Generalising IR Operations

Many operations in an information retrieval pipeline can be thought of a "transformer" functions.

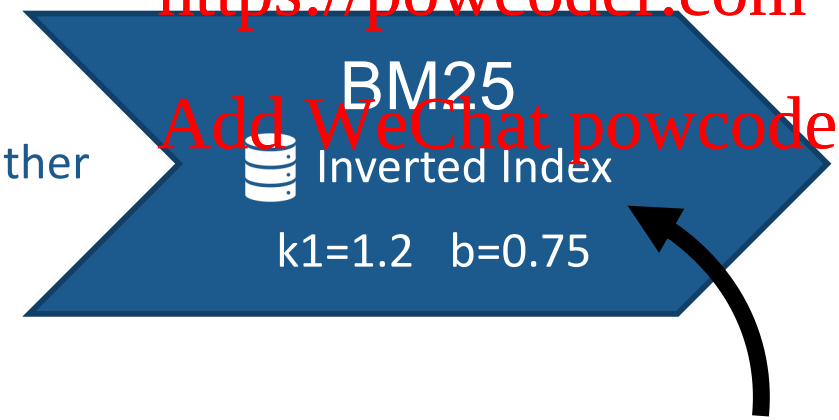
Example: BM25 Retrieval

docno	score
15213	0.5134
42635	0.3742
26340	0.3223
…	…

Generalising IR Operations

Many operations in an information retrieval pipeline can be thought of a "transformer" functions.

Example: BM25 Retrieval



Query: Glasgow weather

Results:

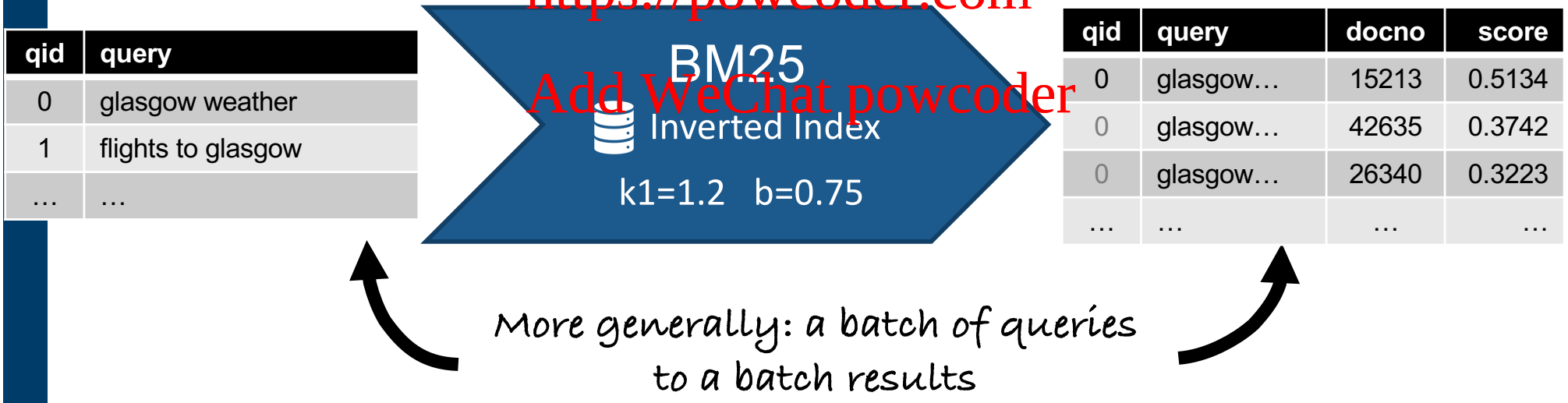
docno	score
15213	0.5134
42635	0.3742
26340	0.3223
…	…

Transformers can have parameters.

Generalising IR Operations

Many operations in an information retrieval pipeline can be thought of a "transformer" functions.

Example: BM25 Retrieval



Generalising IR Operations

Many operations in an information retrieval pipeline can be thought of a "transformer" functions.

Example: BM25 Retrieval

qid	query
0	glasgow weather
1	flights to glasgow
…	…

qid	query	docno	score
0	glasgow…	15213	0.5134
0	glasgow…	42635	0.3742
0	glasgow…	26340	0.3223
…	…	…	…

$$Q \mapsto R$$

Shorthand: BM25 maps a Query (Q) frame to a Result (R) frame.

Generalising IR Operations

The PyTerrier Data Model:

Q

qid	query
0	glasgow weather
1	flights to glasgow
…	…

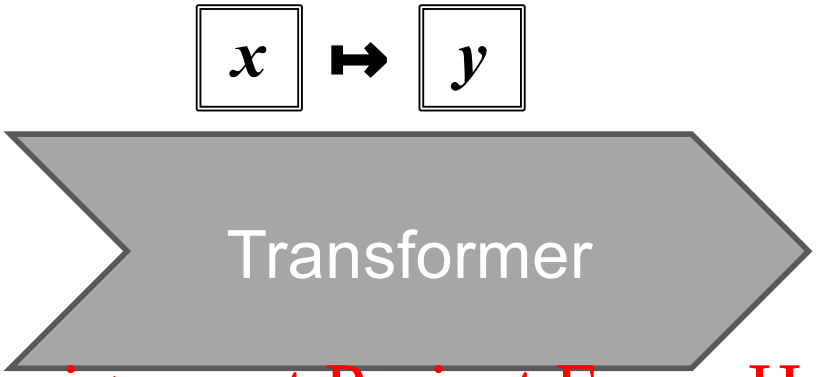
D

docno	text	title
0	Science & Mathematics Physi…	The hot glowing…
1	School-Age Kids Growth &…	Developmental…
…	…	…

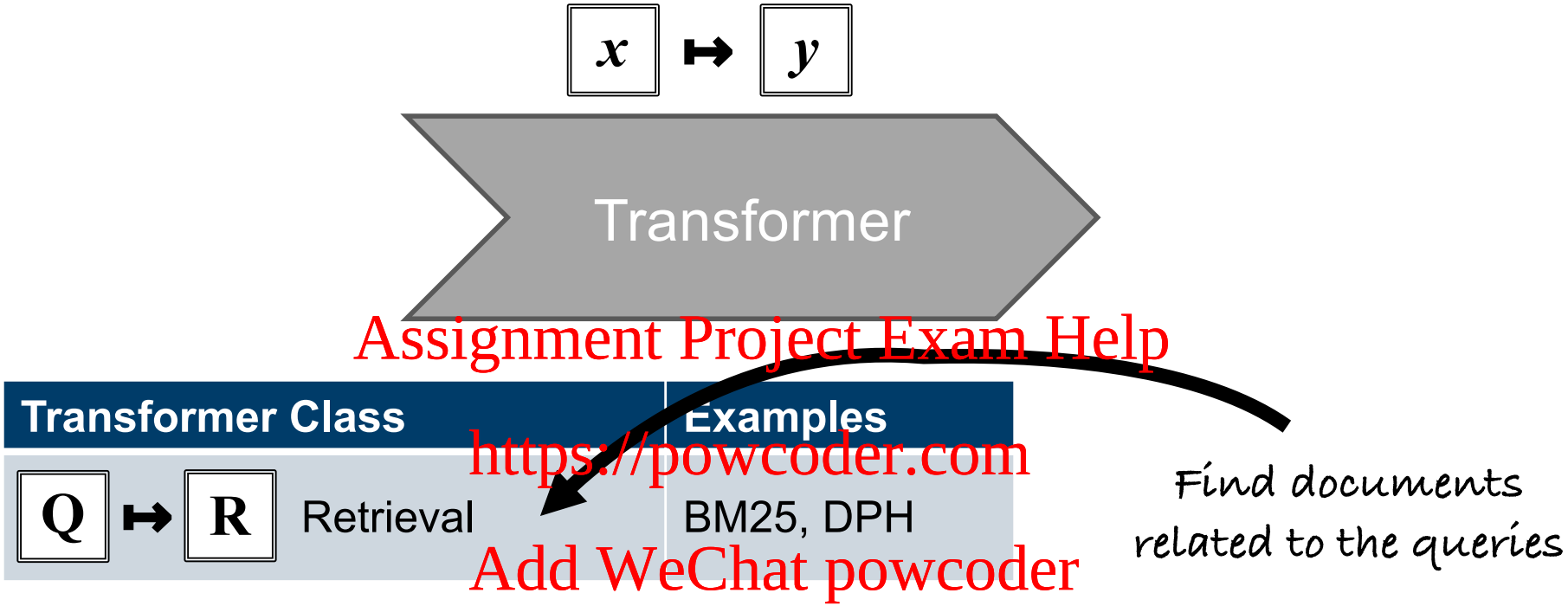
R

qid	query	docno	score
0	glasgow weather	15213	0.5134
0	glasgow weather	42635	0.3742
0	glasgow weather	26340	0.3223
…	…	…	…

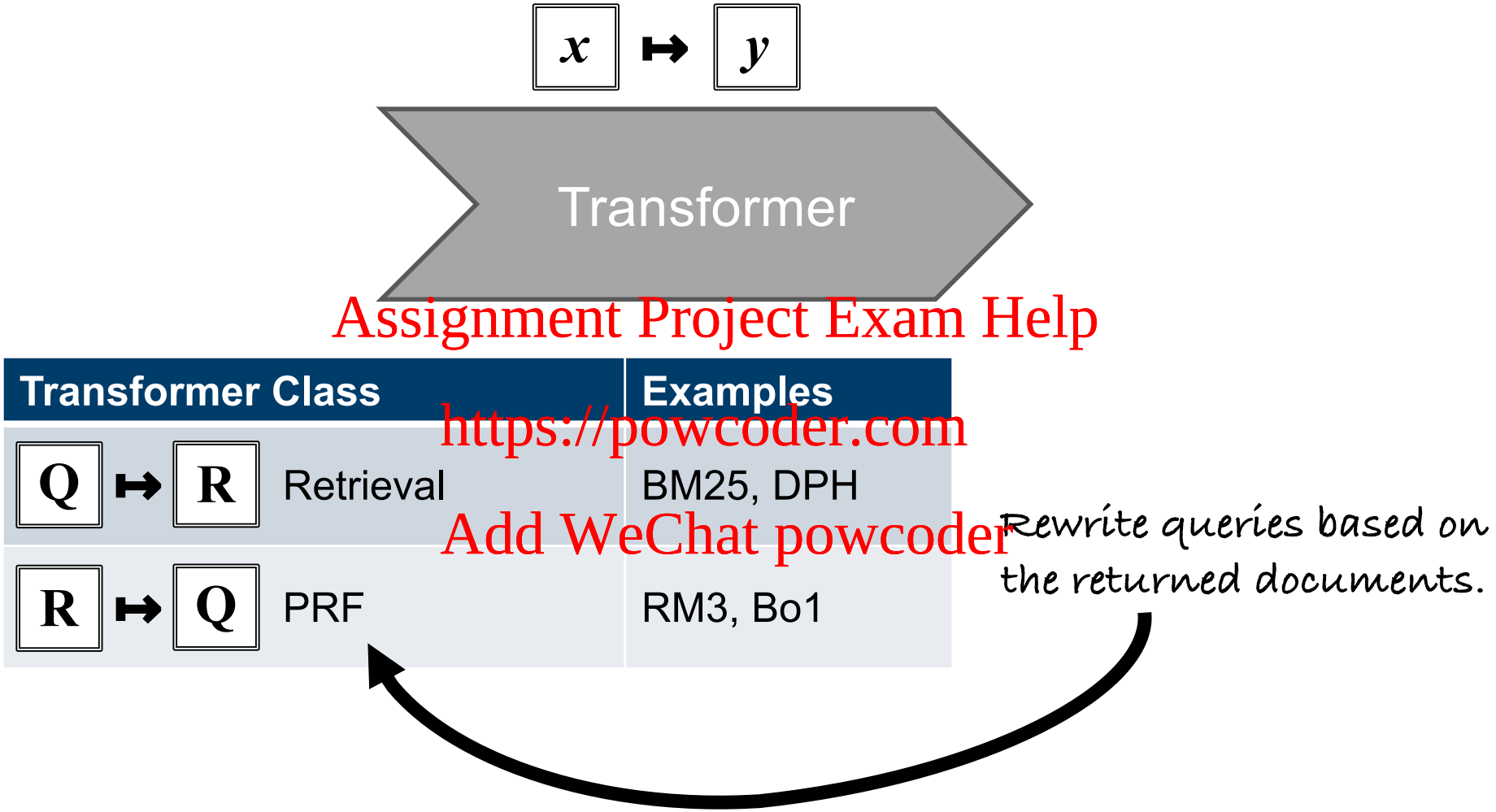
Generalising IR Operations



Generalising IR Operations



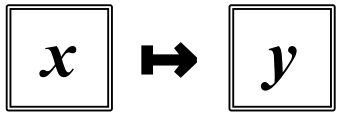
Generalising IR Operations



Transformer Class	Examples
Q ↦ R Retrieval	BM25, DPH
R ↦ Q PRF	RM3, Bo1

Rewrite queries based on the returned documents.

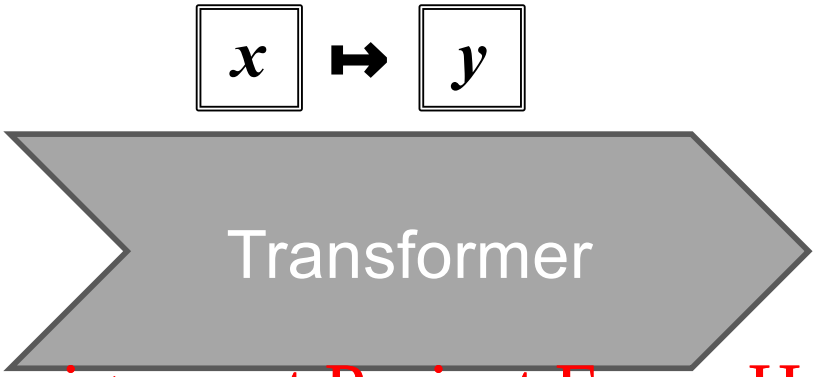
Generalising IR Operations



Transformer Class	Examples
Q ↦ R Retrieval	BM25, DPH
R ↦ Q PRF	RM3, Bo1
R ↦ R Re-ranking	LambdaMART

Find a better order for the results.
(Usually a more expensive method than retrieval.)

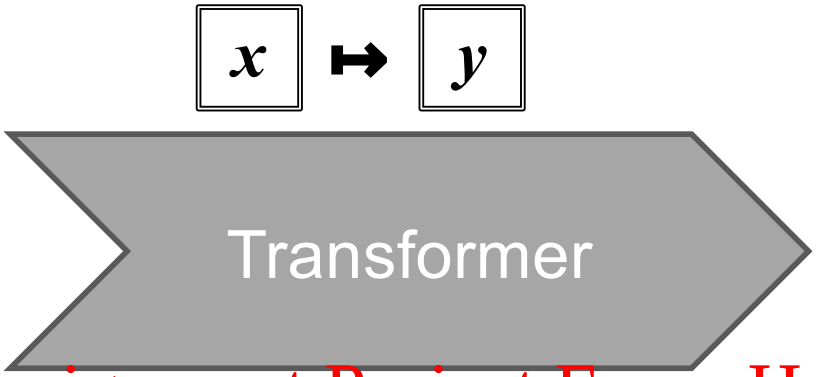
Generalising IR Operations



Transformer Class	Examples
Q ↦ R Retrieval	BM25, DPH
R ↦ Q PRF	RM3, Bo1
R ↦ R Re-ranking	LambdaMART
Q ↦ Q Query Re-writing	SDM

Build a better version of the user's query

Generalising IR Operations



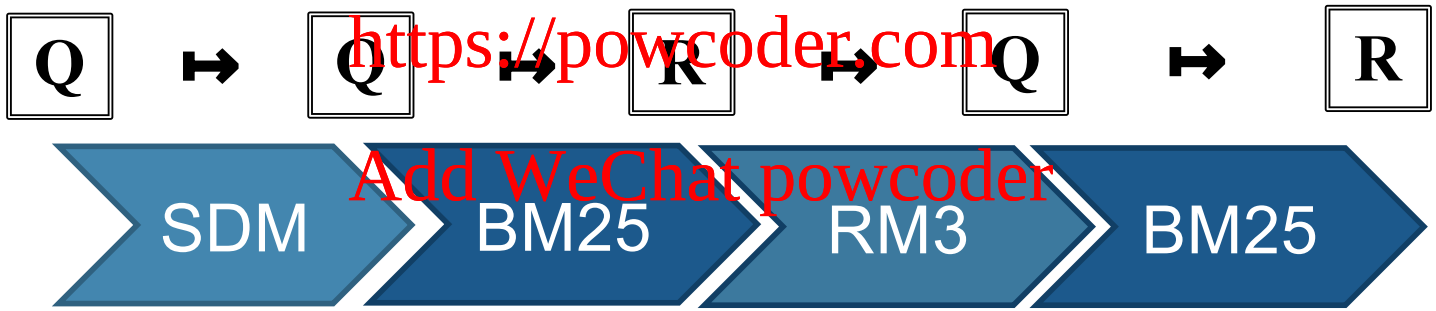
Transformer Class	Examples
Q ↦ R Retrieval	BM25, DPH
R ↦ Q PRF	RM3, Bo1
R ↦ R Re-ranking	LambdaMART
Q ↦ Q Query Re-writing	SDM
D ↦ D Doc. Re-writing	Passaging

Build a better version of documents to index.

Generalising IR Operations

These operations are composable!

Example: Perform SDM, then BM25, then RM3, then BM25 again.



What's the point of all this?

Transformer Class		Examples
Q ↦ R	Retrieval	BM25, DPH
R ↦ Q	PRF	RM3, Bo1
R ↦ R	Re-ranking	LambdaMART
Q ↦ Q	Query Re-writing	SDM
D ↦ D	Doc. Re-writing	Passaging

What's the point of all this?

All of these transformations can be replaced with a Neural Network!

Transformer Class	Examples	Neural Examples
Q ↦ R Retrieval	BM25, DPH	ANCE, ColBERT
R ↦ Q PRF	RM3, Bo1	ColBERT-PRF
R ↦ R Re-ranking	LambdaMART	CEDR, monoT5
Q ↦ Q Query Re-writing	SDM	T5-QE, IntenT5
D ↦ D Doc. Re-writing	Passaging	Doc2Query, DeepImpact

Why Neural Networks?

- Neural Natural Language Processing (NLP) techniques are highly effective: state-of-the-art at most NLP tasks.
  - Particularly: Using models trained on a "language modeling" objective transfer the knowledge well to other tasks.

- Able to learn complex & subtle patterns from training data

  - Automatically learns to overcome the lexical gap, proximity relations, term salience (importance), coreference, etc.
  - This means less manual "feature engineering"

Building comprehensive rules/heuristics for language is challenging.

What about other usages of "where"?

Where was iodine discovered?

BM25=12.05

Iodine was discovered by Bernard Courtois in 1811 **in France**. Courtois used sulfuric acid to extract potassium compounds from seaweed. After crystallizing the potassium chloride, he added sulfuric acid to the remaining liquid and, rather surprisingly, produced a purple vapor, which condensed into dark crystals.

BM25=12.18

Iodine was discovered in 1811 by French chemist Bernard Courtois (1777-1838). The element occurs primarily in seawater and in solids formed when seawater evaporates. Its most important property may be the ability to

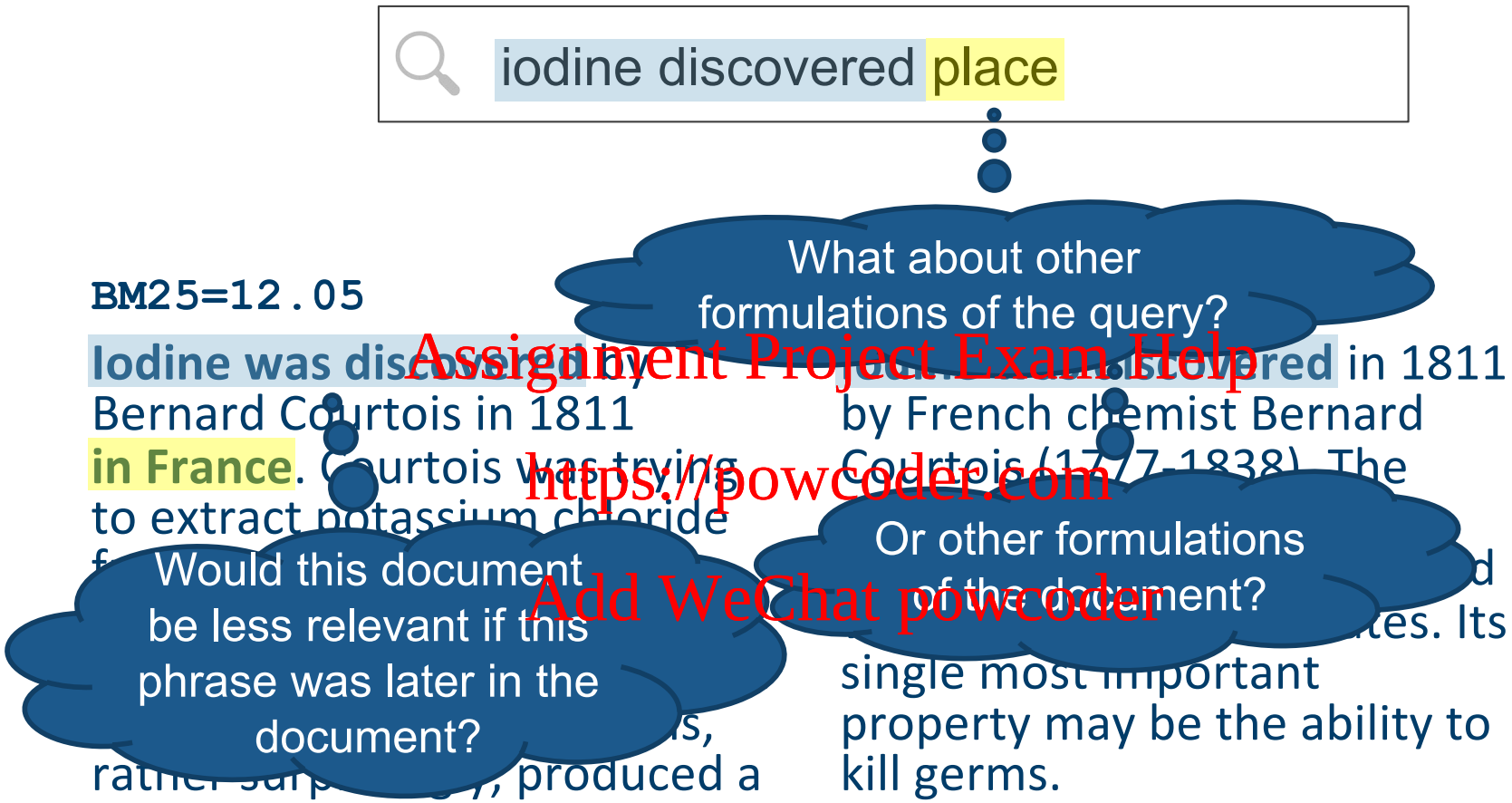
Should "where" queries match "in"?

What about other usages of "in"?

How close does it need to be to other query terms?

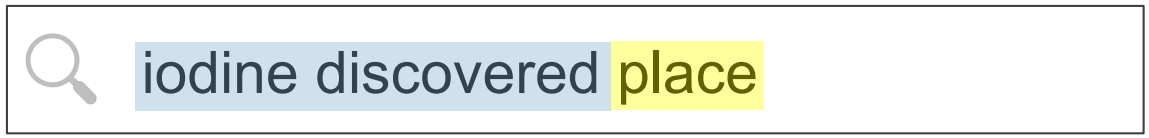
Example from MS-MARCO collection

Building comprehensive rules/heuristics for language is challenging.



Example from MS-MARCO collection

Building comprehensive rules/heuristics for language is challenging.



BM25=12.05

Iodine was discovered by Bernard Courtois in 1811 **in France**. Courtois was trying to extract potassium chloride from seaweed. After crys… to… rather surprisingly, produced a purple vapor, which condensed into dark crystals.

Is "France" really enough? Would the city be better?

BM25=12.18

Iodine was discovered in 1811 by French chemist Bernard Courtois (1777-1838). The element occurs primarily in seaw… …property may be the ability to kill germs.

Is the fact that the discoverer was French close enough?

Example from MS-MARCO collection

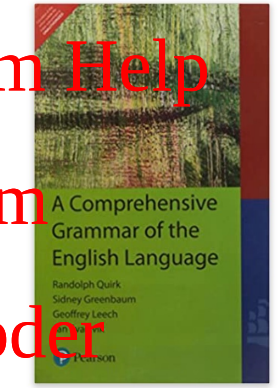
Natural language is messy

The same idea can be described by many sequences of words.

Meanwhile… A sequence of words can describe different ideas (ambiguity).

Even the grammar of languages themselves are challenging to fully describe.

We can instead let a neural network learn how to deal with this mess.



See all 3 images

A Comprehensive Grammar o

Paperback – 1 Jan. 2011

by Geoffrey Leech & Jan Svartvik Randolph Quirk, Sidney G

126 ratings

See all formats and editions

Hardcover	Paperback
from £96.95	£39.95
1 Used from £96.95	4 Used from £26.98 6 New from £26.99

BRAND NEW, perfect condition.

Report incorrect product information.

Print length	Language	Publisher
1779 pages	English	Pearson

Today

1. Review of LTR & Basics of Neural Networks for NLP
2. Neural Re-ranking R ↦ R
3. Neural Retrieval Q ↦ R
4. Neural Query Rewriting & PRF Q ↦ Q & R ↦ Q
5. Neural Document Rewriting D ↦ D
6. Neural IR in PyTerrier

Review of Supervised Learning

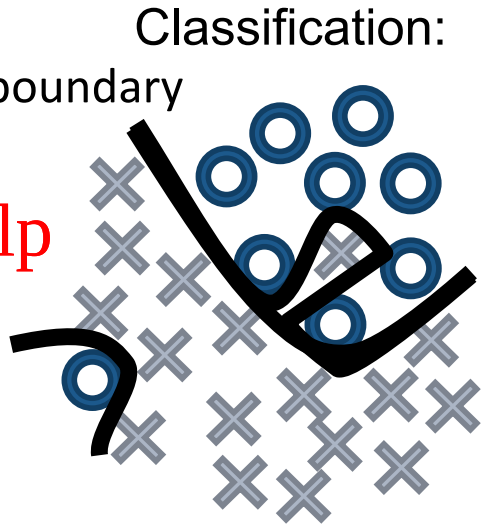
Use **training data** to make a model that is able to predict something about new inputs.

Many algorithms exist (e.g., Logistic Regression, LambdaMART, etc.)

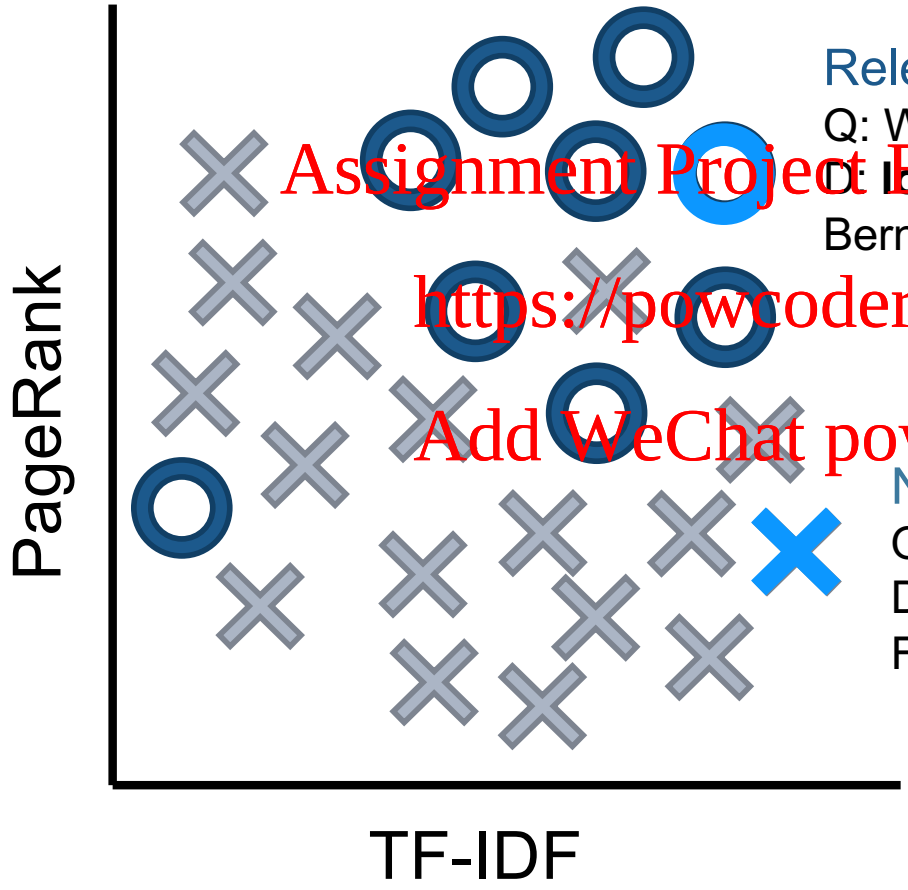
- Today we're focusing on neural networks

These algorithms have settings (hyper-parameters) that affect how the model is built

In reality: these algorithms operate over many features (figure shows 2: x and y axis)



Learning to Rank

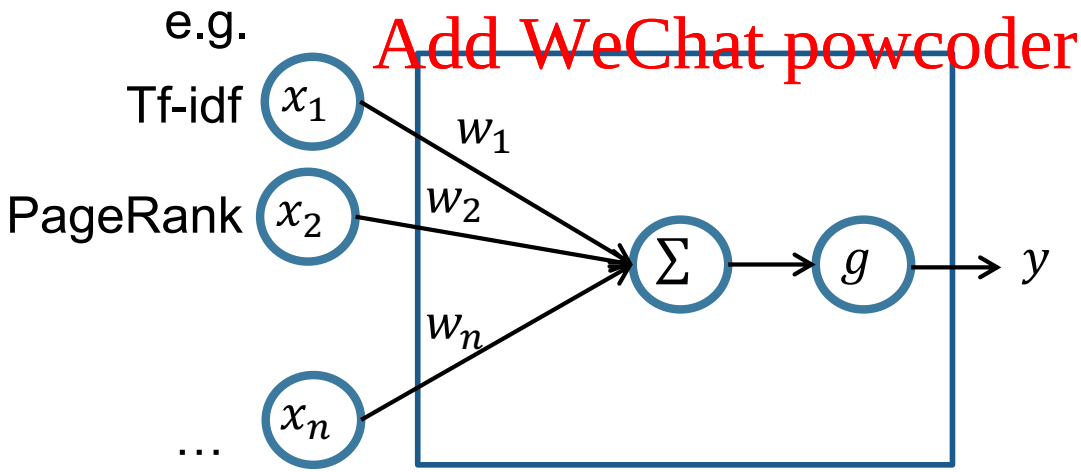


Relevant Query-doc pair
Q: Where was iodine discovered?
D: **Iodine was discovered** by Bernard Courtois in 1811 **in France**…

Non-Relevant Query-doc pair
Q: Where was iodine discovered?
D: **Iodine** was **discovered** in 1811 by French chemist Bernard Courtois …

Neural Networks (Intro.)

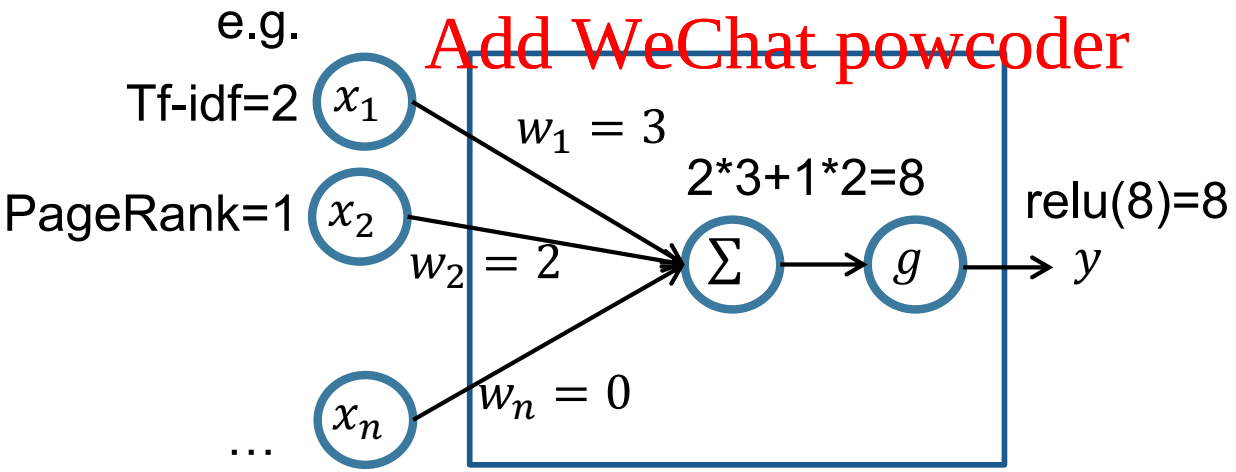
- Consist of computational elements (neurons)
- A neuron receives INPUT from other nodes and each INPUT is associated with a weight w (learned)
- The unit computes some function f of the weighted sum of it's INPUTS: $y = g(\sum_{i=1}^{n} w_i x_i)$
- Also sometimes referred to as a perceptron



g is called the "activation function". It transforms the OUTPUT shape of the weighted sum.

Neural Networks (Intro.)

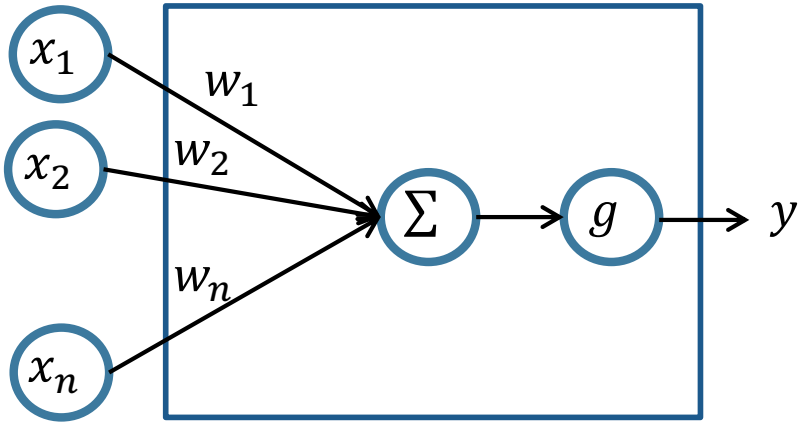
- Consist of computational elements (neurons)
- A neuron receives INPUT from other nodes and each INPUT is associated with a weight w (learned)
- The unit computes some function f of the weighted sum of it's INPUTS: $y = g(\sum_{i=1}^{n} w_i x_i)$
- Also sometimes referred to as a perceptron

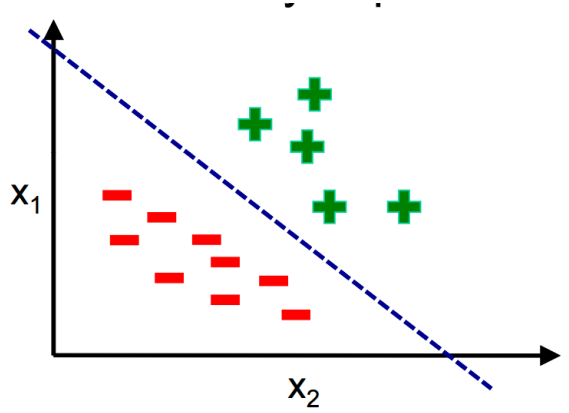


Neural Networks (Intro.)

Training

- Feeding in learning examples $D = (x_i, y_i)$
- The network adjusts the weights based on training samples
- Uses a loss function to determine how "wrong" the predicated value is
- Propagates loss signals obtained by gradients of OUTPUT with respect to INPUT





Neural Networks (Intro.)

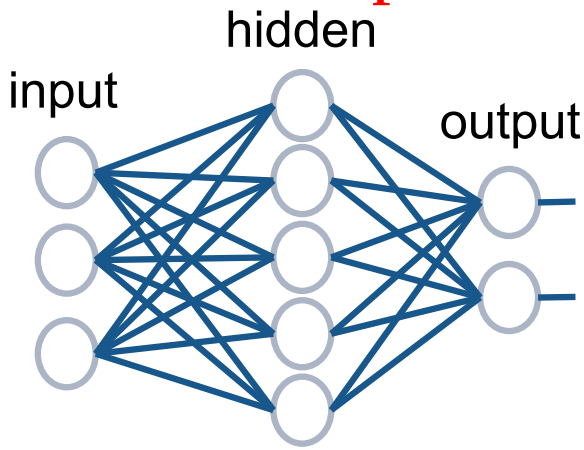
A neural network consists of multiple layers of neurons

Helps in capture non-linear target functions

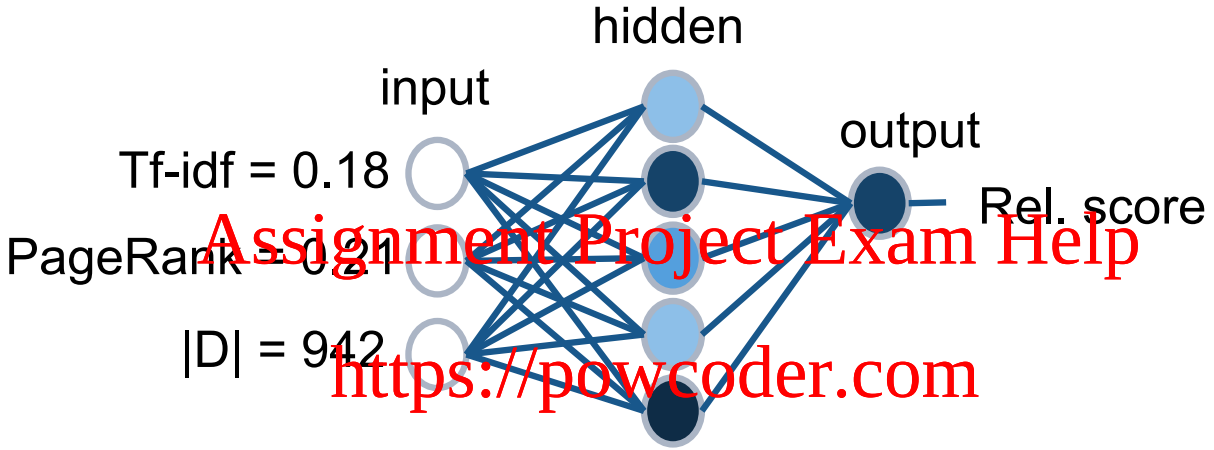
- Many interesting problems are non-linear in nature!

Several special structures allow functionality on certain types of data: e.g., convolutional, recurrent, transformer

This is the basic "feed forward" neural network:



Learning to rank with neural networks



This could work, but doesn't provide much beyond what more basic methods can do (e.g., LambdaMART).

Learning to rank with neural networks

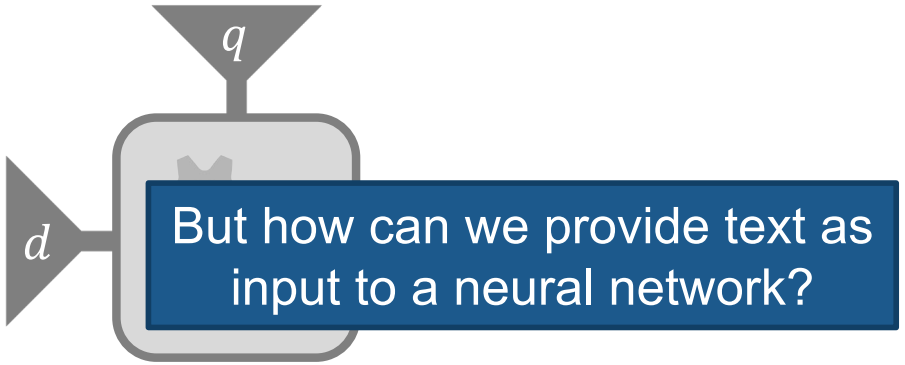
We explore feeding the text of the query and document as features directly into the neural networks.

Goal: Determine relevance scores based on the **query and document text itself**.

Let the neural network learn rules for vocabulary mismatch, proximity, etc.

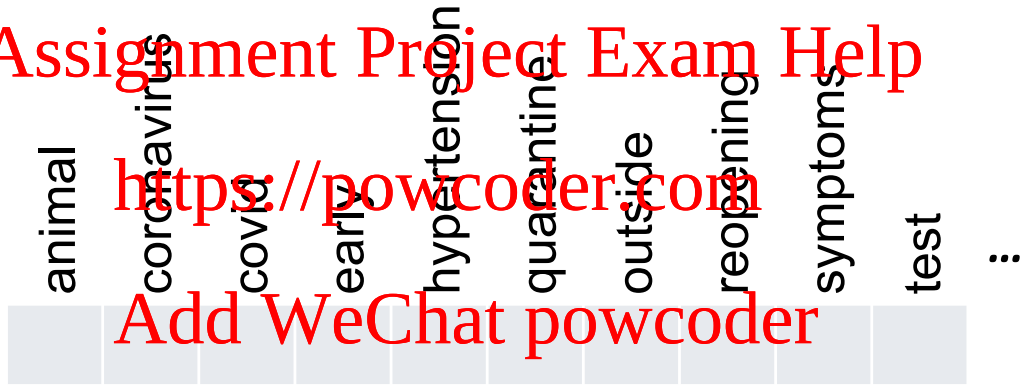
Q: Where was iodine discovered?

D2: **Iodine was discovered** by Bernard Courtois in 1811 **in France**…



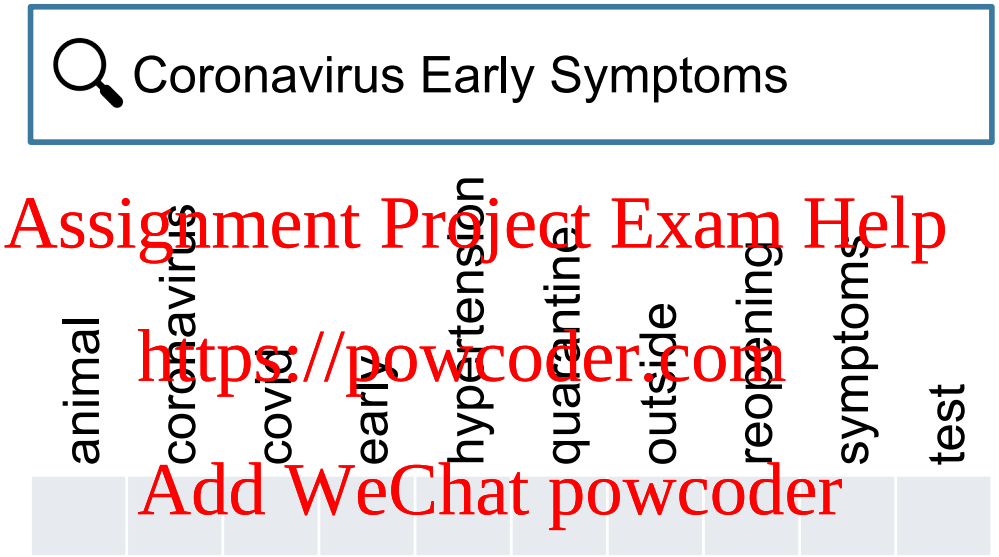
Representing Text

One option: One-hot encoding



Representing Text

One option: One-hot encoding



Representing Text

One option: One-hot encoding

	animal	coronavirus	covid	early	hypertension	quarantine	outside	reopening	symptoms	test	...
Bag of words		1		1					1		

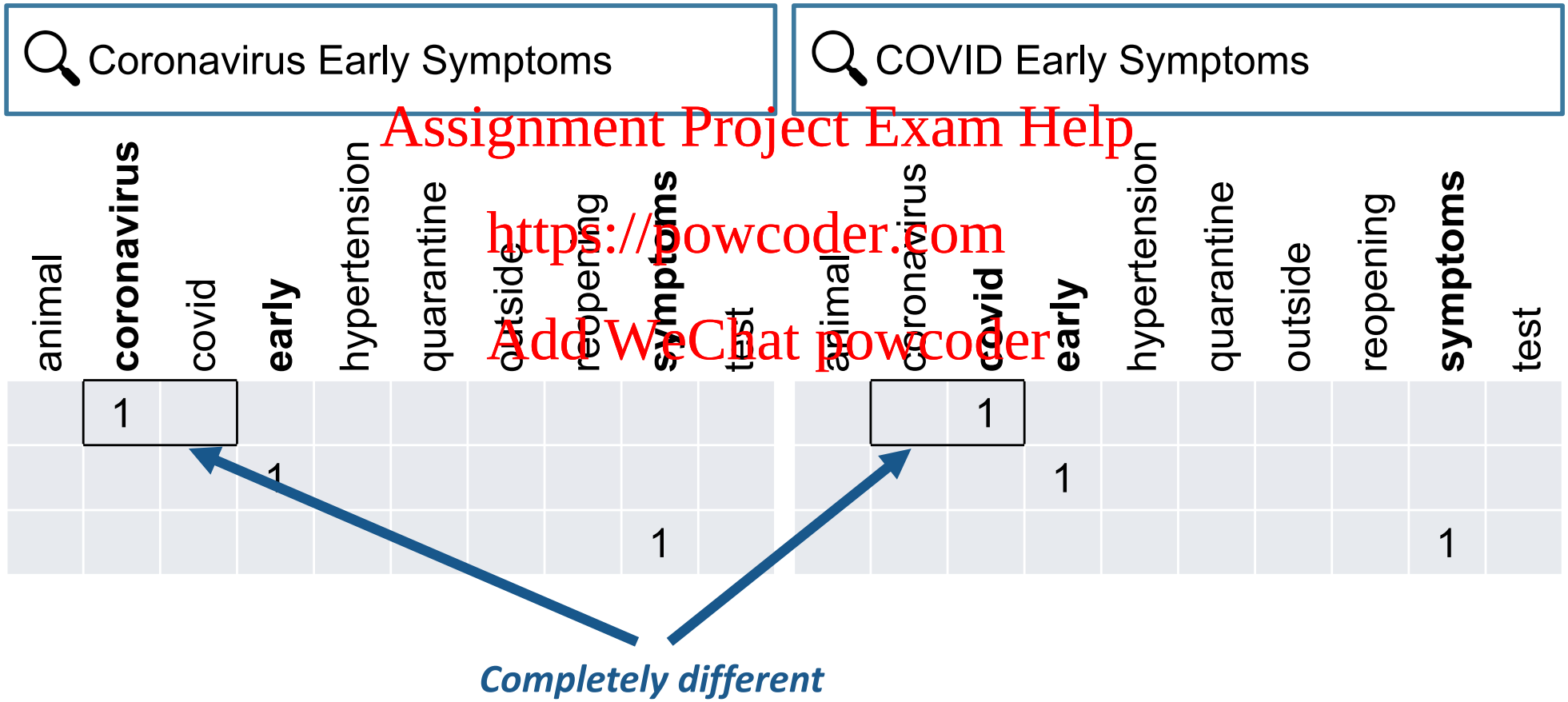
Representing Text

One option: One-hot encoding

	animal	**coronavirus**	covid	**early**	hypertension	quarantine	outside	reopening	**symptoms**	test	...
		1									
Sequence				1							
									1		

Representing Text

Problem: no relationship between words



Representing Text

Word Vectors: Map each word to a dense vector

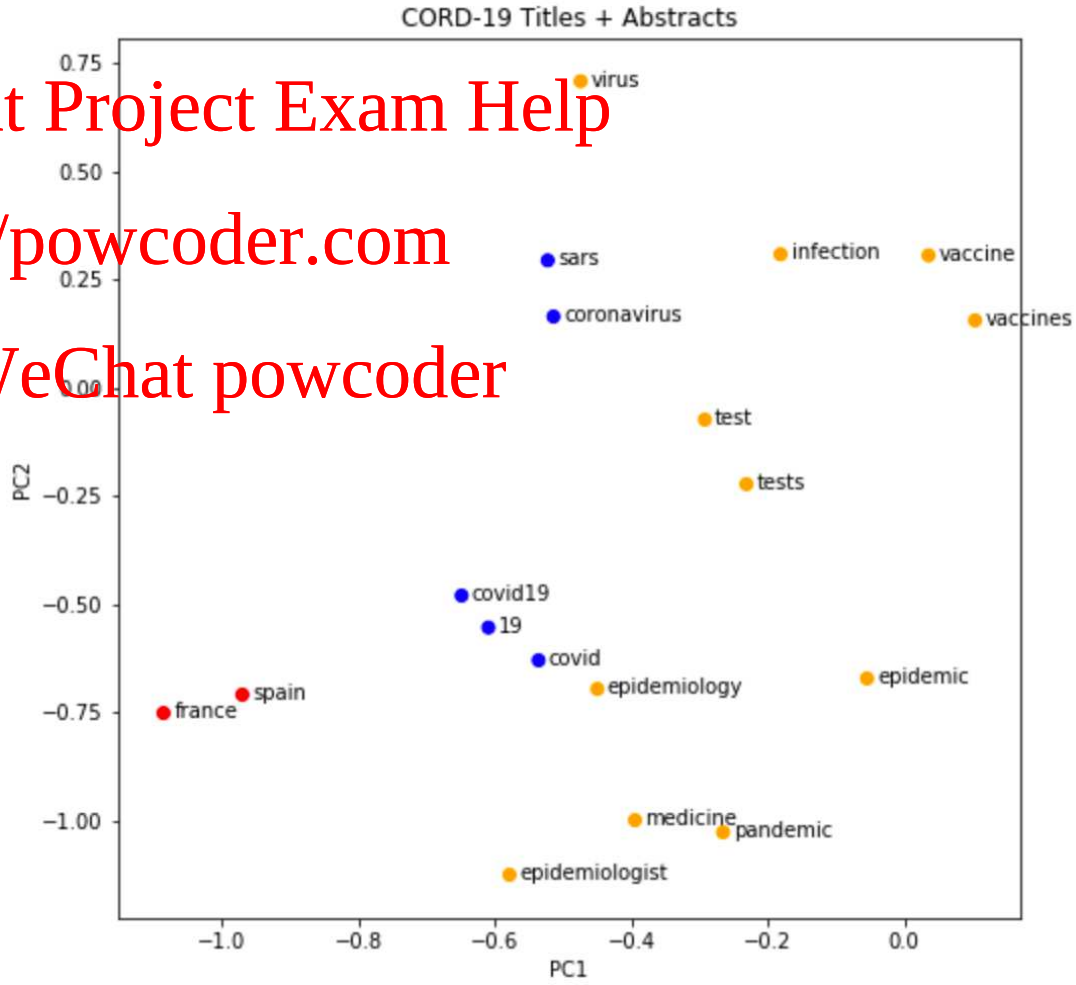
Also known as a word “embedding”

covid =

```
[0.60586,
 0.04596,
 0.12191,
-0.18414,
-0.04422,
 0.13495,
 0.31471,
 0.33992,
 0.01285,
-0.18592,
-0.43352,
-0.62741,
 0.24341,
 0.07149,
...]
```

coronavirus =

```
[0.39883,
-0.27798,
 0.08317,
-0.19729,
-0.49235,
 0.26514,
 0.03004,
 0.25704,
-0.38031,
-0.32722,
-0.47273,
-0.01596,
 0.32322,
-0.04947,
...]
```



Representing Text

Building word vectors

- Based on word co-occurrences
- Trained using neural network
- e.g. word2vec, gloVe, etc.
- Important: these vectors are the same, regardless of context that the word appears in (i.e., “static”).

covid =

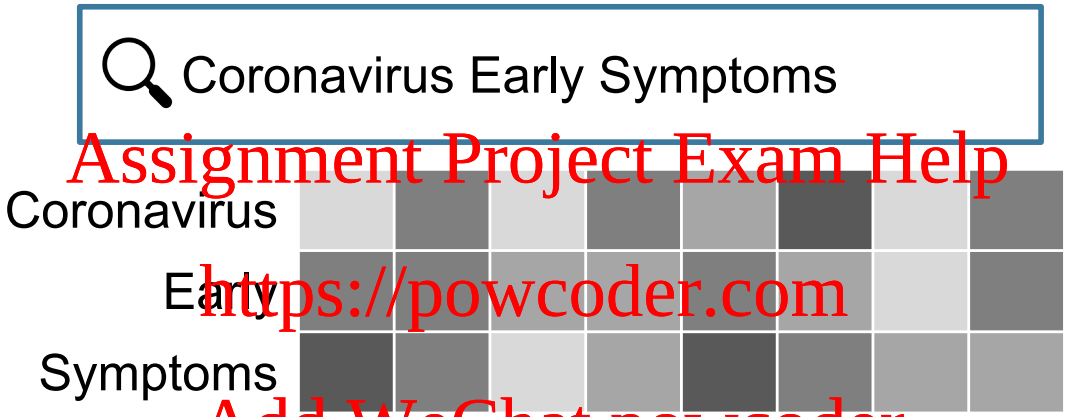
```
[0.60586,
 0.04596,
 0.12191,
-0.18414,
-0.04422,
 0.13495,
 0.31471,
 0.33992,
 0.01285,
-0.18592,
-0.43352,
-0.62741,
 0.24341,
 0.07149,
...]
```

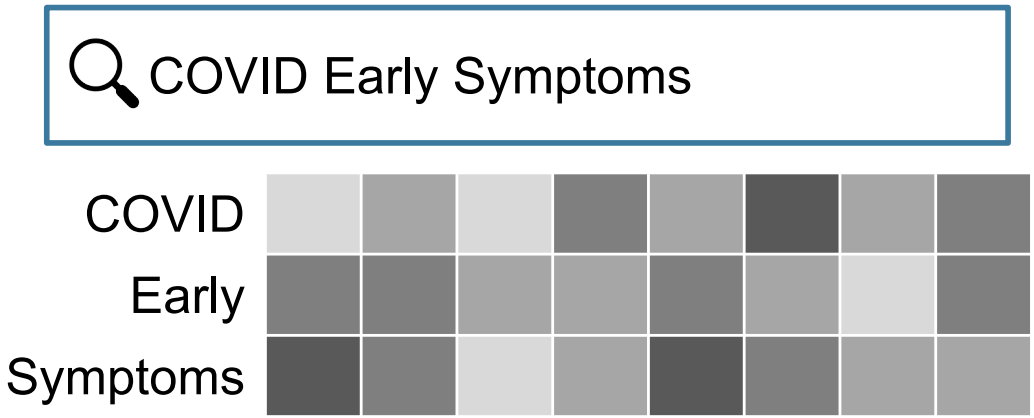
coronavirus =

```
[0.33853,
-0.27798,
 0.08317,
-0.19729,
-0.49235,
 0.26514,
 0.03004,
 0.25704,
-0.38031,
-0.32722,
-0.47273,
-0.01596,
 0.32322,
-0.04947,
...]
```

Representing Text

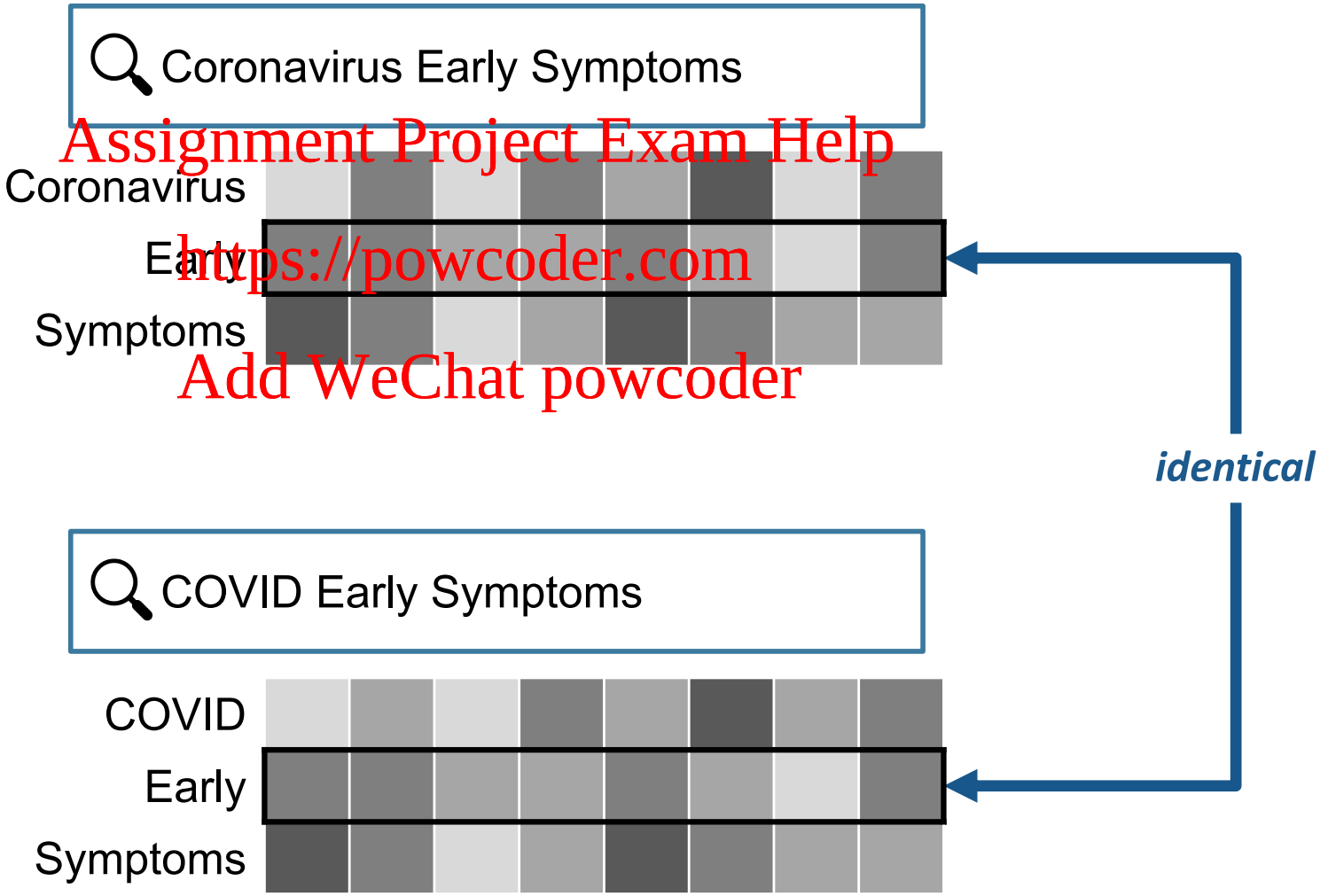
Word Vectors: Map each word to a dense vector





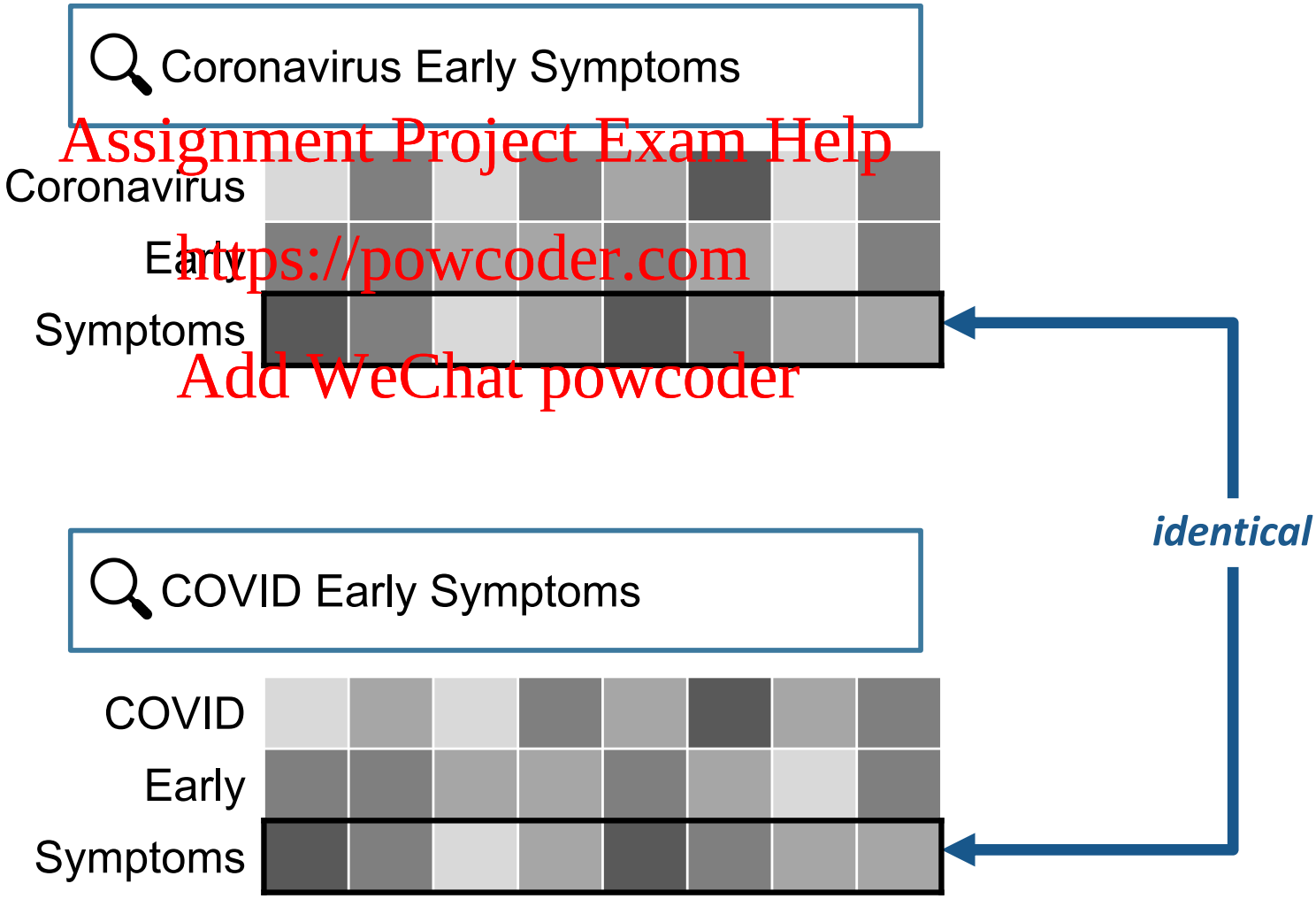
Representing Text

Word Vectors: Map each word to a dense vector



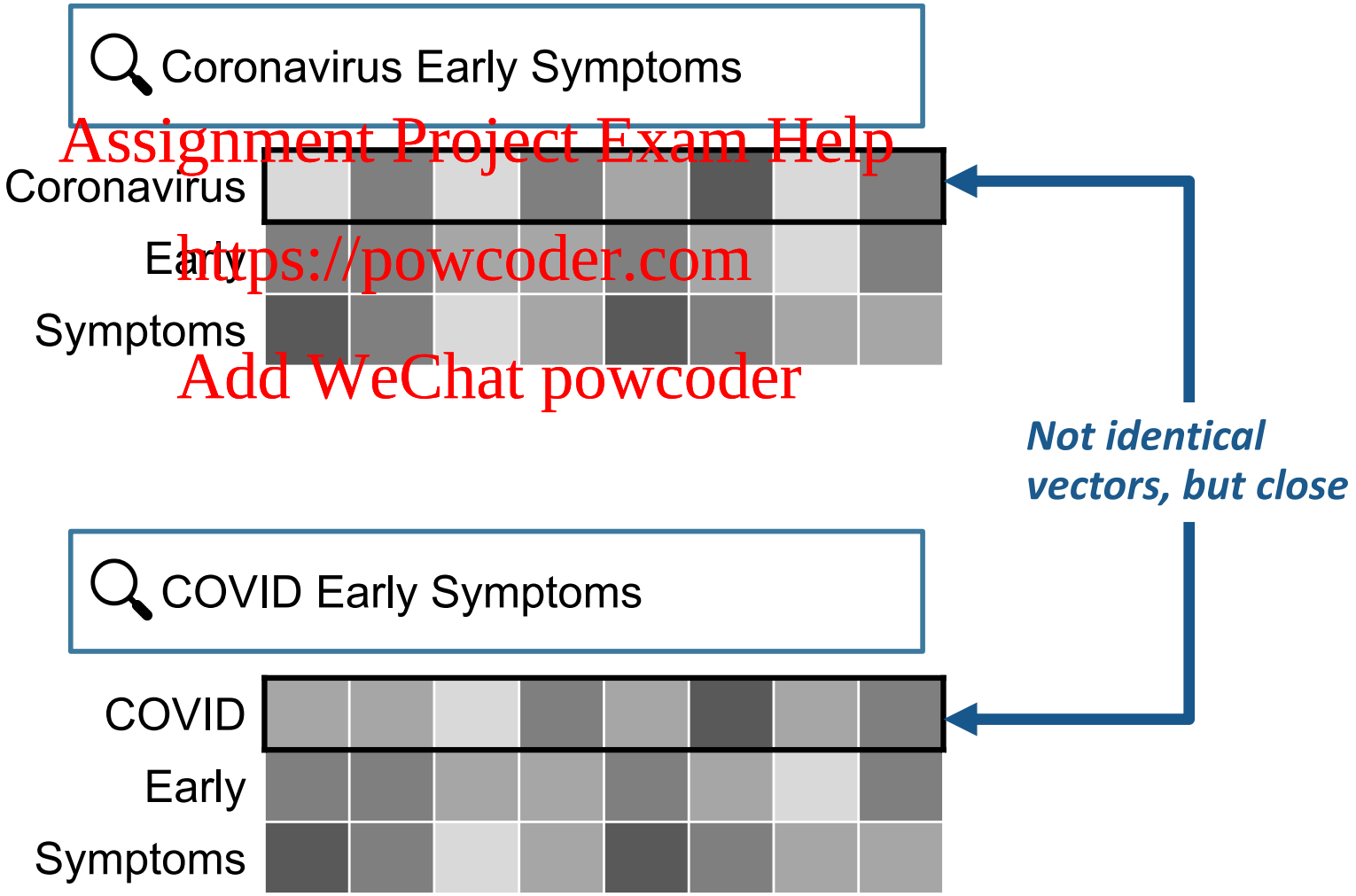
Representing Text

Word Vectors: Map each word to a dense vector



Representing Text

Word Vectors: Map each word to a dense vector



Problem: Not Context-Aware

✅ Handles different tokens with similar meanings

"**Coronavirus**" *has a similar vector than* "**COVID**"

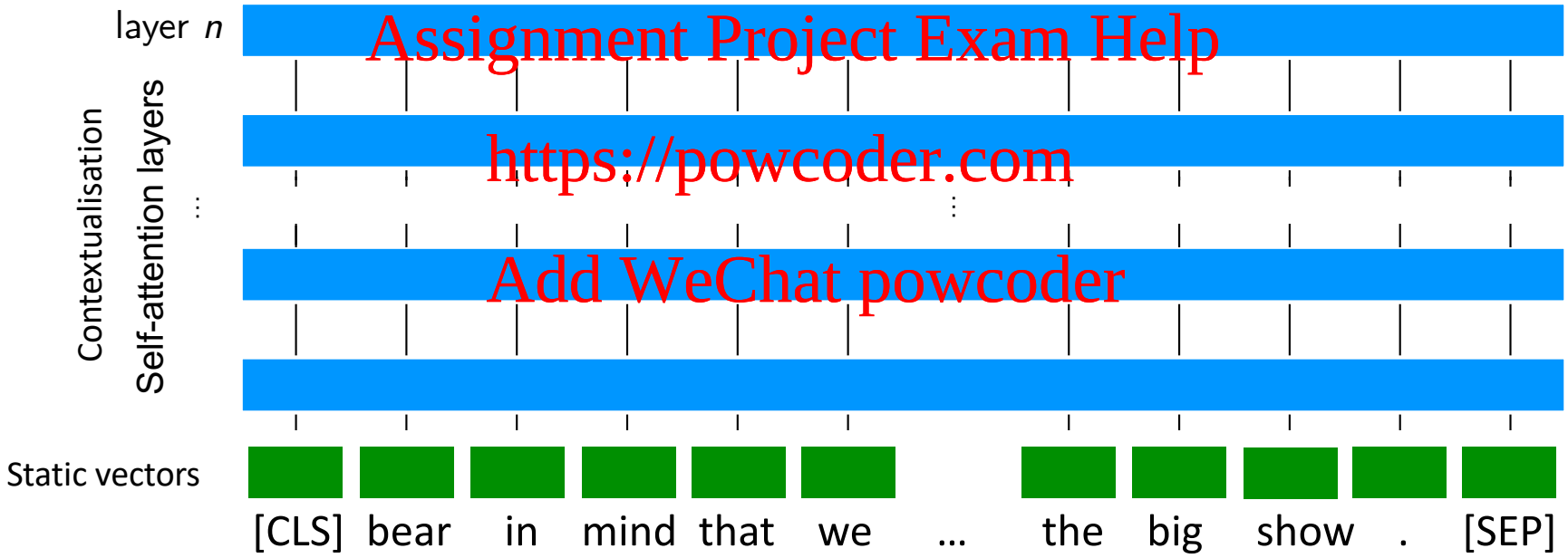

❌ Doesn't handle a single token with multiple possible meanings

"A **bear** is raiding homes in California."

has the same vector than

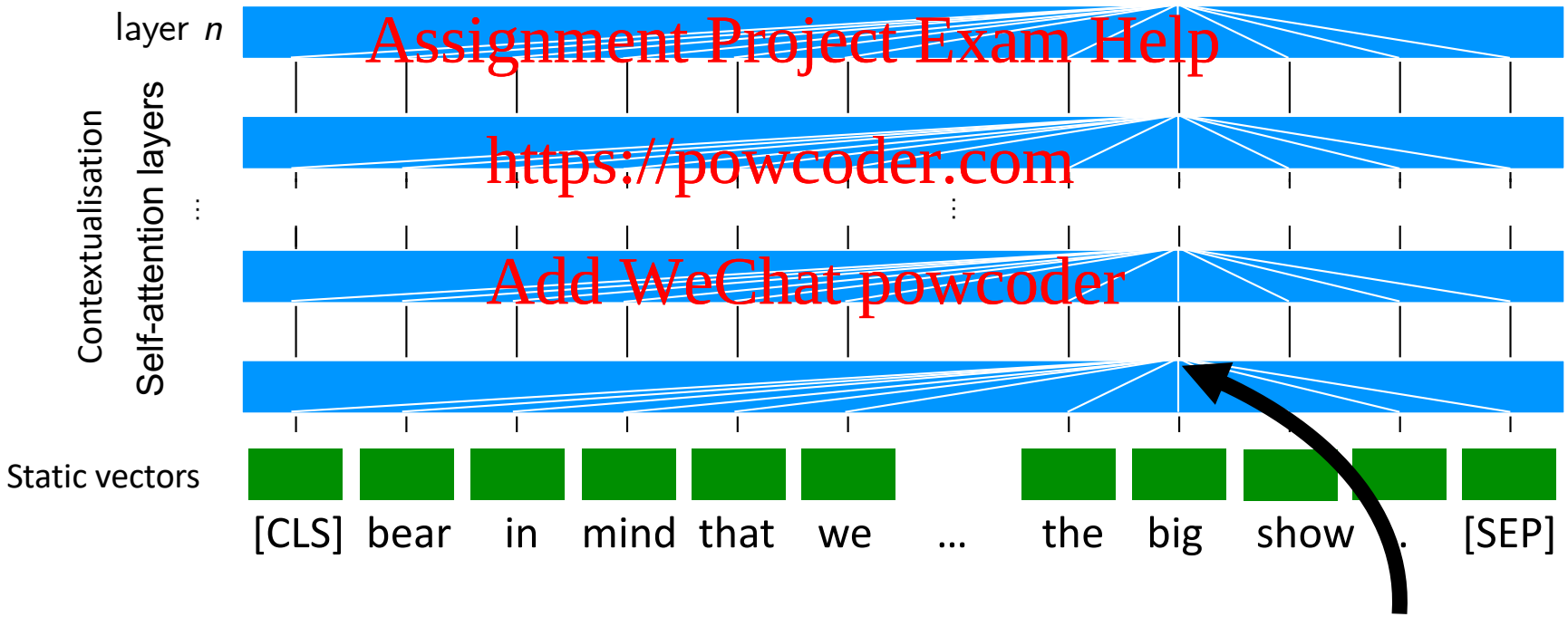
"**Bear** in mind that we do not have much time before the big show."

BERT & Transformer Networks



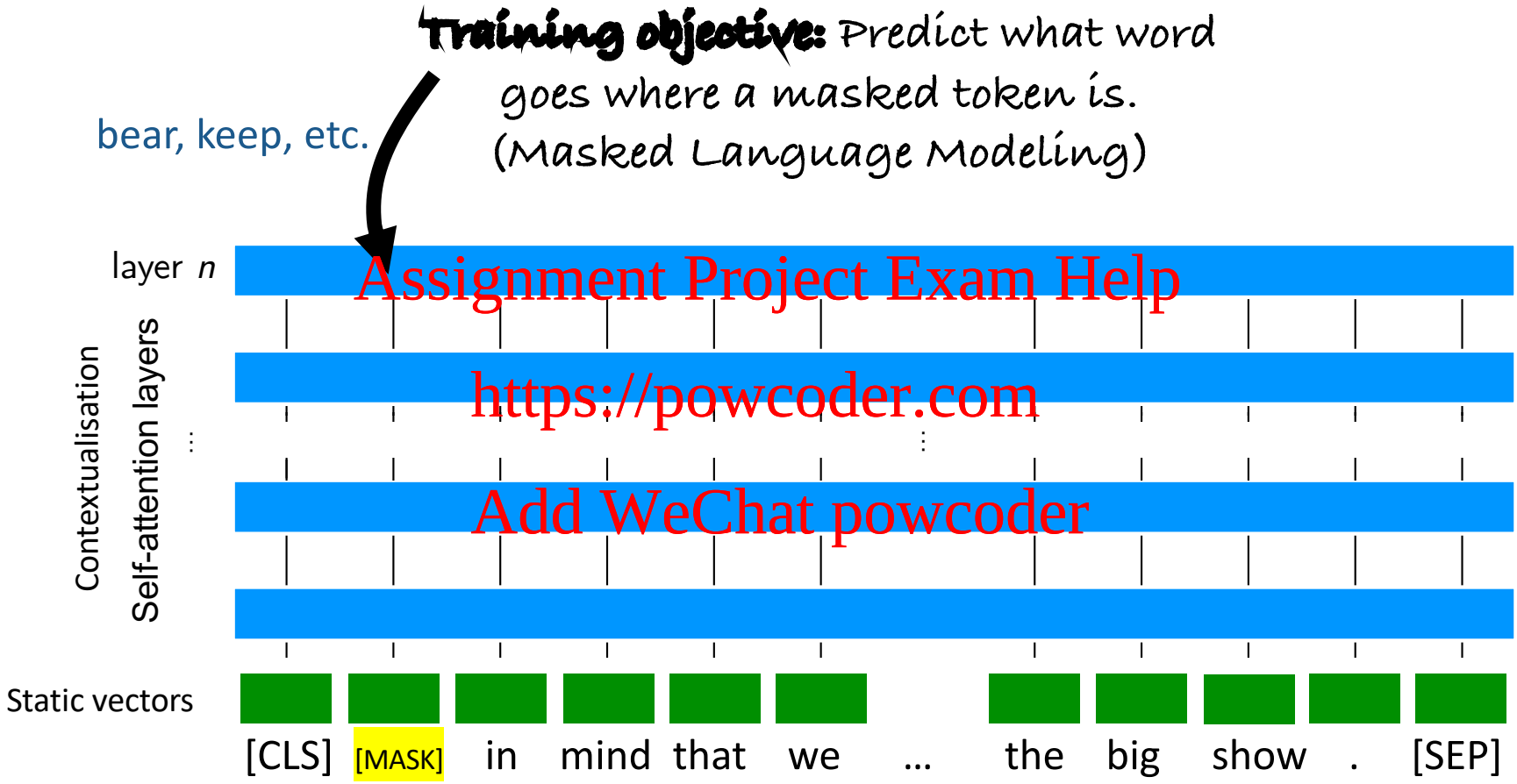
BERT & Transformer Networks

By the end, a "contextualised" vector is produced – one that knows how the word is used in context.

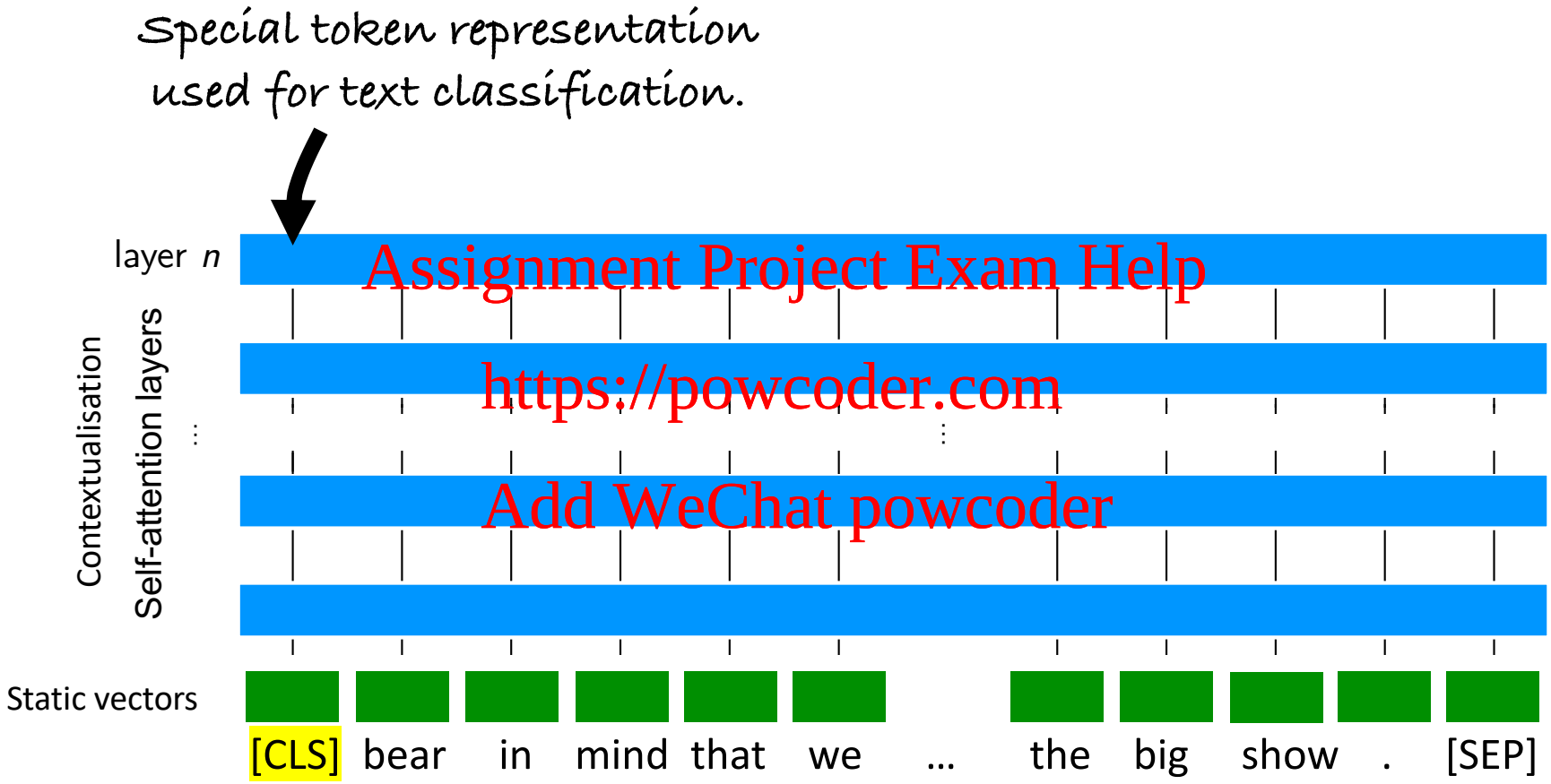


At each layer, a token's vector is a learned combination of all the other vectors in the text.

BERT & Transformer Networks



BERT & Transformer Networks



Problem: Not Context-Aware

✅ Handles different tokens with similar meanings

"**Coronavirus**" *has a similar vector than* "**COVID**"


✅ Doesn't handles a single token with multiple possible meanings

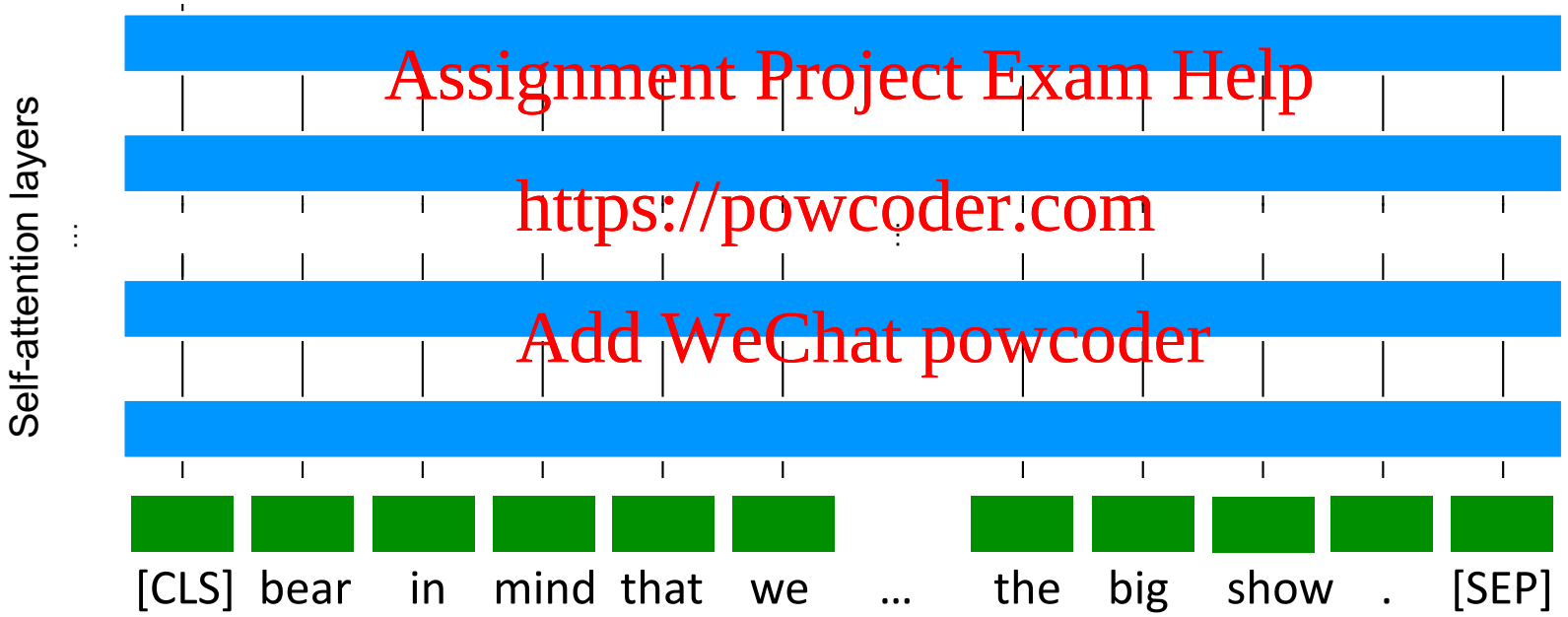
"A **bear** is raiding homes in California."

has a different vector than

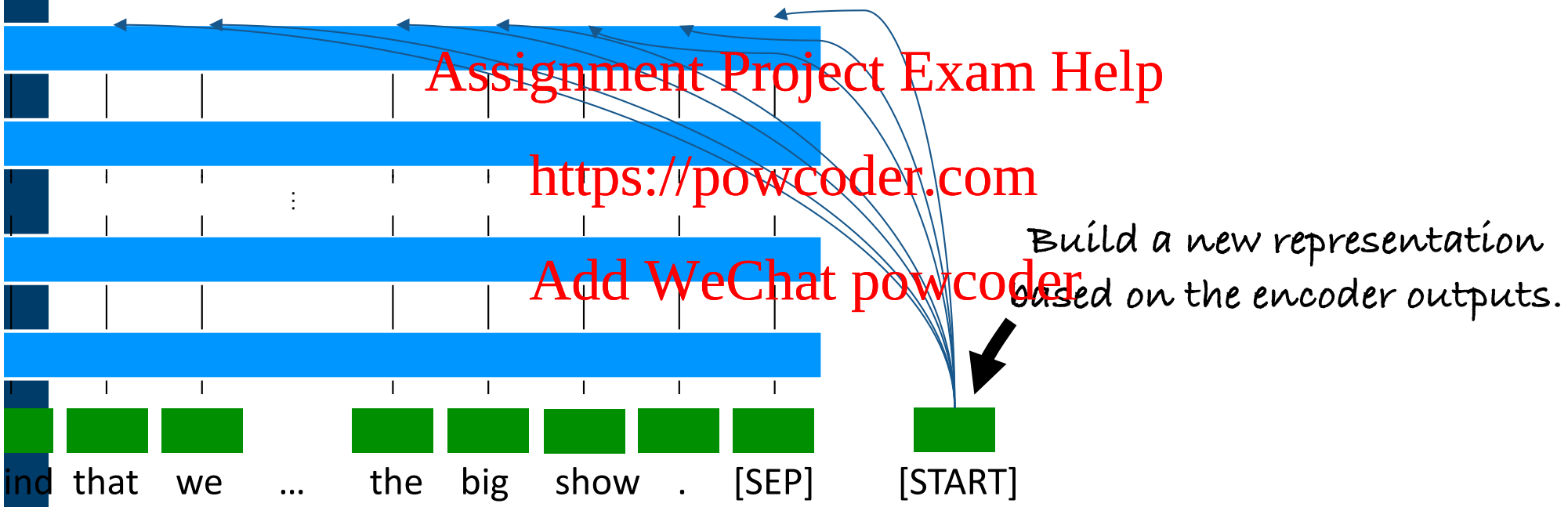
"**Bear** in mind that we do not have much time before the big show."

T5 & Text Generation

Transformer "Encoder"

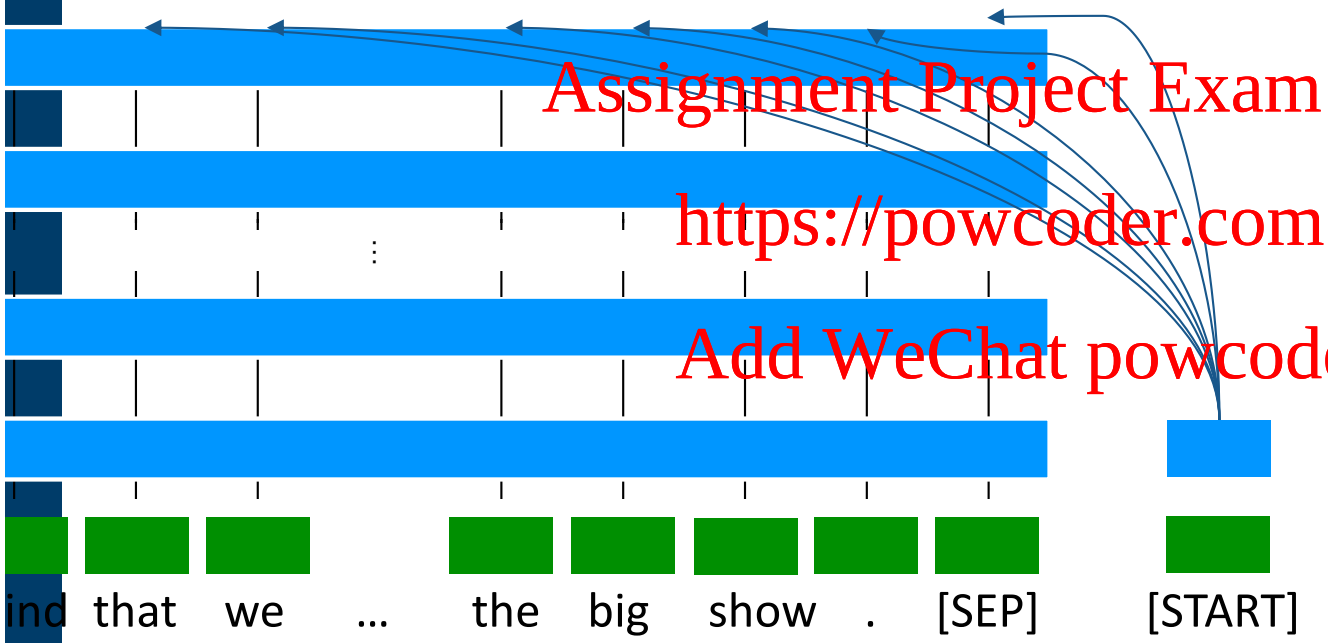


T5 & Text Generation

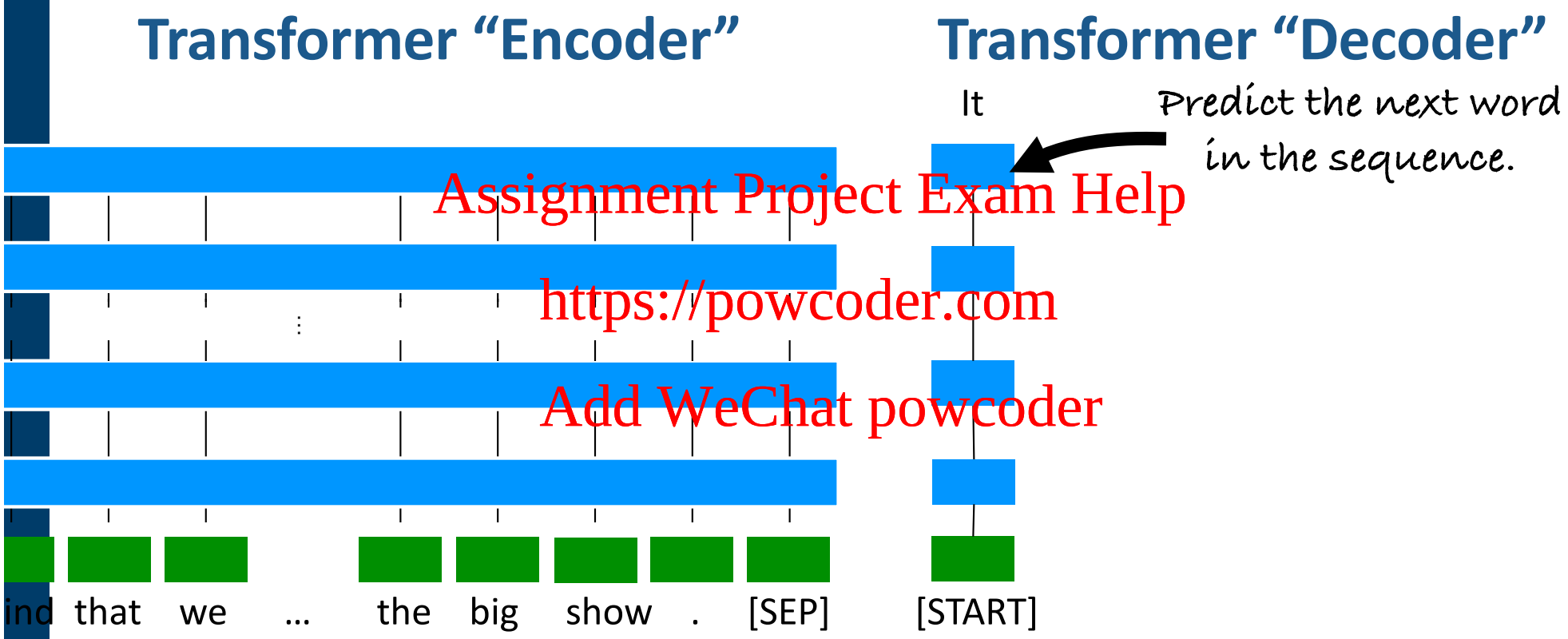


T5 & Text Generation

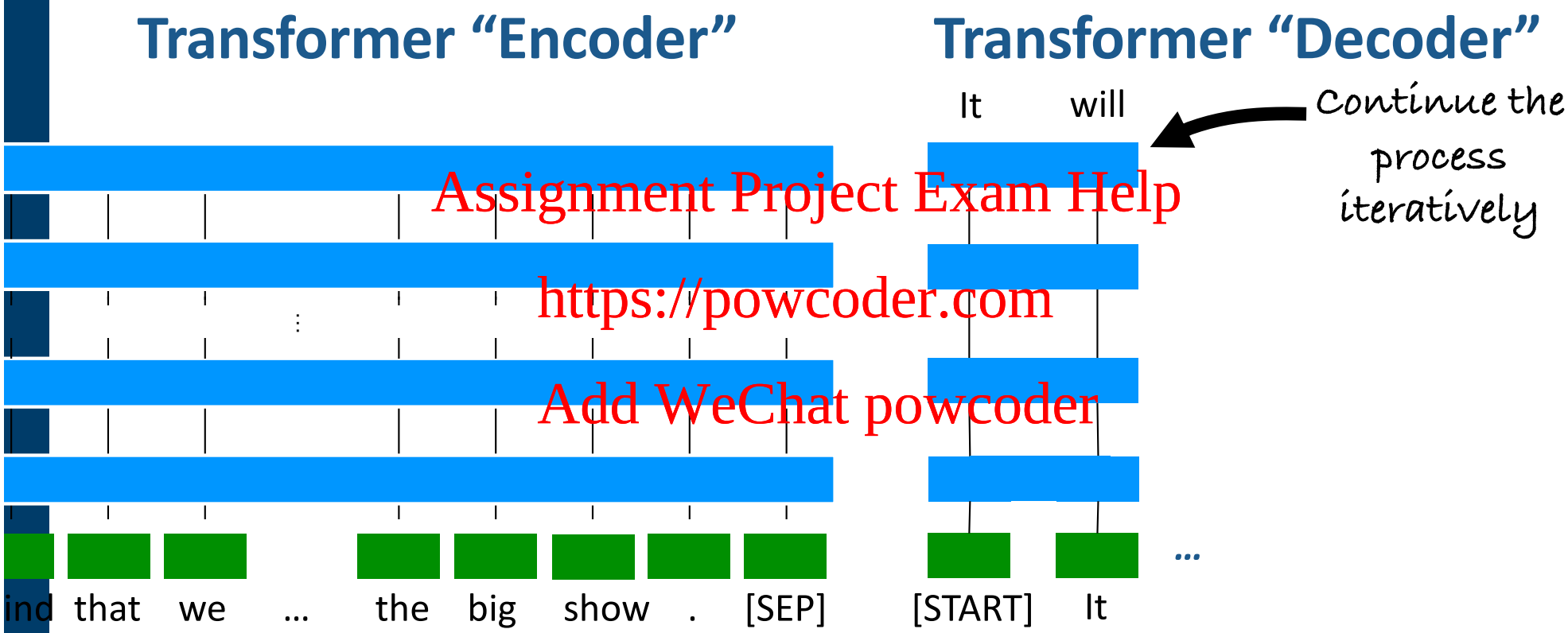
Transformer "Encoder" **Transformer "Decoder"**



T5 & Text Generation



T5 & Text Generation



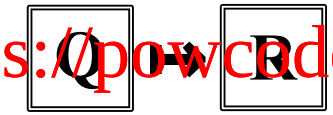
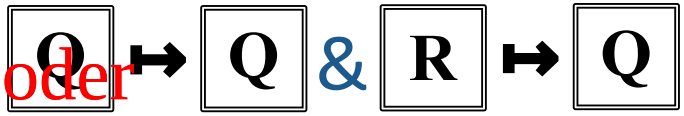
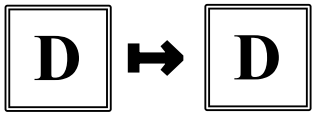
Summary

Neural NLP techniques can:

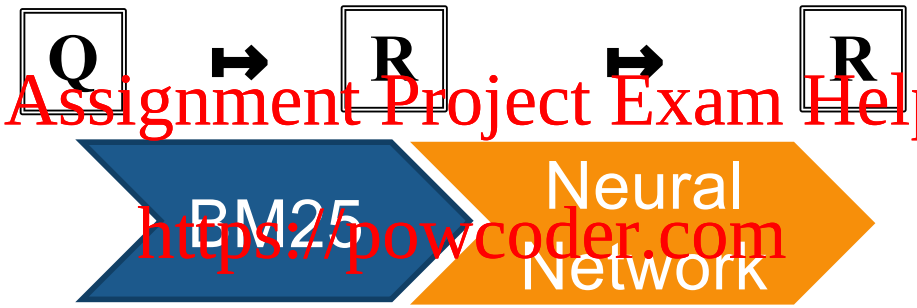
- Build representations that place similar words near each other
- Build representations that can distinguish different meanings a single word may have
- Generate text sequences

We will now use these tools to perform IR operations!

Today

1. Review of LTR & Basics of Neural Networks for NLP
2. **Neural Re-ranking** 
3. Neural Retrieval 
4. Neural Query Rewriting & PRF 
5. Neural Document Rewriting 
6. Neural IR in PyTerrier

Neural Re-Ranking



Why Neural Re-Ranking?

Simple formulation: Given a query and document, assign a new score.

$$R \mapsto R$$

qid	query	docno	score
0	glasgow weather	15213	0.5134
0	glasgow weather	42635	0.3742
0	glasgow weather	26340	0.3223
…	…	…	…

qid	query	docno	score
0	glasgow weather	26340	0.9312
0	glasgow weather	15213	0.1151
0	glasgow weather	52363	0.0215
…	…	…	…

Computational cost: Lexical retrieval method like BM25 are simple and achieve reasonably recall already; you just need to re-score a set of 100-1000 query-document pairs using an expensive NN.

Test Collection Suitability: Most benchmarks are based on pooling of lexical systems like BM25, so there is a higher likelihood that documents in this set have relevance assessments.

Formulation

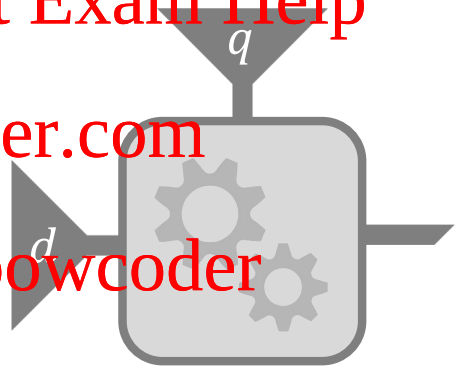
(query, document) -> score

Coronavirus Early Symptoms

Document:

Title: How can we evaluate an interrelation of symptoms?

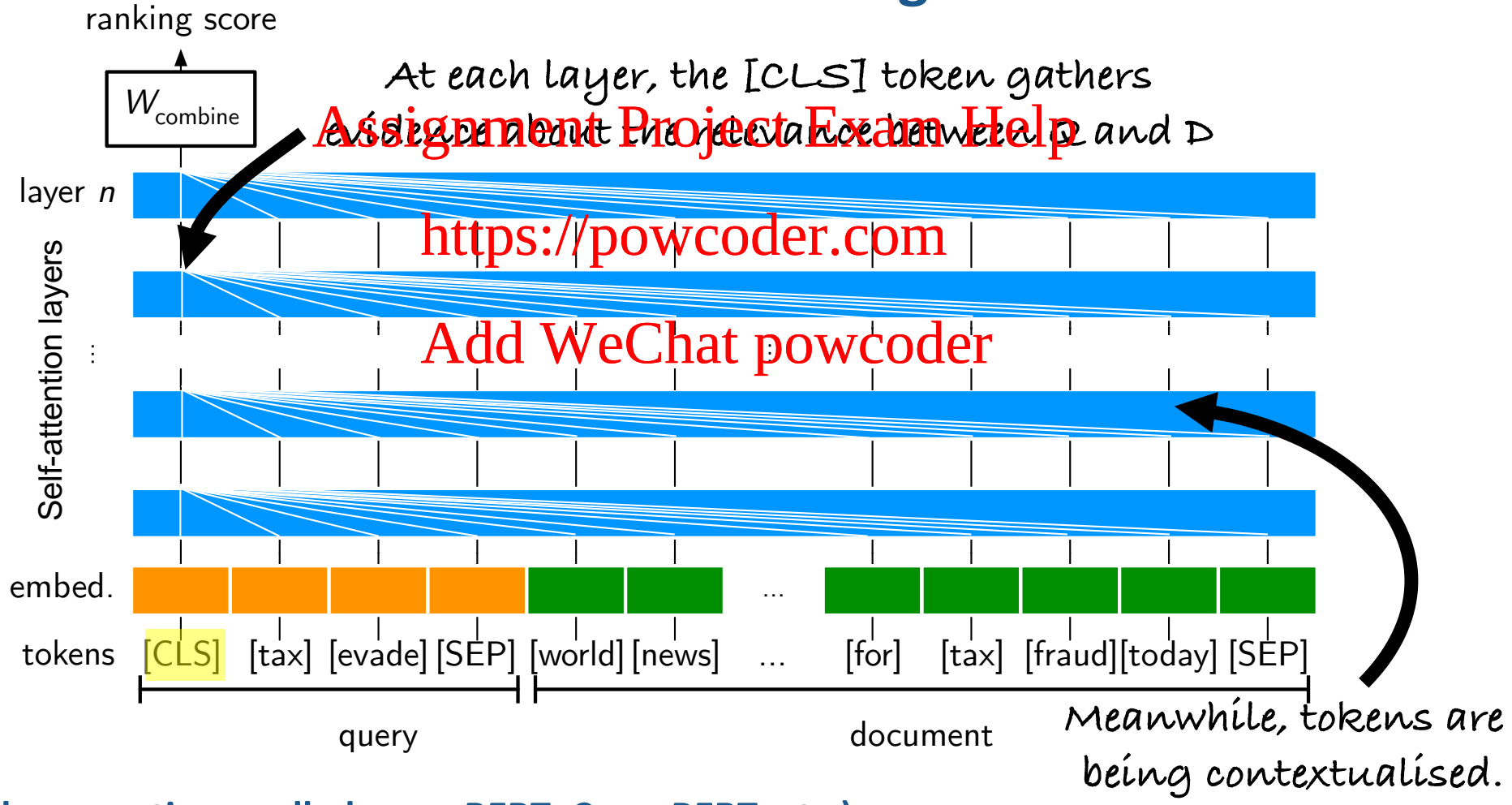
Abstract: A pandemic of 2019 novel coronavirus (COVID-19) is an international problem and factors associated with increased risk of mortality have been reported. However, there exists limited statistical method to estimate a comprehensive risk for a case in which a patient has several characteristics…



$rel = 2.142$

"Vanilla BERT" – The simplest approach that works (really well)!

Idea: Concatenate query and document; let the model learn how to combine into a ranking score.

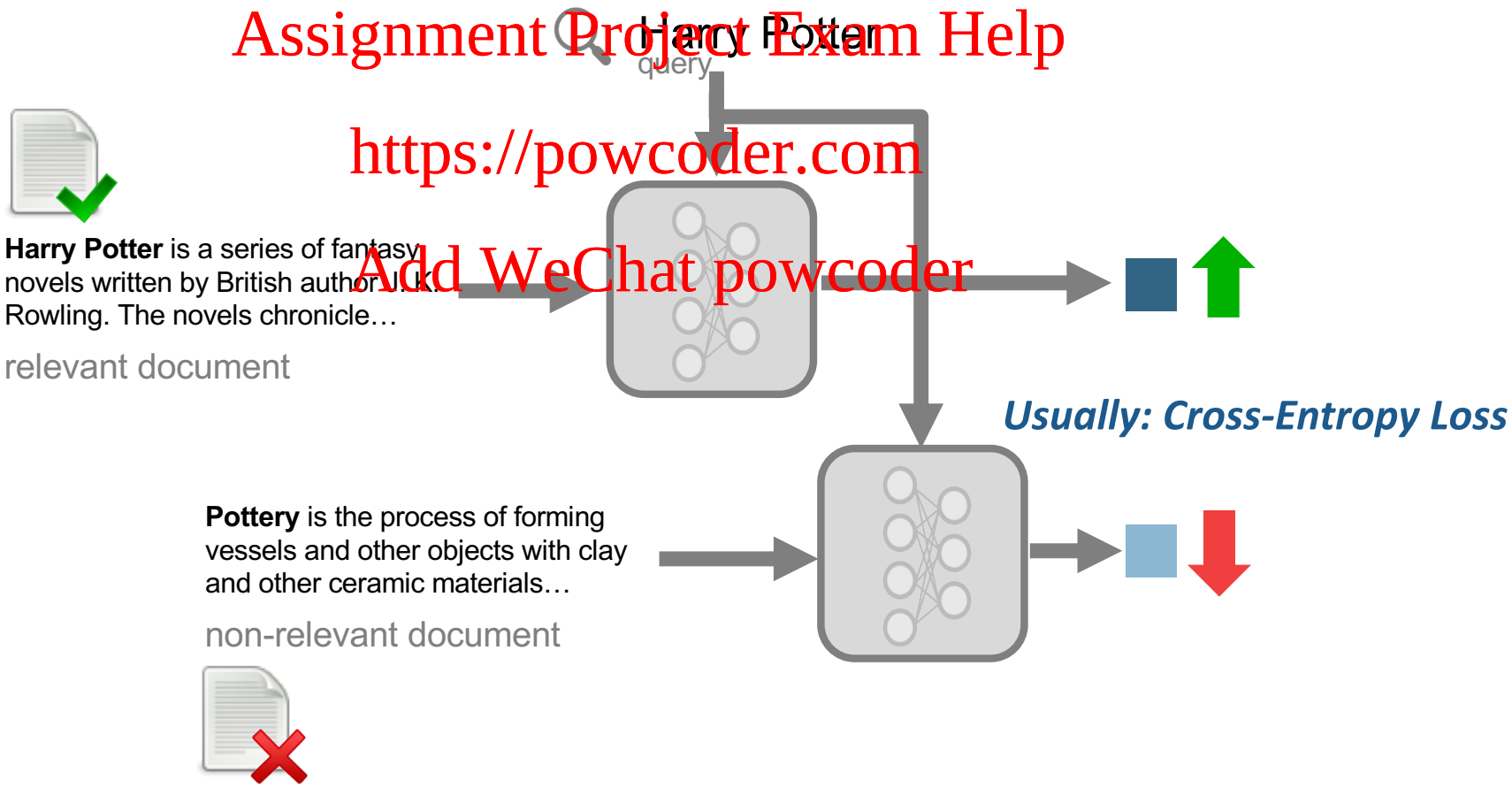


(Also sometimes called monoBERT, Cross BERT, etc.)

Training

Usually: Pairwise training.

With a query, a relevant doc and non-relevant doc, maximize the relative score between them.



Handling Long Documents

The cost of models like BERT are quadratic over the number of tokens. There is also an inherit maximum length (usually 512).

Strategies for long documents:

- **FirstP** – Only take the first 512 tokens

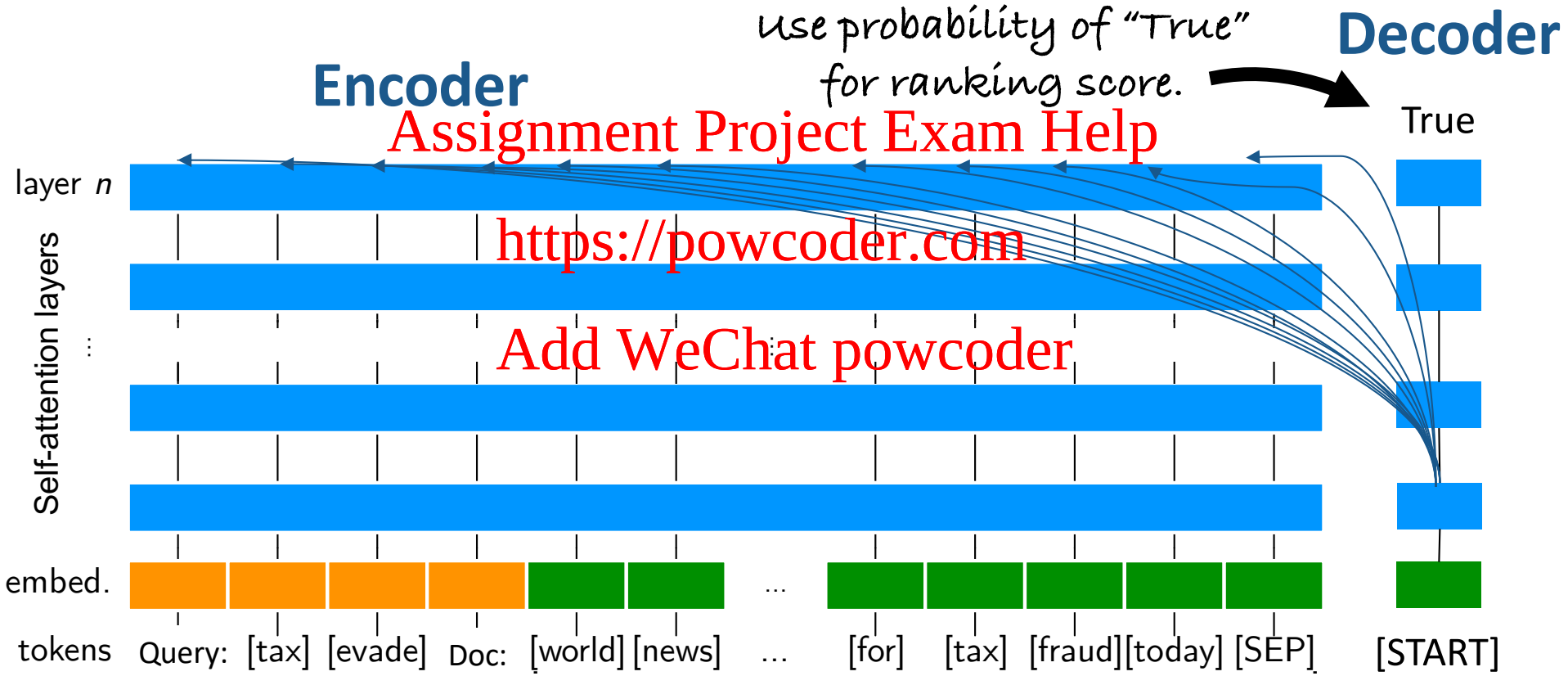
- **MaxP** – Split the document into passages; take the maximum score over all the passages

- **MeanCLS** – Take the average over [CLS] representations of each passage before computing a relevance score

- **PARADE** – Use another transformer to combine the [CLS] representations of each passage

Re-Ranking with a Text Generation Model (e.g., monoT5)

Idea: Generate "True" or "False", prompted by query and document.



Approaches for Re-Ranking

Technique	Example
Use [CLS] Token	Vanilla BERT [1]
Generate True/False Token	monoT5 [2]

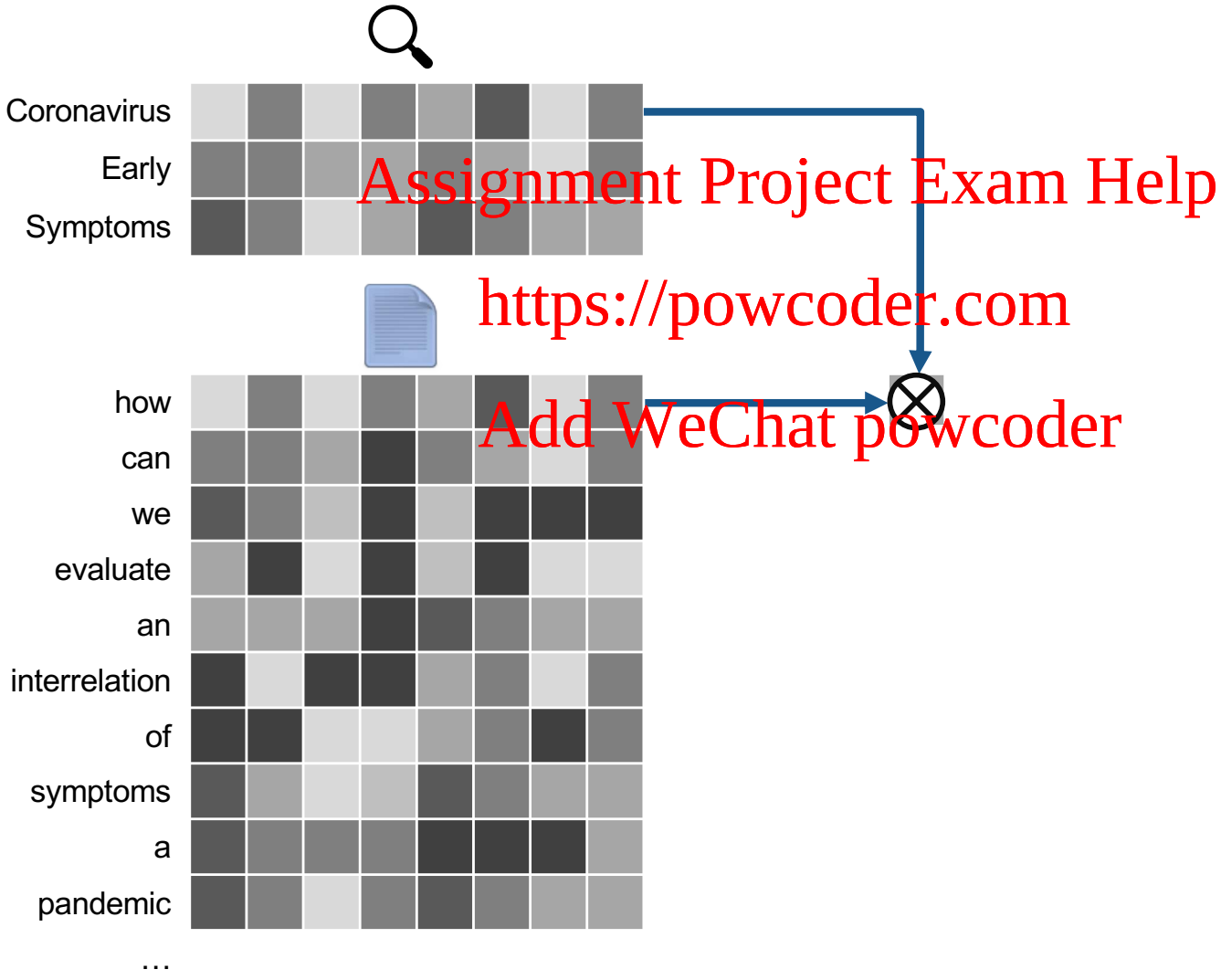
[1] MacAvaney et al. CEDR: Contextualized Embeddings for Document Ranking. SIGIR 2019.
[2] Nogueira et al. Document Ranking with a Pretrained Sequence-to-Sequence Model. xrxiv 2020.

Approaches for Re-Ranking

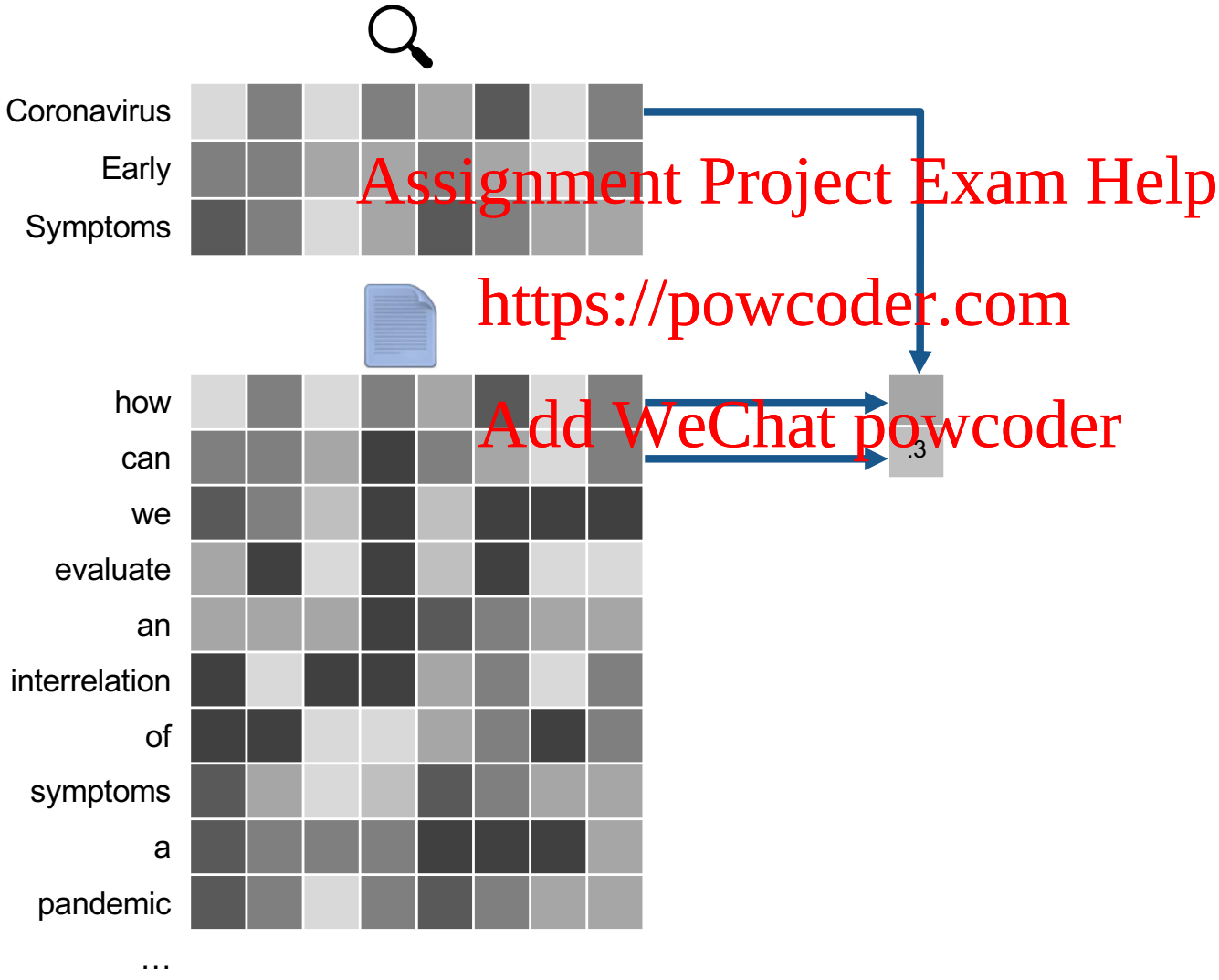
Technique	Example
Use [CLS] Token	Vanilla BERT [1]
Generate True/False Token	monoT5 [2]
Compare token representations	CEDR [1]

[1] MacAvaney et al. CEDR: Contextualized Embeddings for Document Ranking. SIGIR 2019.
[2] Nogueira et al. Document Ranking with a Pretrained Sequence-to-Sequence Model. xrxiv 2020.

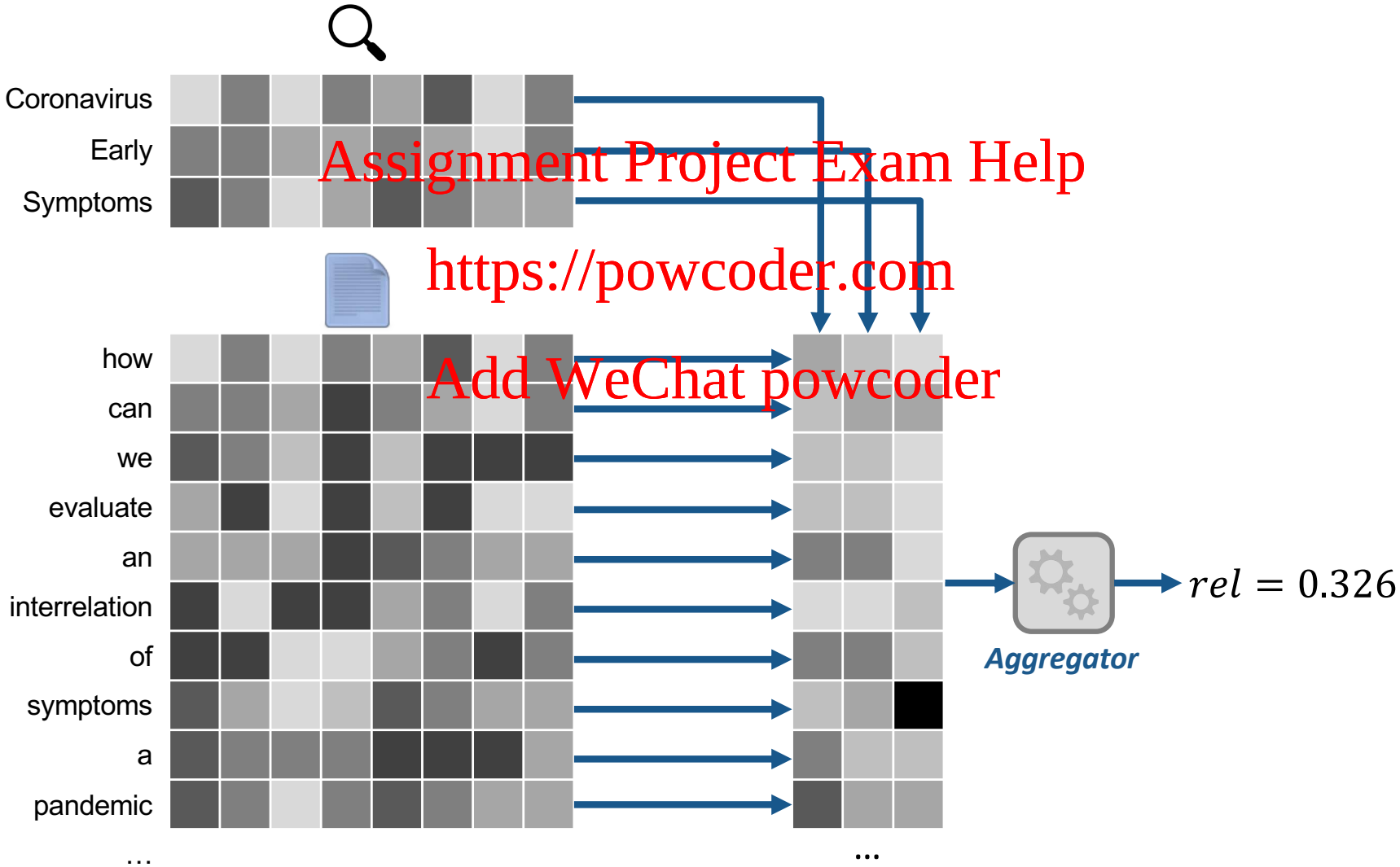
Comparing Token Representations



Comparing Token Representations

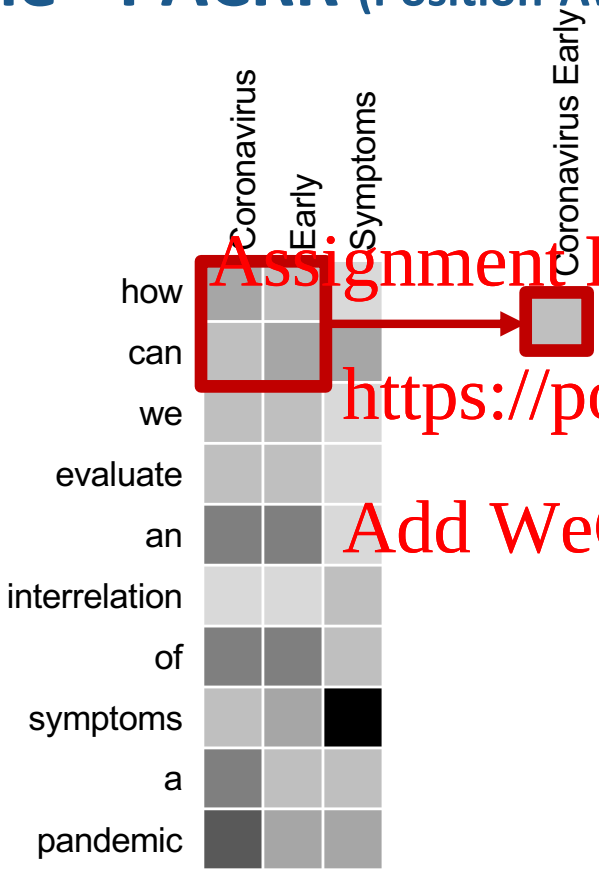


Comparing Token Representations



Interaction Aggregation

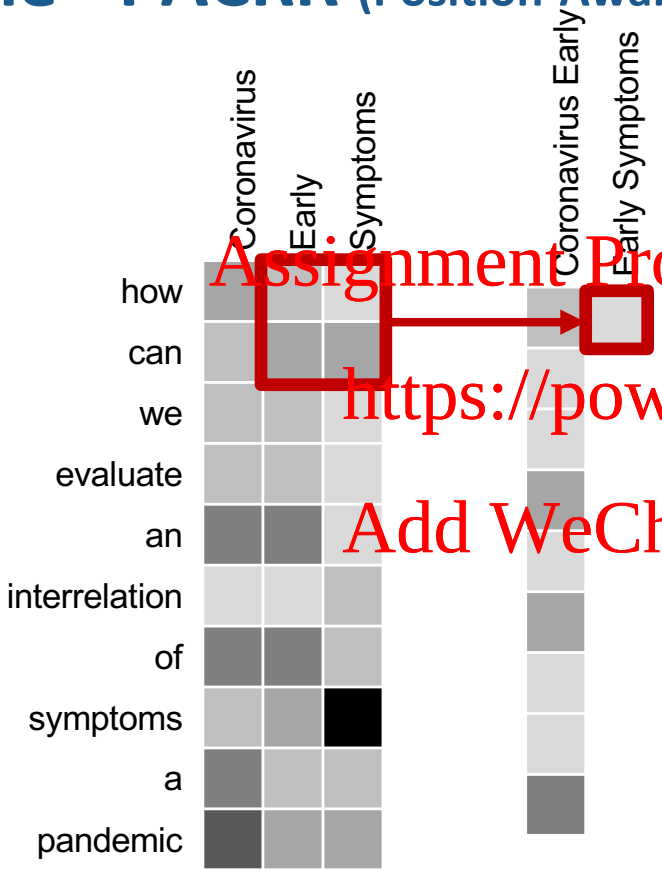
Example - PACRR (Position-Aware Convolutional-Recurrent Relevance)



Hui et al. PACRR: A Position-Aware Neural IR Model for Relevance Matching. EMNLP 2017.

Interaction Aggregation

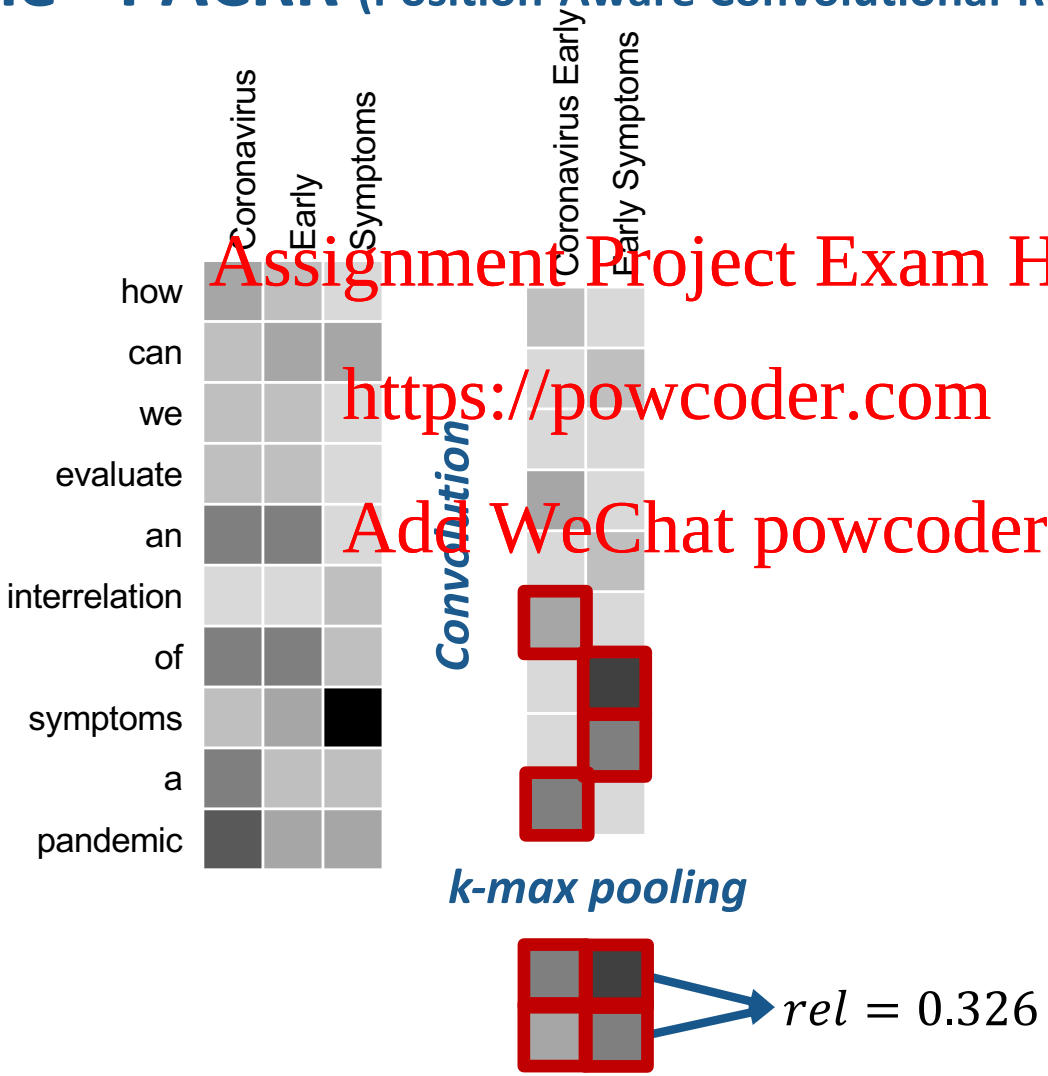
Example - PACRR (Position-Aware Convolutional-Recurrent Relevance)



Hui et al. PACRR: A Position-Aware Neural IR Model for Relevance Matching. EMNLP 2017.

Interaction Aggregation

Example - PACRR (Position-Aware Convolutional-Recurrent Relevance)



Hui et al. PACRR: A Position-Aware Neural IR Model for Relevance Matching. EMNLP 2017.

Problem: Efficiency

Using methods like BERT and T5 for ranking is effective, but also very slow.

E.g., BERT takes 45x longer than BM25: (1.7 seconds)

	name	map	ndcg	ndcg_cut_10	mrt
0	BM25	0.077680	0.177728	0.644374	38.567969
1	BM25 >> BERT	0.085371	0.185821	0.740331	1748.576758

Approaches for Re-Ranking

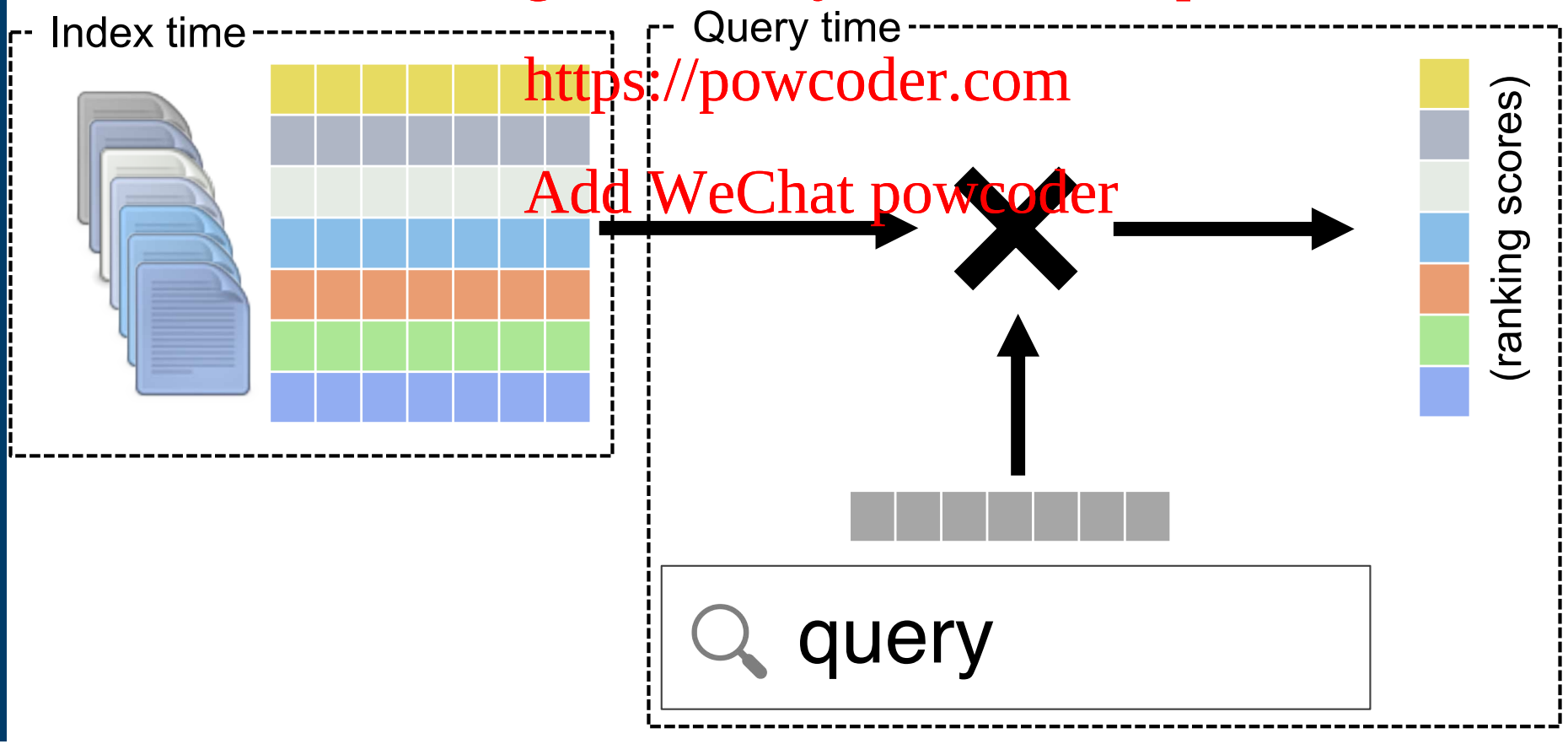
Technique	Example
Use [CLS] Token	Vanilla BERT [1]
Generate True/False Token	monoT5 [2]
Compare token representations	CEDR [1]
Predict token importance	EPIC [3]

For inexpensive re-ranking

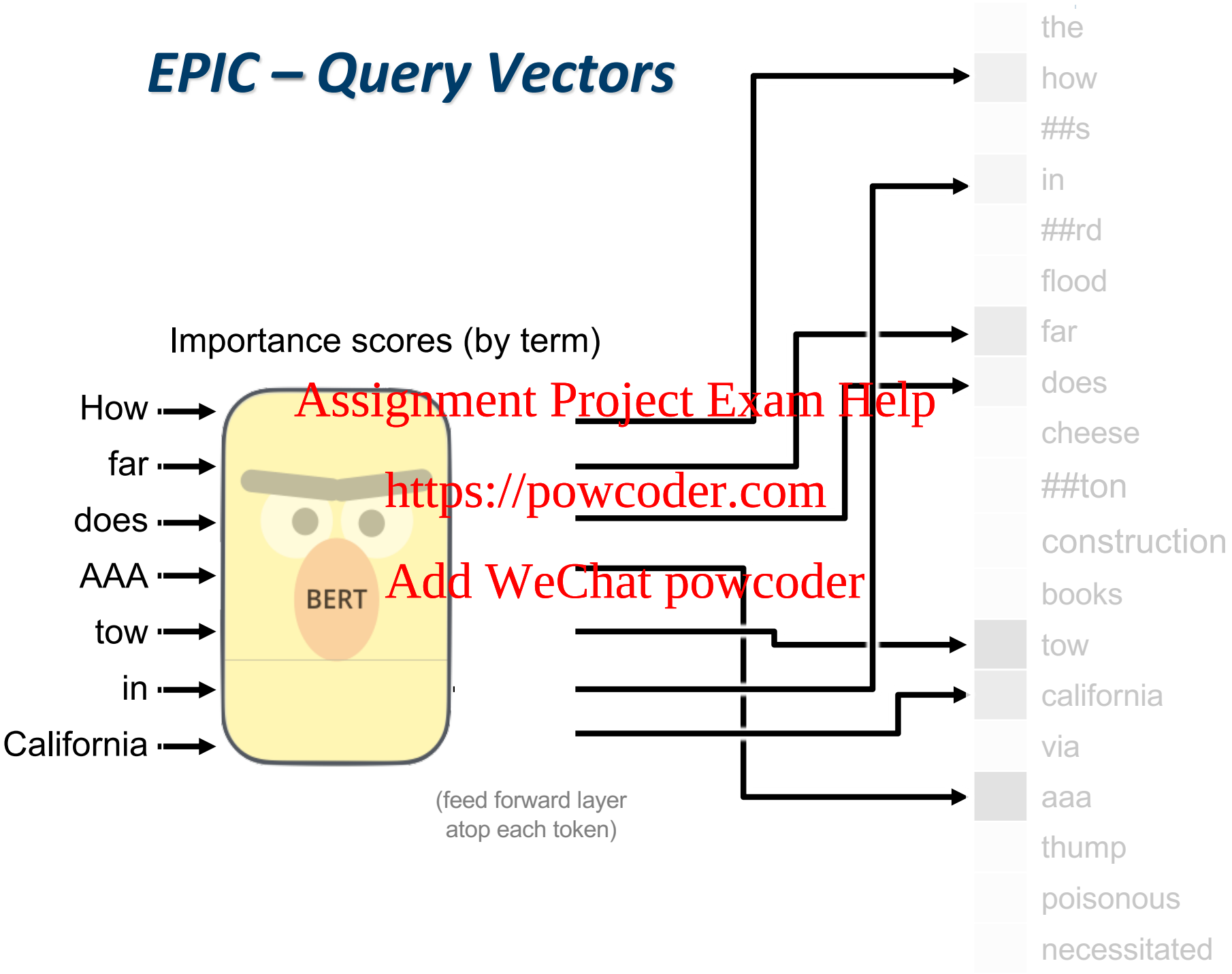
[1] MacAvaney et al. CEDR: Contextualized Embeddings for Document Ranking. SIGIR 2019.
[2] Nogueira et al. Document Ranking with a Pretrained Sequence-to-Sequence Model. xrxiv 2020.
[3] MacAvaney et al. Expansion via Prediction of Importance with Contextualization. SIGIR 2020.

EPIC

Main idea: Learn representations that are fast to score.

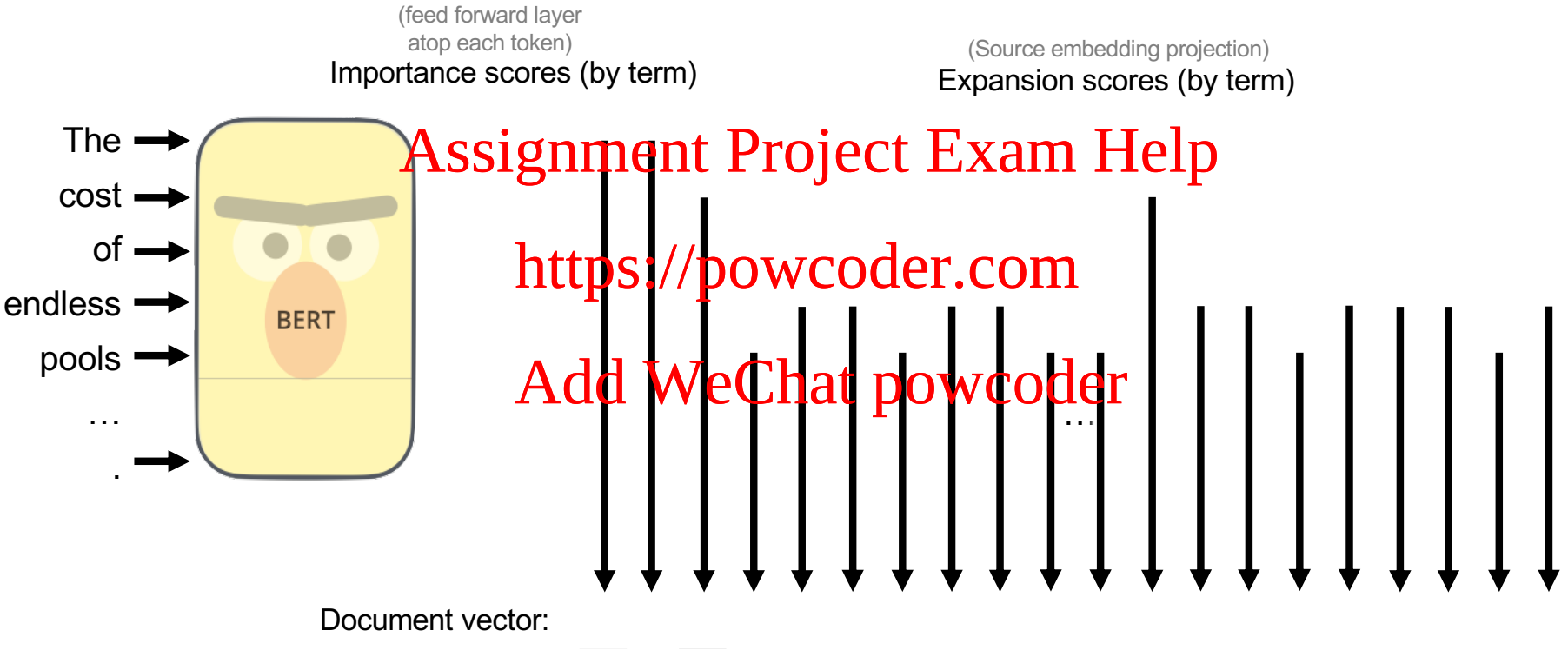


EPIC – Query Vectors



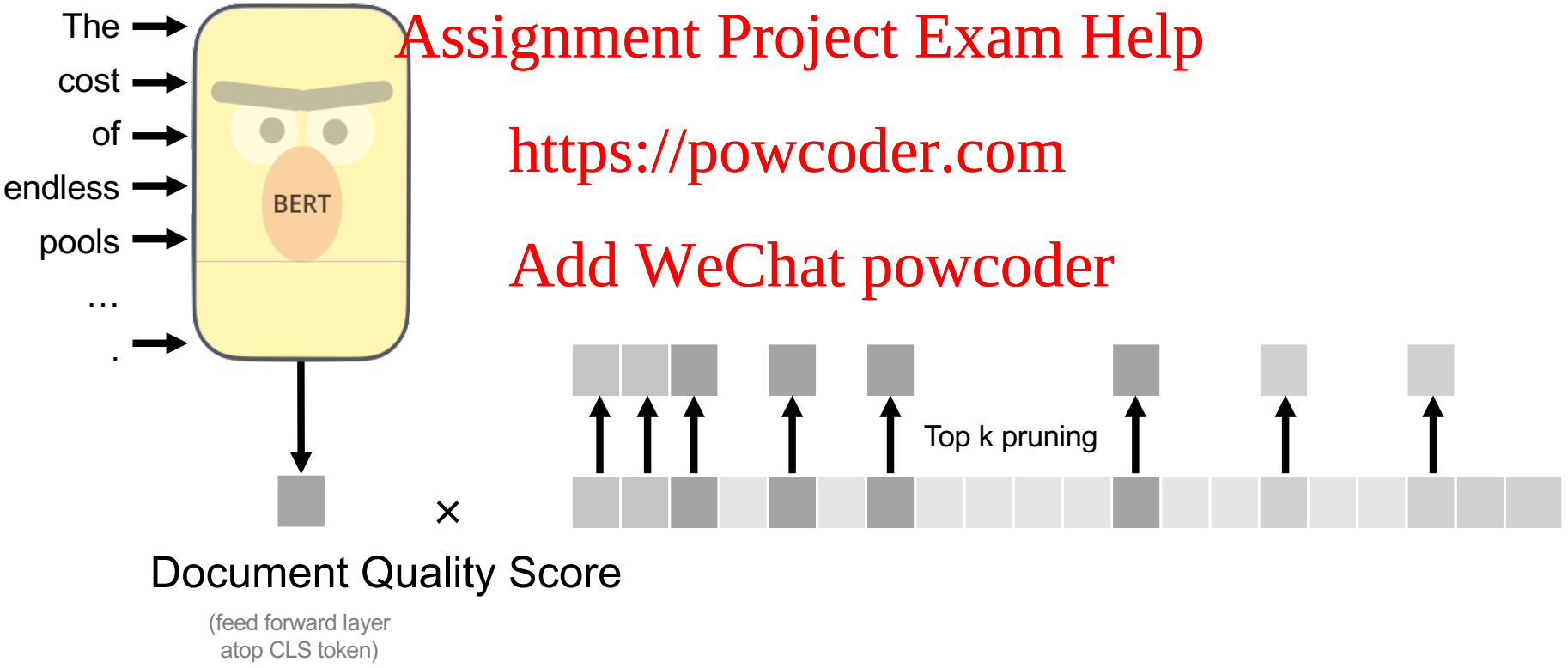
BERT Graphics from Jay Alammar, http://jalammar.github.io/illustrated-bert/

EPIC – Document Vectors



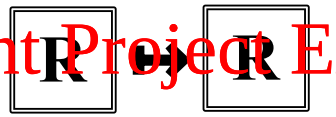
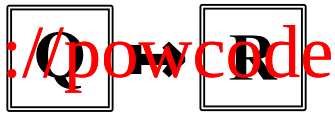
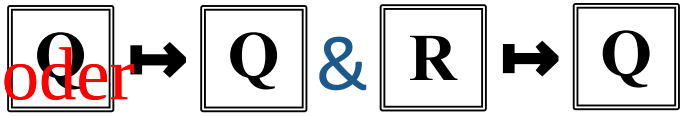
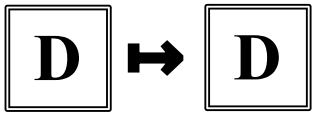
BERT Graphics from Jay Alammar, http://jalammar.github.io/illustrated-bert/

EPIC – Document Vectors



BERT Graphics from Jay Alammar, http://jalammar.github.io/illustrated-bert/

Today

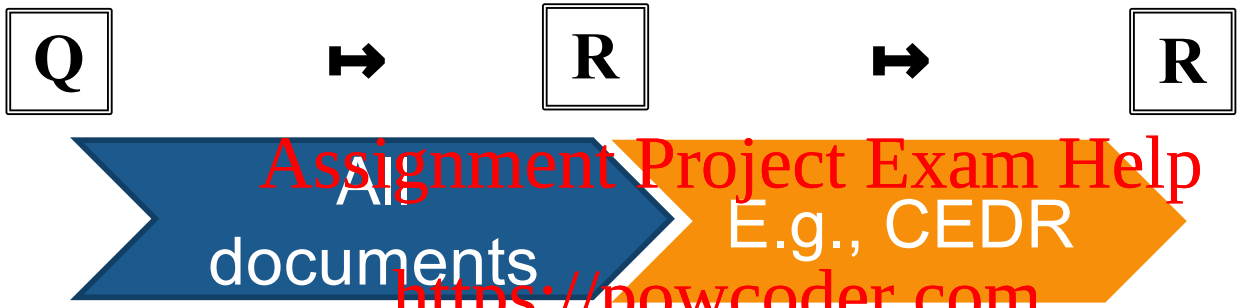
1. Review of LTR & Basics of Neural Networks for NLP
2. Neural Re-ranking 
3. **Neural Retrieval** 
4. Neural Query Rewriting & PRF 
5. Neural Document Rewriting 
6. Neural IR in PyTerrier

Neural Ranking

Can we replace the lexical scoring function with a neural network that better reflects what's important for the neural re-ranker?

Neural Ranking

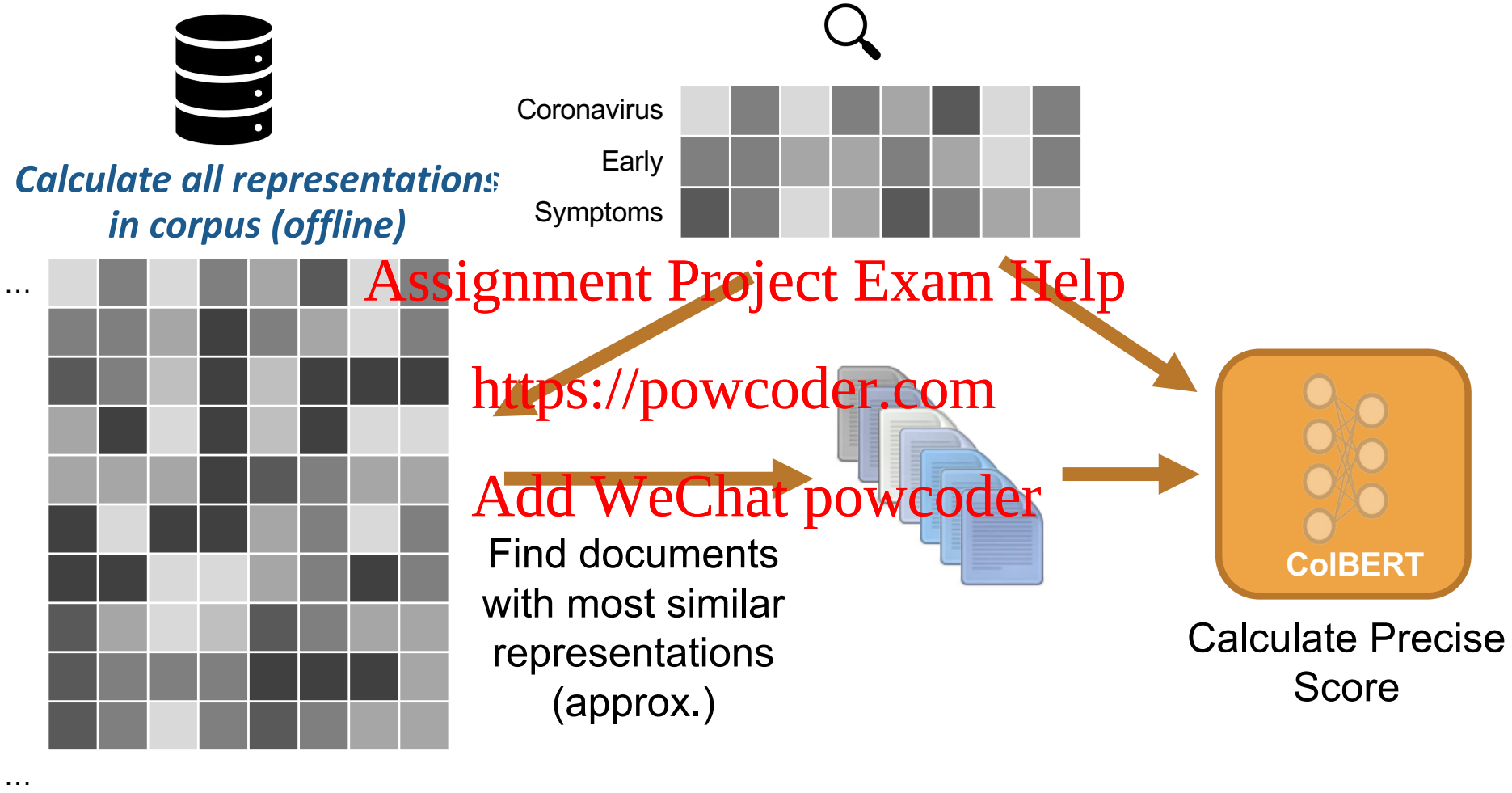
One solution: Score all documents in the corpus:



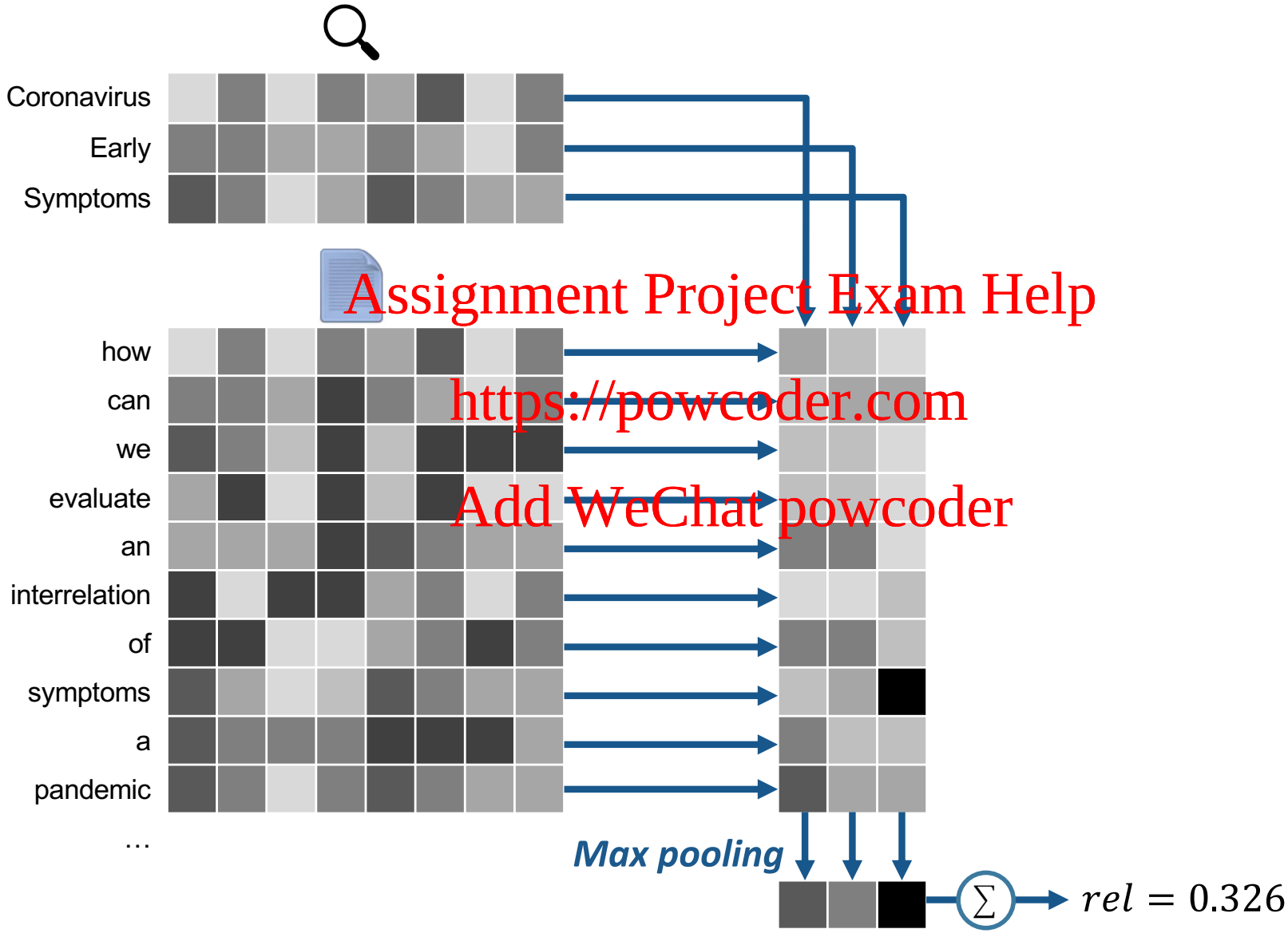
Major Problem: Remember that these approaches are very expensive – time-prohibitive to score all

Solution: Pre-compute representations that are fast to score.

ColBERT: Approx. Nearest Neighbour



Contextualized Late Interaction over BERT (ColBERT)



Khattab & Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. SIGIR 2020.

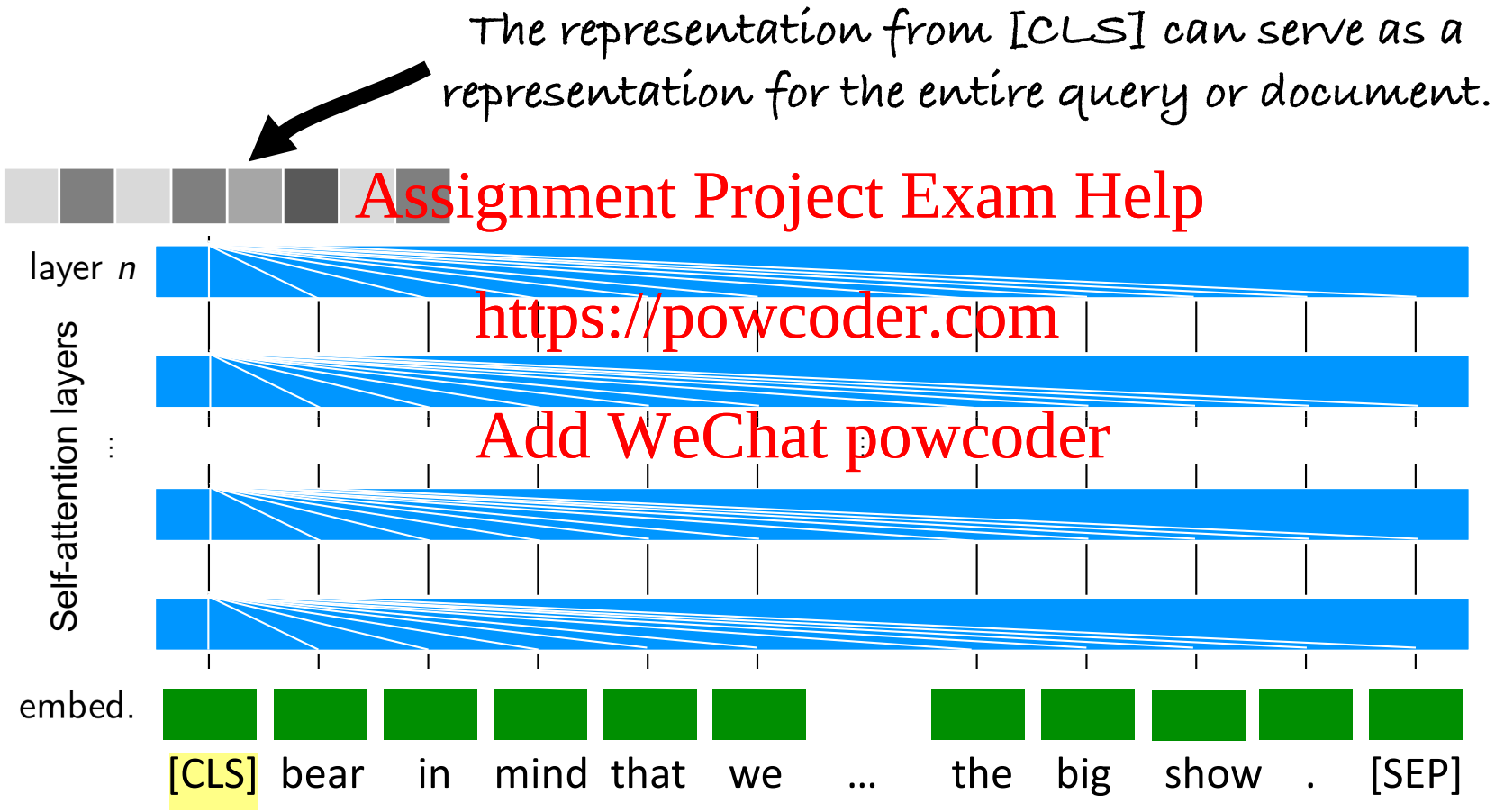
Problem with ColBERT

The token representations are very large (128 floats per token in each document)

This means a lot of storage is required.

Alternative: build just a single representation for each document.

Single-Representation BERT (Example: ANCE)



Overview: Dense Retrieval

Two main strategies:

1. Multiple-Representation

- One vector for each term
- Find similar vectors to those in the query
- Second stage required: score entire document
- Tends to be more robust across datasets than single-rep

2. Single Representation

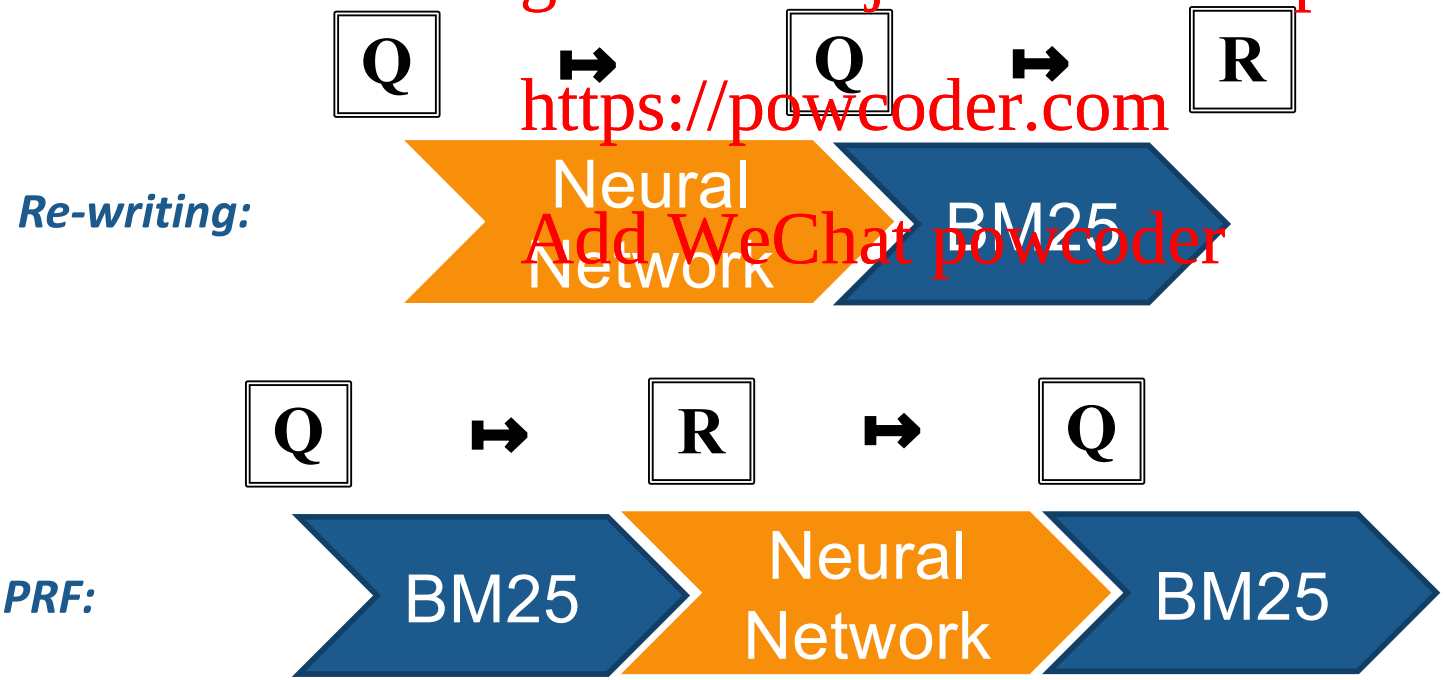
- One vector for document
- Find similar vectors to a representation for the query
- Tends to be faster & smaller than multi-rep

Today

1. Review of LTR & Basics of Neural Networks for NLP
2. Neural Re-ranking R ↦ R
3. Neural Retrieval Q ↦ R
4. **Neural Query Rewriting & PRF** Q ↦ Q & R ↦ Q
5. Neural Document Rewriting D ↦ D
6. Neural IR in PyTerrier

Neural Query Rewriting & PRF

Can we use a neural network to build better queries?



T5-QR & T5-PRF

Main idea: use a text generation model to generate better queries either based on:

(1) The query text alone

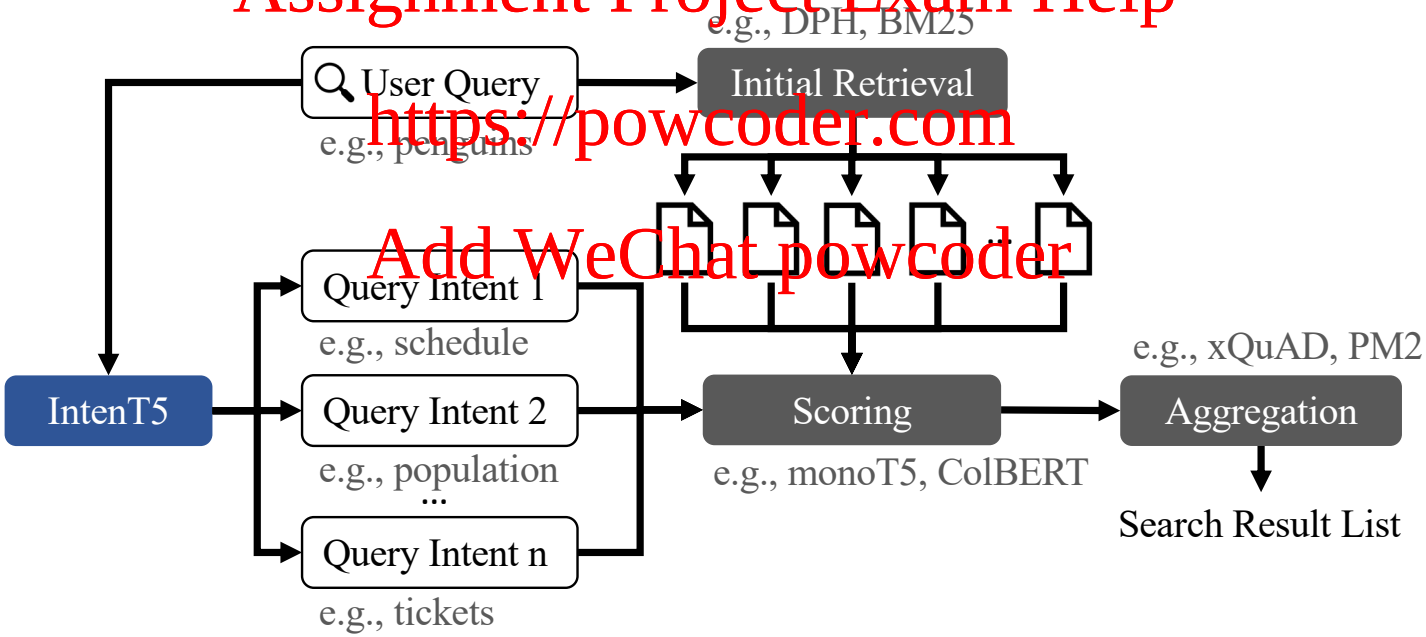
Example: *Do goldfish Grow -> How long do goldfish grow*

(2) The query text and the top-retrieved results

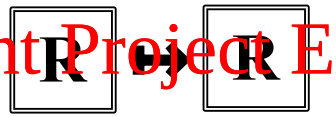
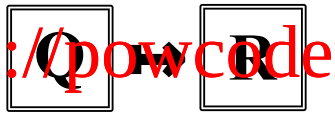
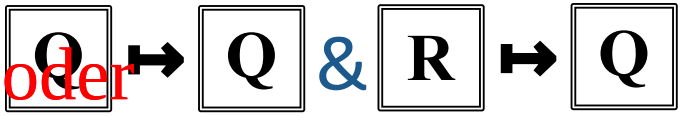
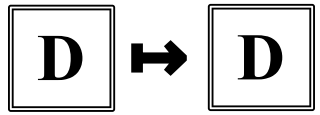
Example: *do goldfish grow. Context: A: The conditions goldfish are kept in plus their diet determine how large they will grow. I have seen goldfish grow ridiculously large in very small containers when their water was changed frequently. Goldfish will not grow if water conditions are poor. Fancy goldfish don't grow as large as Common goldfish. A good size would be around 5 inches body length for most fancy varieties, 8 inches for Comets and 12 inches for Common Goldfish. These sizes are usually only attained by pond grown fish.*
-> large kept container water conditions

IntenT5

Generating multiple queries and aggregating them can help improve search result diversity.

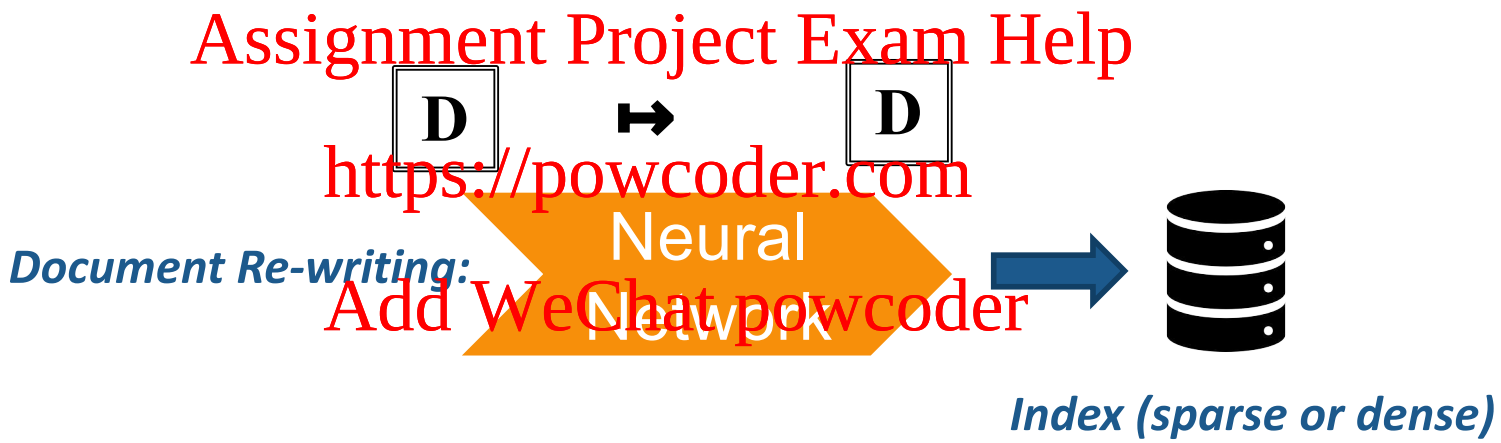


Today

1. Review of LTR & Basics of Neural Networks for NLP
2. Neural Re-ranking 
3. Neural Retrieval 
4. Neural Query Rewriting & PRF 
5. **Neural Document Rewriting** 
6. Neural IR in PyTerrier

Document Re-Writing

Can we find better representations of documents to index?



DeepCT

Main idea: Boost the term frequency of important words by repeating them in the text.

Abstract:
Nidovirus subgenomic mRNAs contain a leader sequence derived from the 5′ end of the genome fused to different sequences (‘bodies’) derived from the 3′ end. Their generation involves a unique mechanism of discontinuous subgenomic RNA synthesis that resembles copy-choice RNA recombination. During this process, the nascent RNA strand is transferred from one site in the template to another, during either plus or minus…

Abstract:
Nidovirus Nidovirus Nidovirus subgenomic subgenomic subgenomic mRNAs mRNAs mRNAs contain a leader sequence derived from the 5′ end of the genome fused to different sequences (‘bodies’) derived from the 3′ end. Their generation involves a unique mechanism of discontinuous **subgenomic subgenomic RNA RNA** synthesis that resembles copy-choice **RNA RNA RNA** recombination. During this process, the nascent **RNA RNA RNA** strand is transferred from one site in the template to another, during either plus or minus…

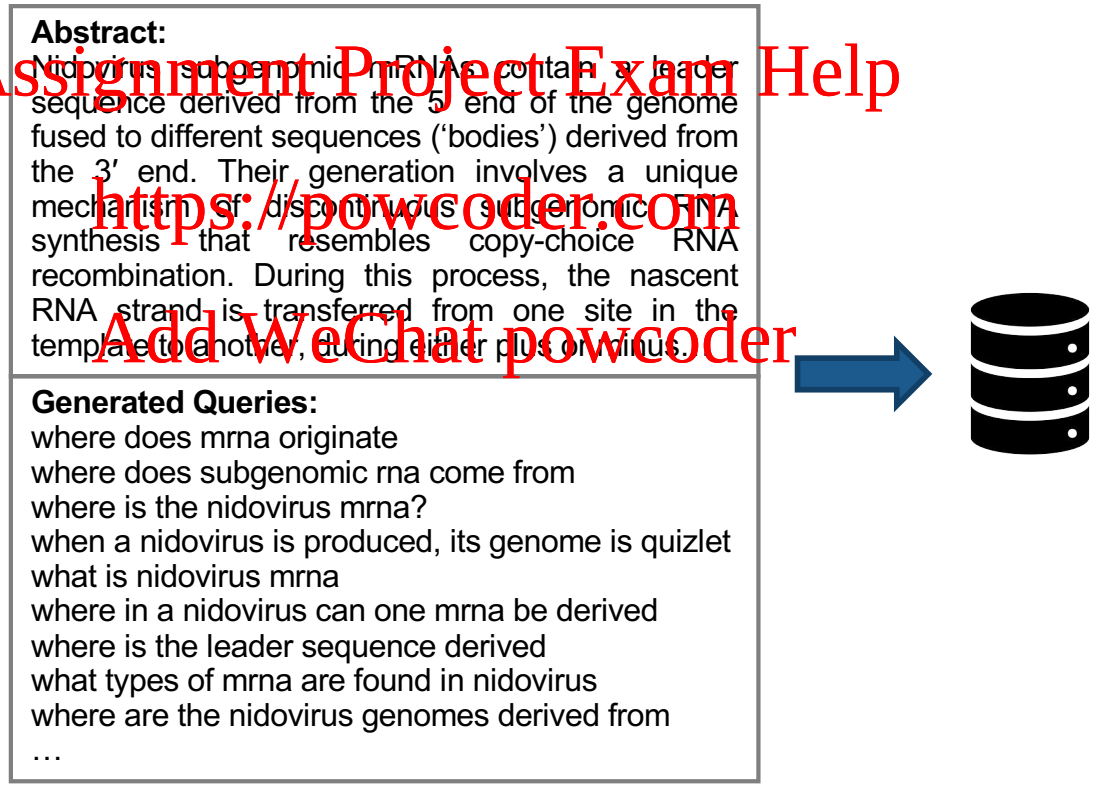
Training: Predict the terms that match relevant query terms.

Dai & Callan. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. arXiv 2019.

Doc2Query

Main idea: Use a causal language model to generate additional text to add to documents when indexing.

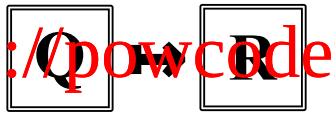
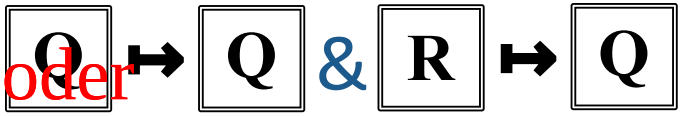
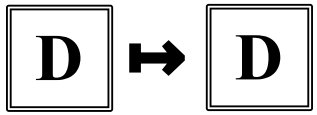
At retrieval time, use standard retrieval model (e.g., BM25).



Training: optimize to generate queries from passages on a collection with many queries (e.g., MS-MARCO)

Nogueira & Lin. From doc2query to docTTTTTquery. 2019.

Today

1. Review of LTR & Basics of Neural Networks for NLP
2. Neural Re-ranking 
3. Neural Retrieval 
4. Neural Query Rewriting & PRF 
5. Neural Document Rewriting 
6. **Neural IR in PyTerrier**

Neural IR In PyTerrier

Ranker: Which model to use?

Which doc text field to use?

```
drmm = onir_pt.reranker('drmm', 'wordvec_hash', text_field='abstract')
knrm = onir_pt.reranker('knrm', 'wordvec_hash', text_field='abstract')
pacrr = onir_pt.reranker('pacrr', 'wordvec_hash', text_field='abstract')
```

Vocab: Which word embeddings to use?


```
br = pt.BatchRetrieve(index) % 100
pipeline = br >> pt.text.get_text(dataset, 'abstract') >> reranker
```


These models need the document text. You can fetch it using **pt.text.get_text** *or* **BatchRetrieve**


```
pt.Experiment(
    [br, drmm_pipeline, knrm_pipeline, pacrr_pipeline],
    dataset.get_topics('title'),
    dataset.get_qrels(),
    names=['DPH', 'DPH >> DRMM', 'DPH >> KNRM', 'DPH >> PACRR'],
    eval_metrics=["recip_rank", "P.5", 'ndcg_cut.10', 'mrt']
)
```

	name	recip_rank	P_5	ndcg_cut_10	mrt
0	DPH	0.767259	0.684	0.584309	33.919908
1	DPH >> DRMM	0.515536	0.420	0.377125	88.168377
2	DPH >> KNRM	0.488852	0.340	0.329785	79.464181
3	DPH >> PACRR	0.623139	0.532	0.457561	89.766008

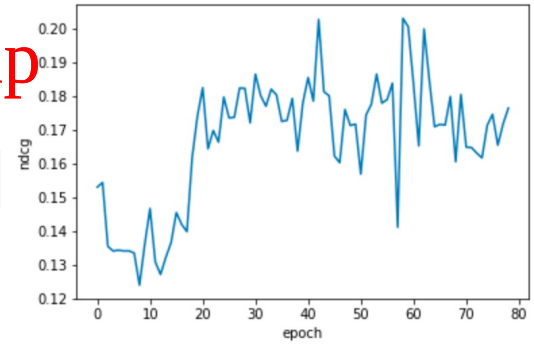
Neural IR In PyTerrier - Training

Ranker: Train model using medical subset of MS MARCO [1,2]

```
train_ds = pt.datasets.get_dataset('irds:msmarco-passage/train/medical')
```

```
fit_res = knrm.fit(
    train_ds.get_topics(),
    train_ds.get_qrels(),
    valid_ds.get_topics(),
    valid_ds.get_qrels())
```

```
plt.plot(fit_res['ndcg'])
```



	name	map	recip_rank	ndcg	ndcg_cut_10	mrt
0	DPH	0.075329	0.767259	0.164584	0.584309	30.752108
1	DPH >> KNRM	0.075105	0.810732	0.164964	0.619237	77.134939

[1] Bajaj et al. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset}. InCoCo@NeuIPS 2016.
[2] MacAvaney et al. SLEDGE-Z: A Zero-Shot Baseline for COVID-19 Literature Search. EMNLP 2020.

Neural IR In PyTerrier - Pretrained

Load trained model from URL

```
bert = onir_pt.reranker.from_checkpoint('https://macavaney.us/scibert-medmarco.tar.gz',
    text_field='title_abstract',
    expected_md5='854966d0b61543ffffa44cea627ab63b')
```

(optional) verification of download's hash

```
def cat_title_abstract(df):
    df['title_abstract'] = df['title'].str.cat(df['abstract'], sep=' ')
    return df

bert_pipeline = (br >>
                 pt.text.get_text(dataset, ['title', 'abstract']) >>
                 pt.apply.generic(cat_title_abstract) >>
                 bert)
```


Include both the title and abstract

```
pt.Experiment(
    [br, bert_pipeline],
    dataset.get_topics('title'),
    dataset.get_qrels(),
    names=['BM25', 'BM25 >> BERT'],
    eval_metrics=["map", "ndcg", 'ndcg_cut.10', 'mrt']
)
```

TREC COVID results

Title queries

	name	map	ndcg	ndcg_cut_10	mrt
0	BM25	0.073623	0.162657	0.583665	29.487896
1	BM25 >> BERT	0.077678	0.168394	0.650975	1637.205799

Description queries

	name	map	ndcg	ndcg_cut_10	mrt
0	BM25	0.077880	0.177728	0.644374	38.557969
1	BM25 >> BERT	0.085371	0.185821	0.740331	1748.576758

Neural IR In PyTerrier - EPIC

EPIC (indexed)

```
indexed_epic = onir_pt.indexed_epic.from_checkpoint(
    'https://macavaney.us/epic.msmarco.tar.gz',
    index_path='./epic_cord19')
```



```
indexed_epic.index(dataset.get_corpus_iter(), fields=('abstract',))
```



```
pipeline = br >> indexed_epic.reranker()
pt.Experiment(
    [br, pipeline],
    dataset.get_topics('title'),
    dataset.get_qrels(),
    names=["DPH", "DPH >> EPIC (indexed)"],
    eval_metrics=["recip_rank", "P.5", "mrt"]
)
```


	name	recip_rank	P_5	mrt
0	DPH	0.766833	0.684	30.500175
1	DPH >> EPIC (indexed)	0.821500	0.700	53.264584

Neural IR In PyTerrier – Indexing Pipelines

```
deepct = pyterrier_deepct.DeepCTTransformer(
    "bert-base-uncased/bert_config.json",
    "marco/model.ckpt-65816")
```


Example DeepCT output:

```
df.iloc[0]['text']
```

```
'OBJECTIVE: This retrospective chart review describes the epidemiology and clinical
features of 40 patients with culture-proven Mycoplasma pneumoniae infections at King
Abdulaziz University Hospital, Jeddah, Saudi Arabia. METHODS: Patients with positive
M. pneumoniae cultures from respiratory specimens from January 1997 through December
```

```
deepct.transform(df).iloc[0]["text"]
```

```
'objective objective retrospective retrospective chart chart chart chart chart revie
w describes epidemiology epidemiology epidemiology epidemiology epidemiology epidemi
ology epidemiology epidemiology epidemiology clinical clinical clinical clinical cli
nical clinical clinical clinical features features features features patients patien
```

Neural IR In PyTerrier – Indexing Pipelines

```
dataset = pt.get_dataset("irds:cord19/trec-covid")
index_loc = "./deepct_index_path"
indexer = deepct >> pt.IterDictIndexer(index_loc)
```

Build an *indexing* pipeline

```
indexref = indexer.index(dataset.get_corpus_iter())
```

Index the collection

	name	map	ndcg	ndcg_cut_10
0	BM25	0.181478	0.373328	0.583665
1	BM25_deepct	0.132069	0.304521	0.538936

Overview

We covered techniques for replacing various transformations in IR to use neural networks!

Transformer Class	Neural Examples
Q ↦ R Retrieval	ANCE, ColBERT
R ↦ Q PRF	ColBERT-PRF
R ↦ R Re-ranking	CEDR, monoT5
Q ↦ Q Query Re-writing	T5-QE, IntenT5
D ↦ D Doc. Re-writing	Doc2Query, DeepImpact

Closing Remarks / Future Directions

This was just a survey of recent advances.

1. **Are there inherent trade-offs among:**
   - Effectiveness
   - Efficiency
   - Interpretability
2. **Can we prevent unintended biases/side-effects?**
3. **NLP techniques have helped IR, can IR help NLP?**
   - In many models (particularly long-form seq2seq), models have the tendency to produce coherent text
   - But these often contain factual or common-sense errors
   - Can IR systems be incorporated into these models?
     - Example: K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. REALM: Retrieval-augmented language model pre-training. arXiv:2002.08909, 2020.

Bibliography

Z. Dai, C. Xiong, J. Callan, and Z. Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018), pages 126–134, 2018.

J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM 2016), pages 55–64, Indianapolis, Indiana, 2016.

K. Hui, A. Yates, K. Berberich, and G. de Melo. PACRR: A position-aware neural IR model fore levance matching. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1049–1058, Copenhagen, Denmark, 2017.

S. MacAvaney, A. Yates, A. Cohan, and N. Goharian. CEDR: Contextualized embeddings for document ranking. In Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019), pages 1101–1104, Paris, France, 2019a.

S. MacAvaney, A. Cohan, and N. Goharian. SLEDGE: A simple yet effective baseline for coronavirus scientific knowledge search.arXiv:2005.02365, 2020a.

S. MacAvaney, F. M. Nardini, R. Perego, N. Tonellotto, N. Goharian, and O. Frieder. Expansion via prediction of importance with contextualization. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020), pages1573–1576, 2020c.

S. MacAvaney, S. Feldman, N. Goharian, D. Downey, and A. Cohan. ABNIRML: Analyzing the Behavior of Neural IR Models. In TACL 2022.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. REALM: Retrieval-augmented language model pre-training.arXiv:2002.08909, 2020.

R. Nogueira and J. Lin. From doc2query to docTTTTTquery, 2019.

R. Nogueira and K. Cho. Passage re-ranking with BERT.arXiv:1901.04085, 2019.

C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017), pages 55–64, Tokyo, Japan, 2017.