IR From Bag-of-words to BERT and Beyond through Practical Experiments

An ECIR 2021 tutorial with PyTerrier and OpenNIR

Sean MacAvaney*
Craig Macdonald*
Nicola Tonellotto*

*(*Alphabetical ordering)*


Part 3: Contemporary Retrieval Architectures:
Neural re-rankers such as
BERT, T5, EPIC, ColBERT

From Features to Text

Università di Pisa | University of Glasgow

Learning to rank approaches require you to design effective features.

Neural methods adapted from NLP can learn these patterns from the text itself, reducing the need for manually designed features.

In this session, we will review:

- the basics of neural re-ranking methods.
- some of the current SOTA methods using BERT and T5 for ranking.
- approaches for reducing cost at query-time.

You will experience:

- re-ranking with models like BERT and T5 using PyTerrier and OpenNIR.
- performing re-ranking thresholding tuning.
- re-ranking with pre-computed representations to reduce query latency
- visualizing the internal representation of a ranking model

Intended Learning Outcomes

Part 3 – *Contemporary* Retrieval Architectures: Neural re-rankers such as BERT, EPIC, ColBERT

ILO 3A. Understand contemporary retrieval architectures, such as BERT re-rankers.

ILO 3B. perform BERT and T5 re-ranking experiments within a Python notebook.

Outline of Part 3

Part 3A: Background

Part 3B: Contextualized Language Models (e.g., BERT)

Part 3C: Managing Efficiency

Part 3D: Wrap up and move to practical session

University of Glasgow

UNIVERSITÀ DI PISA

Part 3A

NEURAL RE-RANKING BACKGROUND

From Features to Text

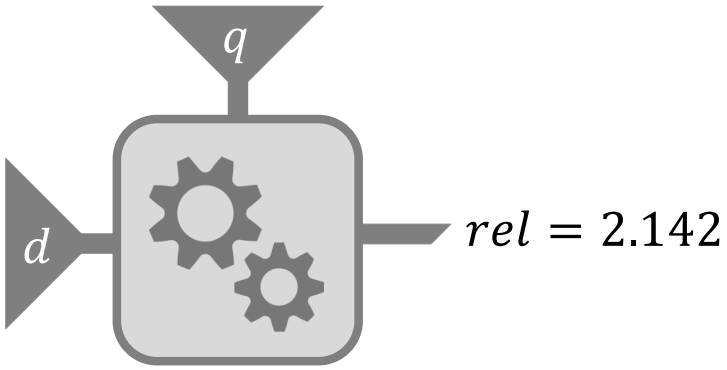
Università di Pisa | University of Glasgow

Coronavirus Early Symptoms

Document:

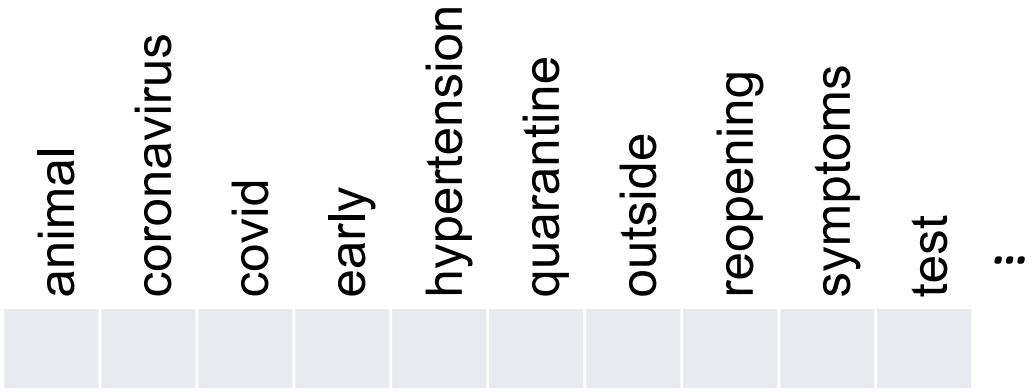
Title: How can we evaluate an interrelation of symptoms?

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…



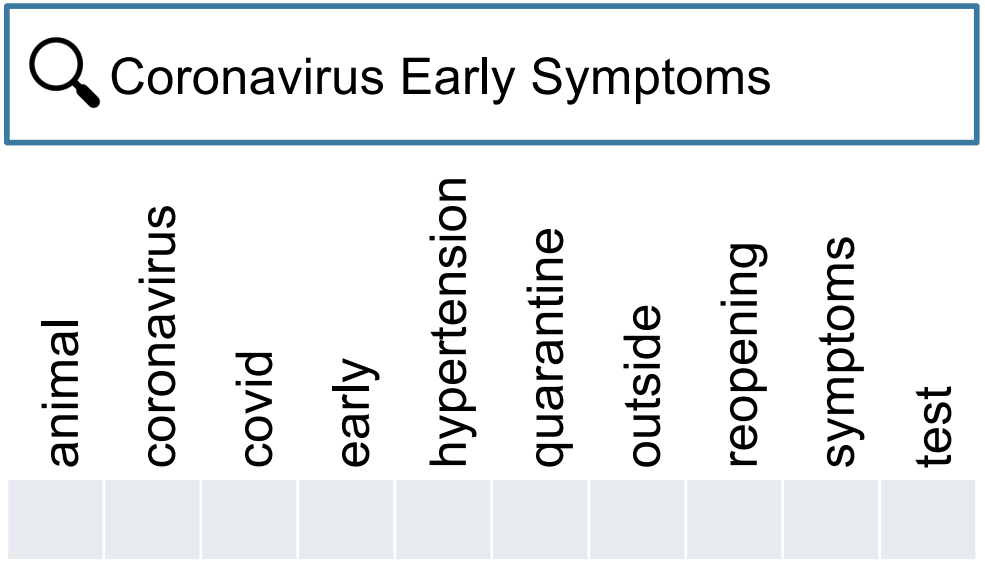
Representing Text

Option 1: One-hot encoding



Representing Text

Option 1: One-hot encoding



Representing Text

Option 1: One-hot encoding

Coronavirus Early Symptoms

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

Representing Text

Option 1: One-hot encoding

Coronavirus Early Symptoms

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

Representing Text

Problem: no relationship between words

Coronavirus Early Symptoms

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

COVID Early Symptoms

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

Completely different

Representing Text

Università di Pisa | University of Glasgow

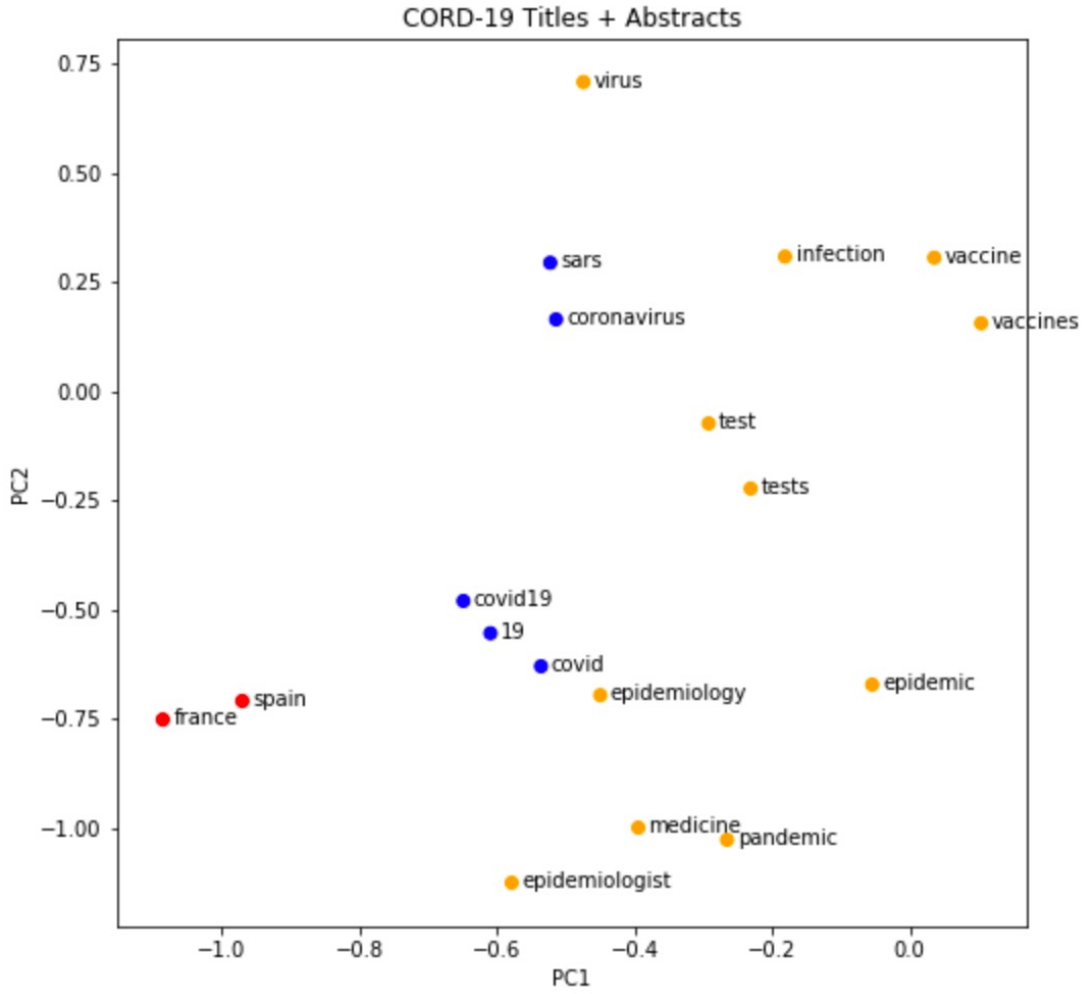
Word Vectors: Map each word to a dense vector

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,
...]
```

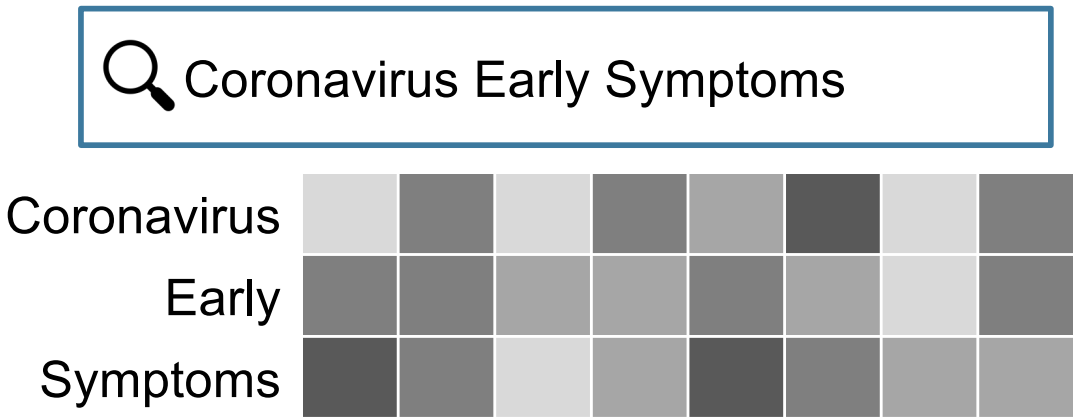
coronavirus =

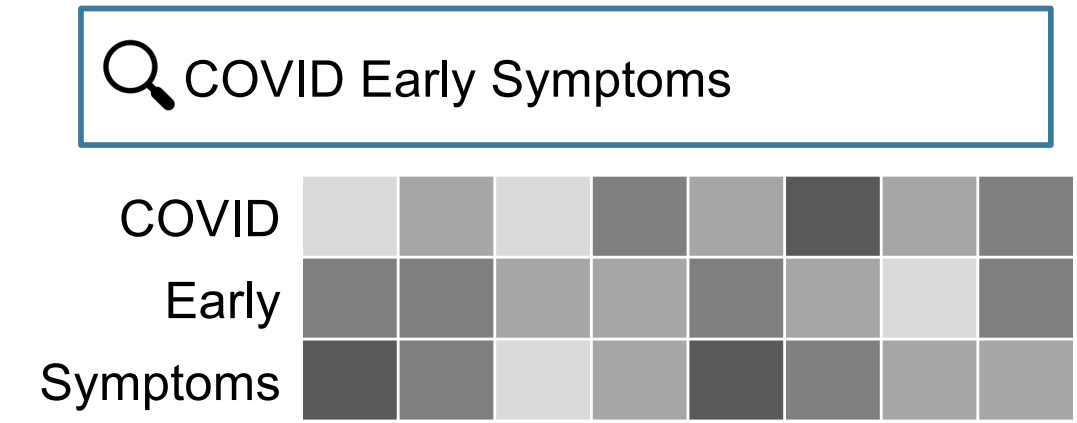
```
[0.33853,
-0.27798,
 0.08317,
-0.19729,
-0.49235,
 0.26514,
 0.03004,
 0.25704,
-0.38031,
-0.32722,
-0.47273,
-0.01596,
 0.32322,
-0.04947,
...]
```



Representing Text

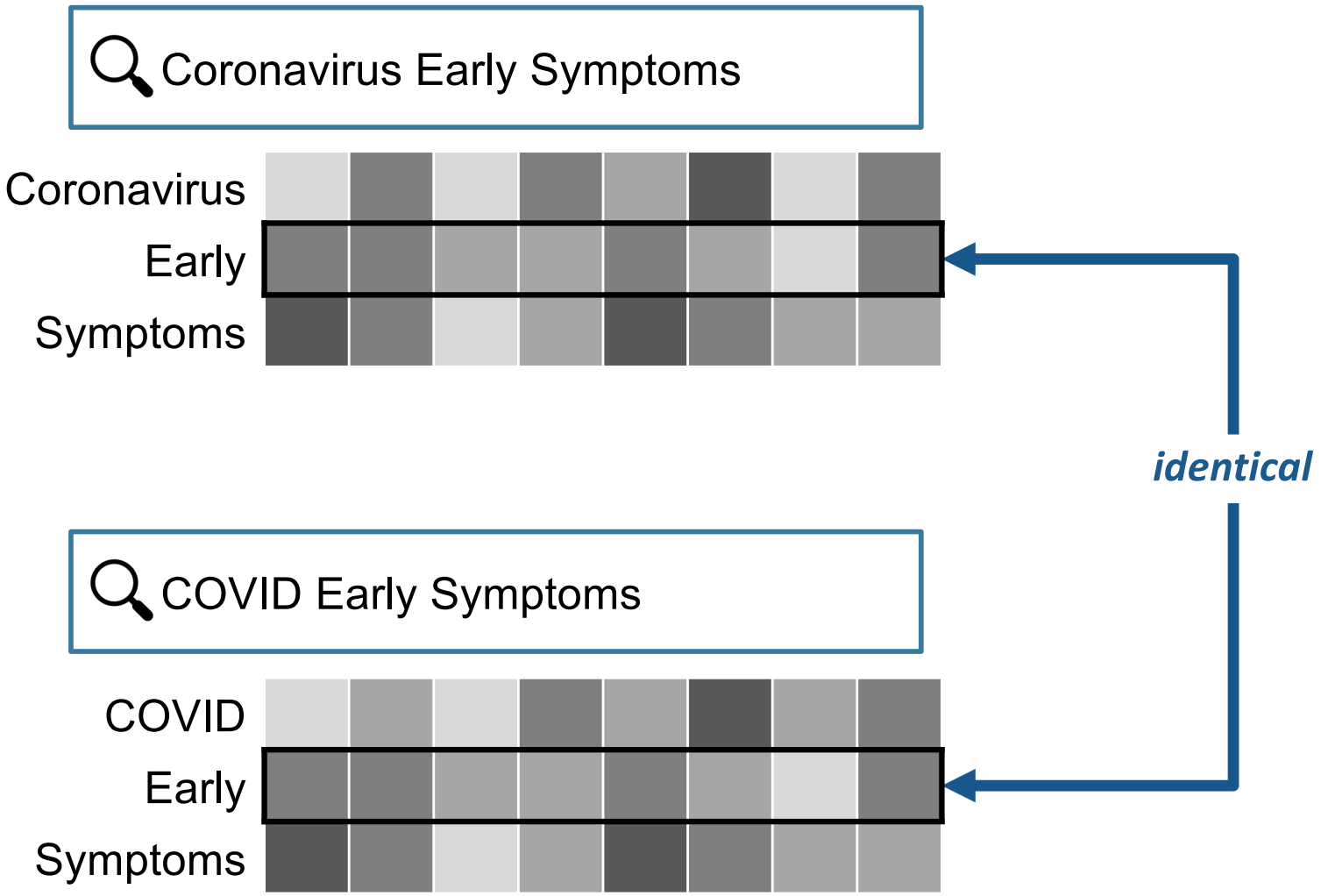
Word Vectors: Map each word to a dense vector





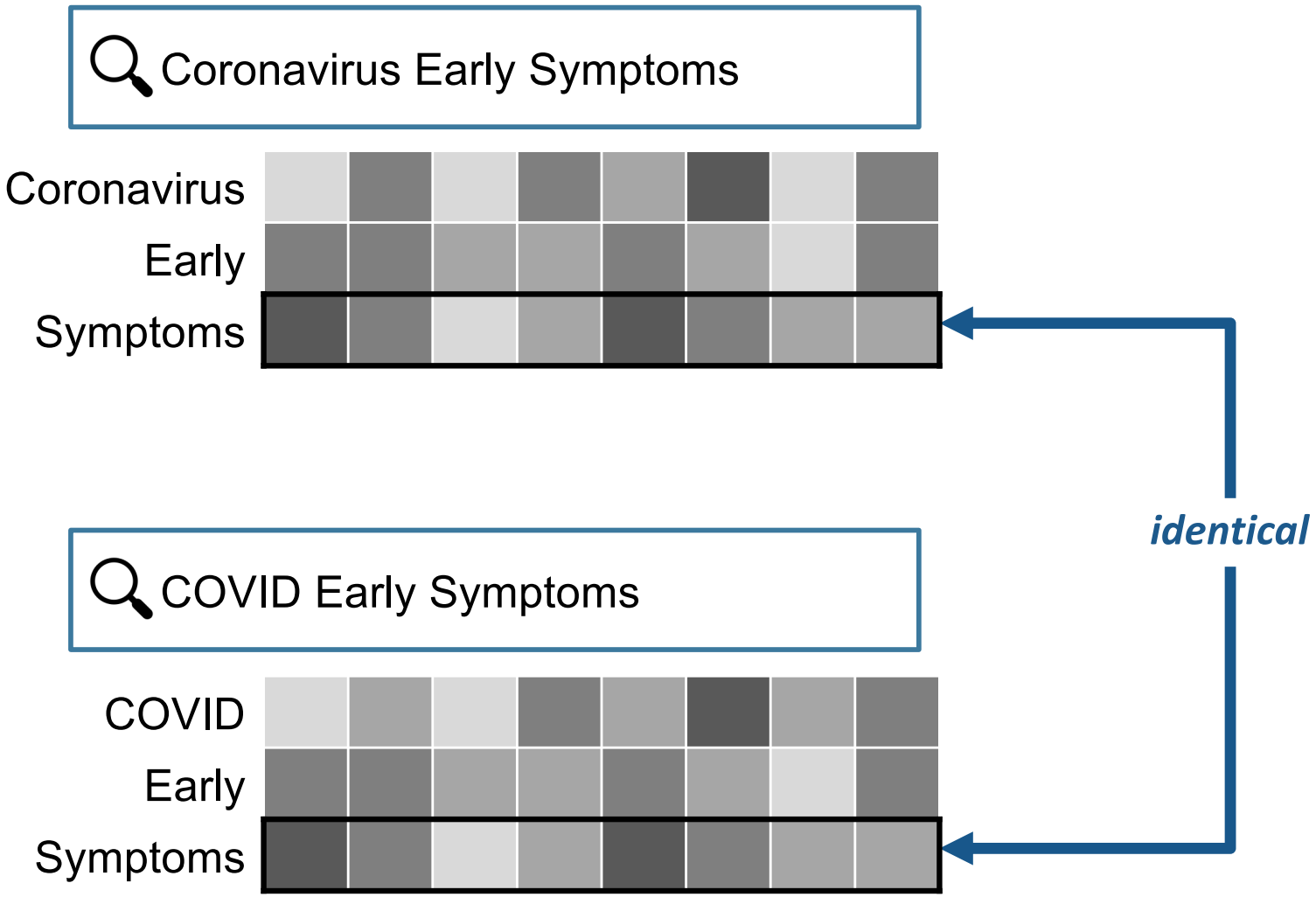
Representing Text

Word Vectors: Map each word to a dense vector



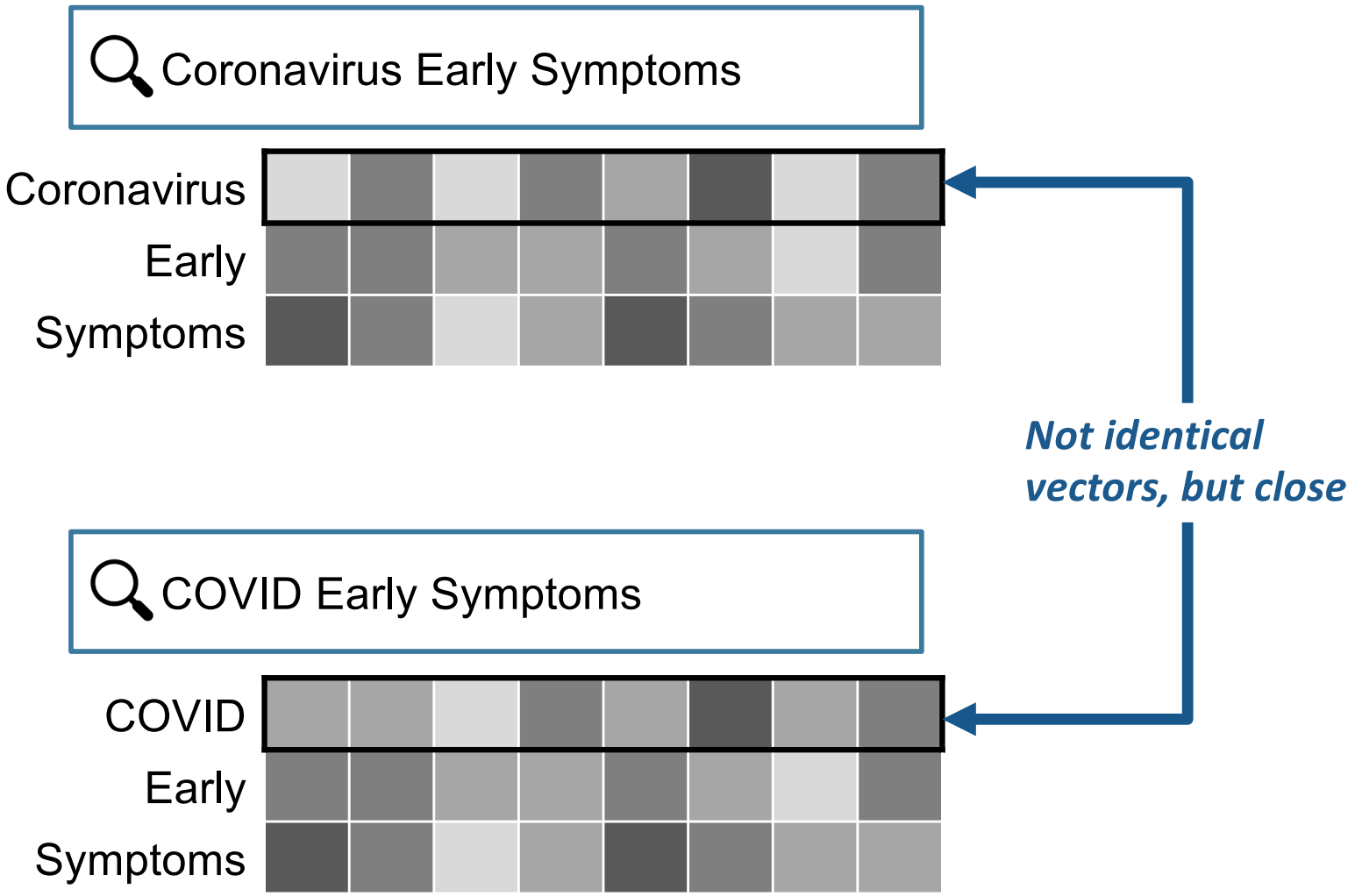
Representing Text

Word Vectors: Map each word to a dense vector



Representing Text

Word Vectors: Map each word to a dense vector



Representing Text

Building word vectors

- Based on word co-occurrences
- Trained using neural network
- e.g. word2vec, gloVe, etc.

Recent: “contextualized” word vectors – different based on the context that they appear in the text

- More on this later…

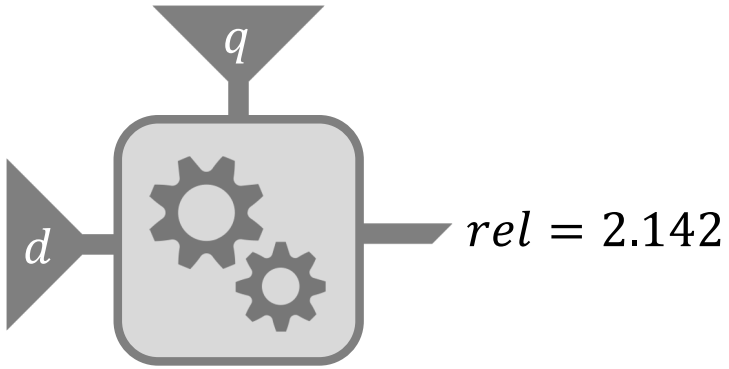
From Text to Features

Coronavirus Early Symptoms

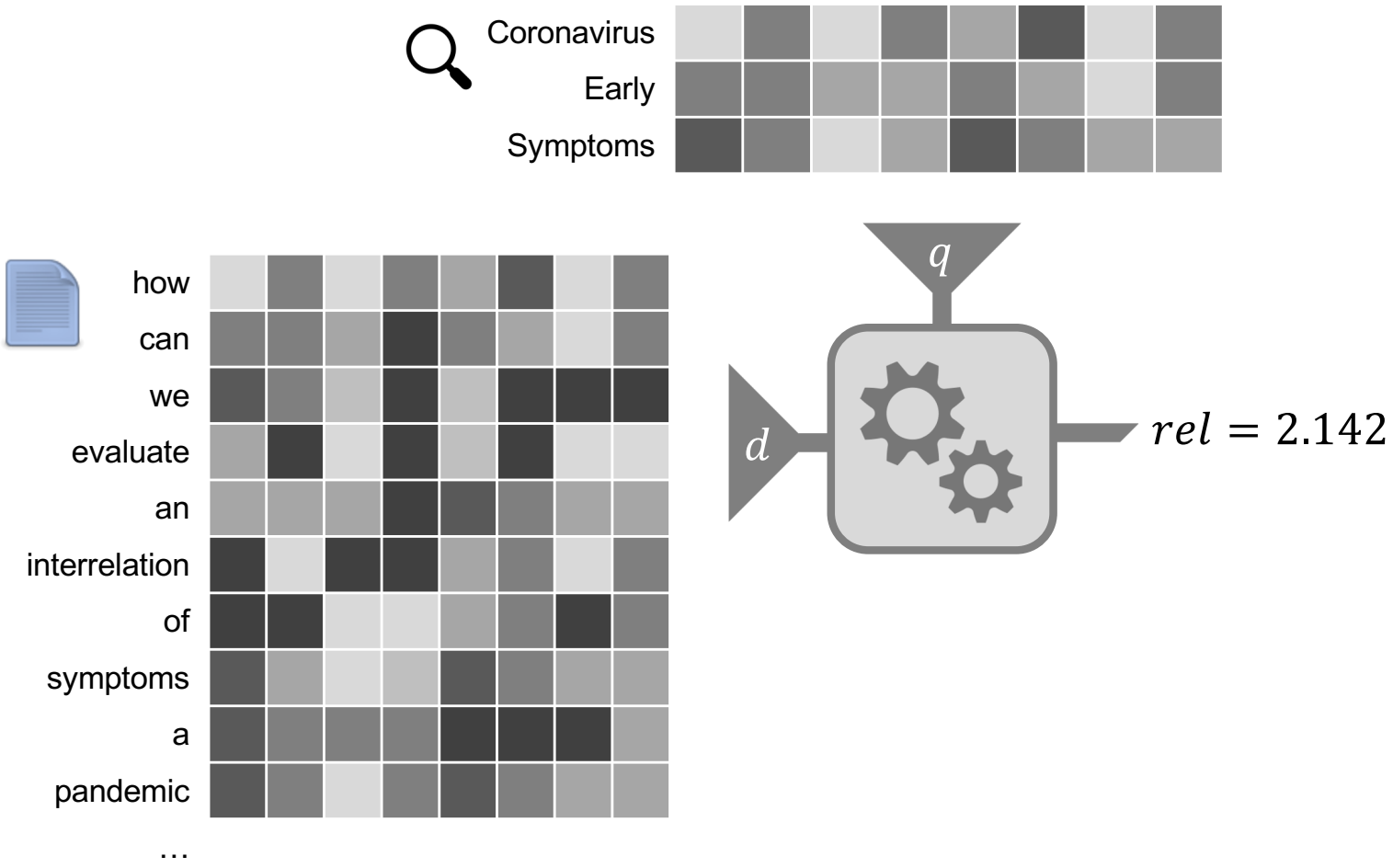
Document:

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…

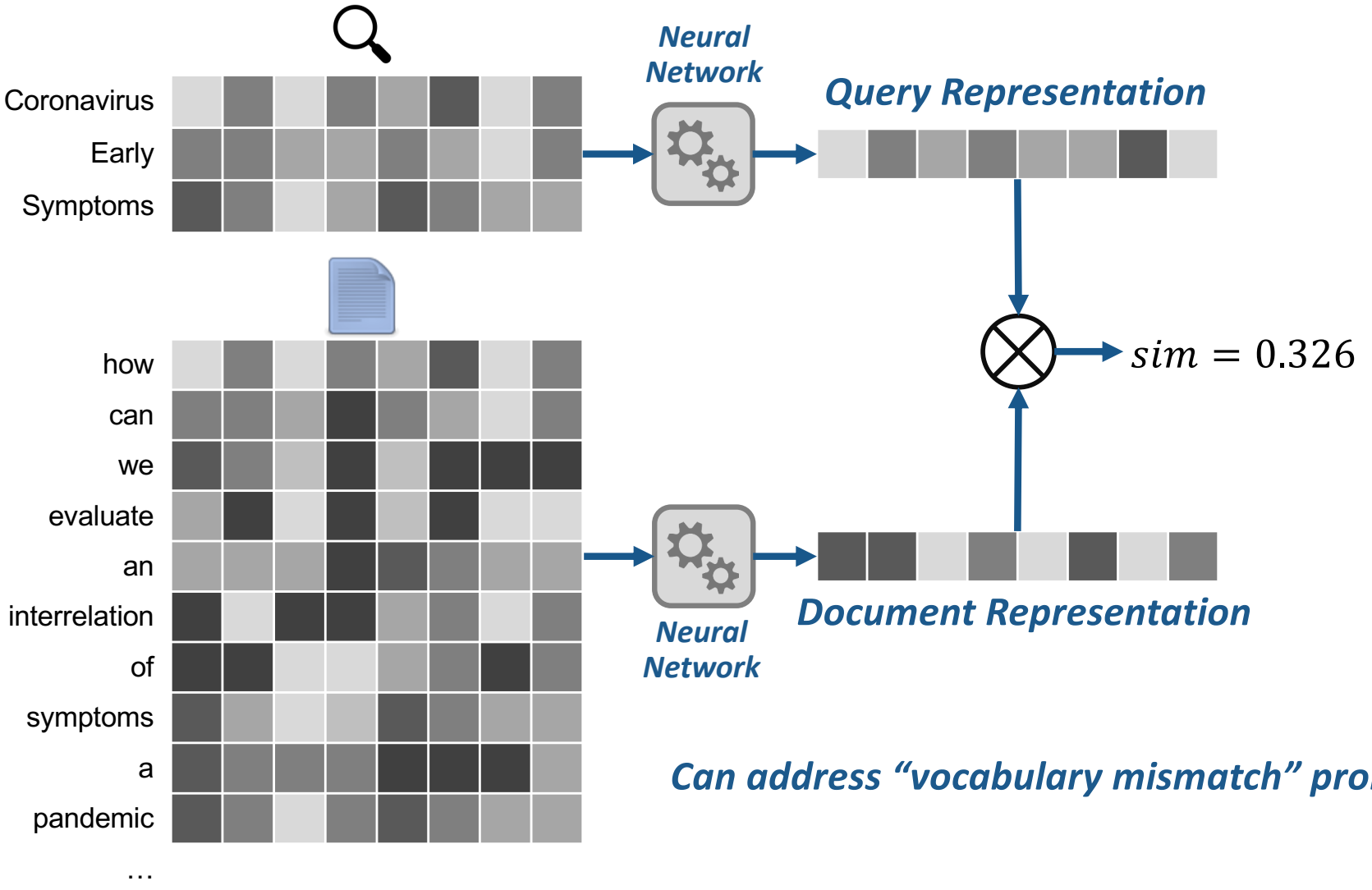


From Text to Features



From Text Features to Scores

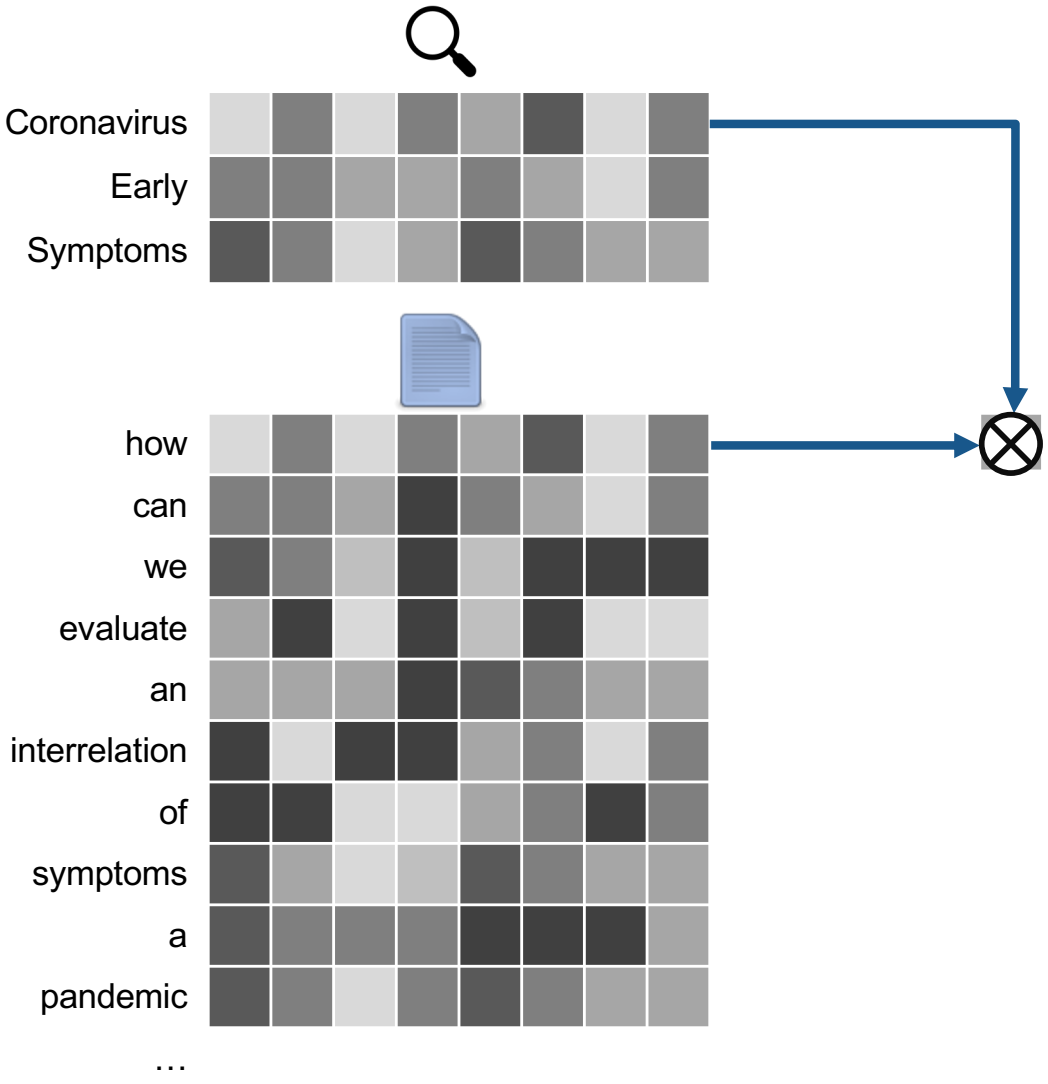
Two main approaches: **Representation** and Interaction



Can address "vocabulary mismatch" problem.

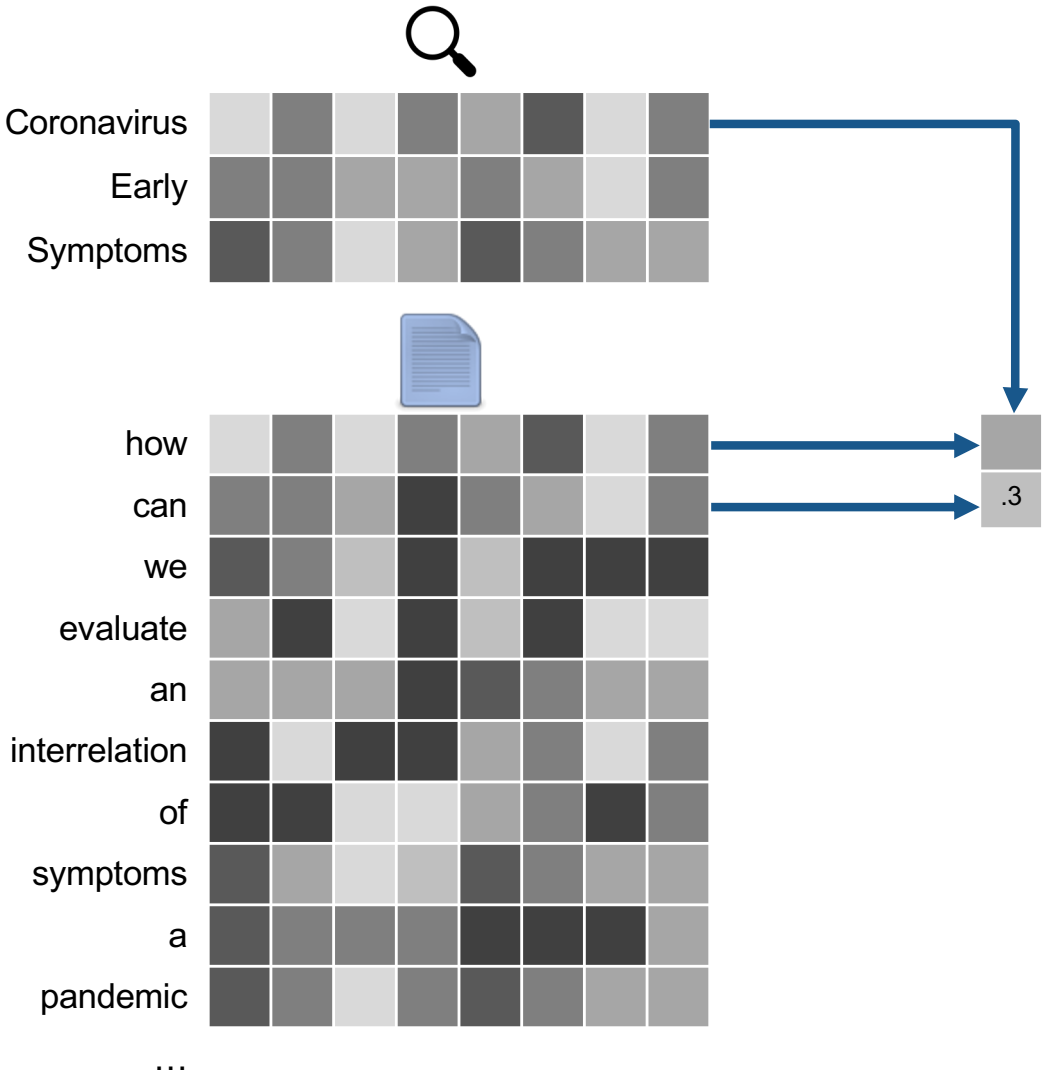
From Text Features to Scores

Two main approaches: Representation and **Interaction**



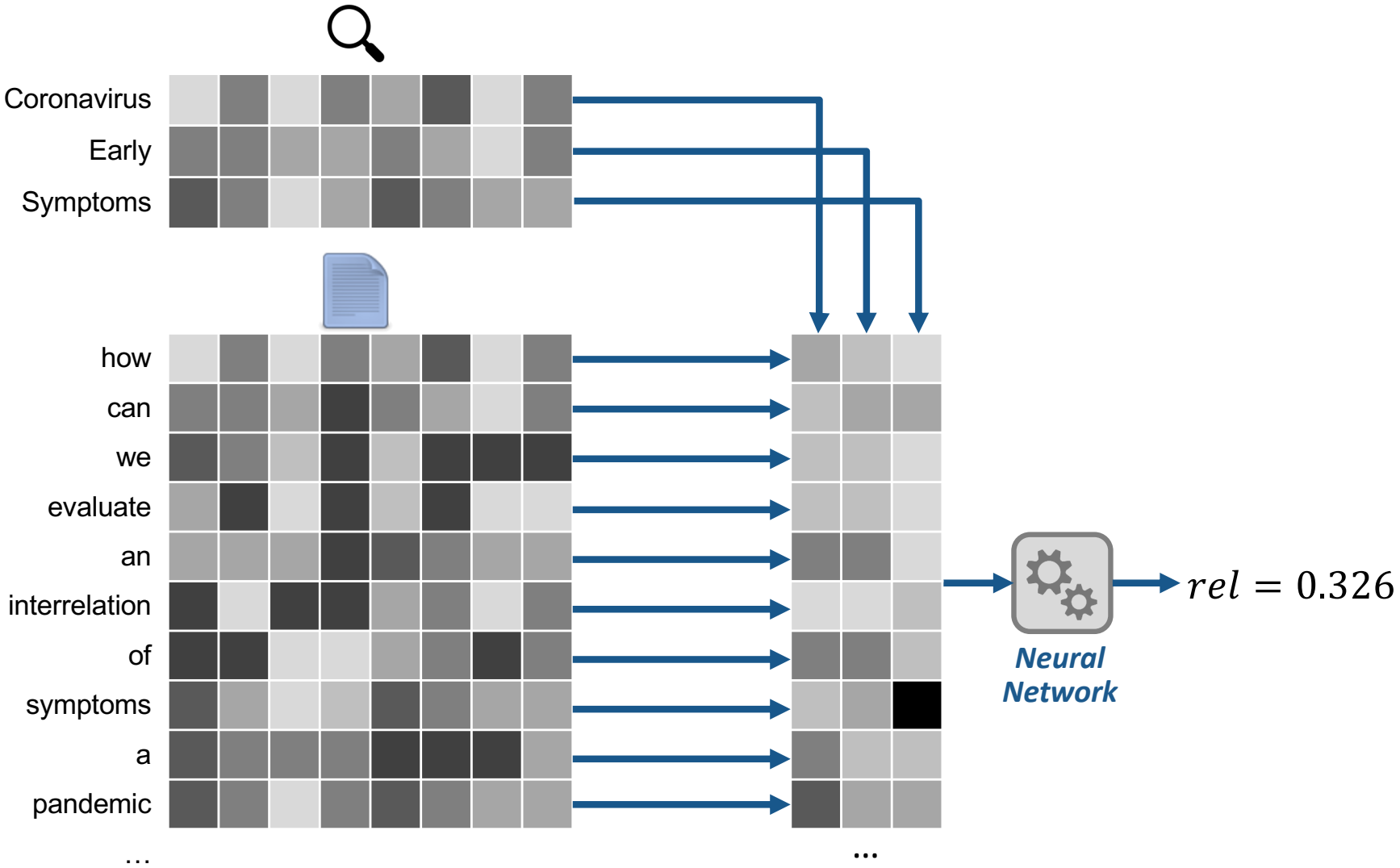
From Text Features to Scores

Two main approaches: Representation and **Interaction**



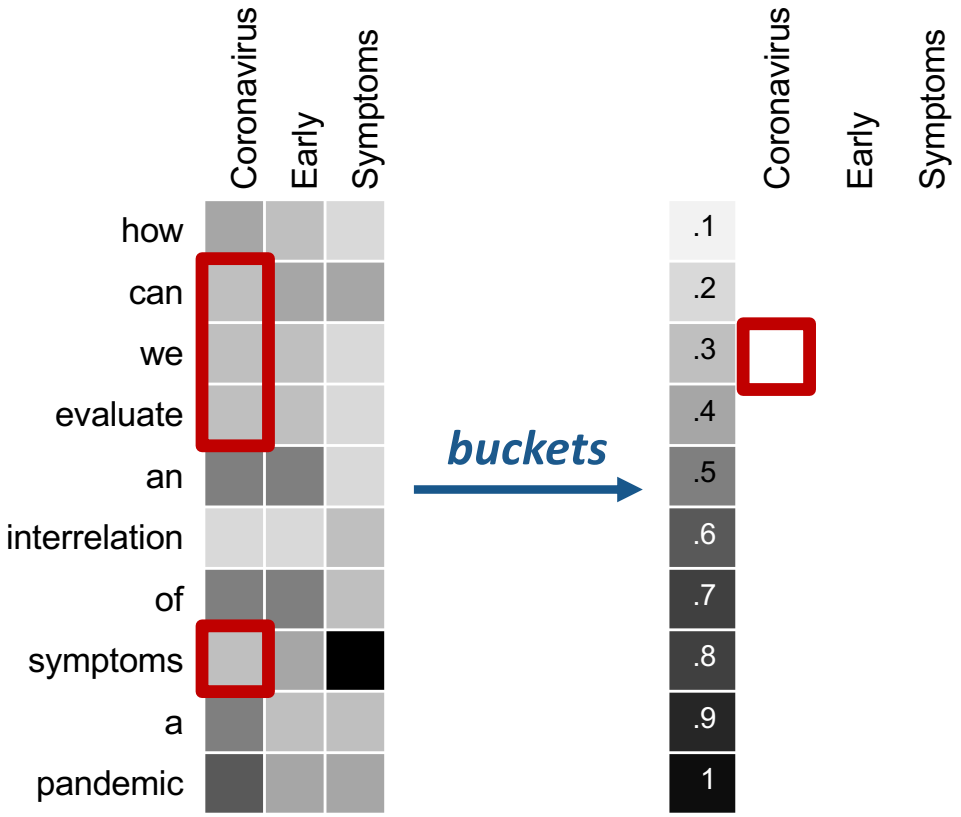
From Text Features to Scores

Two main approaches: Representation and **Interaction**



Interaction Aggregation

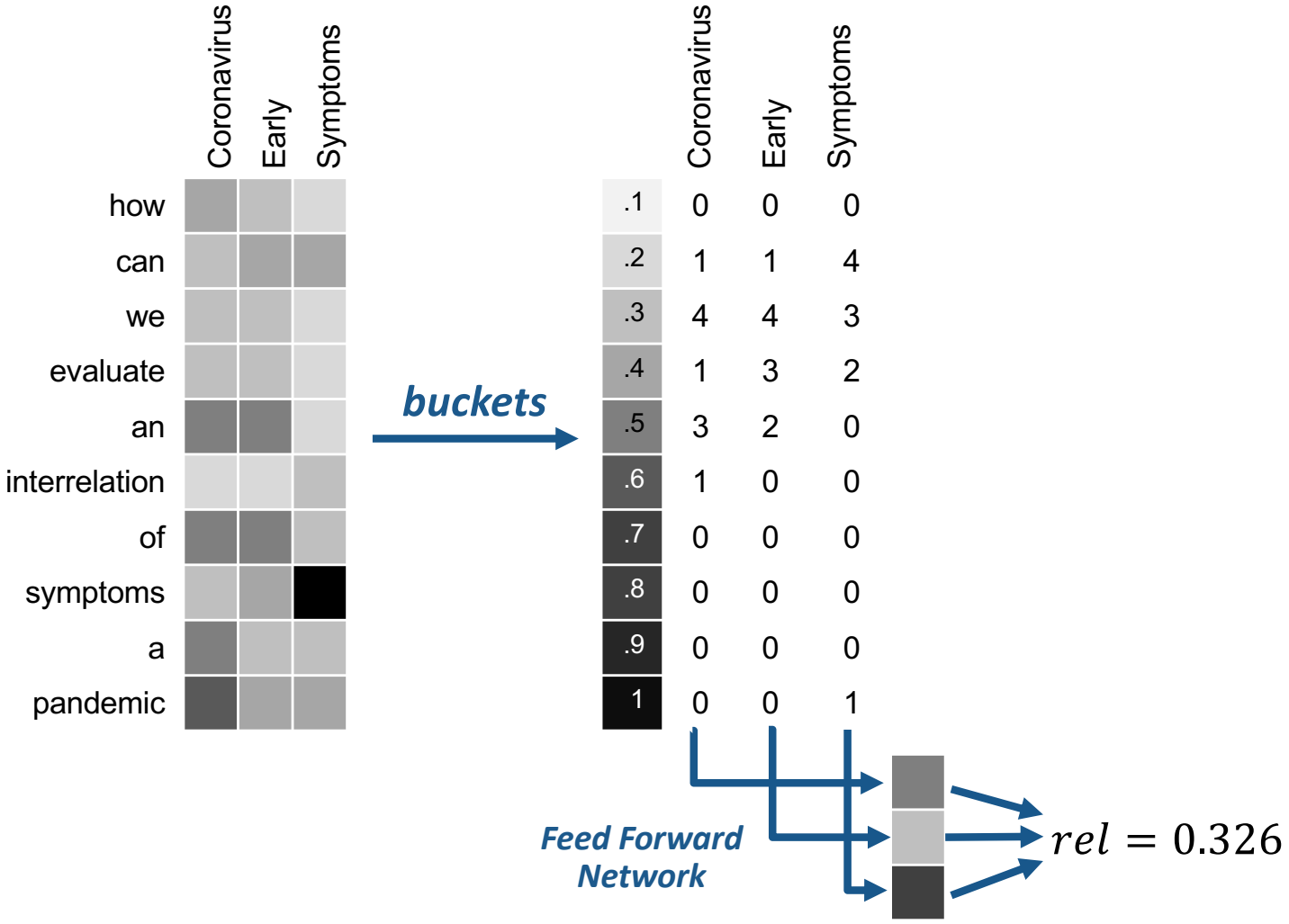
Bucketing - DRMM (Deep Relevance Matching Model)



Guo et al. A Deep Relevance Matching Model for Ad-hoc Retrieval. CIKM 2017.

Interaction Aggregation

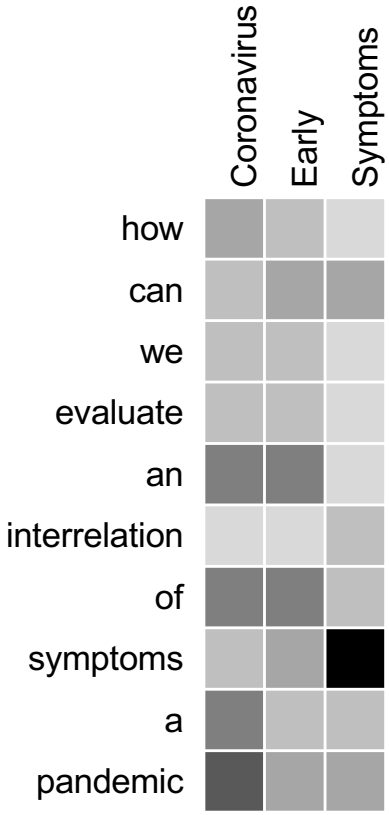
Bucketing - DRMM (Deep Relevance Matching Model)



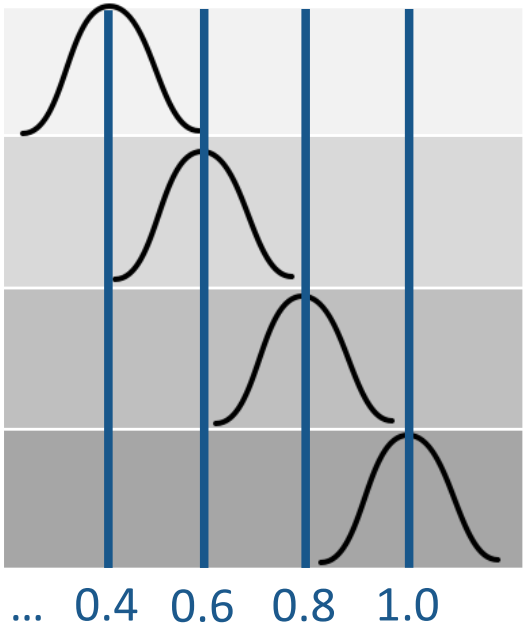
Guo et al. A Deep Relevance Matching Model for Ad-hoc Retrieval. CIKM 2017.

Interaction Aggregation

Soft Bucketing - KNRM (Kernel-based Neural Ranking Model)



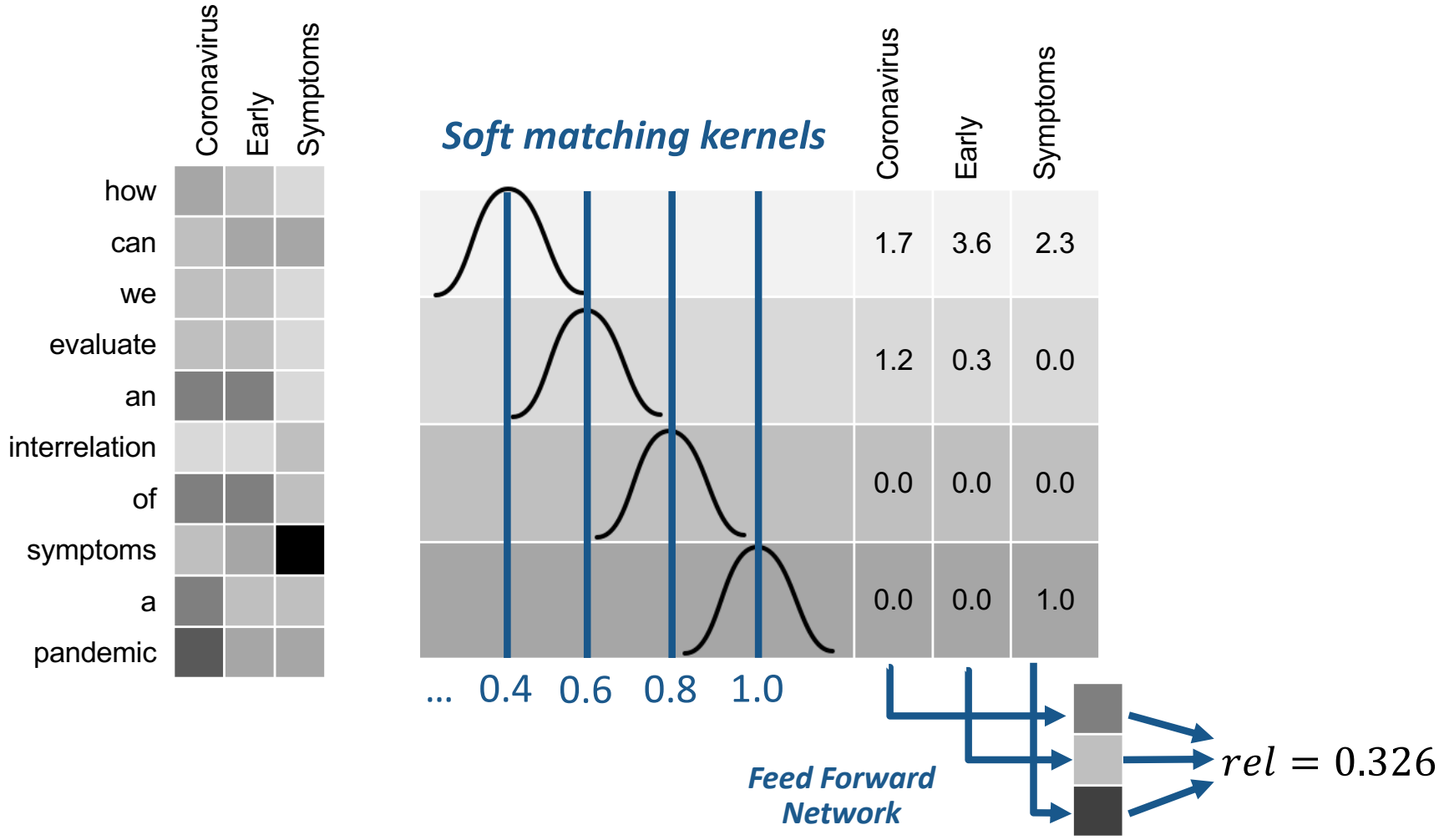
Soft matching kernels



Xiong et al. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. SIGIR 2017.

Interaction Aggregation

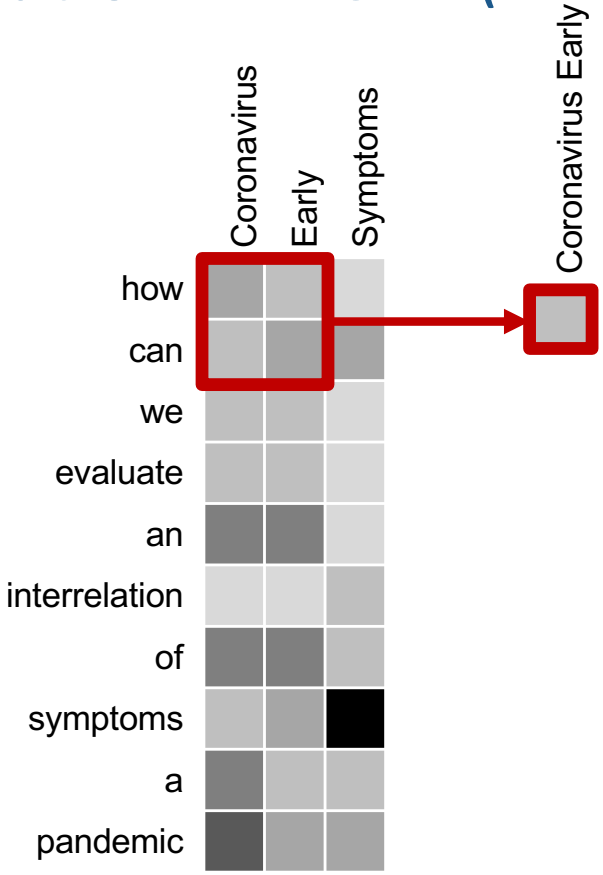
Soft Bucketing - KNRM (Kernel-based Neural Ranking Model)



Xiong et al. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. SIGIR 2017.

Interaction Aggregation

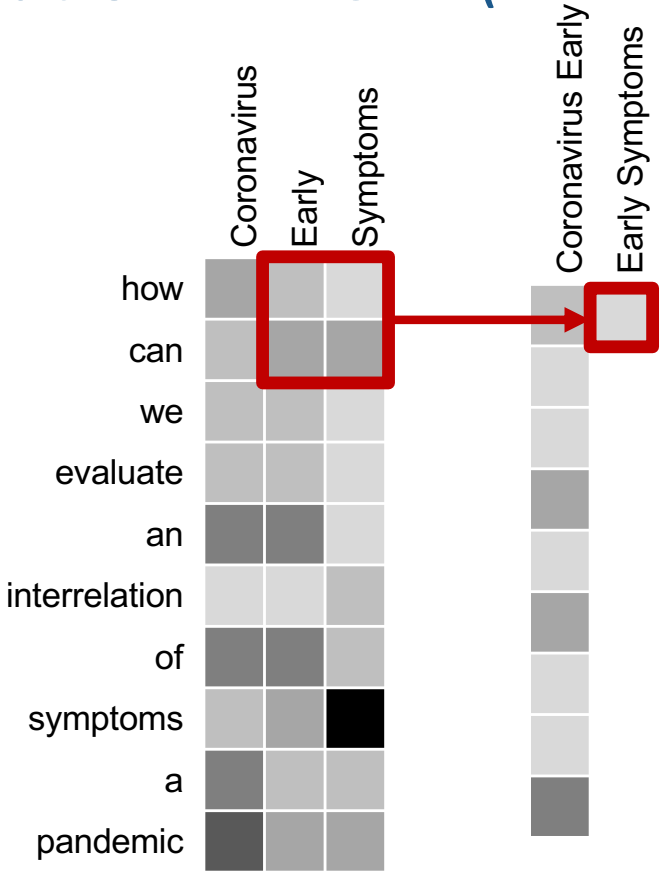
Convolution - PACRR (Position-Aware Convolutional-Recurrent Relevance)



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

Interaction Aggregation

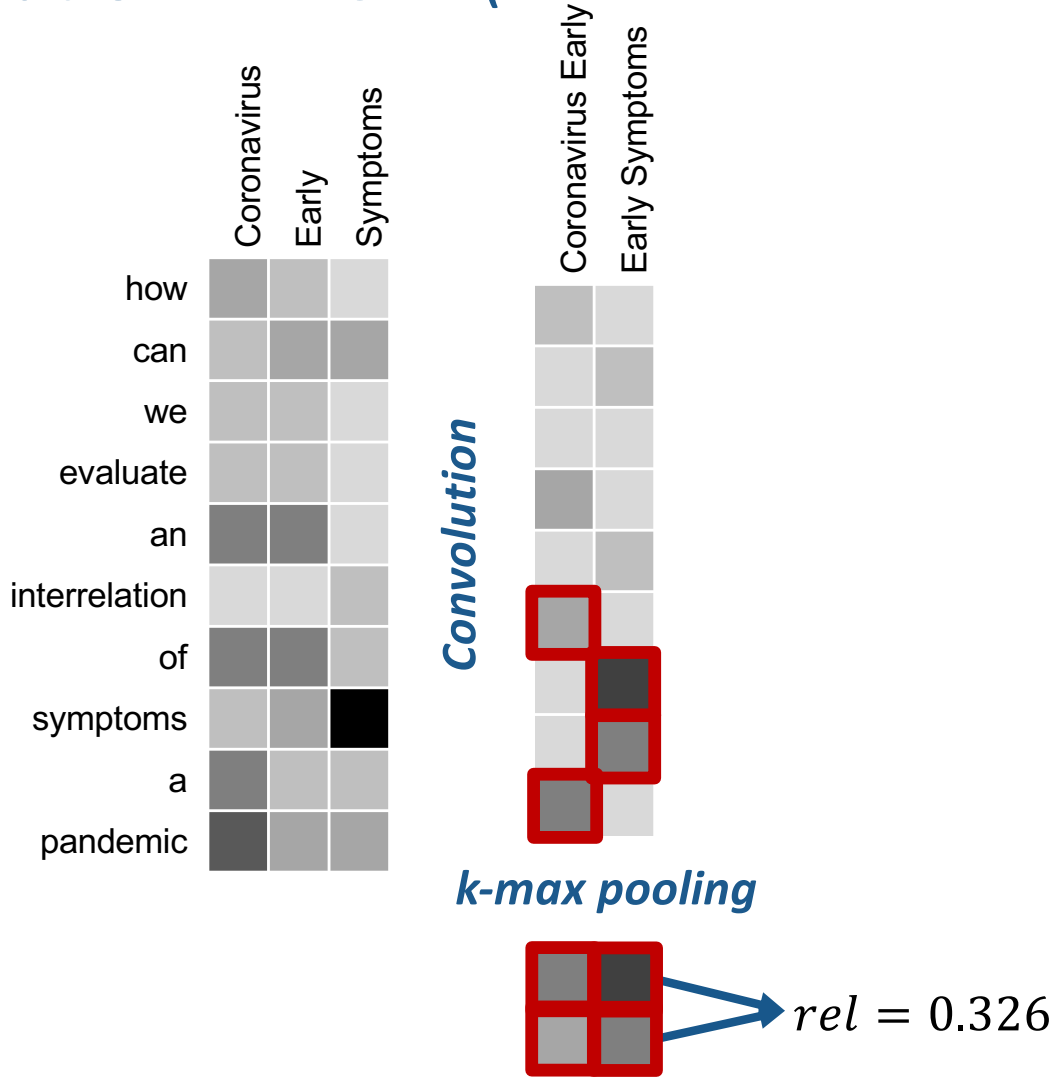
Convolution - PACRR (Position-Aware Convolutional-Recurrent Relevance)



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

Interaction Aggregation

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



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

Interaction Aggregation

Interaction Models

- Soft term matching: DRMM, KNRM, PACRR
- Bucketing: DRMM, KNRM
- Proximity Matching: PACRR

nDCG@20

Model	WT 2012	WT 2013	WT 2014
DRMM	0.197	0.228	0.300
KNRM	0.222	0.251	0.324
PACRR	**0.243**	**0.295**	**0.339**

Results reported by Hui, et al. 2017. (PACRR)

OpenNIR

A toolkit for training and evaluating neural IR models

Historically: Script-based

Today: Can use with PyTerrier!

https://OpenNIR.net/

MacAvaney. OpenNIR: A Complete Neural Ad-Hoc Ranking Pipeline. WSDM 2020.

Interaction Aggregation - Practical

Università di Pisa | University of Glasgow

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`

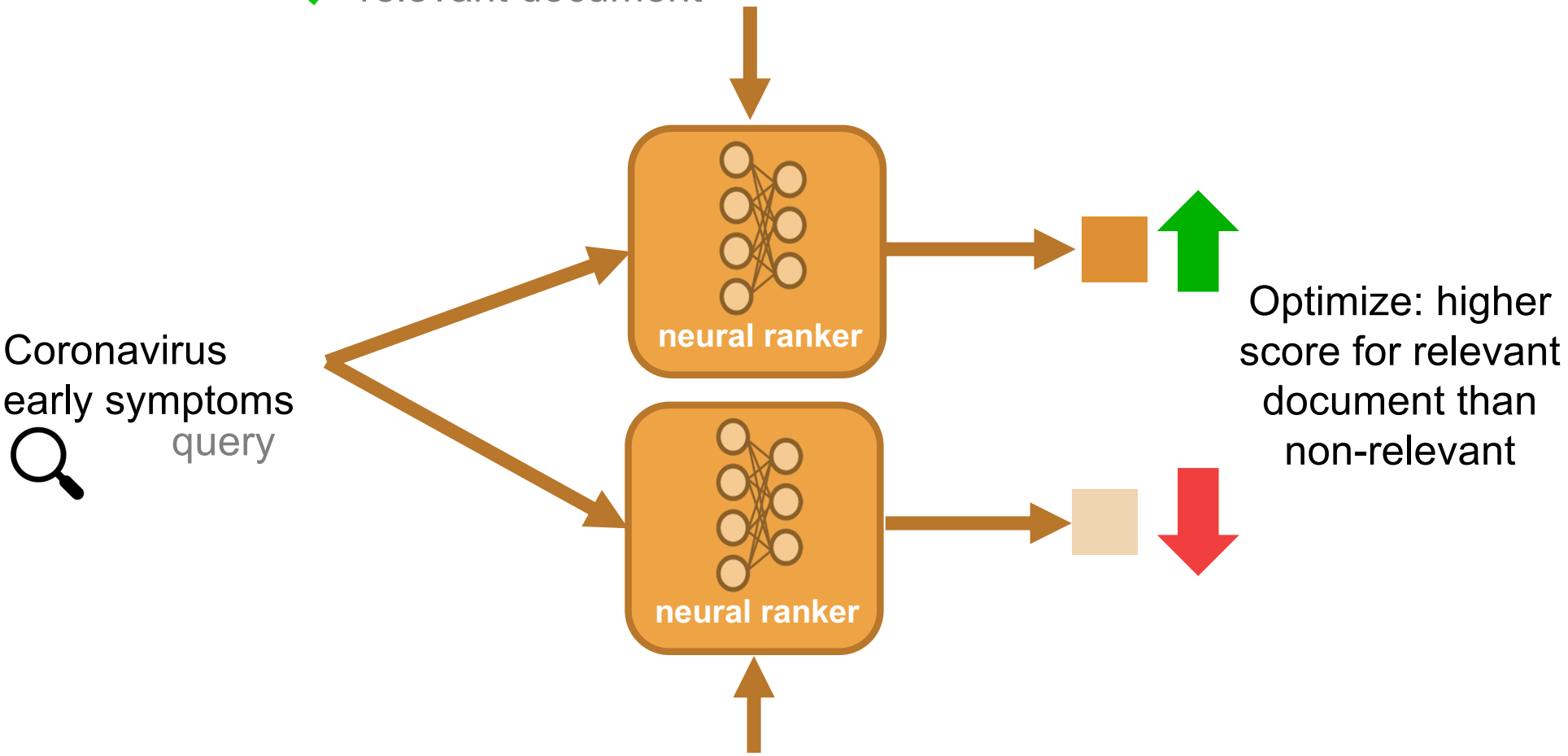
```
pt.Experiment(
    [br, drmm_pipeline, knrm_pipeline, pacrr_pipeline],
    dataset.get_topics('title'),
    dataset.get_qrels(),
    names=['DPH', 'DPH >> DRMM', 'DPH >> KNRM', 'DPH >> PACRR'],
    eval_metrics=["recip_rank", "P.5", 'ndcg_cut.10', 'mrt']
)
```

	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

Pairwise Training

The importance of olfactory and gustatory disorders as early symptoms of coronavirus disease

relevant document



Assessing Viral Shedding and Infectivity of Tears in Coronavirus Disease 2019 (COVID-19) Patients

non-relevant document

Pairwise Training

Training data sources:

- Annotated collections
- Behavioral data
- Weak supervision / relevance transfer

Graphic from trec.nist.gov

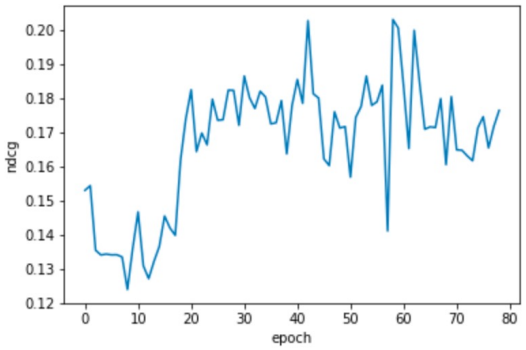
Pairwise Training

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.

University of Glasgow

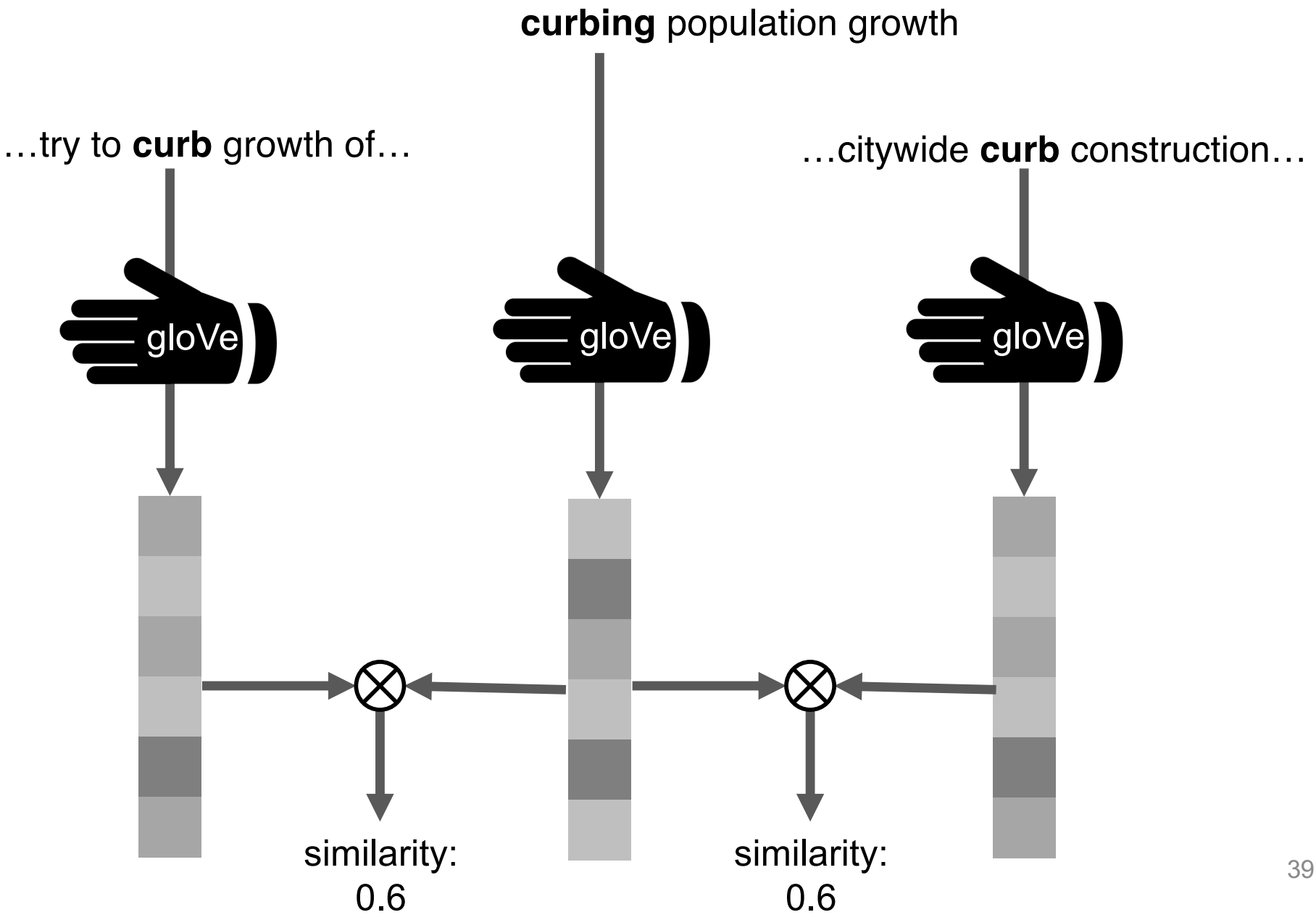
UNIVERSITÀ DI PISA

ECIR 2021

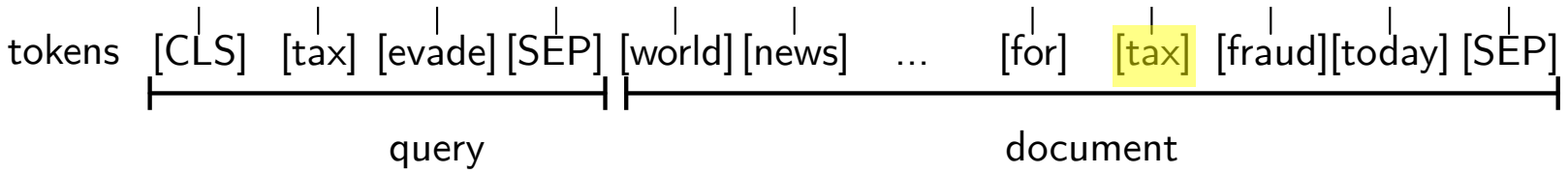
Part 3B

CONTEXTUALIZED LANGUAGE MODELS FOR RE-RANKING

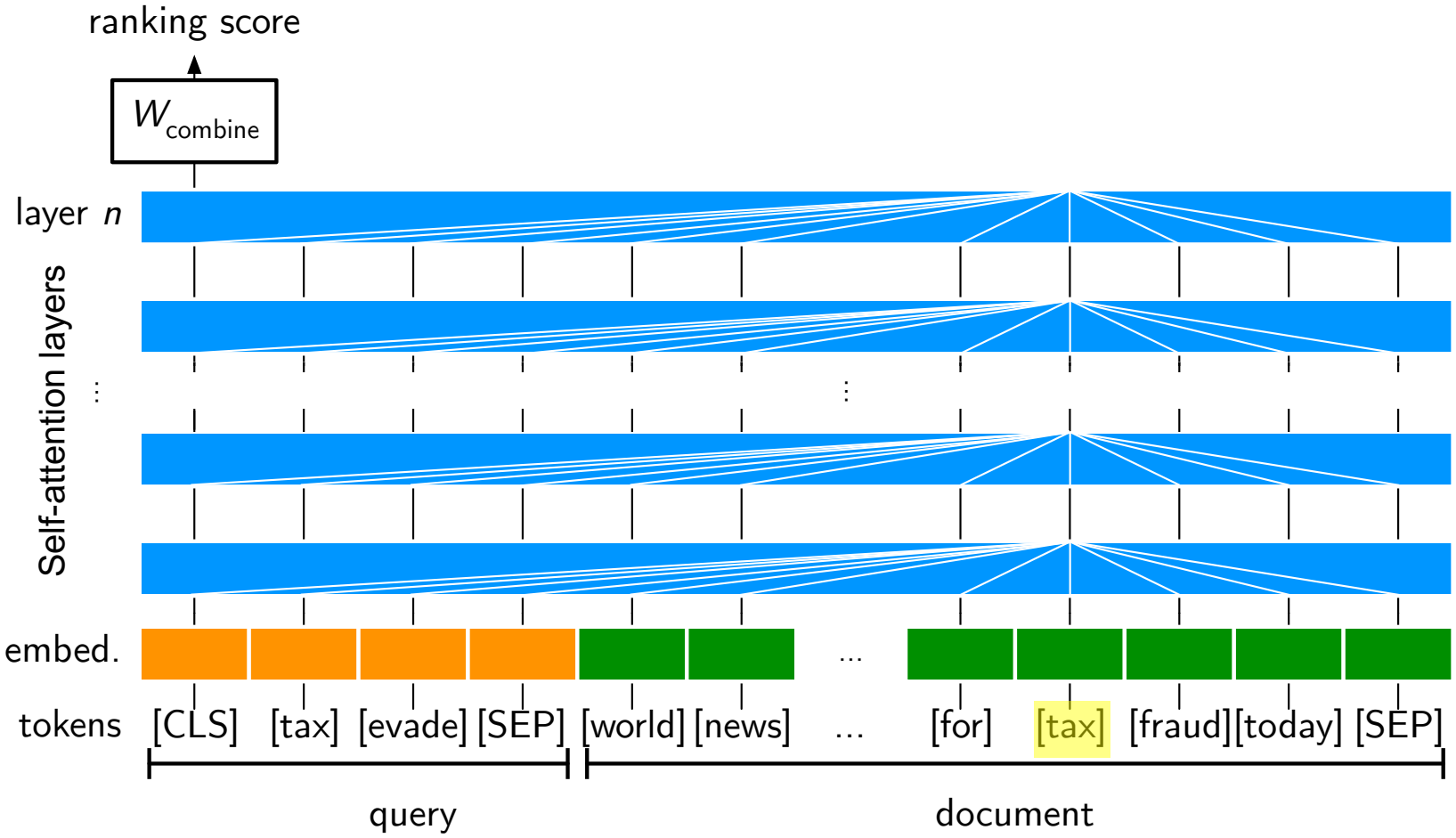
Static Word Vectors



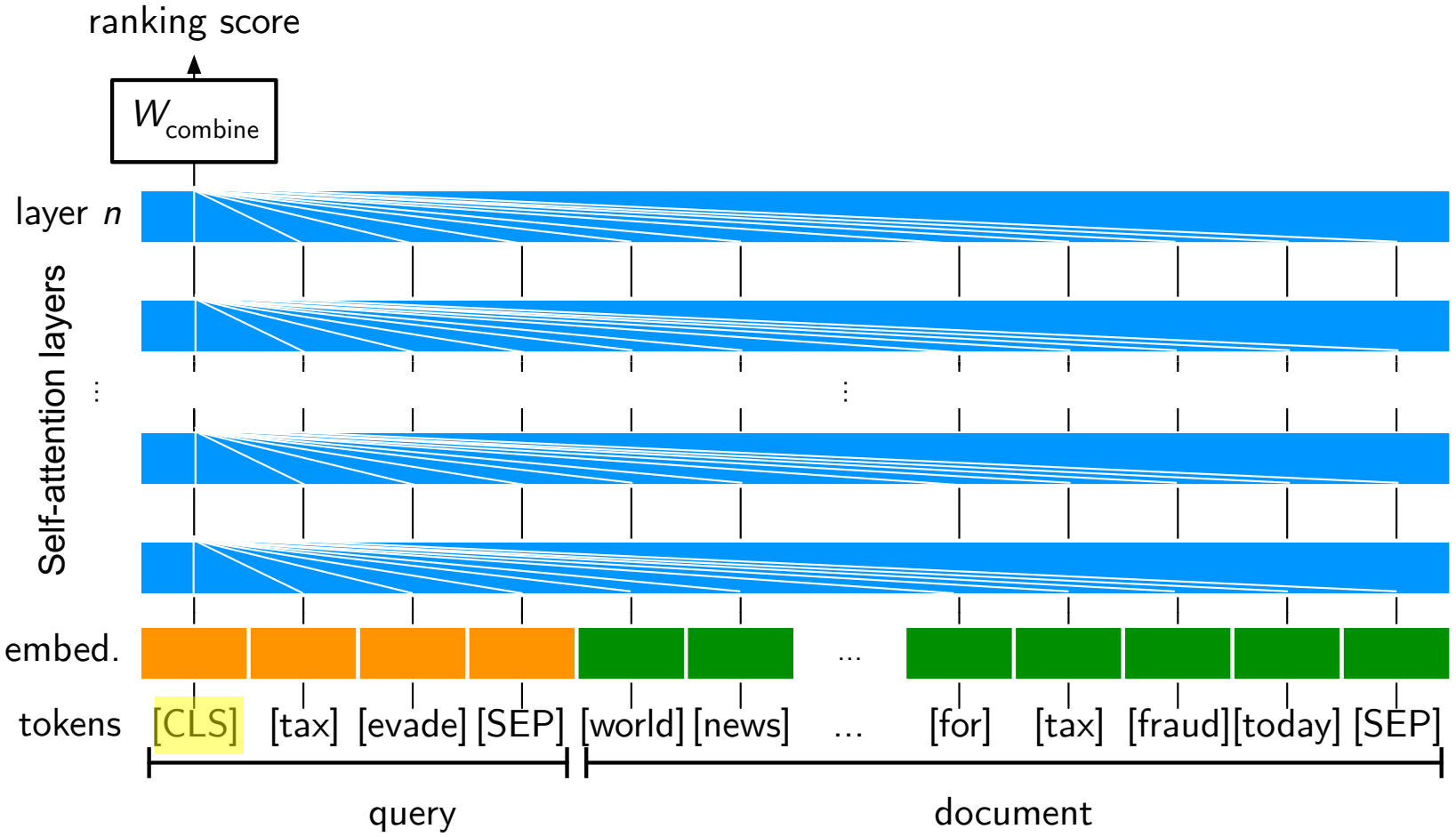
BERT & Self-Attention Networks



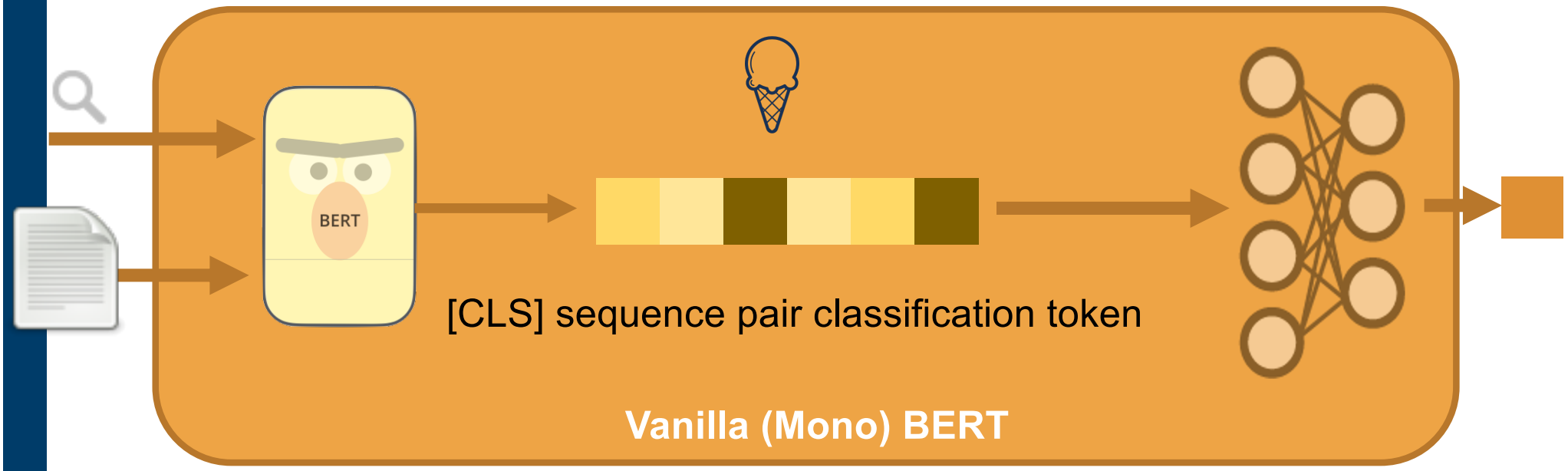
BERT & Self-Attention Networks



BERT & Self-Attention Networks

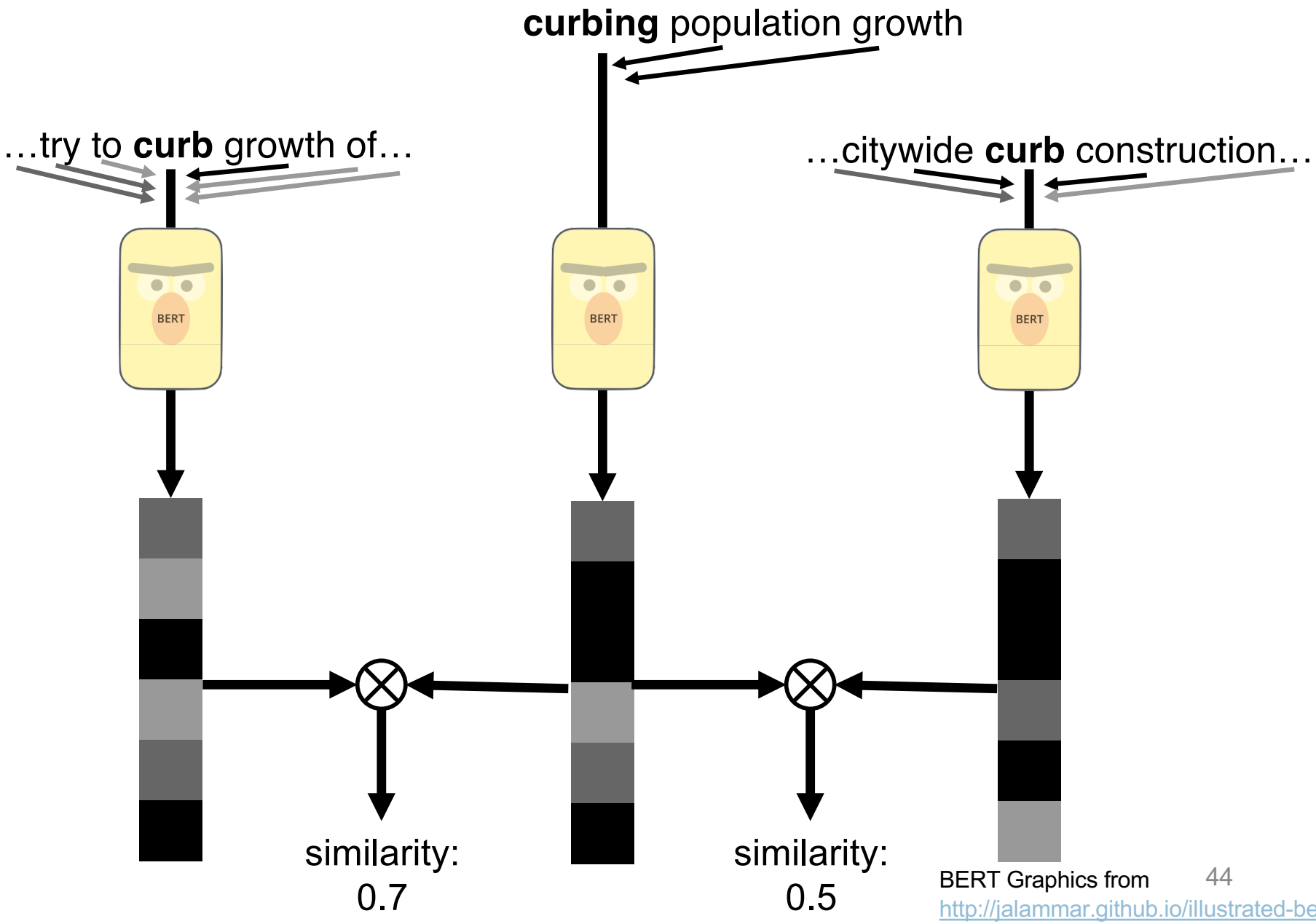


Vanilla BERT



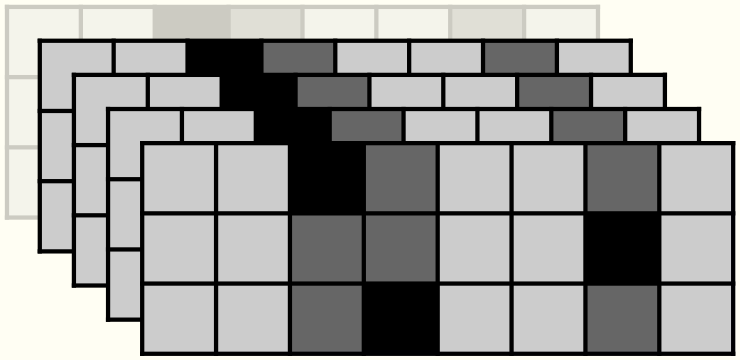
BERT Graphics from Jay Alammar, http://jalammar.github.io/illustrated-bert/

Contextualized Word Vectors



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

BERT consists of multiple layers. We build a similarity matrix for each layer.

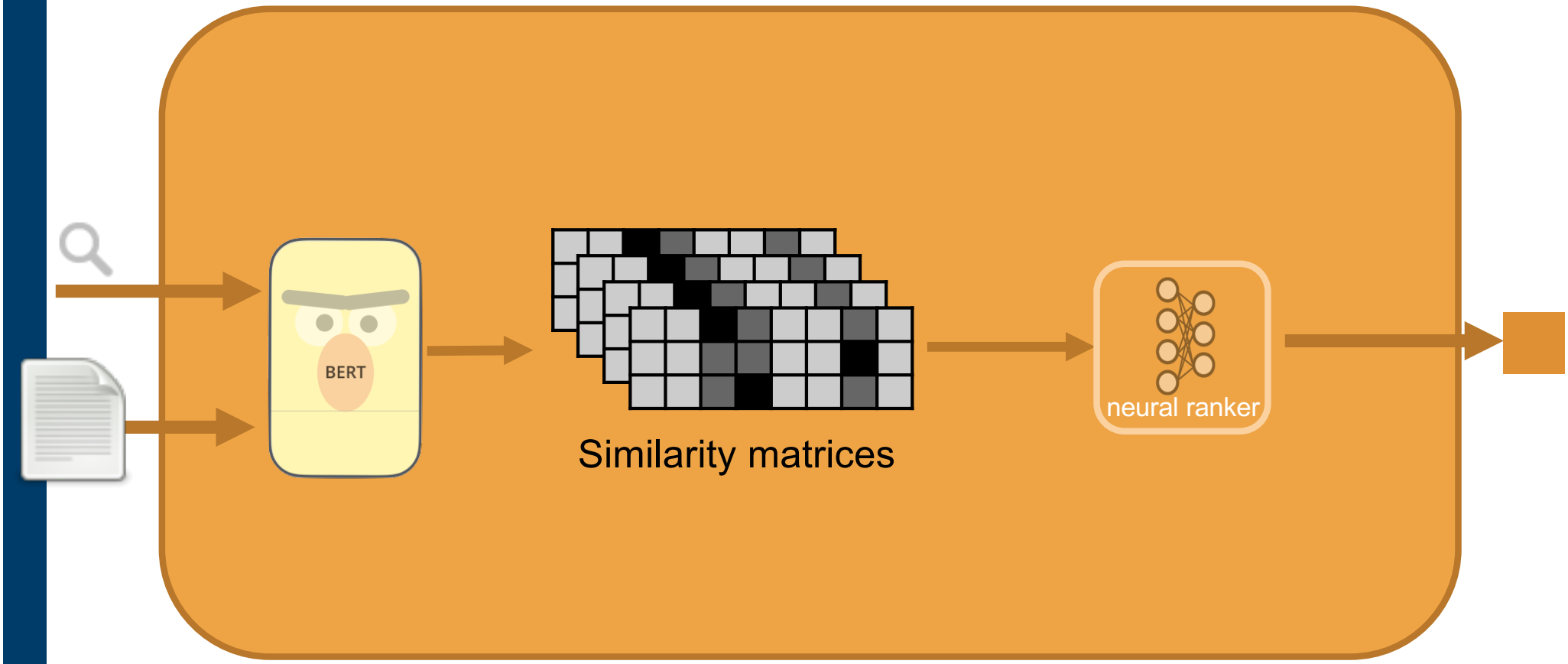


$BERT_{BASE}$: 13 Layers

* Including 1 input layer (non-transformed)

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

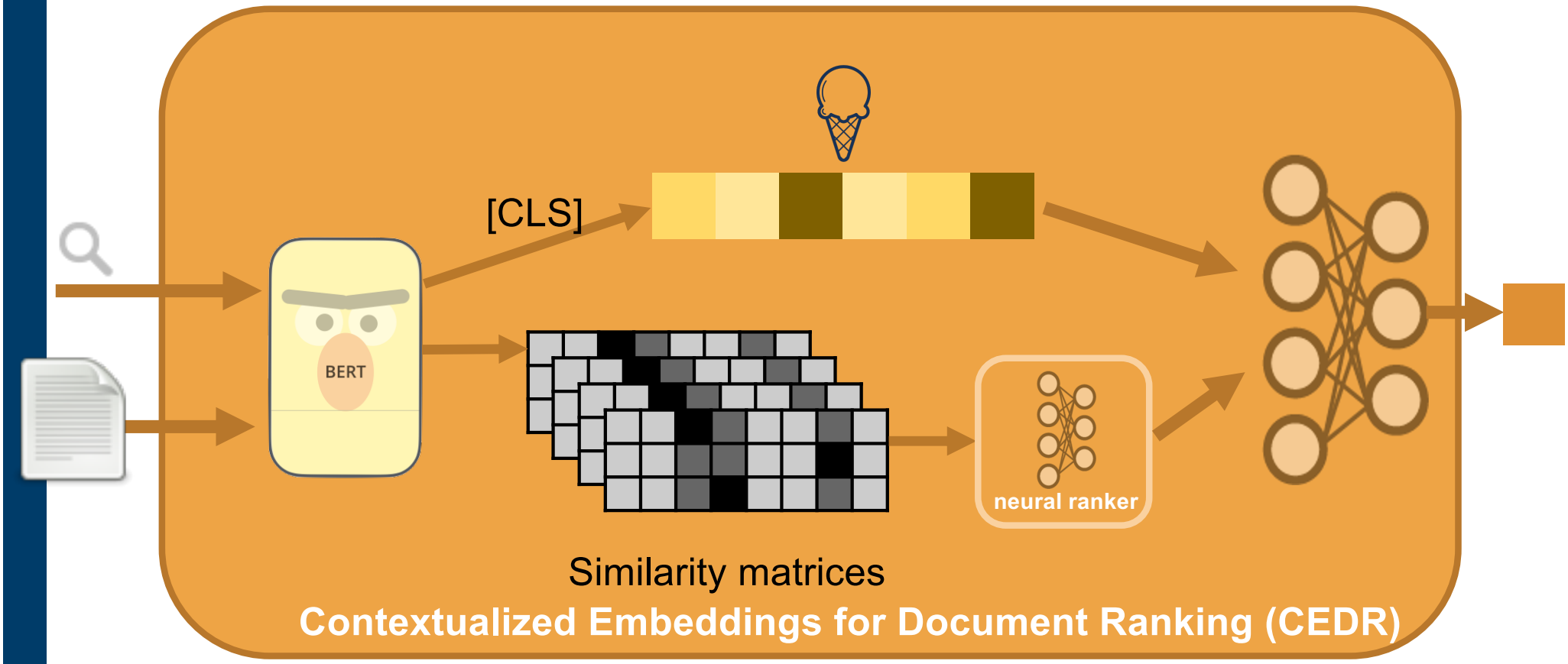
Contextualized Word Vectors



MacAvaney et al. CEDR. SIGIR 2019.

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

CEDR



MacAvaney et al. CEDR. SIGIR 2019.

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

BERT Practical

Università di Pisa | University of Glasgow

Replace KNRM's vocabulary with BERT

```
bert_knrm = onir_pt.reranker('knrm', 'bert', text_field='title_abstract')
```

Vanilla model that uses BERT's [CLS] for ranking

```
vbert = onir_pt.reranker('vanilla_transformer', 'bert', text_field='title_abstract')
cedr_knrm = onir_pt.reranker('cedr_knrm', 'bert', text_field='title_abstract')
```

CEDR KNRM model (using both KNRM and [CLS])

	name	map	ndcg	ndcg_cut_10	mrt
0	BM25	0.077880	0.177728	0.644374	36.795420
1	BM25 >> BERT	0.065846	0.160599	0.416803	1847.491717
2	BM25 >> BERT KNRM	0.059786	0.151265	0.277798	1906.480065
3	BM25 >> CEDR KNRM	0.067430	0.165405	0.507656	1917.520533

You still need to train these models!

BERT Practical

Università di Pisa | University of Glasgow

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

Causal Language Models

BERT is trained with a Masked Language Modeling (MLM) objective.

- Words are masked out in the input and it's trained to fill in the missing words

T5, GPT, etc. are trained with a Causal Language Modeling (CLM) objective.

- Models learn to predict the next word in a sequence.
- This allows them to be used in text generation settings.

Scoring with a CLM

Università di Pisa | University of Glasgow

Main idea: Train a model to generate the text "true" or "false", prompted with the query and document:

Query: coronavirus early symptoms **Document:** 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… **Relevant:**

Ask the model: Should the next word be "true" or "false"?

$$P(rel \mid q, d) \approx P(\text{“}true\text{”} \mid \text{“}Query\text{:”}\ \$q\ \text{“}Document\text{:”}\ \$d\ \text{“}Relevant\text{:”})$$

Train on relevant and non-relevant pairs.

Nogueira et al. Document Ranking with a Pretrained Sequence-to-Sequence Model. arXiv 2020.

MonoT5 Practical

Università di Pisa | University of Glasgow

```
from pyterrier_t5 import MonoT5ReRanker
monoT5 = MonoT5ReRanker()
```

```
br = pt.BatchRetrieve(indexref, wmodel='BM25') % 100
pt.Experiment(
  [br, br >> pt.text.get_text(dataset, 'text') >> monoT5],
  dataset.get_topics(),
  dataset.get_qrels(),
  names=["BM25", "BM25 >> monoT5"],
  eval_metrics=["map", "recip_rank", "P.10", "ndcg_cut.10"]
)
```

monoT5 batches: 100% 2325/2325 [01:36<00:00, 24.00it/s]

	name	map	recip_rank	P_10	ndcg_cut_10
0	BM25	0.272523	0.725587	0.352688	0.446609
1	BM25 >> monoT5	0.295071	0.750839	0.413978	0.490703

Part 3C

EFFICIENCY

Problem: Efficiency

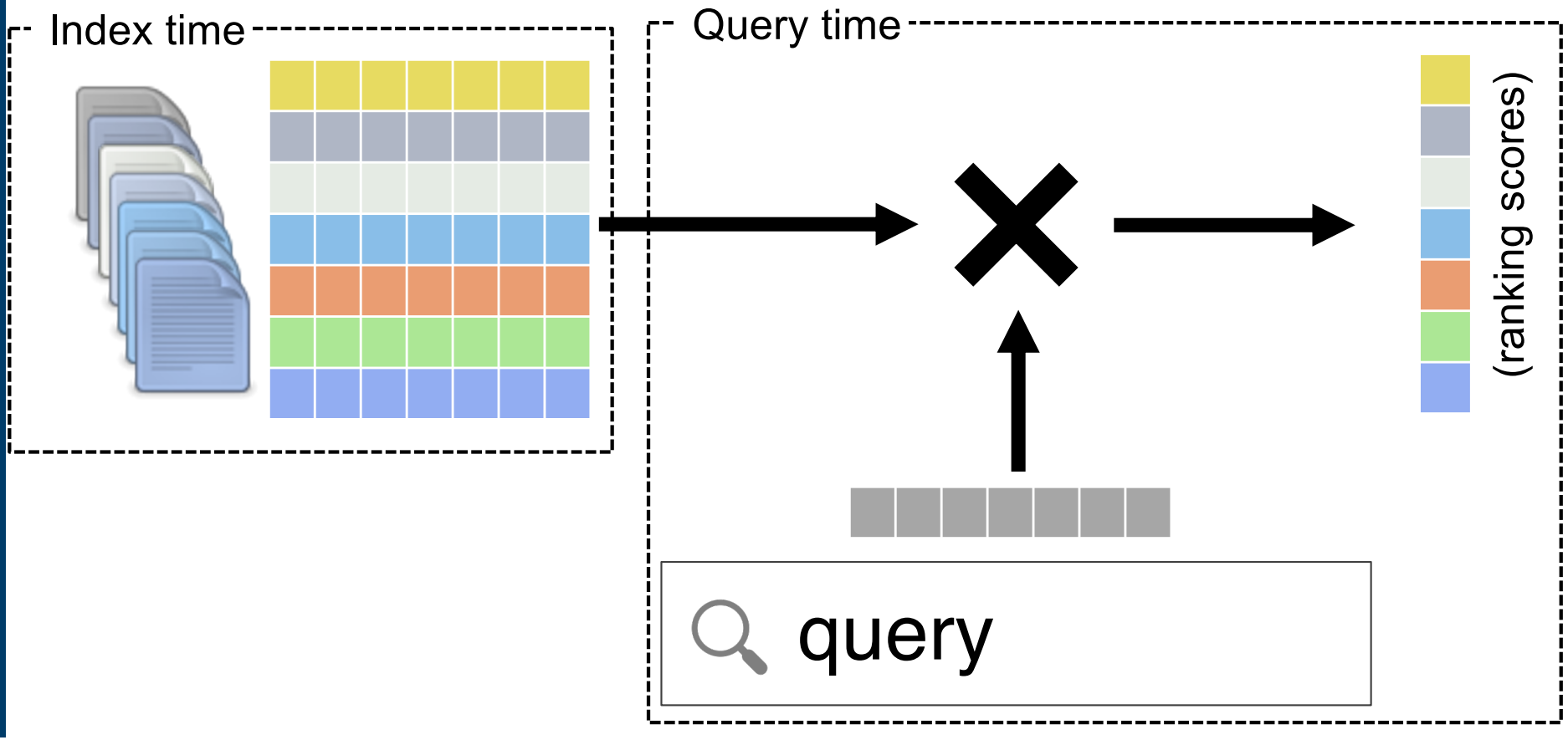
Università di Pisa | University of Glasgow

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

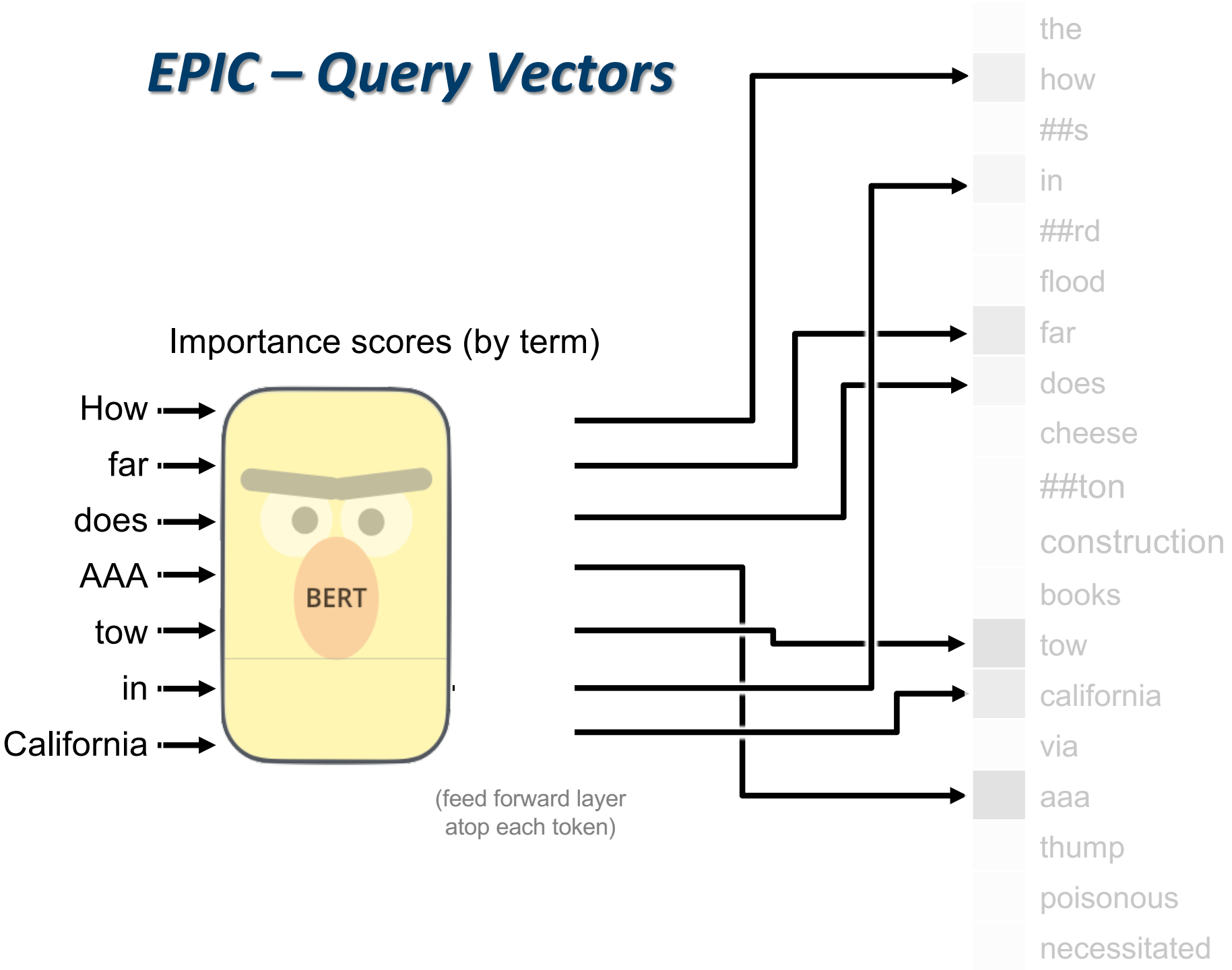
E.g., BERT takes 45x longer than BM25: (1.7 seconds)

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

Main idea: Learn representations that are fast to score.

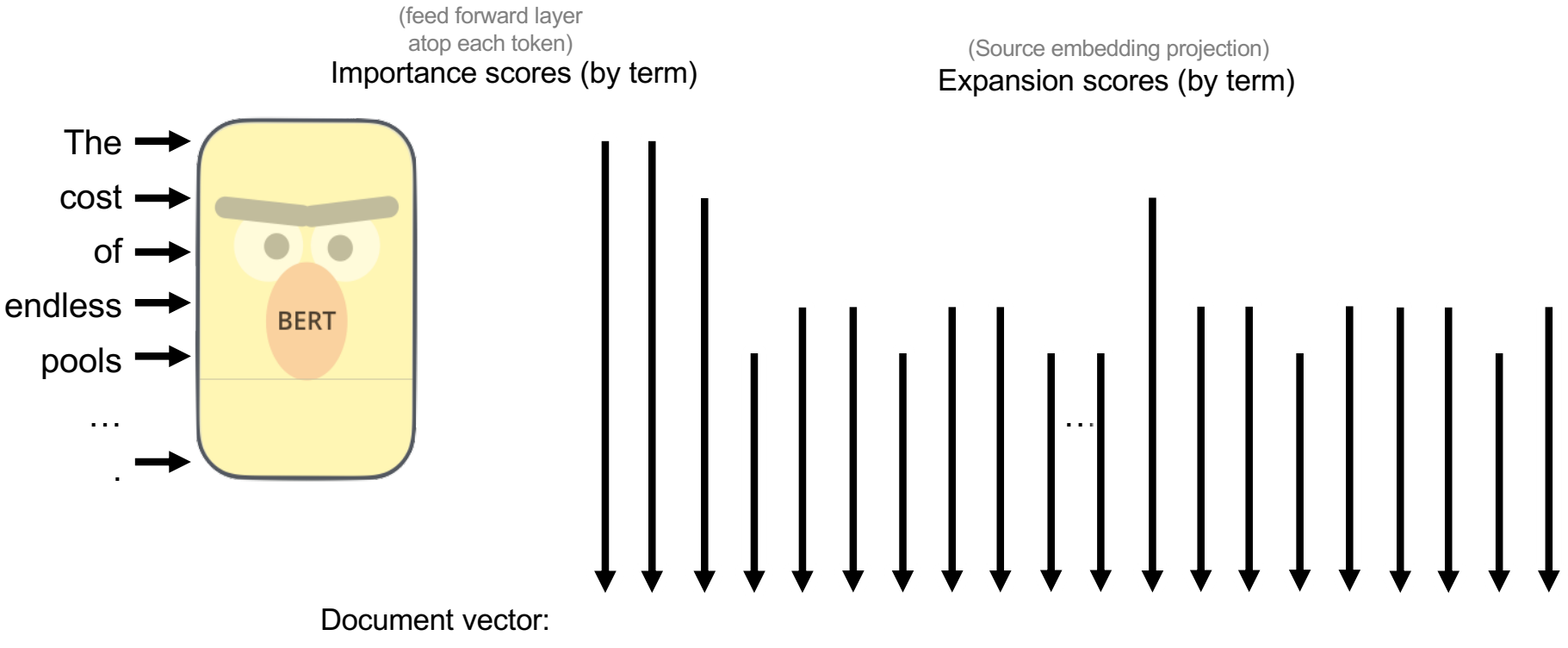


EPIC – Query Vectors



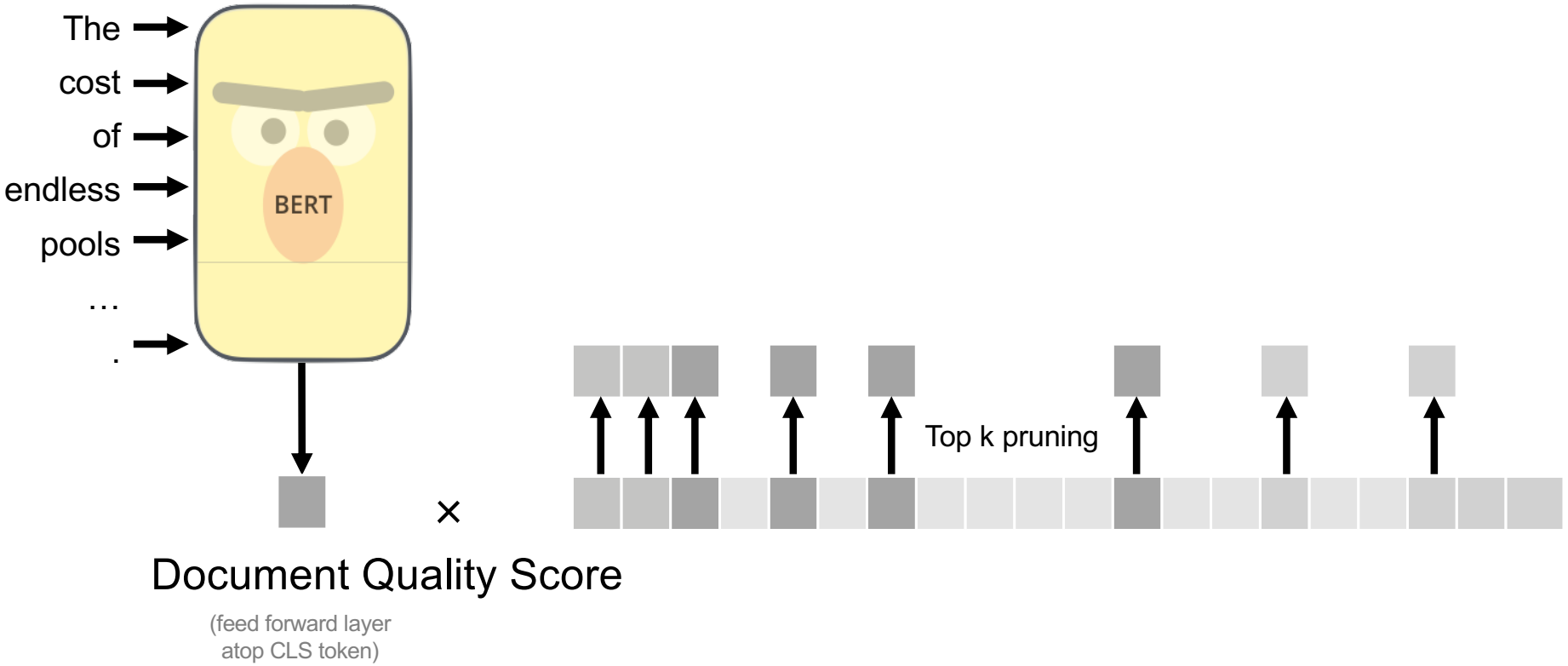
BERT Graphics from Jay Alammar, http://jalammar.github.io/illustrated-bert/

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/

EPIC Practical

Lazy EPIC (not indexed)

```
# Load a version of EPIC trained on the MS-MARCO dataset
lazy_epic = onir_pt.reranker.from_checkpoint(
    'https://macavaney.us/epic.msmarco.tar.gz',
    expected_md5="2f6a16be1a6a63aab1e8fed55521a4db")
```

```
br = pt.BatchRetrieve(index) % 30
pipeline = (br >> pt.text.get_text(dataset, 'abstract')
               >> pt.apply.generic(lambda x: x.rename(columns={'abstract': 'text'}))
               >> lazy_epic)
pt.Experiment(
    [br, pipeline],
    dataset.get_topics('title'),
    dataset.get_qrels(),
    names=['DPH', 'DPH >> EPIC (lazy)'],
    eval_metrics=["recip_rank", "P.5", "mrt"]
)
```

	name	recip_rank	P_5	mrt
0	DPH	0.766833	0.684	36.734074
1	DPH >> EPIC (lazy)	0.817889	0.724	801.524438

EPIC Practical

Università di Pisa | University of Glasgow

EPIC (indexed)

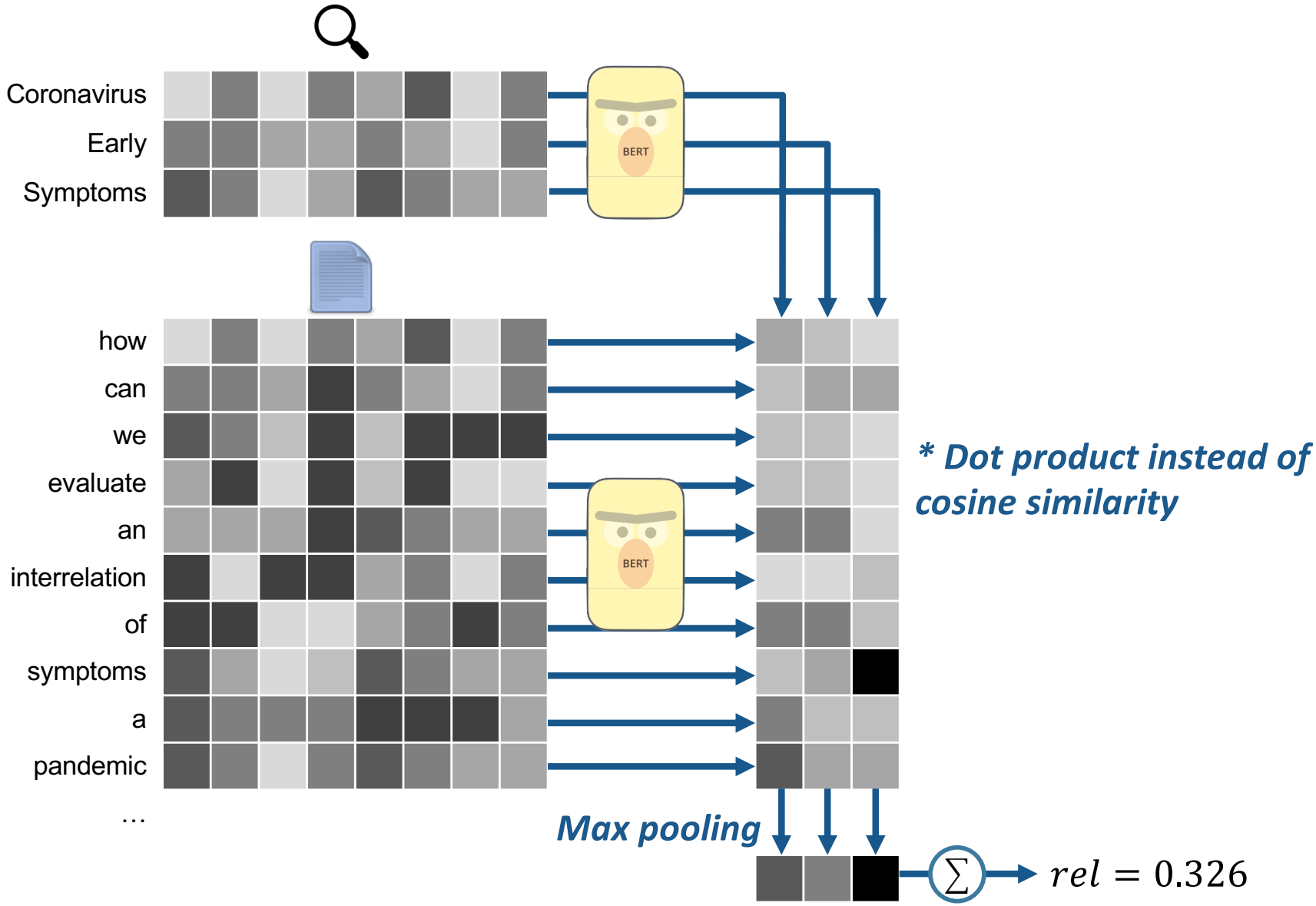
```
indexed_epic = onir_pt.indexed_epic.from_checkpoint(
    'https://macavaney.us/epic.msmarco.tar.gz',
    index_path='./epic_cord19')
```

```
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

Contextualized Late Interaction over BERT (ColBERT)



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

ColBERT Re-Ranking practical

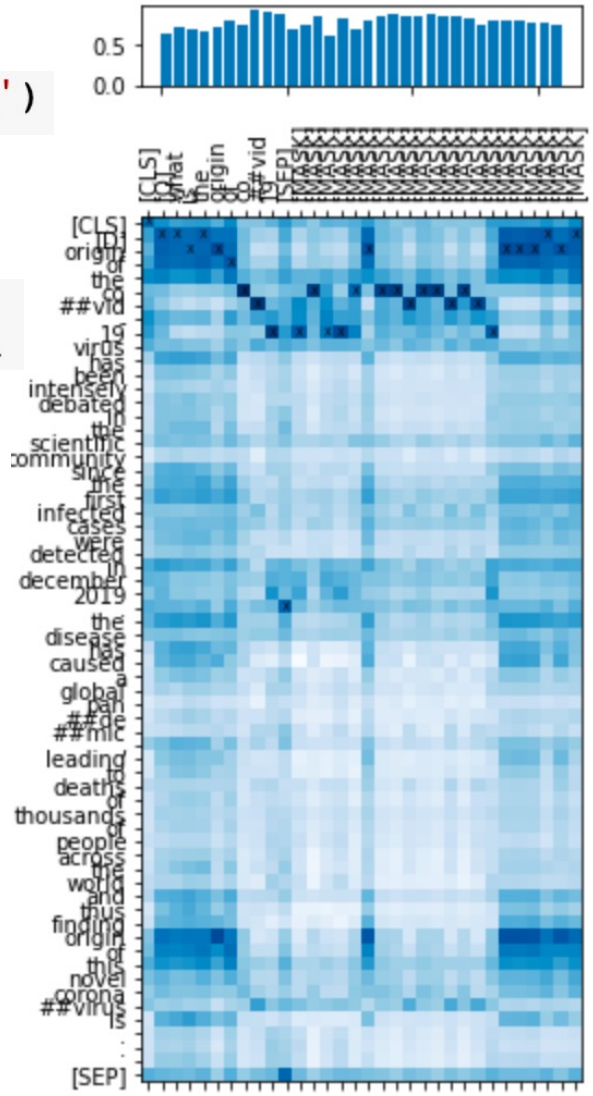
```
colbert_factory = pyterrier_colbert.ranking.ColBERTFactory(
    "http://www.dcs.gla.ac.uk/~craigm/colbert.dnn.zip", None, None)
```

```
colbert = colbert_factory.text_scorer(doc_attr='abstract')
```

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

	name	map	P_10	ndcg	ndcg_cut_10	mrt
0	DPH	0.068006	0.658	0.165607	0.609058	52.517634
1	DPH >> ColBERT	0.074670	0.752	0.172034	0.690447	615.640942

```
colbert_factory.explain_text(query, text)
```



Other Models

There's a bunch of other great re-ranking models:

- **TK/TKL – Contextualized kernel-based ranking.**
  - Hofstätter et al. Interpretable & time-budget-constrained contextualization for re-ranking. ECAI 2020.
- **PARADE – Handling long documents through passage representation aggregation**
  - Li et al. PARADE: Passage Representation Aggregation for Document Reranking. arXiv 2020.
- **Duet – combine representation and interaction models.**
  - Mitra et al. Learning to match using local and distributed representations of text for web search. WWW 2017.
- **And others…**

University of Glasgow | Università di Pisa

ECI 2021

Part 3D

WRAPUP

PyTerrier Supported Re-Rankers

- From OpenNIR (supports training & inference):
  - Supporting training & inference: DRMM, KNRM, PACRR, Duet (Local), MatchPyramid, Vanilla BERT, CEDR
  - Training, inference, and indexing: EPIC
  - https://github.com/Georgetown-IR-Lab/OpenNIR
- monoT5 (supports inference)
  - Using PyTerrier plugin & huggingface transformers library
  - https://github.com/terrierteam/pyterrier_t5
- ColBERT (supports inference)
  - Using PyTerrier plugin & author's repository
  - https://github.com/terrierteam/pyterrier_colbert

Summary

Re-ranking approaches continue to be an appealing and active area of research:

- Handle vocabulary mismatch and text semantics
- Straightforward setting & used in industry
- Highly effective
- Query latency can be managed

Limitations:

- Documents not retrieved in the first stage cannot be re-ranked
- A higher-quality initial ranking means you need a lower retrieval threshold (which means less compute)
- Can we improve first-stage ranking using neural methods? (Coming in Part 4)

QUESTIONS?

What's the task in the notebooks?

Università di Pisa | University of Glasgow

In the notebooks, you will experience:

- Building re-ranking models from scratch
- Using pre-trained BERT and T5 models
- Scoring with EPIC using pre-computed document vecs
- Tuning re-ranking thresholds
- Visualising ColBERT's scoring as a re-ranker

Practical Time

The tutorial Github repo has links to the notebook for Part 3

- https://github.com/terrier-org/ecir2021tutorial
- Press the 

link for each notebook to start a Colab session

Timings:

- Practical – in breakout rooms – until 1615
- Coffee break 1615-1645
- Part 4 resumes at 1645

If you are leaving us here, please complete our feedback quiz https://forms.office.com/r/2WbpLiQmWV

References

DRMM: *Guo, et al. A Deep Relevance Matching Model for Ad-hoc Retrieval. CIKM 2017.*

KNRM: *Xiong, et al. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. SIGIR 2017.*

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

BERT: *Devlin, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019.*

CEDR: *MacAvaney, et al. CEDR: Contextualized Embeddings for Document Ranking. SIGIR 2019*.

MonoT5: *Nogueira, et al. Document Ranking with a Pretrained Sequence-to-Sequence Model. arXiv 2020.*

EPIC: *MacAvaney, et al. Expansion via Prediction of Importance with Contextualization. SIGIR 2020.*

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