整段序列的聚合表示**，常被拿来接一个线性层做**分类/打分**（情感分类、文本匹配、检索重排等）；也常被当作"句向量/段向量"的候选表示（有时会用全序列平均池化替代）。

SI650-Week-08-Deep-Learning-for... SI650-Week-08-Deep-Learning-for...

更多细节

- 训练时，[CLS] 和普通词一样会被编码、更新参数；但它**不代表真实词**，只是"让模型有个专门的槽位来汇聚全局信息"。
- 句对任务通常格式：`[CLS] sent1 [SEP] sent2 [SEP]`，再用 `[CLS]` 向量做分类/相关性判断。
- 抽取式问答一般用起止位置预测，不直接用 `[CLS]`，但有些设置会用它判"无答案"。
- 做句向量时，**[CLS] 还是平均池化谁更好**取决于是否为该目的做过微调；直接拿预训练 BERT 的 [CLS] 做句向量往往不如平均池化或专门的句向量模型。

一句话：**[CLS] 就是序列级表示的"占位符"**，方便模型把整段话的信息集中到一个向量上，再用于下游任务。

SI650-Week-08-Deep-Learning-for...

2023 © University of Michigan

Deep Learning Representations for IR

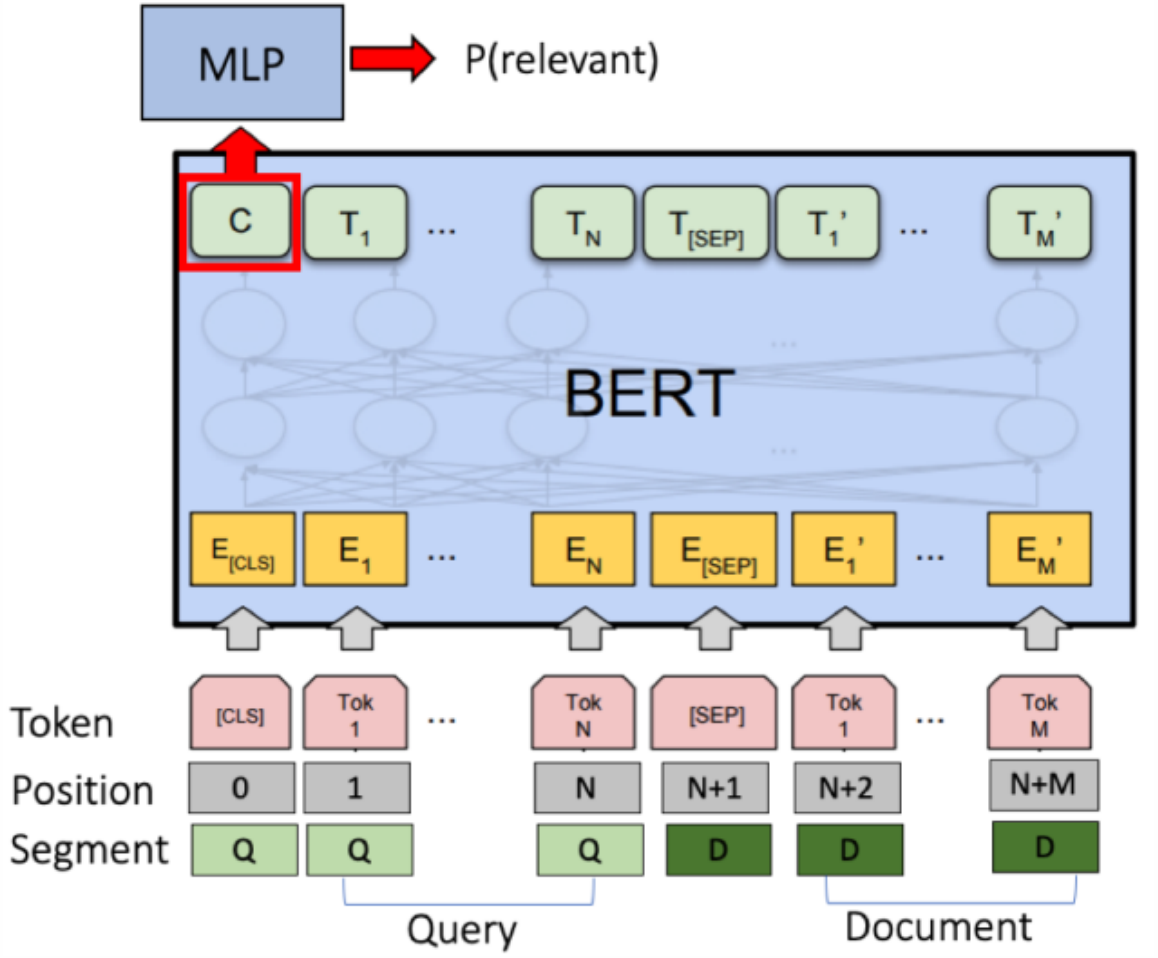
2023 © *University of Michigan*

Contextualized Representations for IR

- Example Query: “nova for sale”
  - Probably searching for the car
- Documents:
  - “A nova (plural novae or novas) is a transient astronomical event that causes the sudden appearance of a bright, apparently "new" star…”
  - “The Chevrolet Chevy II/Nova is a small automobile manufactured by Chevrolet…”
- Idea: vector representations that take contextual meaning into account can give better retrieval

2023 © University of Michigan

Use BERT to encode document and query jointly



2023 © University of Michigan

How to handle long documents?

- Problem: BERT is $O(n^2)$ in the length of the input
  - Most documents are hundreds of tokens
- Idea:
  - Break documents into overlapping passages
  - Score relevance of each pass
  - Aggregate passages' scores somehow…
    - max? mean? sum?

2023 © University of Michigan

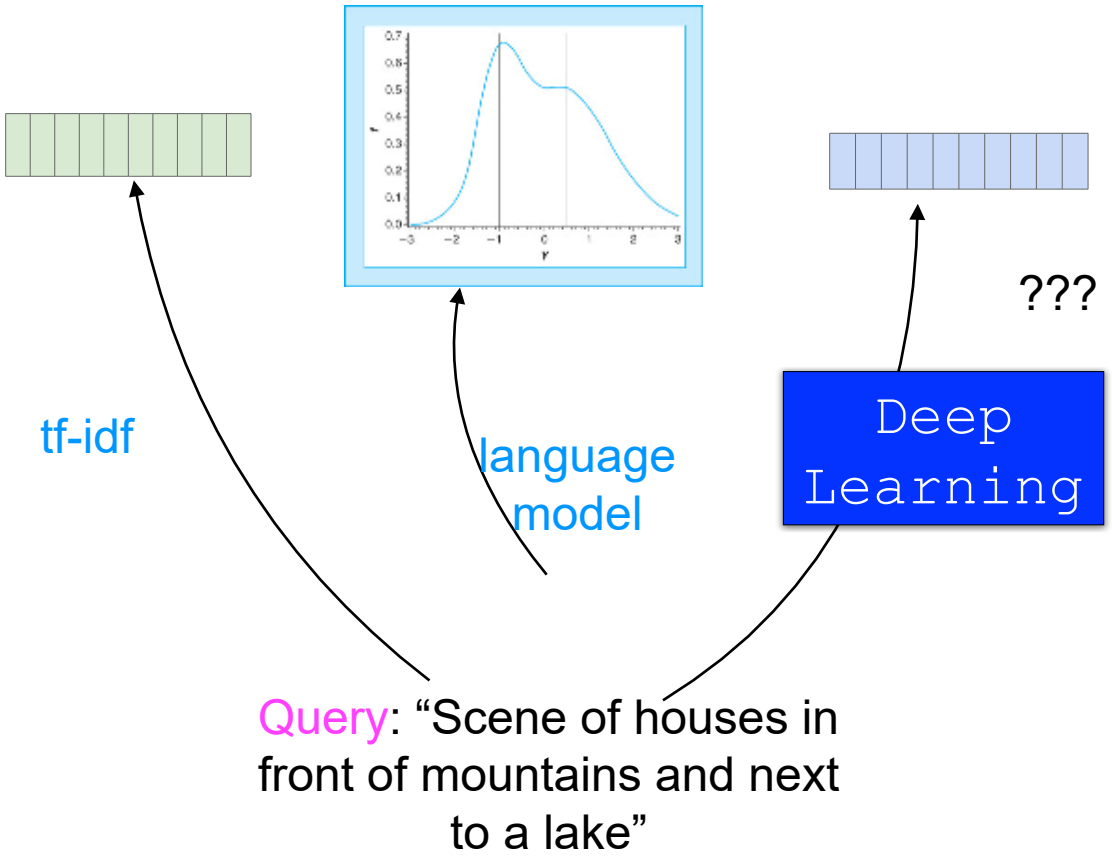
BERT for IR

Table 2: Search accuracy on Robust04 and ClueWeb09-B. † indicates statistically significant improvements over Coor-Ascent by permutation test with p< 0.05.

	nDCG@20			
	Robust04		ClueWeb09-B	
Model	Title	Description	Title	Description
BOW	0.417	0.409	0.268	0.234
SDM	0.427	0.427	0.279	0.235
RankSVM	0.420	0.435	0.289	0.245
Coor-Ascent	0.427	0.441	**0.295**	0.251

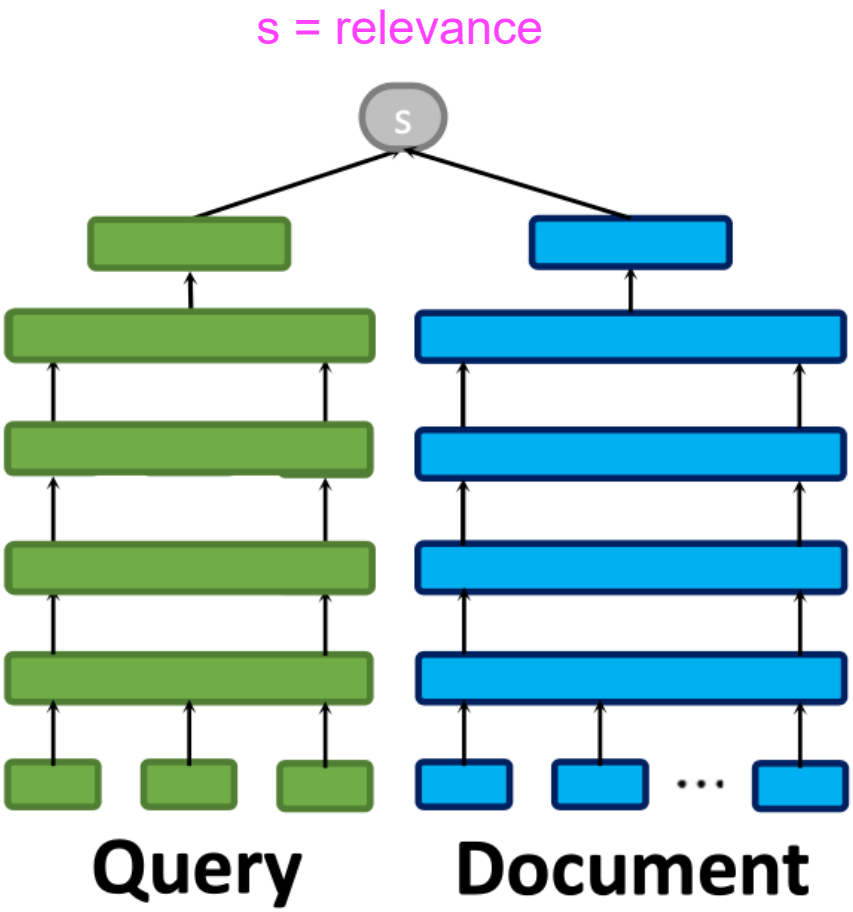
2023 © *University of Michigan*

We can think of an encoder as a function of some input to a vector output



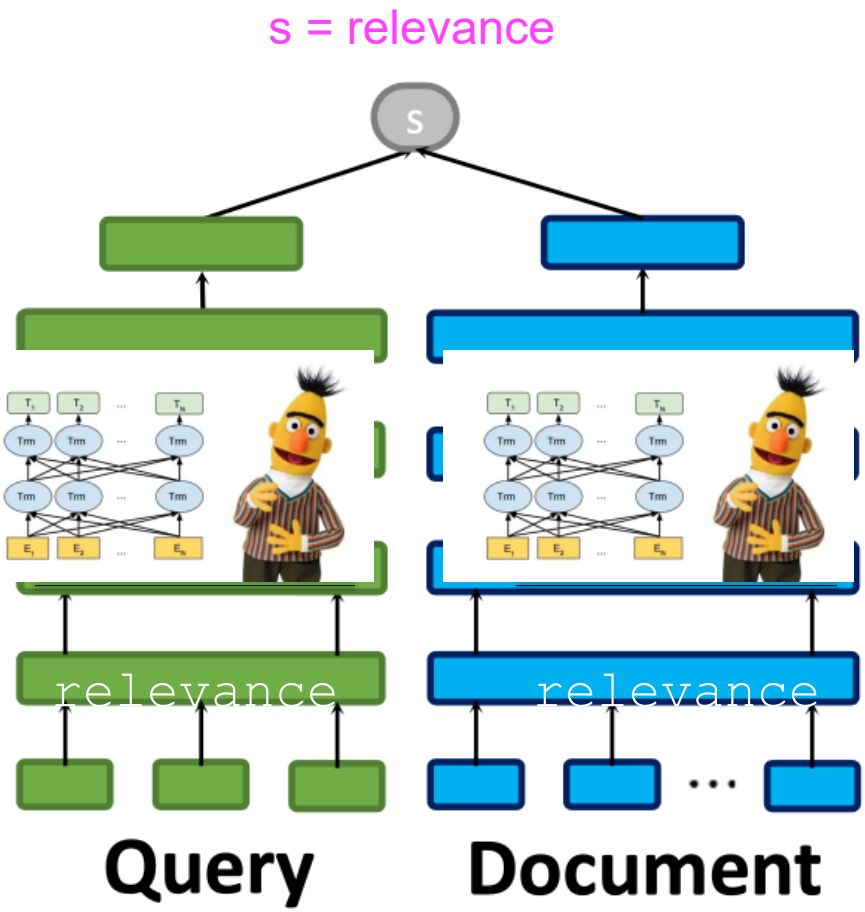
2021 © *University of Michigan*

We could try another formulation that *learns* how to encode queries and documents



2021 © *University of Michigan*

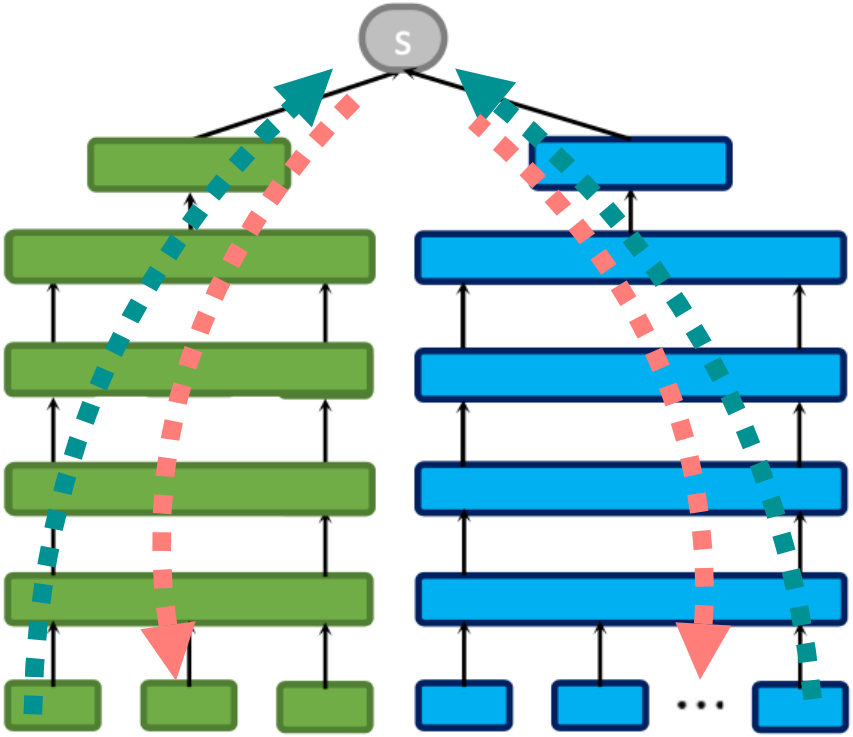
We could try another formulation that *learns* how to encode queries and documents



2021 © *University of Michigan*

By training to predict relevance, we update how the models learn to represent a query and document

Predict



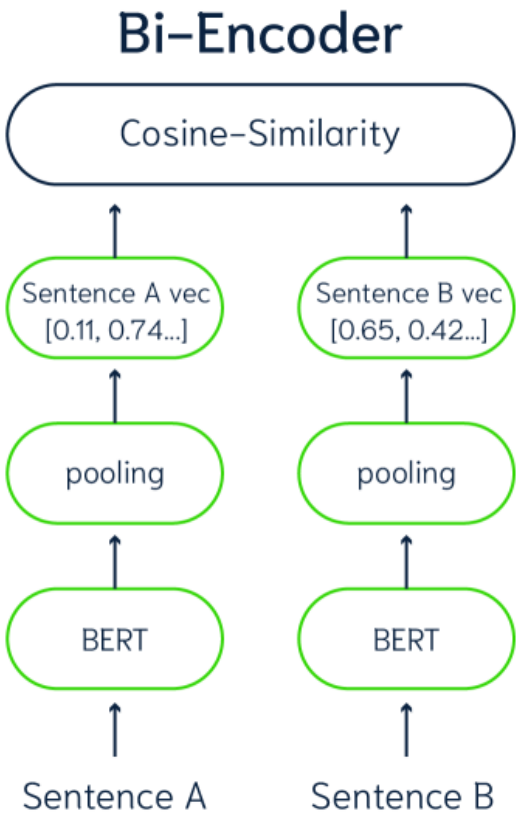
Update weights with Stochastic Gradient Descent

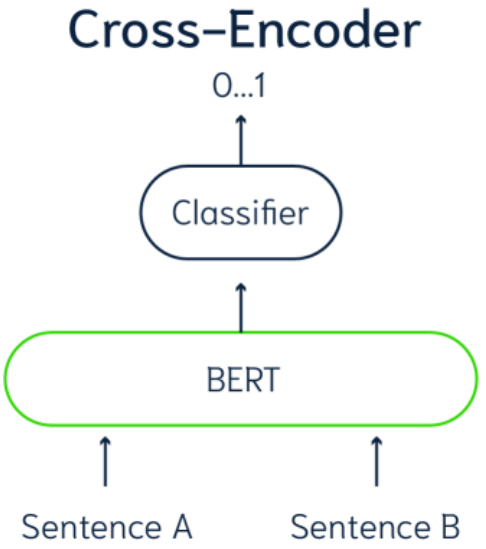
Query: “How do magnets work?”

Document: “Magnetized metals consist of …”

2022 © University of Michigan

IR Architectures: Bi-Encoder vs. Cross-Encoder





2022 © University of Michigan

量/分数汇聚成更短的表示的一步，用于后续打分或分类。它解决"长度不一、

token 表示做池化得到一个固定维度向量
效 token 取平均（最常用、稳健）
最大，突出最强信号
用 [CLS] 的向量作为序列表示（需为此目标微调）
ing-for...
用 [CLS] 接 MLP 得相关性分数，也可在需要时对token分数再做聚合
...。
g：不是对 token 向量做均值/最大，而是
相似度矩阵；
词相似度区间做"软计数"（Soft-TF）；
送入学习排序层输出最终分数 SI650-Week-08-Deep-Learning-for... 。

好！给你三个直观小例子👇

例子1：Mean pooling（最常见）

句子："**red car**"

设两个 token 的向量分别是：

- red → $[1,\ 0]$
- car → $[0,\ 1]$

Mean pooling：对每维取平均

$$h_{\text{mean}} = \frac{[1,0]+[0,1]}{2} = [0.5,\ 0$$

得到一个固定长度的句向量，可拿去跟文档向量做余弦相似度/分类。

例子2：Max pooling（取最强信号）

句子："**not bad**"

向量：

- not → $[0.2,\ 0.1,\ 0.9]$
- bad → $[0.8,\ 0.3,\ 0.2]$

Max pooling：逐维取最大

$$h_{\text{max}} = [\max(0.2, 0.8),\ \max(0.1,$$ ↓ $$),\ \max(0.9, 0$$

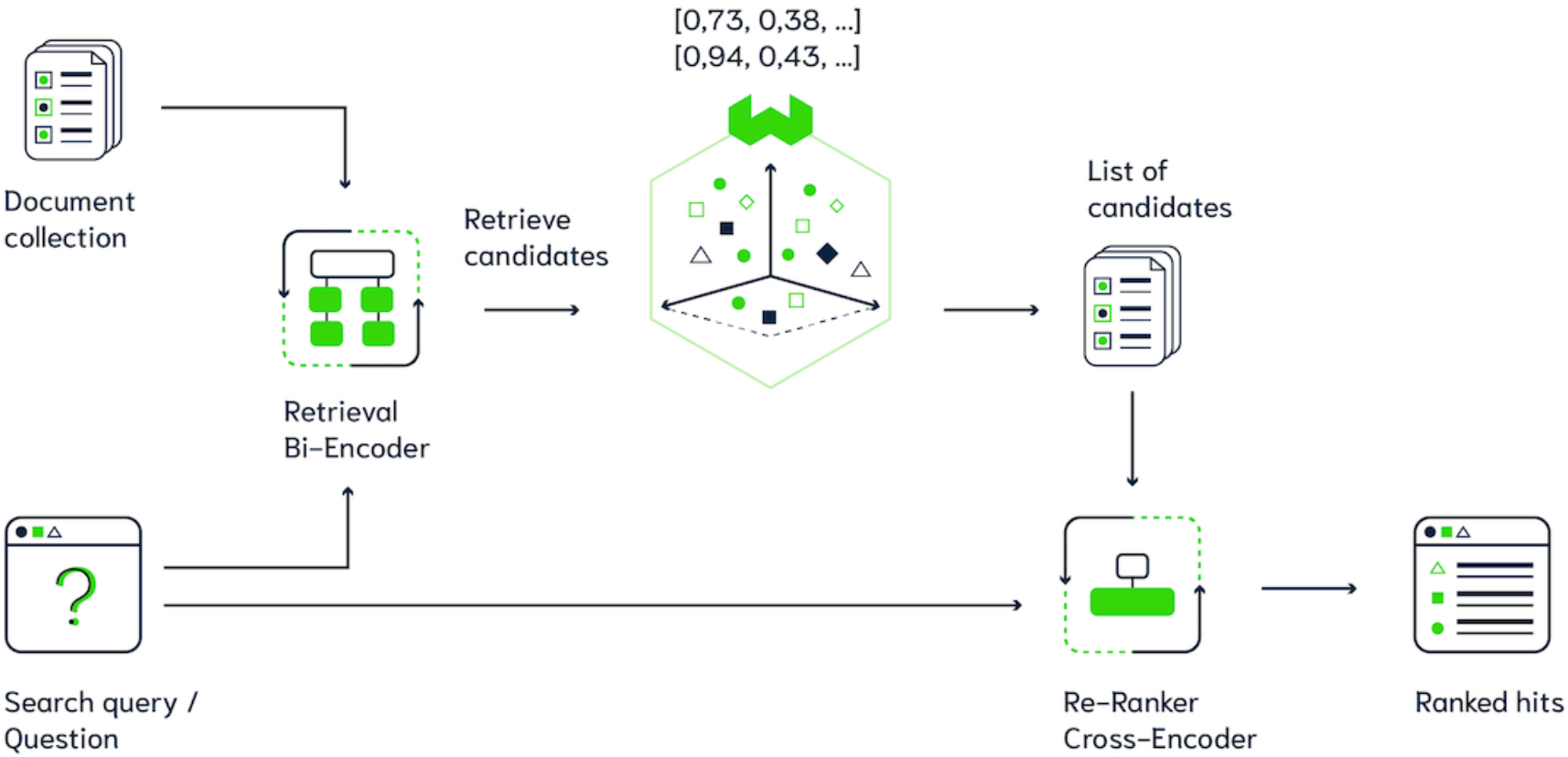
2022 © *University of Michigan*

Architecture Trade-offs

- Bi-encoders:
  - Good: Can precompute document vectors
  - Good: Fast similarity comparison: it's just vector similarity!
  - Bad: No query-document word interaction
- Cross-encoder:
  - Good: Can better interpret/compare query and document contents
  - Bad: Requires re-encoding Q+D every time (slow)

2023 © *University of Michigan*

Both setups are are typically used together in practice



2023 © University of Michigan

Brief look into how this will work in practice… (optional)

You will look into this in more detail during the discussion sections.

2023 © *University of Michigan*

Beyond BERT

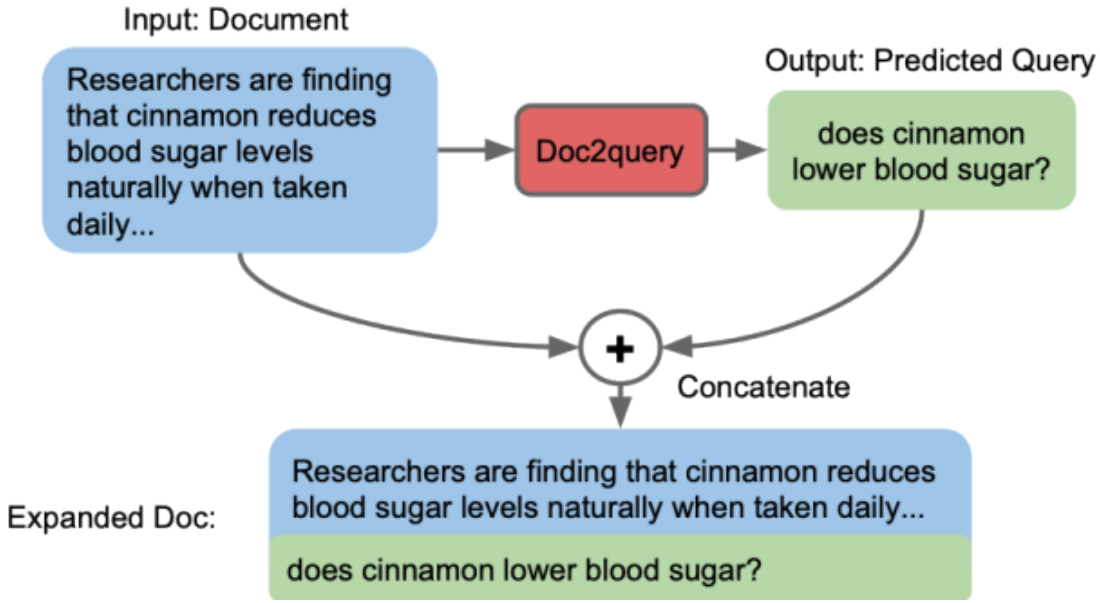
2023 © *University of Michigan*

Doc2Query Idea

- Problem: Deep learning is slow and expensive
- Idea: Could we use deep learning to augment our documents at indexing time, rather than query time
  - Retrieval stays the same (BM25, L2R, …)

2023 © University of Michigan

Augmenting Documents with Doc2Query



2023 © *University of Michigan*

How to predict queries for documents??

- Use an encoder-decoder framework similar to translation:
  - encode document, decode queries
- Truncate documents to 400 tokens, queries to 100
- Train on relevant query-document pairs

2023 © *University of Michigan*

Performance when adding machine-generated queries

	TREC-CAR MAP	MS MARCO MRR@10	
	Test	Test	Dev
Single Duet v2 (Mitra and Craswell, 2019b)	-	24.5	24.3
Co-PACRR♠ (MacAvaney et al., 2017)	14.8	-	-

2023 © University of Michigan

Things not covered today

- What can go wrong when using pre-trained language models (covered next week!)
- How these models are actually trained
- Many technical details
- Lots of recent advances in neural networks

2023 © *University of Michigan*

What You Should Know

- Know what deep learning is about
- Know the intuitions of why deep learning works
- Know the basic models of deep learning and how they are applied to text
- Know the basic ideas of Word2Vec
- Know how deep/large language models can be applied to IR

2023 © *University of Michigan*