

# **Fine-tuning and information retrieval with LMs**

**NLP Week 9**

**Thanks to Dan Jurafsky for most of the slides this week!**

# Plan for today

1. Contextualized embeddings
2. Fine-tuning for classification
3. Fine-tuning for sequence labeling
4. Practical considerations for fine-tuning
5. Information retrieval powered by language models
6. *Group exercises*

# Problem with static embeddings (word2vec)

They are **static**! The embedding for a word doesn't reflect how its meaning changes in context.

The chicken didn't cross the road because **it** was too tired

What is the meaning represented in the static embedding for "it"?

# Contextual Embeddings

- **Intuition:** a representation of meaning of a word should be different in different contexts!
- **Contextual Embedding:** each word has a different vector that expresses different meanings depending on the surrounding words
- **How to compute contextual embeddings?**
  - Deep neural networks
  - Bi-LSTMs
  - Self-attention

# Contextual Embeddings

The chicken didn't cross the road because it \_\_\_\_\_

What should be the properties of "it"?

The chicken didn't cross the road because it was too **tired**

The chicken didn't cross the road because it was too **wide**

At this point in the sentence, it's probably referring to either the chicken or the street

# Word sense

Words are ambiguous

A **word sense** is a discrete representation of one aspect of meaning

**mouse<sup>1</sup>** : .... a *mouse* controlling a computer system in 1968.

**mouse<sup>2</sup>** : .... a quiet animal like a *mouse*

**bank<sup>1</sup>** : ...a *bank* can hold the investments in a custodial account ...

**bank<sup>2</sup>** : ...as agriculture burgeons on the east *bank*, the river ...

Contextual embeddings offer a continuous high-dimensional model of meaning that is more fine grained than discrete senses.

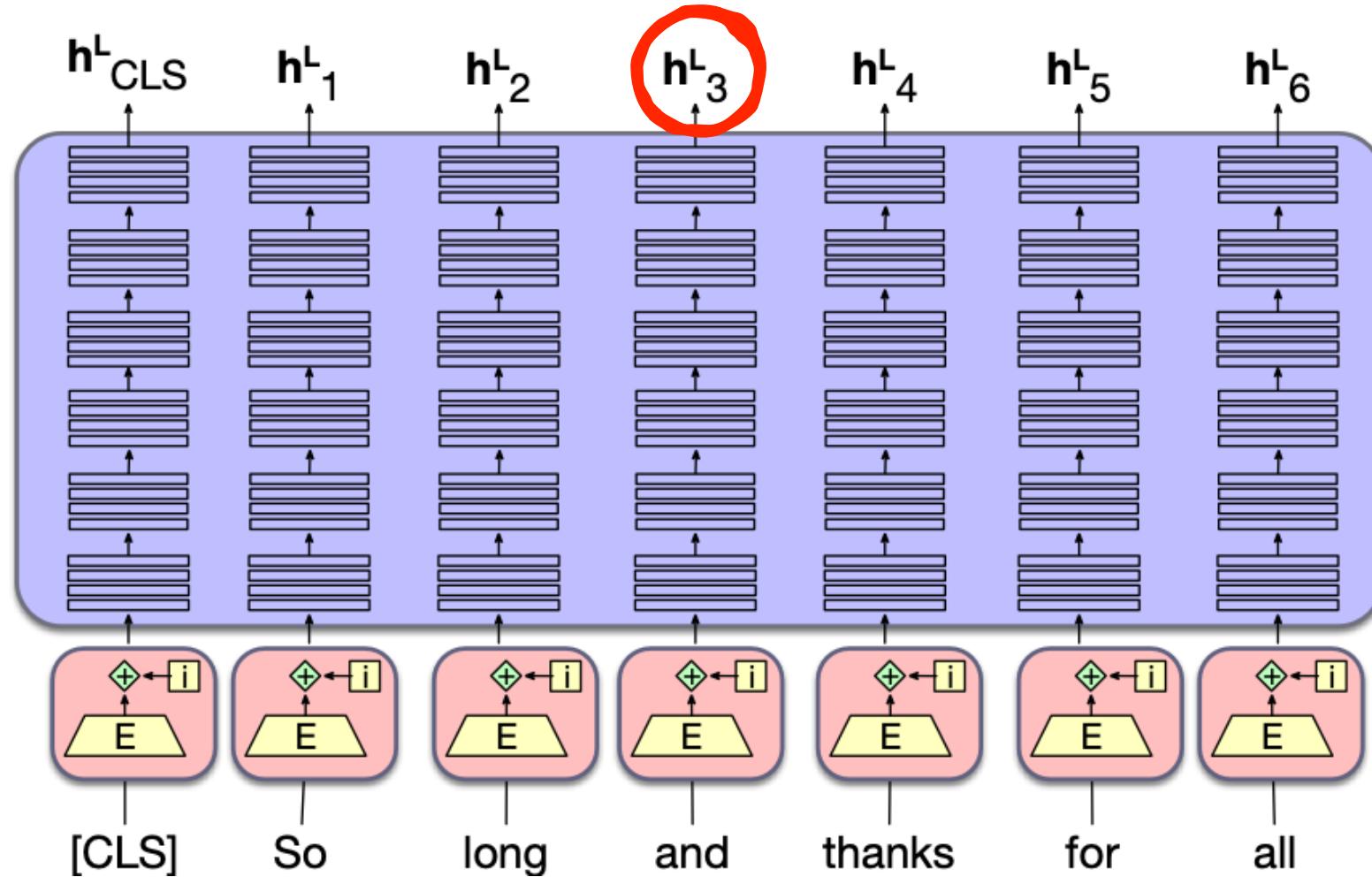
# Static vs Contextual Embeddings

Static embeddings represent **word types** (dictionary entries)

Contextual embeddings represent **word instances** (one for each time the word occurs in any context/sentence)

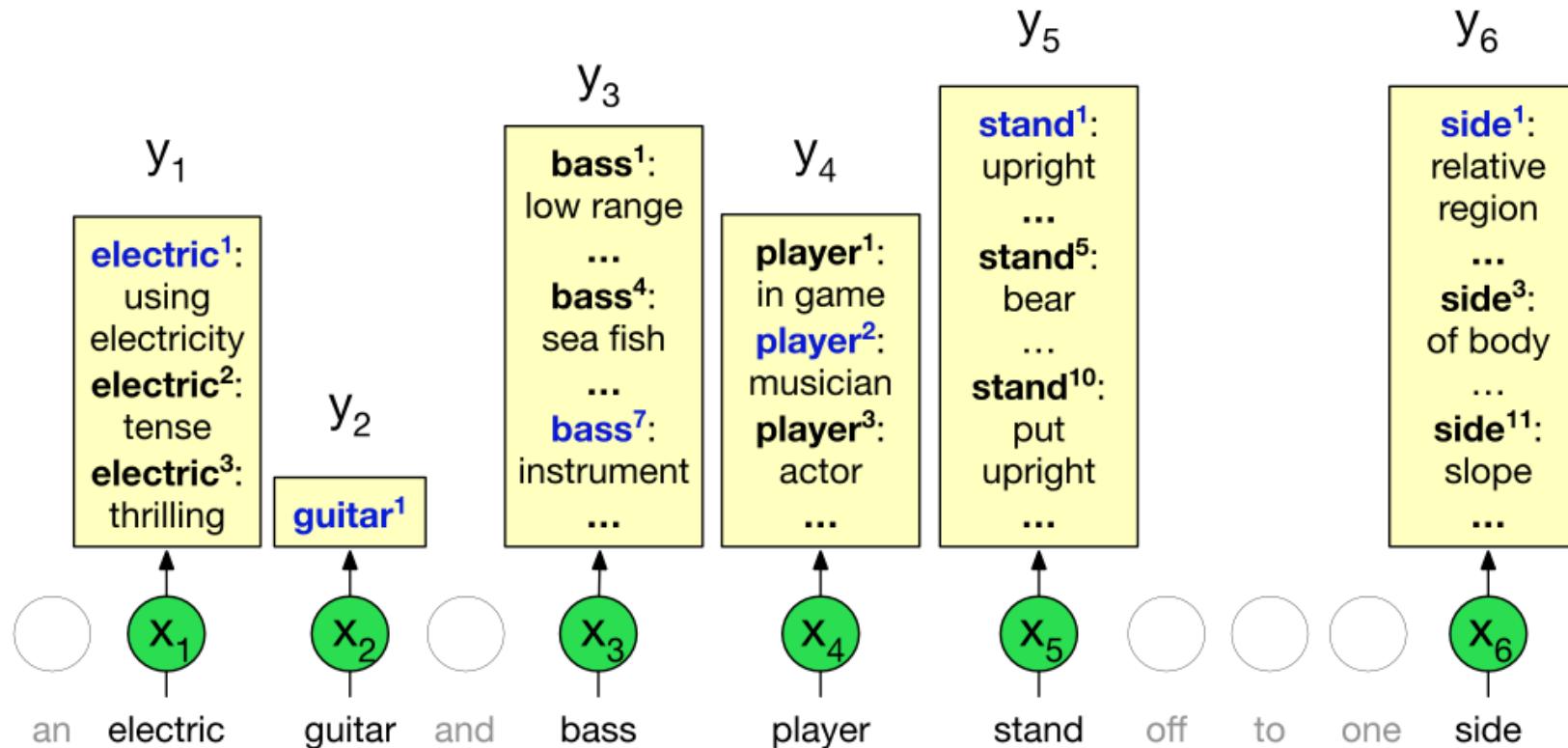


# Contextual Embeddings from BERT



# Word sense disambiguation (WSD)

The task of selecting the correct sense for a word.



# 1-nearest neighbor algorithm for WSD

Melamud et al (2016), Peters et al (2018)

At training time, take a sense-labeled corpus like SEMCOR

Run corpus through BERT to get contextual embedding for each token

- E.g., pooling representations from last 4 BERT transformer layer

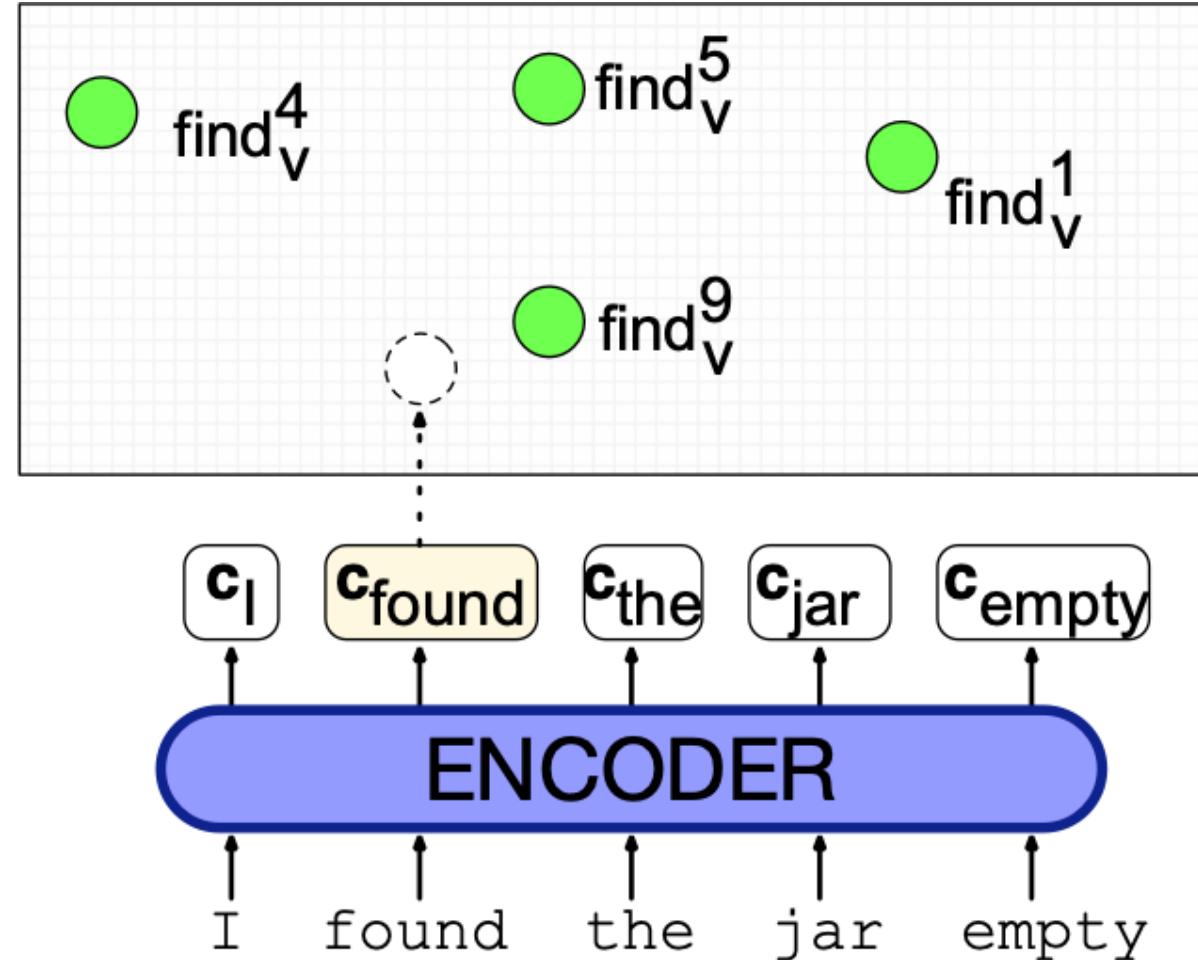
Then for each sense  $s$  of word  $w$  for  $n$  tokens of that sense, pool embeddings:

$$\mathbf{v}_s = \frac{1}{n} \sum_{i=0}^n \mathbf{v}_i \quad \forall \mathbf{v}_i \in \text{tokens}(s)$$

At test time, given a token of a target word  $t$ , compute contextual embedding  $\mathbf{t}$  and choose its nearest neighbor sense from training set

$$\text{sense}(t) = \underset{s \in \text{senses}(t)}{\operatorname{argmax}} \cos(\mathbf{t}, \mathbf{v}_s)$$

# 1-nearest neighbor algorithm for WSD



# Similarity and contextual embeddings

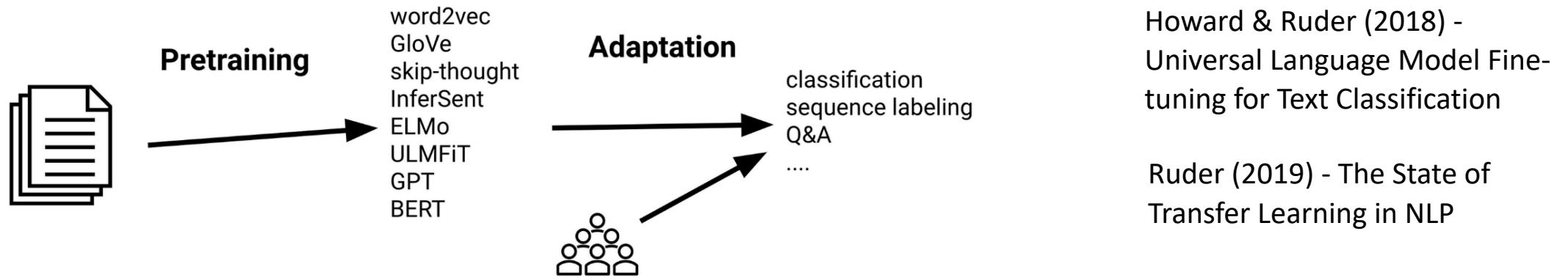
- We generally use cosine as for static embeddings
- But some issues:
  - Contextual embeddings tend to be **anisotropic**: all point in roughly the same direction so have high inherent cosines (Ethayarajh 2019)
  - Cosine measure are dominated by a small number of "rogue" dimensions with very high values (Timkey and van Schijndel 2021)
  - Cosine tends to underestimate human judgments on similarity of word meaning for very frequent words (Zhou et al., 2022)

# Fine-tuning pre-trained models

- Recap: Basic ingredients for training ML models
  - A model  $p_{\theta}(y \mid \mathbf{x})$
  - A dataset  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$
  - An objective function, e.g., cross-entropy loss
  - A learning algorithm: gradient descent

# Fine-tuning pre-trained models

- So far, models were initialized randomly, i.e.,  $\theta \sim \mathcal{N}(\mu, \sigma)$
- Fine-tuning → Initialize weights from a pre-trained model
- Why should this work? → Transfer learning
  - Pre-training encodes useful information about language
  - This is helpful for various NLP downstream tasks



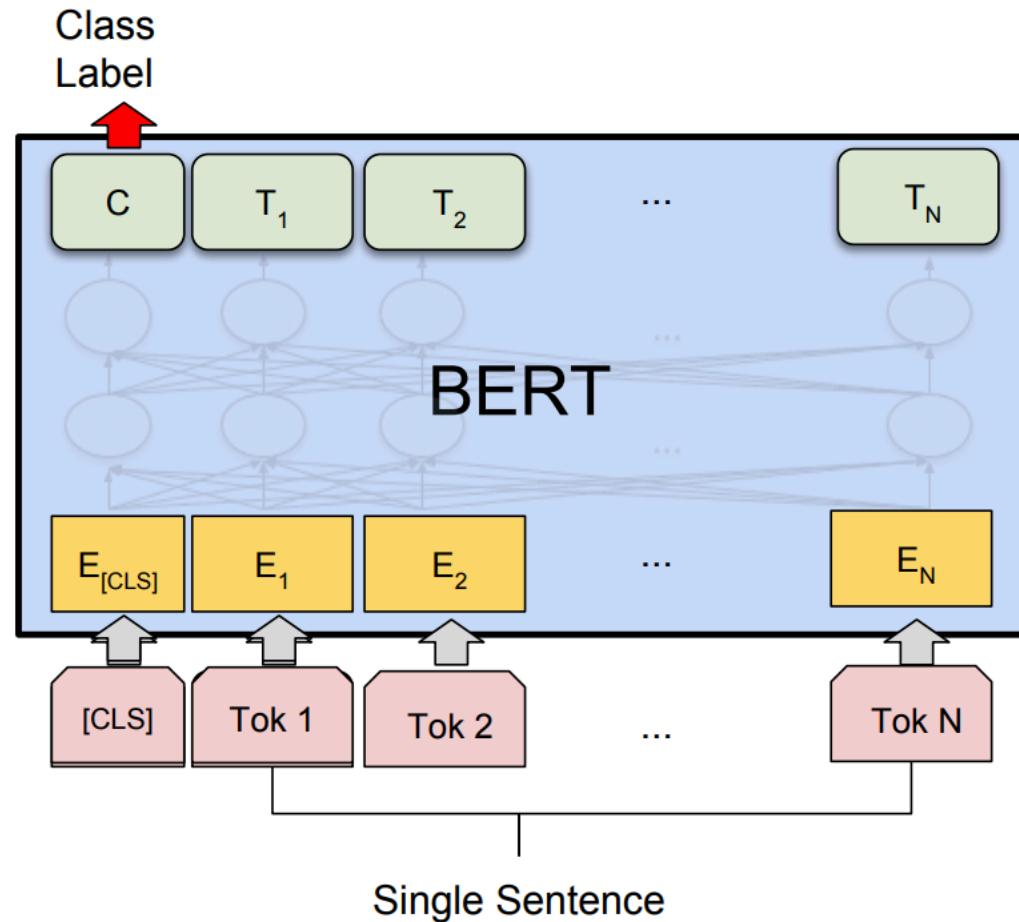
# Fine-tuning pre-trained models

- Nowadays, fine-tuning means many things
  - Adaptation to a single task, e.g., classification
  - Continual pre-training
  - Language adaptation
  - Instruction fine-tuning
  - Safety fine-tuning, RLHF
  - Post-training
  - Mid-training
  - Reinforcement fine-tuning
  - etc.

# Fine-tuning pre-trained models

- Nowadays, fine-tuning means many things
  - **Adaptation to a single task**, e.g., classification
  - Continual pre-training
  - Language adaptation
  - Instruction fine-tuning
  - Safety fine-tuning, RLHF
  - Post-training
  - Mid-training
  - Reinforcement fine-tuning
  - etc.

# Using BERT for classification

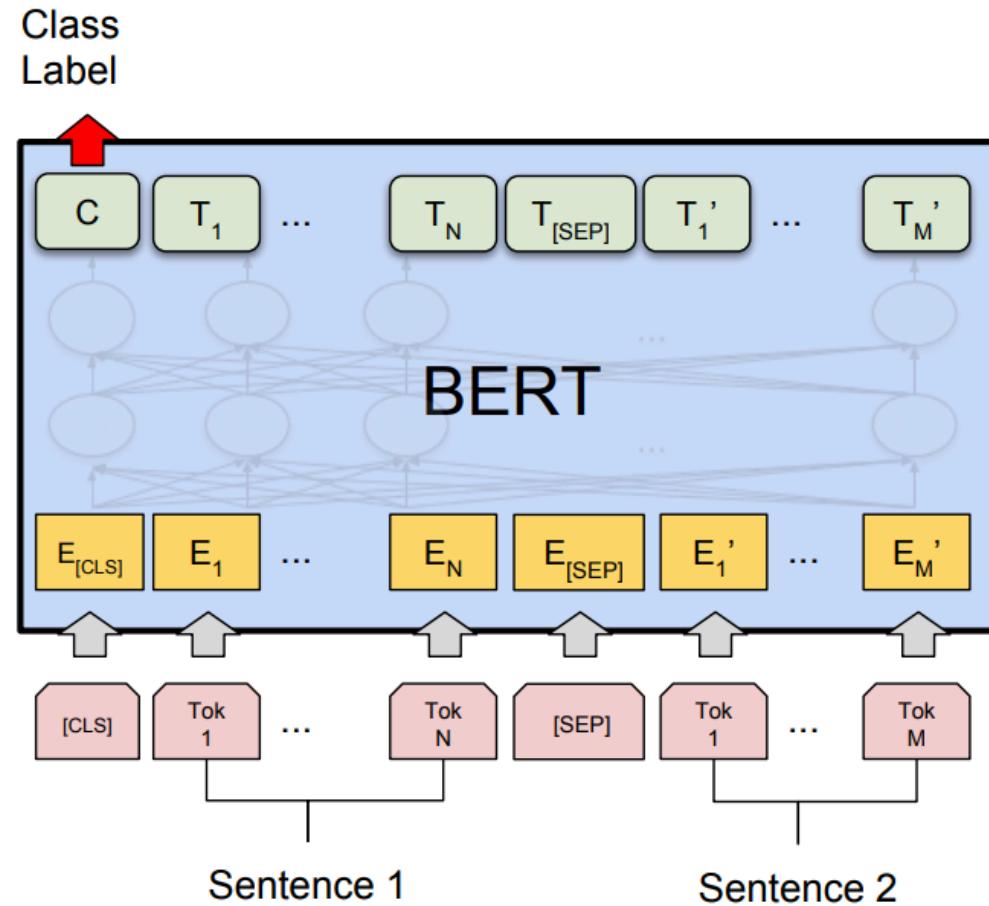


# Sentence classification

Assign a label to a single sentences:

- **Sentiment analysis** (does a given sentence have positive or negative sentiment?)
- **Acceptability judgements** (is a given sentence grammatical?)
- **Hate speech detection** (e.g, does a given sentence express hate or encourages violence?)

# Using BERT for sentence pair classification



# Sequence-Pair classification

Assign a label to pairs of sentences:

- **Paraphrase detection** (are the two sentences paraphrases of each other?)
- **Logical entailment** (does sentence A logically entail sentence B?)
- **Discourse coherence** (how coherent is sentence B as a follow-on to sentence A?)

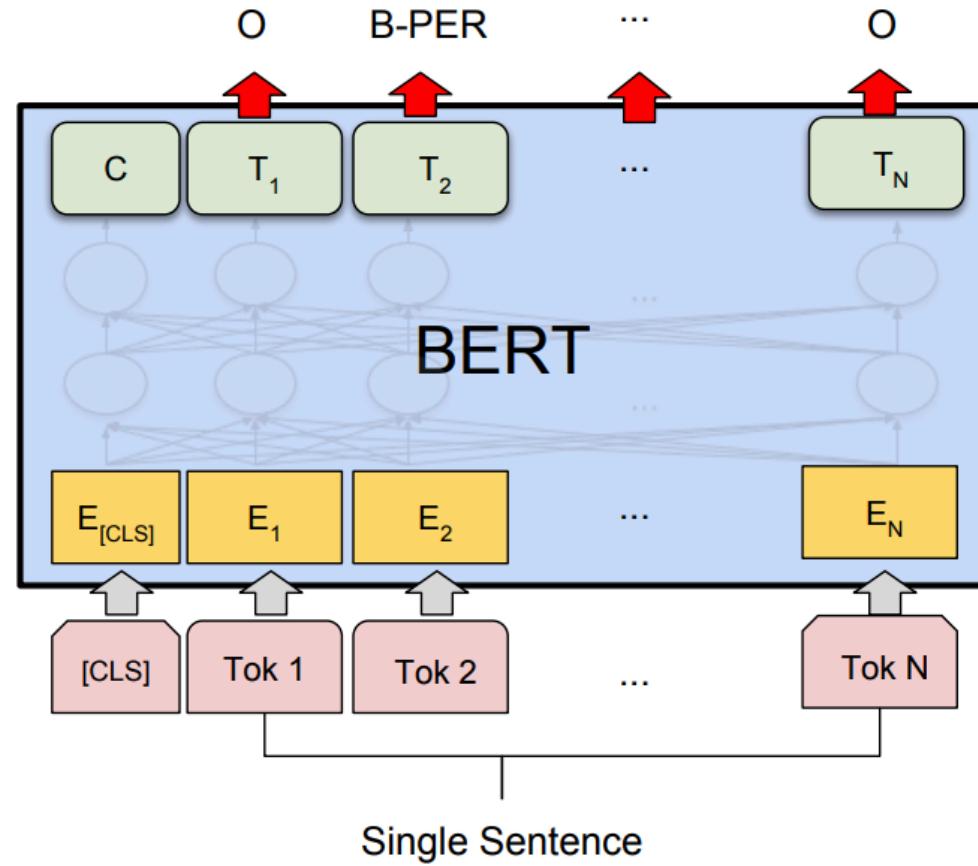
# Example: Natural Language Inference

Pairs of sentences are given one of 3 labels

- Neutral
  - a: Jon walked back to the town to the smithy.
  - b: Jon traveled back to his hometown.
- Contradicts
  - a: Tourist Information offices can be very helpful.
  - b: Tourist Information offices are never of any help.
- Entails
  - a: I'm confused.
  - b: Not all of it is very clear to me.

How? → pass the premise/hypothesis pairs through BERT and use the output vector for the [CLS] token as the input to the classification head .

# Using BERT for sequence labeling



# Using BERT for sequence labeling

Assign a label from a small fixed set of labels to each token in the sequence.

- Named entity recognition
- Part of speech tagging

# Named Entity Recognition

A **named entity** is anything that can be referred to with a proper name: a person, a location, an organization

**Named entity recognition (NER)**: find spans of text that constitute proper names and tag the type of the entity

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	Turing is a giant of computer science.
Organization	ORG	companies, sports teams	The IPCC warned about the cyclone.
Location	LOC	regions, mountains, seas	Mt. Sanitas is in Sunshine Canyon.
Geo-Political Entity	GPE	countries, states	Palo Alto is raising the fees for parking.

# Named Entity Recognition

Citing high fuel prices, [ORG United Airlines] said [TIME Friday] it has increased fares by [MONEY \$6] per round trip on flights to some cities also served by lower-cost carriers. [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said. [ORG United], a unit of [ORG UAL Corp.], said the increase took effect [TIME Thursday] and applies to most routes where it competes against discount carriers, such as [LOC Chicago] to [LOC Dallas] and [LOC Denver] to [LOC San Francisco].

# BIO Tagging

Ramshaw and Marcus (1995)

A method that lets us turn a segmentation task (finding boundaries of entities) into a classification task

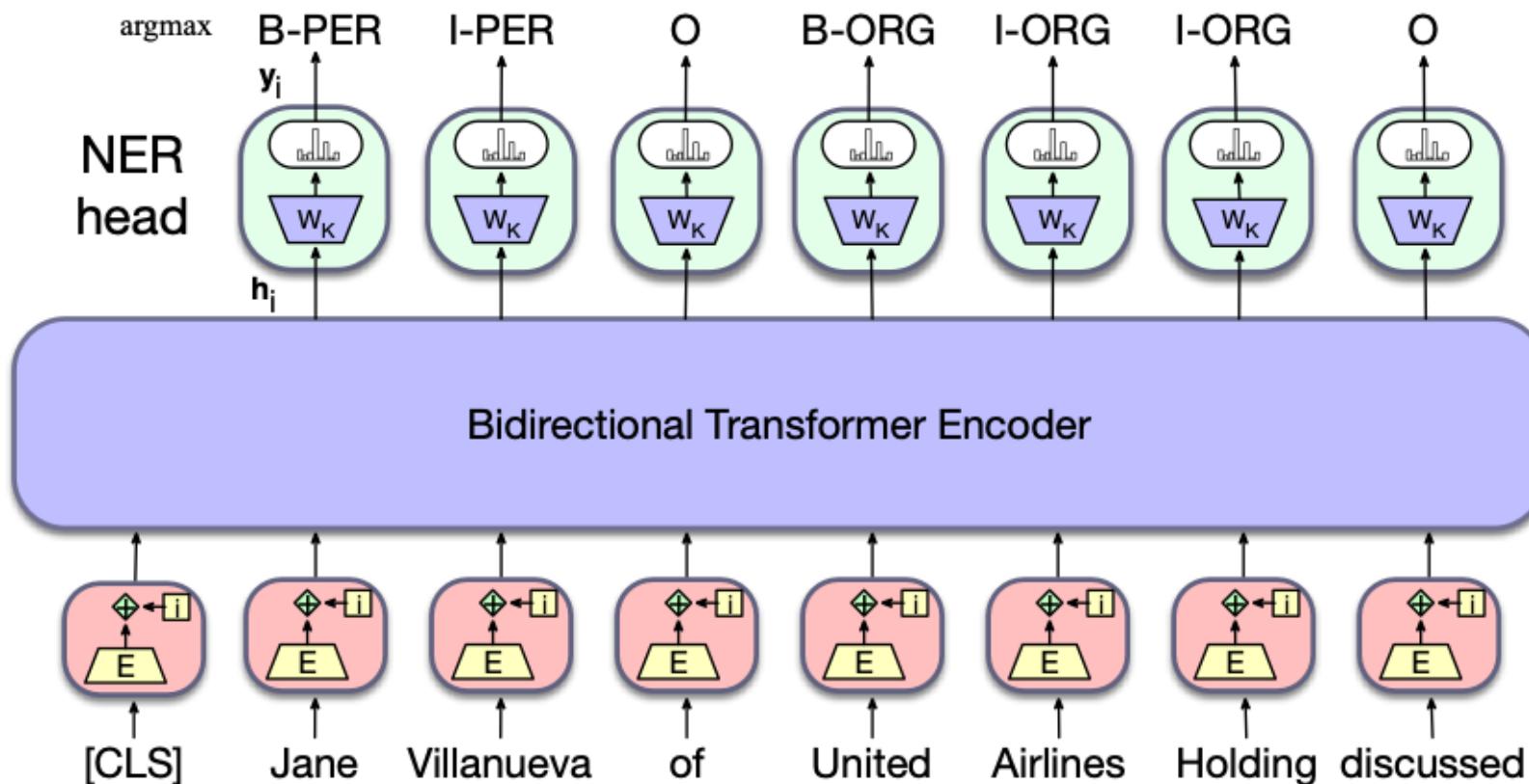
[PER Jane Villanueva ] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago ] route.

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

# Sequence labeling

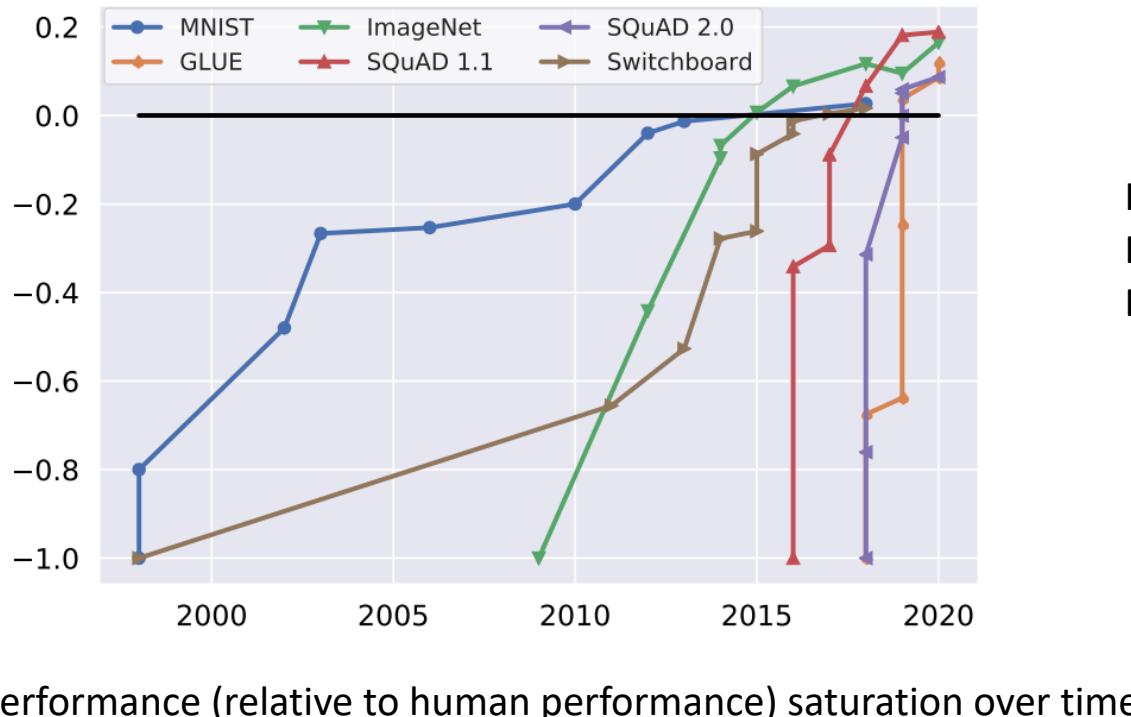
$$\mathbf{y}_i = \text{softmax}(\mathbf{h}_i^L \mathbf{W}_K)$$

$$t_i = \text{argmax}_k(\mathbf{y}_i)$$



# Evaluating pre-trained language models

- How good are fine-tuning language models?
- Glue, SuperGLUE, SQuAD, MasakhaNER, etc.

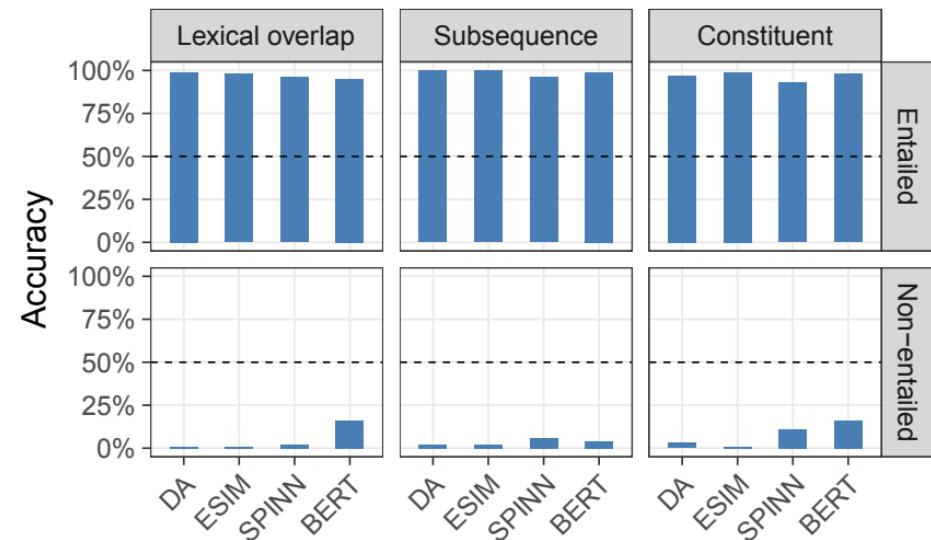


Kiela et al. (2021) -  
Dynabench: Rethinking  
Benchmarking in NLP

# Practical considerations: Out-of-domain generalization

- Do models *truly* learn the task they are supposed to learn?
- What happens if we evaluate models *outside the training distribution*?
- Models can learn shortcuts present in the training data

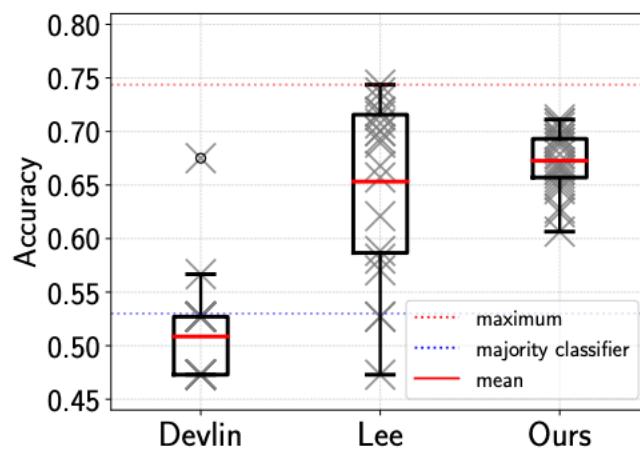
McCoy et al. (2019) - Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference



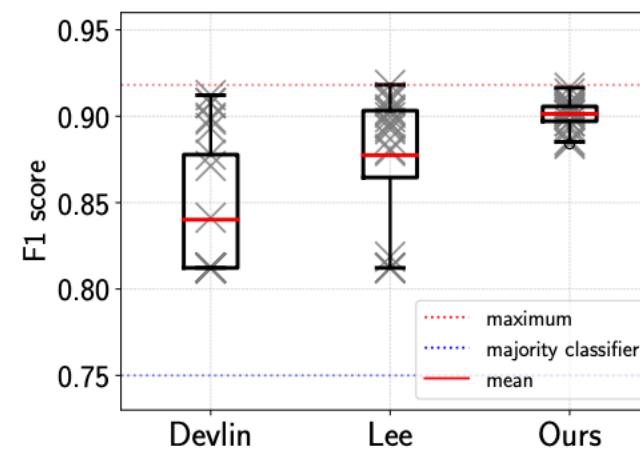
Heuristic	Premise	Hypothesis	Label
Lexical overlap heuristic	The banker near the judge saw the actor. The lawyer was advised by the actor. The doctors visited the lawyer. The judge by the actor stopped the banker.	The banker saw the actor. The actor advised the lawyer. The lawyer visited the doctors. The banker stopped the actor.	E E N N

# Practical considerations: Fine-tuning (in)stability

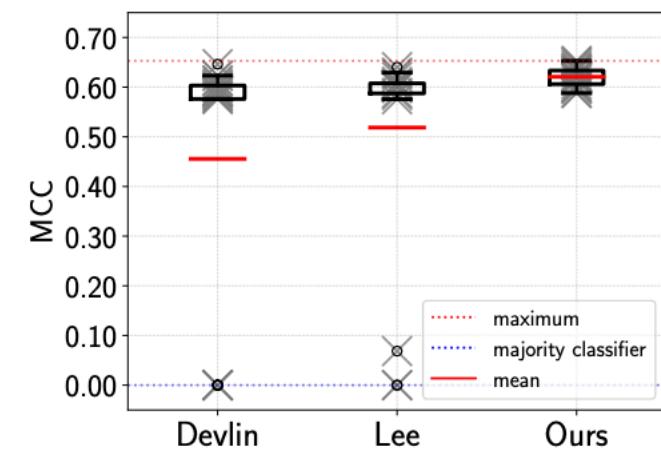
- Repeat fine-tuning many times and change only the data order + weights of the classification layer
- Substantial variations in task performance



(a) RTE



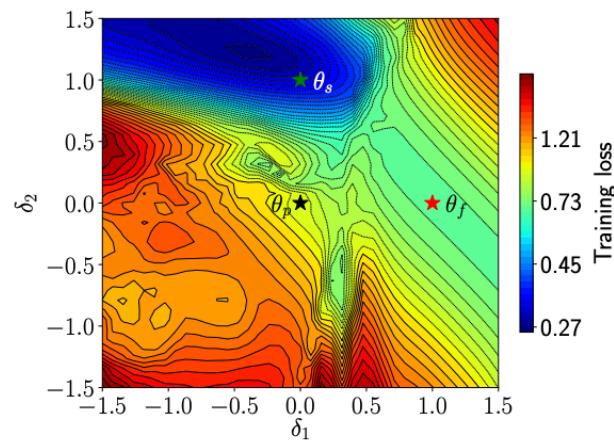
(b) MRPC



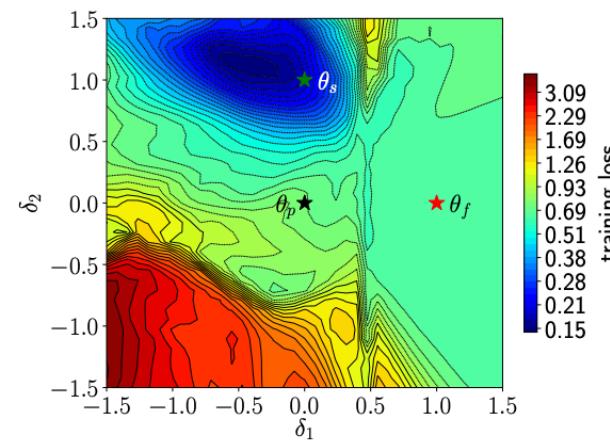
(c) CoLA

# Practical considerations: Fine-tuning (in)stability

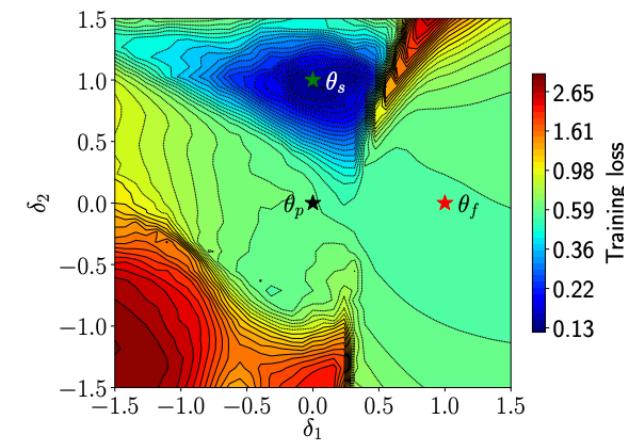
- Fine-tuning can fail due to optimization problems
- You have to choose your step-size (learning rate) carefully



(a) RTE



(b) MRPC



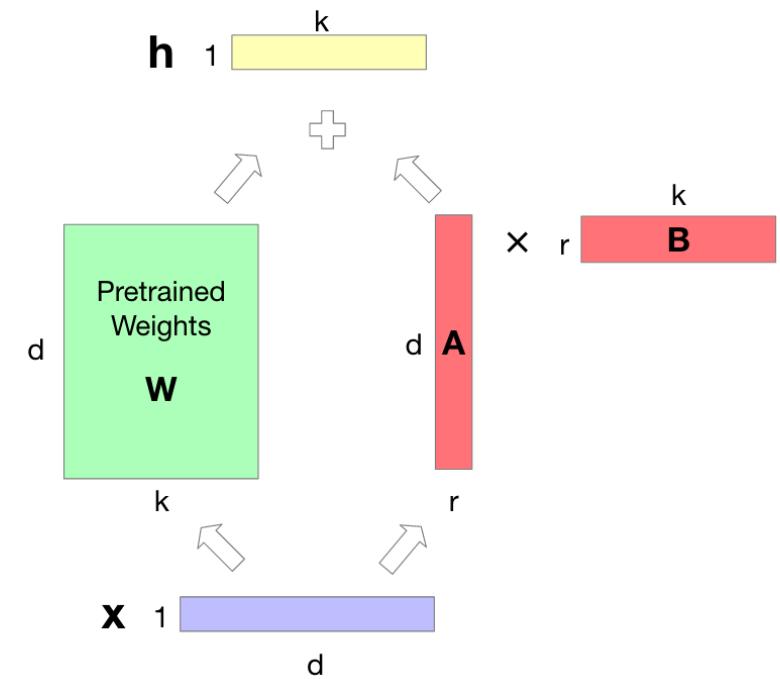
(c) CoLA

## Practical considerations: Parameter-efficient fine-tuning

- As models grow in size, fine-tuning can be expensive both in memory and time
- But do we really need to fine-tune all weights of a pre-trained model?
- There exist many method for fine-tuning only a subset of the weights and these models often achieve very similar performance compare to fine-tuning all weights

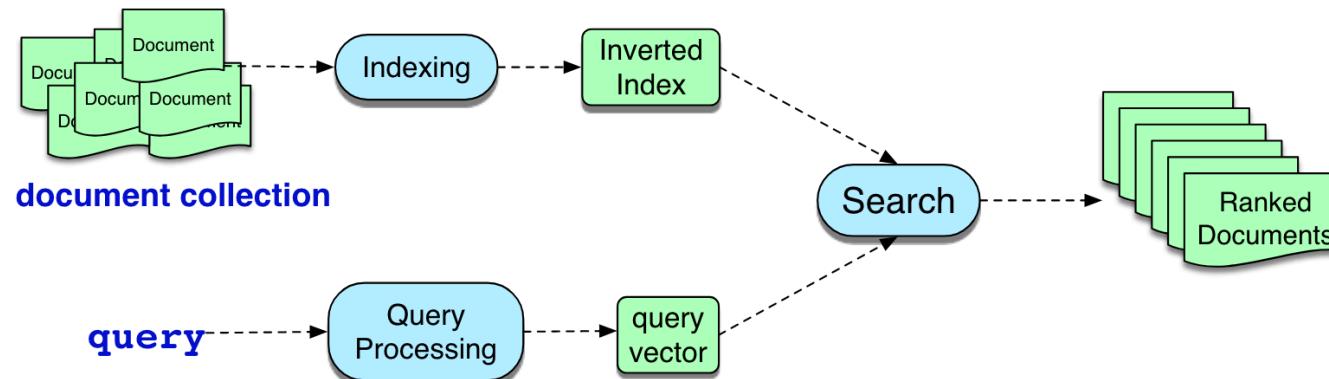
# LoRA

- Low-rank adaptation (LoRA) is a widely used parameter-efficient fine-tuning method (PEFT)
- Fine-tuning introduces changes to the models weights  
$$\mathbf{W}' = \mathbf{W} + \Delta \mathbf{W}$$
- Let's assume  $\Delta \mathbf{W}$  is low rank
- Parameterize  $\Delta \mathbf{W} = \mathbf{AB}$
- Now learn only  $\mathbf{A}$  and  $\mathbf{B}$



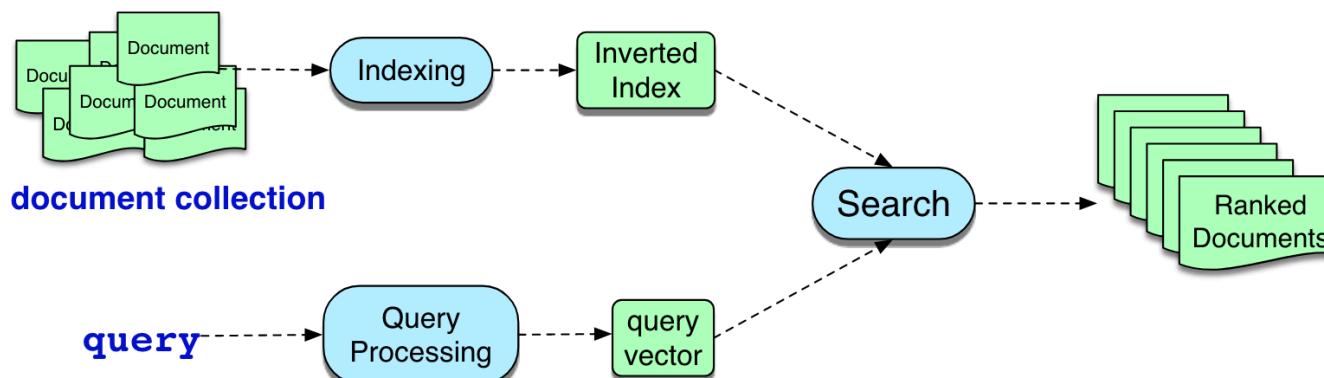
# Information retrieval

- Retrieve *media* based on a user's *information need*
- We consider a special case of IR: **ad-hoc retrieval**
- A user provides a question to a system, which then returns one or more documents from a collection of documents



# Information retrieval

- **Document:** the unit of text that the IR system indexes
- **Collection:** set of documents
- **Term:** a word (or paragraph) that occurs in the collection
- **Query:** information need expressed as a set of terms



Recall from Week 5

# Embedding types

- **There are three main types of embeddings used in NLP and they are created using three different methods:**
  1. Sparse embeddings and term-document frequency
  2. Dense embeddings and Word2Vec algorithms
  3. Contextual word embeddings and language modeling

Recall from Week 5

# Term frequency (tf) in the tf-idf algorithm

We could imagine using raw count:

$$\text{tf}_{t,d} = \text{count}(t,d)$$

But instead of using raw count, we usually squash a bit:

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if } \text{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Recall from Week 5

# Inverse document frequency (idf)

$$\text{idf}_t = \log_{10} \left( \frac{N}{\text{df}_t} \right)$$

$\text{df}_t$  is the number of documents  $t$  occurs in.  
(note this is not collection frequency: total count across all documents)

N is the total number of documents  
in the collection

# tf-idf scoring

- Weighted value for each term

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_f$$

- How do we score queries and documents?

$$\text{score}(q, d) = \cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}| |\mathbf{d}|}$$

# tf-idf scoring

$$\text{score}(q, d) = \cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}| |\mathbf{d}|}$$

- Rewrite the dot-product as a sum of products

$$\text{score}(q, d) = \sum_{t \in \mathbf{q}} \frac{\text{tf-idf}(t, q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i, q)}} \cdot \frac{\text{tf-idf}(t, d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i, d)}}$$

# tf-idf scoring

$$\text{score}(q, d) = \sum_{t \in q} \frac{\text{tf-idf}(t, q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i, q)}} \cdot \frac{\text{tf-idf}(t, d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i, d)}}$$

**Query:** sweet love

**Doc 1:** Sweet sweet nurse! Love?

**Doc 2:** Sweet sorrow

**Doc 3:** How sweet is love?

**Doc 4:** Nurse!

Query						
word	cnt	tf	df	idf	tf-idf	n'lized = tf-idf/ q
sweet	1	1	3	0.125	0.125	0.383
nurse	0	0	2	0.301	0	0
love	1	1	2	0.301	0.301	0.924
how	0	0	1	0.602	0	0
sorrow	0	0	1	0.602	0	0
is	0	0	1	0.602	0	0

$|q| = \sqrt{.125^2 + .301^2} = .326$

# tf-idf scoring

$$\text{score}(q, d) = \sum_{t \in q} \frac{\text{tf-idf}(t, q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i, q)}} \cdot \frac{\text{tf-idf}(t, d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i, d)}}$$

**Query:** sweet love

**Doc 1:** Sweet sweet nurse! Love?

**Doc 2:** Sweet sorrow

**Doc 3:** How sweet is love?

**Doc 4:** Nurse!

word	Document 1					Document 2				
	cnt	tf	tf-idf	n'lized	$\times q$	cnt	tf	tf-idf	n'lized	$\times q$
sweet	2	1.301	0.163	0.357	<b>0.137</b>	1	1.000	0.125	0.203	<b>0.0779</b>
nurse	1	1.000	0.301	0.661	0	0	0	0	0	0
love	1	1.000	0.301	0.661	<b>0.610</b>	0	0	0	0	0
how	0	0	0	0	0	0	0	0	0	0
sorrow	0	0	0	0	0	1	1.000	0.602	0.979	0
is	0	0	0	0	0	0	0	0	0	0

$$|d_1| = \sqrt{.163^2 + .301^2 + .301^2} = .456$$

$$|d_2| = \sqrt{.125^2 + .602^2} = .615$$

Cosine:  $\sum$  of column: **0.747**

Cosine:  $\sum$  of column: **0.0779**

# BM25

$$\text{idf}_t = \log_{10} \left( \frac{N}{\text{df}_t} \right)$$

- Introduce parameters k and b

$$\text{score}(q, d) = \sum_{t \in q} \overbrace{\log \left( \frac{N}{\text{df}_t} \right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k \left( 1 - b + b \left( \frac{|d|}{|\text{d}_{\text{avg}}|} \right) \right) + tf_{t,d}}}^{\text{weighted tf}}$$

TODO: complete this

# Information retrieval

- Efficiently find documents that contain terms of interest
- **Inverted index**
  - Dictionary: list of terms (+ document frequency)
  - Postings list: list of document ids that contain the term (+ term frequency in each of the documents)

**Query:** sweet love

**Doc 1:** Sweet sweet nurse! Love?

**Doc 2:** Sweet sorrow

**Doc 3:** How sweet is love?

**Doc 4:** Nurse!

how {1}	→ 3 [1]
is {1}	→ 3 [1]
love {2}	→ 1 [1] → 3 [1]
nurse {2}	→ 1 [1] → 4 [1]
sorry {1}	→ 2 [1]
sweet {3}	→ 1 [2] → 2 [1] → 3 [1]

# Precision and Recall

- Heuristic classifiers tend to have high **precision** but very low **recall**.
- **Classifier accuracy is measured with precision and recall:**

How many relevant items are retrieved?

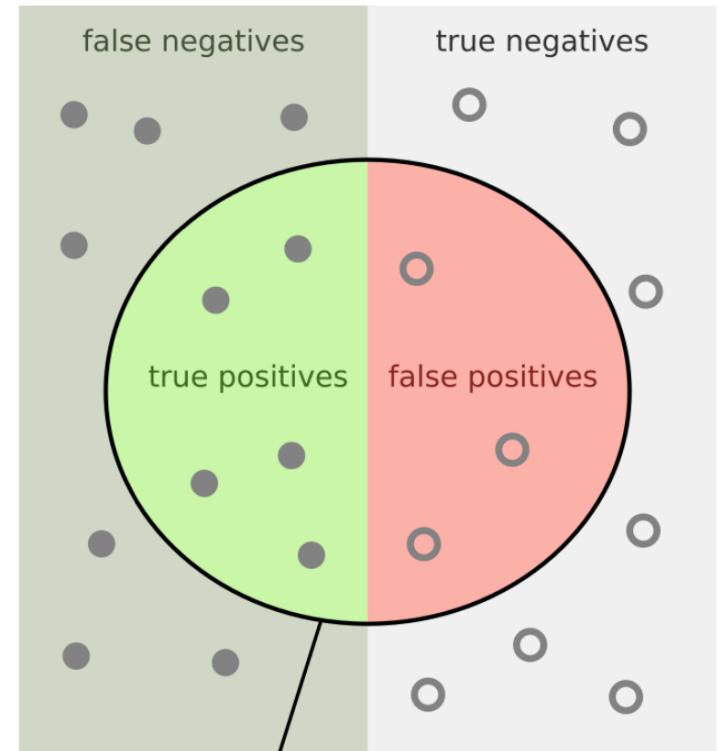
$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{Recall} = \text{TP} / (\text{FN} + \text{TP})$$

How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{Precision} = \text{TP} / (\text{FP} + \text{TP})$$



# Precision and recall in IR

- Each document is either relevant or not relevant
- **Precision:** Fraction of retrieved documents that are relevant
- **Recall:** Fraction of all relevant documents that are retrieved
- Let T be the set of returned documents, R are the relevant documents in T, N are the irrelevant documents in T, U are all relevant documents in the collection

$$Precision = \frac{|R|}{|T|} \quad Recall = \frac{|R|}{|U|}$$

# Evaluation of information retrieval systems

- We care about the rank of the target document(s)
- **Precision@k:** fraction of relevant documents seen at a particular rank k
- **Recall@k:** fraction of relevant documents found at a particular rank k

Rank	Judgment	Precision <sub>Rank</sub>	Recall <sub>Rank</sub>
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

# Evaluation of information retrieval systems

- **Mean average precision (MAP)**
  - Iterate over the ranked list top to bottom
  - Note the precision **only** at positions where a relevant item has been encountered
  - Average these precisions over the return set
  - Formally: Let  $R_r$  be the set of relevant documents at or above rank  $r$
- **Average precision:**
  - $\text{Precision}_r(d)$  precision measured at rank where  $d$  was found

$$\text{AP} = \frac{1}{|R_r|} \sum_{d \in R_r} \text{Precision}_r(d)$$

# Evaluation of information retrieval systems

- **Mean average precision (MAP)**
  - **Average precision:**
    - $\text{Precision}_r(d)$  precision measured at rank where d was found
  - Given a set of queries Q:

$$\text{AP} = \frac{1}{|R_r|} \sum_{d \in R_r} \text{Precision}_r(d)$$

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AP}(q)$$

# Evaluation of information retrieval systems

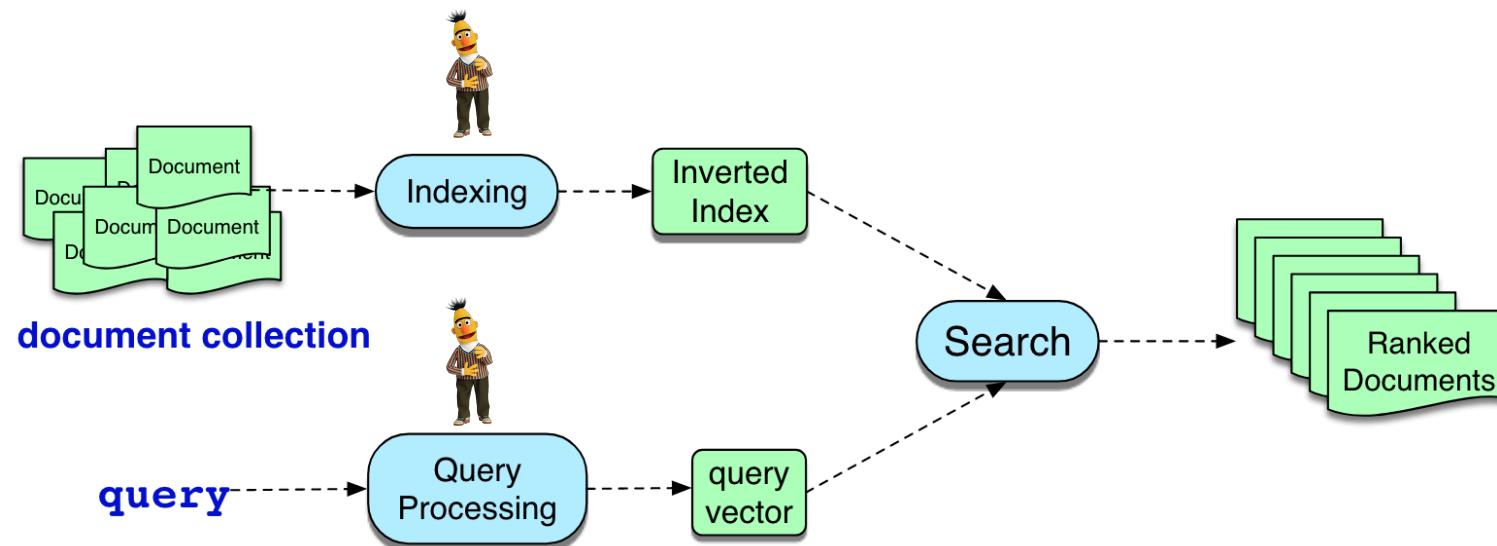
- Mean average precision (MAP)

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AP}(q)$$

Rank	Judgment	Precision <sub>Rank</sub>	Recall <sub>Rank</sub>
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

- Relevant docs at: 1, 3, 5, 6, 8
- Precisions: 1.0, 0.66, 0.60, 0.66, 0.63
- $\text{AP} = (1.0 + 0.66 + 0.60 + 0.66 + 0.63) / 5 = 3.55 / 5 = 0.71$

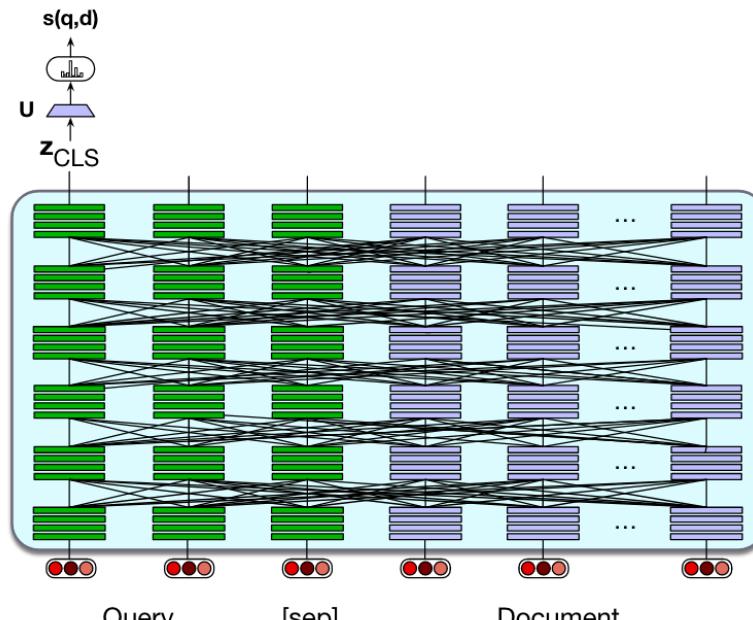
# Information retrieval with pre-trained LMs



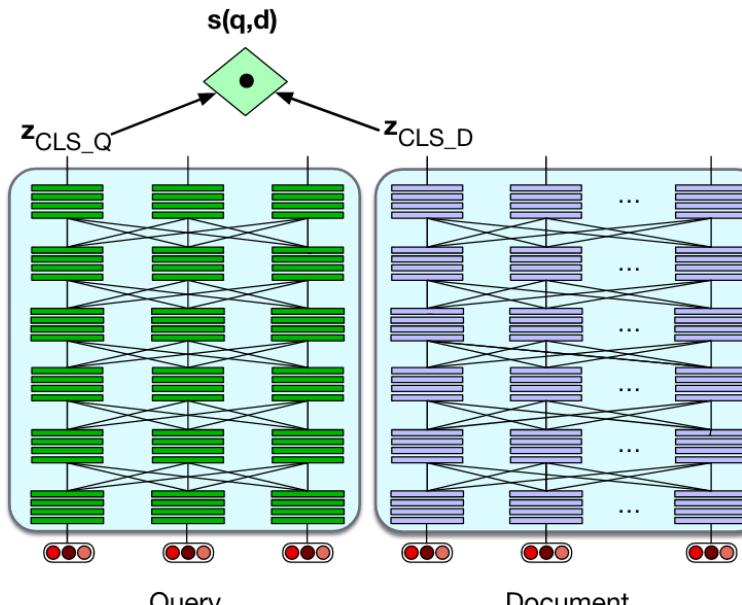
- Instead of using word-count vectors, use dense vectors from pre-trained language models, e.g., BERT

# Information retrieval with pre-trained LMs

- Two potential ways to index queries and documents
  - Jointly encode query and each document
  - Bi-encoder: encode query and documents independently



(a)



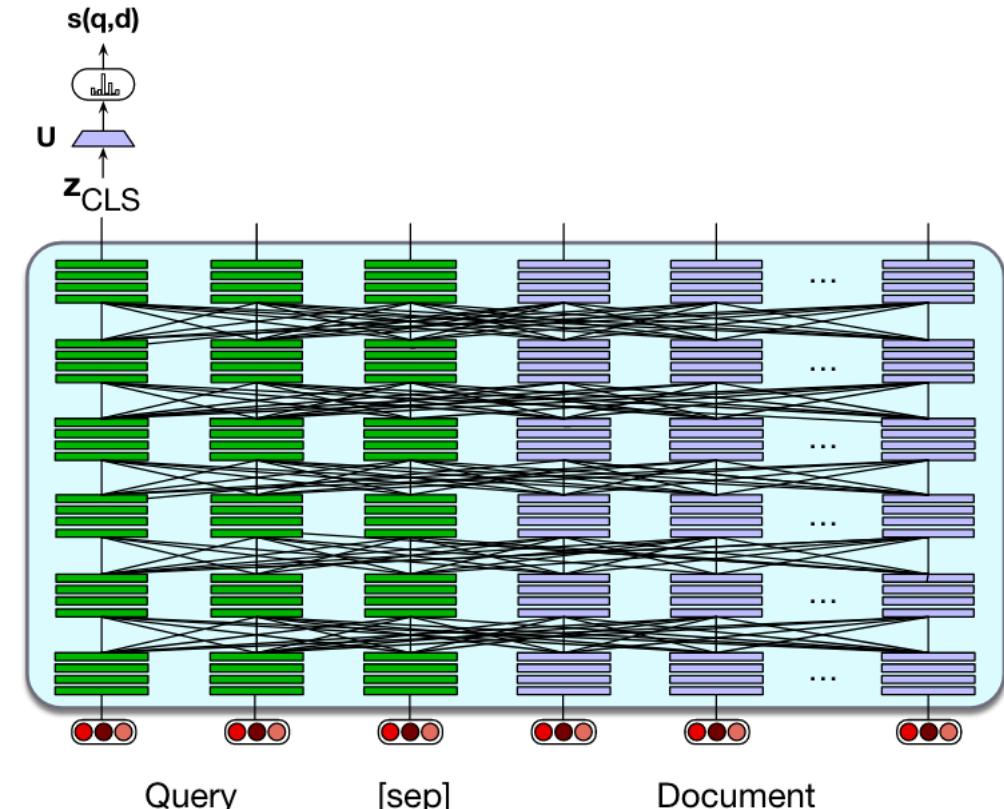
(b)

# Information retrieval with pre-trained LMs

- Jointly encode query and every document

$$\mathbf{z} = \text{BERT}(q; [\text{SEP}]; d)[\text{CLS}]$$

$$\text{score}(\mathbf{z}) = \text{softmax}(\mathbf{U}\mathbf{z})$$

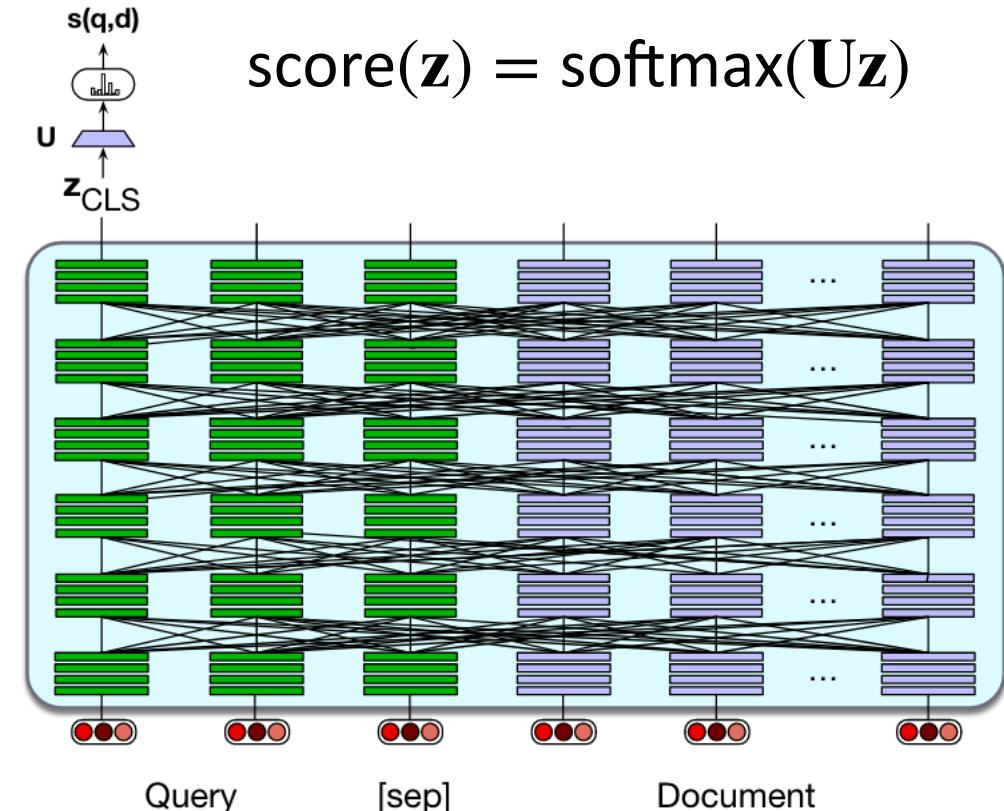


# Information retrieval with pre-trained LMs

- This approach is expensive
- Need to re-encode every document for each query
- This quickly becomes impractical when dealing with large document collections

$$\mathbf{z} = \text{BERT}(q; [\text{SEP}]; d)[\text{CLS}]$$

$$\text{score}(\mathbf{z}) = \text{softmax}(\mathbf{U}\mathbf{z})$$



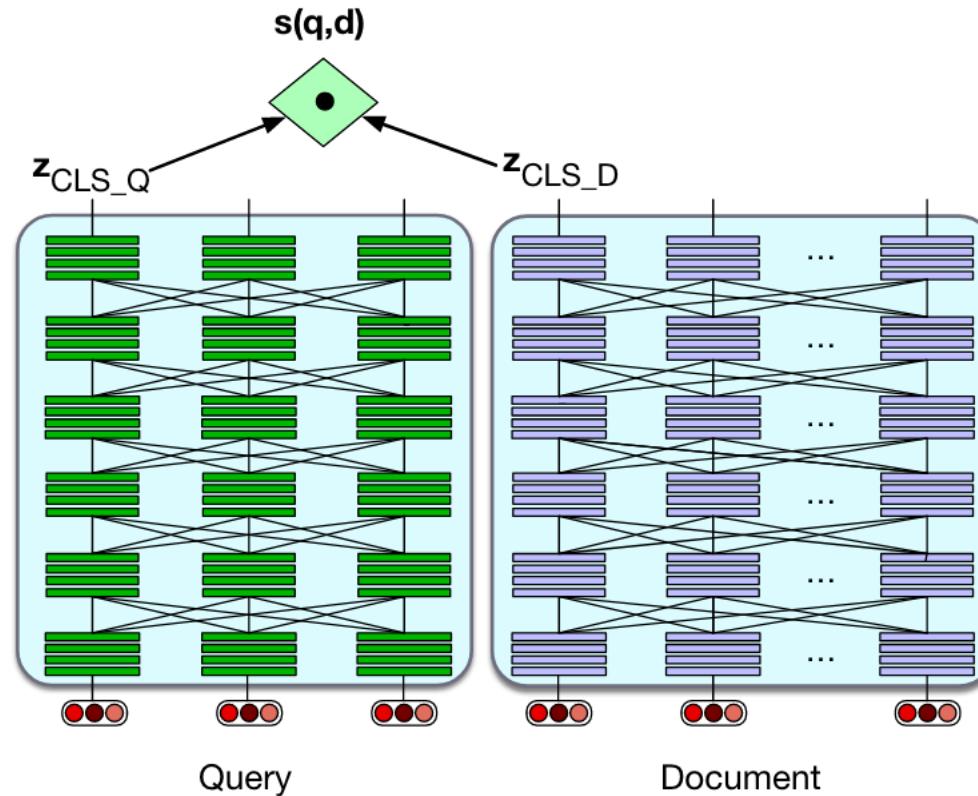
# Information retrieval with pre-trained LMs

- **Bi-encoder:** encode queries and documents independently

$$\mathbf{z}_q = \text{BERT}_{\text{query}}(q)[\text{CLS}]$$

$$\mathbf{z}_d = \text{BERT}_{\text{doc}}(d)[\text{CLS}]$$

$$\text{score}(q, d) = \mathbf{z}_q \cdot \mathbf{z}_d$$



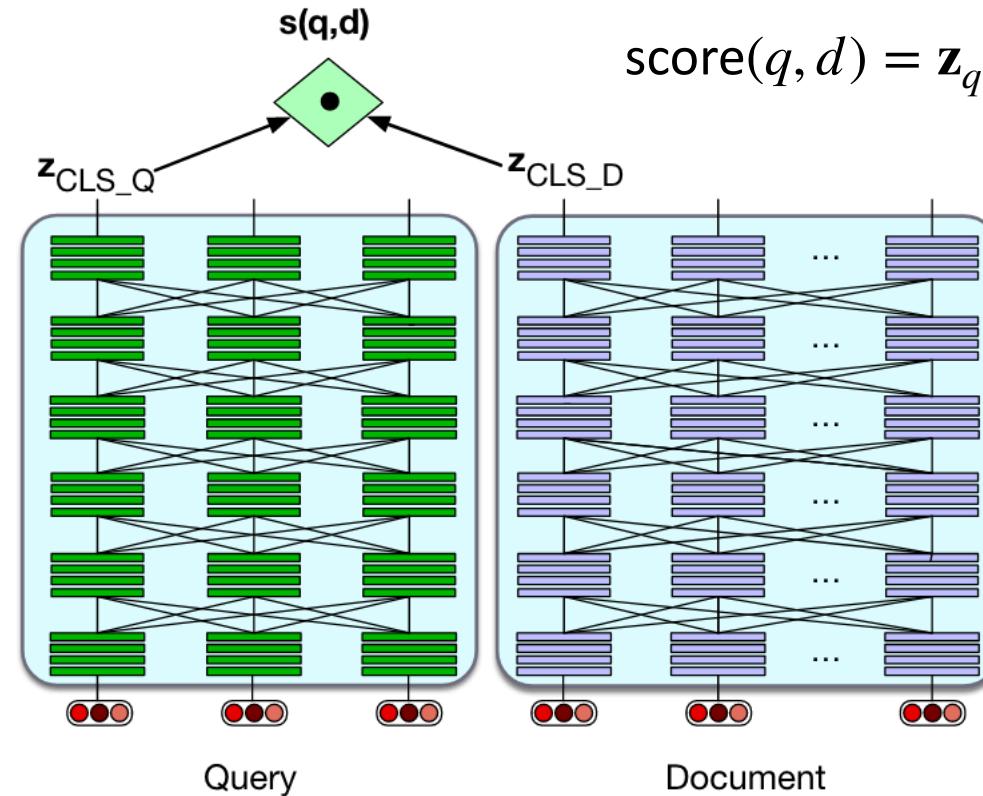
# Information retrieval with pre-trained LMs

- Documents are encoded in advance
- When a new query comes in, we only have to encode that query
- Much more efficient but tends to give worse results
- Question: How to combine the two approaches?

$$\mathbf{z}_q = \text{BERT}_{\text{query}}(q)[\text{CLS}]$$

$$\mathbf{z}_d = \text{BERT}_{\text{doc}}(d)[\text{CLS}]$$

$$\text{score}(q, d) = \mathbf{z}_q \cdot \mathbf{z}_d$$



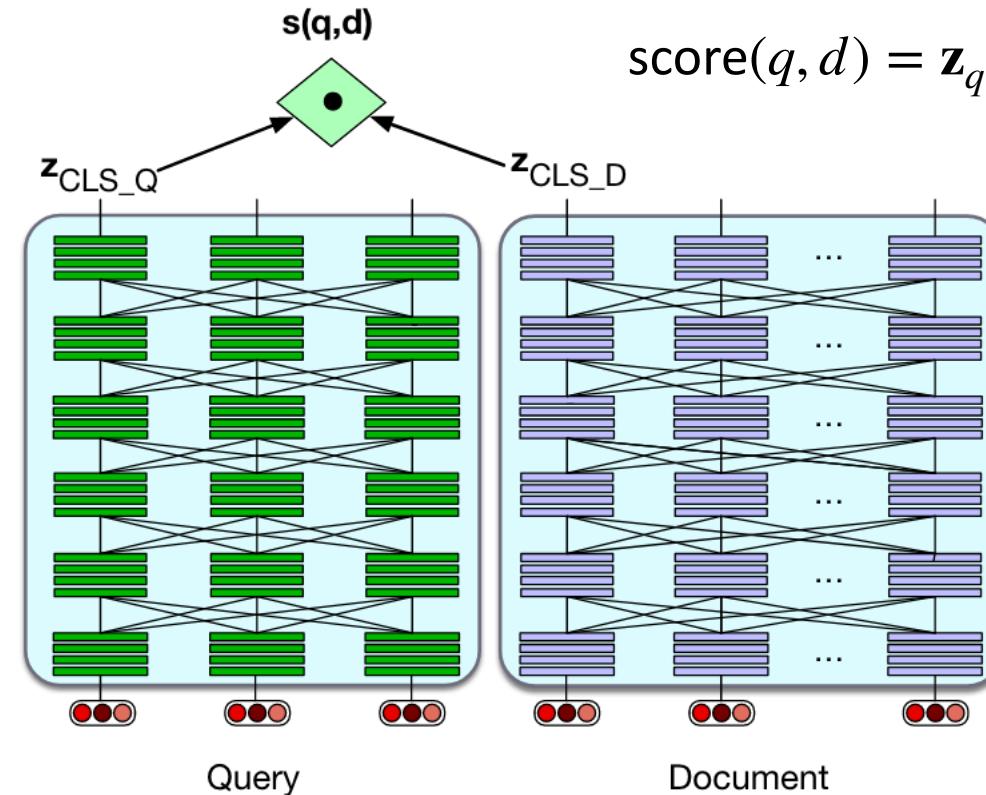
# Information retrieval with pre-trained LMs

- Documents are encoded in advance
- When a new query comes in, we only have to encode that query
- Much more efficient but tends to give worse results
- Question: How to combine the two approaches?
  - **Re-ranking**

$$\mathbf{z}_q = \text{BERT}_{\text{query}}(q)[\text{CLS}]$$

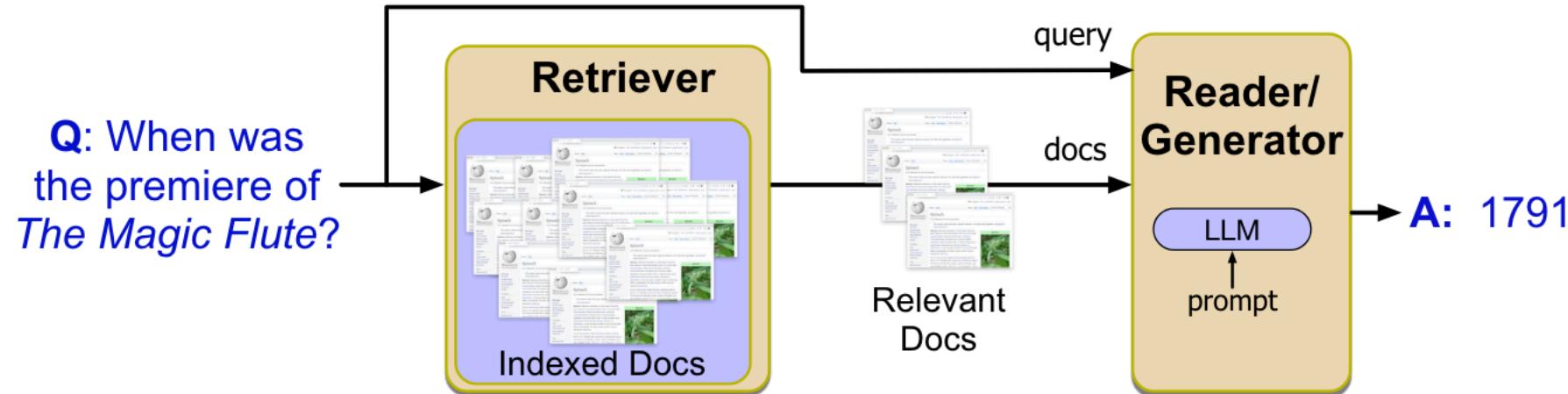
$$\mathbf{z}_d = \text{BERT}_{\text{doc}}(d)[\text{CLS}]$$

$$\text{score}(q, d) = \mathbf{z}_q \cdot \mathbf{z}_d$$



# Information retrieval with pre-trained LMs

- Question Answering (QA) as a retrieval problem
- Find supporting documents for a query
- Extract or generate an answer based on the top-k docs



# Information retrieval with pre-trained LMs

- Consider the document on the right
- There exist many queries for which this document might be relevant
  - When was HEC founded?
  - Can I do a PhD at HEC?
  - What's Viger Square?
  - Etc.
- Ideally, all of this needs to be encoded in the document representation

HEC Montréal ([French](#): *Hautes études commerciales de Montréal*; [English](#): *High Commercial Studies of Montreal*) is a [bilingual](#) public business school located in [Montreal](#), [Quebec](#), Canada. Founded in 1907, HEC Montréal is the graduate business school of the [Université de Montréal](#) and is the first established school of management in Canada.<sup>[2][3]</sup>

HEC Montréal offers undergraduate, graduate, and post-graduate programs, including Bachelor of Business Administration (BBA), Master of Science in Administration (MSc), Master of Management (MM), Master of Business Administration (MBA), and PhD in Administration, in addition to a joint Executive MBA program with [McGill University](#).

HEC Montréal was founded in 1907 by the [Board of Trade of Metropolitan Montreal](#). Its initial building in [Viger Square](#) is now called the [Gilles Hocquart Building](#).<sup>[2]</sup>

In 1988, a group of HEC students established Jeux du Commerce, where more than 1300 students from 14 universities in Eastern Canada gather annually for academic, social, and sports events.<sup>[4][5]</sup> A similar competition has been established in [Western Canada](#) called [JDC West](#).

As of 2021, the centenary of the HEC Montréal Alumni Association, the school had over 100,000 alumni.<sup>[6]</sup>

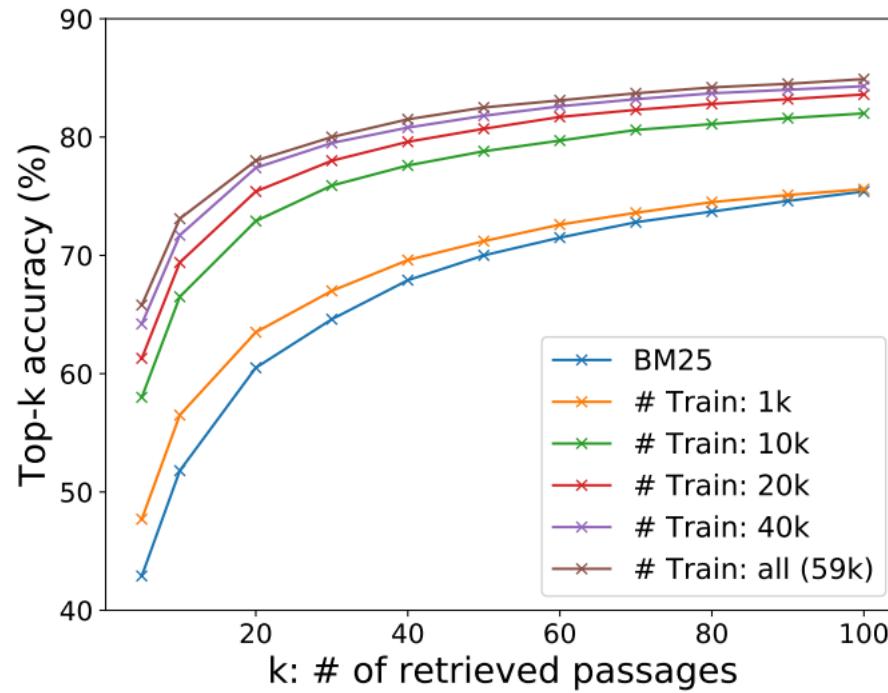
# How to improve dense retrieval?

- Use a collection  $\{(q_i, d_i^+, d_i^-)\}_{i=1}^n$  of queries paired with positive and negative documents
- Learn an implicit relevance definition based on this collection via **contrastive learning**
- Intuition: move positive documents close to query, push negative documents away from query

$$\mathcal{L}(q_i, d_i^+, \{d_{i,j}^-\}_{j=1}^k) = \frac{\exp(\mathbf{q} \cdot \mathbf{d}_i^+)}{\exp(\mathbf{q}_i \cdot \mathbf{d}_i^+) + \sum_{j=1}^k \exp(\mathbf{q}_i \cdot \mathbf{d}_{i,j}^-)}$$

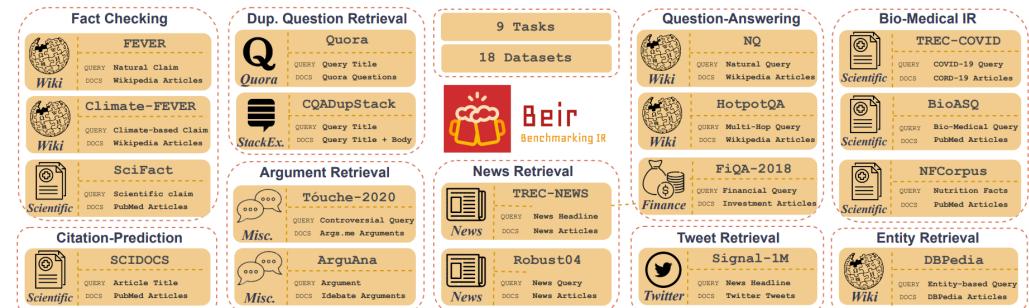
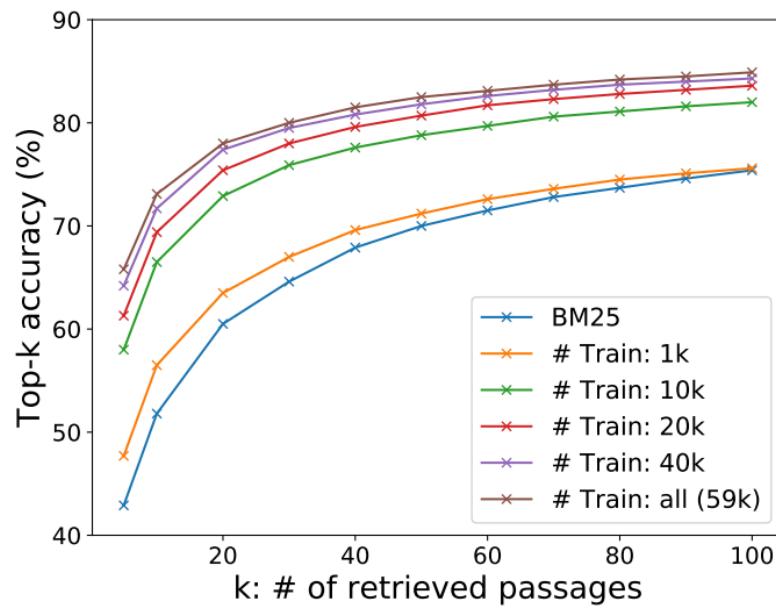
# Dense Passage Retrieval for Open-Domain QA

- How does it compare to BM25?



# Dense Passage Retrieval for Open-Domain QA

- What happens outside of the training distribution?
- Question: what could be the reason?



# Practical considerations

- How to efficiently search over dense representations?
  - Approximate nearest neighbour search
  - Fast embedding search (FAISS) on GPUs

**[15 minute break]**