University of Glasgow | Università di Pisa

SEARCH SOLUTIONS London, UK

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

## A Search Solutions 2022 tutorial with PyTerrier

***Starting at 14:45 GMT***

Sean MacAvaney*
Craig Macdonald*
Nicola Tonellotto*

*(*Alphabetical ordering)*

# PART 3
# INDEX AUGMENTATION & LEARNED SPARSE RETRIEVAL & DENSE RETRIEVAL

# *Intended Learning Outcomes*

**Part 3** – ***Understand the most recent effective retrieval architectures that learn representations of documents, both in leveraging traditional index structures (index augmentation & sparse retrieval), as well as new similarity search systems (dense retrieval)***

ILO 3A. Understand neural augmentation models, such as Doc2Query

ILO 3B Understand learned sparse retrieval methods, such as DeepCT, DeepImpact and SPLADE

ILO 3C. Understand dense retrieval models and how embedding indexes work, such as FAISS, ANCE and ColBERT

ILO 3D. Perform experiments with ANCE and ColBERT in a Python notebook.

# *Outline of Part 3*

Università di Pisa | University of Glasgow

## Part 3A: Neural Index Augmentation

## Part 3B: Learned Sparse Retrieval

## Part 3C: Dense Retrieval

## Part 3D: Embedding Indexes and Vector Search

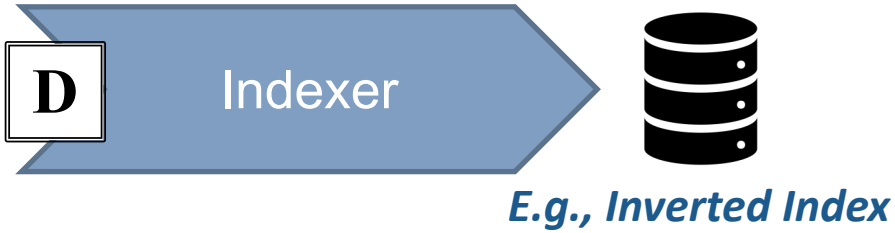
University of Glasgow | Università di Pisa

SEARCH SOLUTIONS London, UK

Part 3A

# NEURAL INDEX AUGMENTATION

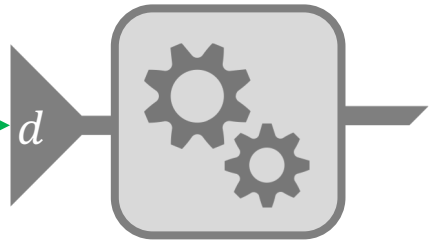
# Document Augmentation




| docno | text |
|---|---|
| ug7v899j | OBJECTIVE: This retrospective chart review describes the… |
| 02tnwd4m | Inflammatory diseases of the respiratory tract are common… |
| ejv2xln0 | Endothelin-1 (ET-1) is a 21 amino acid peptide with diverse… |
| … | … |

# Document Augmentation

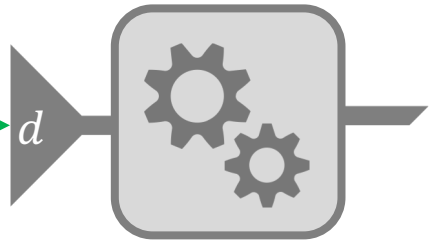

| docno | text |
|---|---|
| ug7v899j | OBJECTIVE: This retrospective chart review describes the… |
| 02tnwd4m | Inflammatory diseases of the respiratory tract are common… |
| ejv2xln0 | Endothelin-1 (ET-1) is a 21 amino acid peptide with diverse… |
| … | … |




```
where is mycoplasma p where is
jeddah hospital located what
culture proves mycoplasma
pneumoniae
```

# Document Augmentation

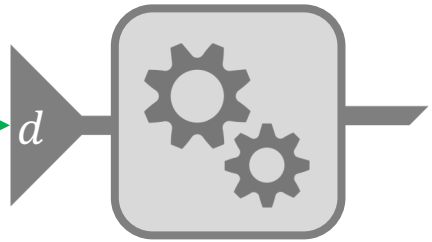

| docno | text | newfield |
|---|---|---|
| ug7v899j | OBJECTIVE: This retrospective chart review describes the… | where is mycoplasma p where is jeddah… |
| 02tnwd4m | Inflammatory diseases of the respiratory tract are common… | |
| ejv2xln0 | Endothelin-1 (ET-1) is a 21 amino acid peptide with diverse… | |
| … | … | |




```
where is mycoplasma p where is
jeddah hospital located what
culture proves mycoplasma
pneumoniae
```

# Document Augmentation


| docno | text |
|---|---|
| ug7v899j | OBJECTIVE: This retrospective chart review describes the… where is mycoplasma p where is jeddah… |
| 02tnwd4m | Inflammatory diseases of the respiratory tract are common… |
| ejv2xln0 | Endothelin-1 (ET-1) is a 21 amino acid peptide with diverse… |
| … | … |




```
where is mycoplasma p where is
jeddah hospital located what
culture proves mycoplasma
pneumoniae
```

# Doc2Query

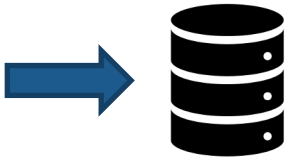
**Main idea:** Use a causal language model to generate additional text to add to documents when indexing.

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

***How are the queries generated?***

**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…

**Generated Queries:**
where does mrna originate
where does subgenomic rna come from
where is the nidovirus mrna?
when a nidovirus is produced, its genome is quizlet
what is nidovirus mrna
where in a nidovirus can one mrna be derived
where is the leader sequence derived
what types of mrna are found in nidovirus
where are the nidovirus genomes derived from
…



*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# *Doc2Query*

Università di Pisa | University of Glasgow

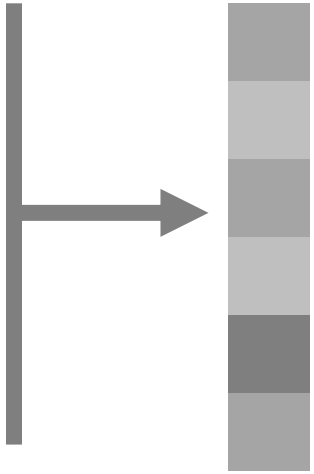
**Training: optimize to generate queries from passages on a collection with many queries (e.g., MS-MARCO)**

**Most recent version: Use T5 model for generation (docTTTTTquery)**

*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# Doc2Query

**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…

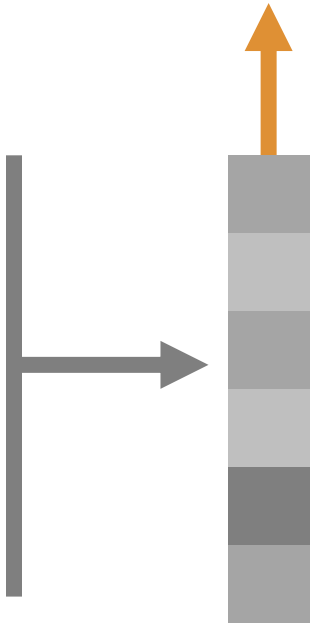


Step 1: Source text encoded
e.g., via RNN or Transformer

*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# Doc2Query

**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…

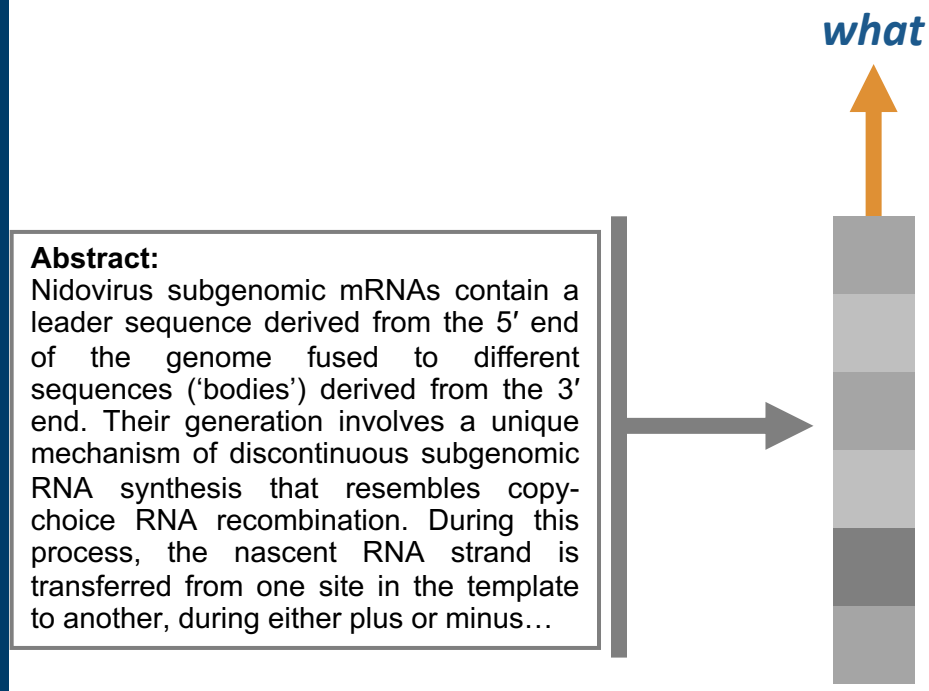


Step 2: Text iteratively generated from encoded document

e.g., via RNN or transformer, using beam search

*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# Doc2Query

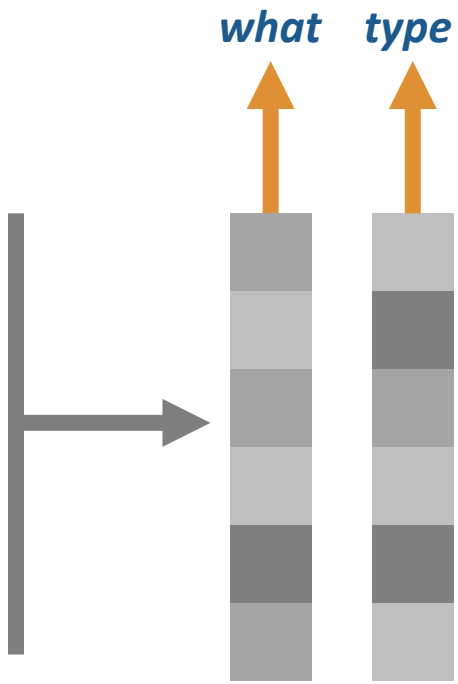



Step 2: Text iteratively generated from encoded document

e.g., via RNN or transformer, using beam search

*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# Doc2Query




**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…

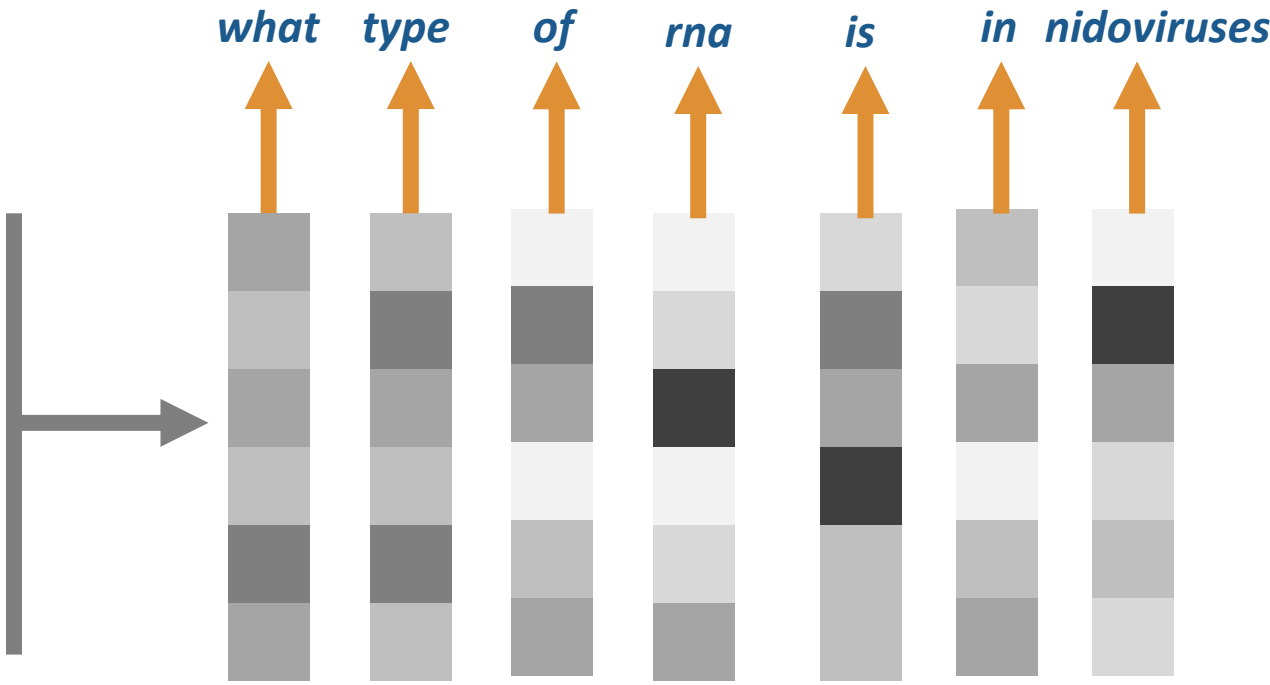
Step 2: Text iteratively generated from encoded document

e.g., via RNN or transformer, using beam search

*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# Doc2Query

**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…




Step 2: Text iteratively generated from encoded document
e.g., via RNN or transformer, using beam search

*Nogueira & Lin. From doc2query to docTTTTTquery. 2019.*

# Doc2Query Practical

Università di Pisa | University of Glasgow

```
from pyterrier_doc2query import Doc2Query
doc2query = Doc2Query(out_attr="text", batch_size=8)
```

## Example doc2query outputs

```
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. MET
HODS: Patients with positive M. pneumoniae cultures from respiratory specim
```

```
doc2query.transform(df).iloc[0]['text']
```

```
'what is culture confirmed mycoplasma where is mycoplasma pneumonia located
in saudi arabia? where is mycoplasma cultured'
```

Part 3B

# LEARNED SPARSE RETRIEVAL

# *Learned Sparse Retrieval*

Università di Pisa | University of Glasgow

**Main question: Can we leverage the neural IR effectiveness improvements together with the sparse retrieval efficiency in query processing?**

- In sparse retrieval, documents are stored in an inverted index data structure
- The inverted index is highly specialized toward efficient query processing
- Major data structure in search engines
- In neural IR, semantic models outperform lexical models in ad-hoc search tasks
- Storing semantic information is expensive
- Processing semantic information is expensive

# DeepCT

Università di Pisa | University of Glasgow

## 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.*

# DeepCT practical

Università di Pisa | University of Glasgow

```
deepct = pyterrier_deepct.DeepCTTransformer(
    "bert-base-uncased/bert_config.json",
    "marco/model.ckpt-65816"
)
deepct(pd.DataFrame([{
  'docno': '0', 'text': '''The presence of communication amid scientific minds
                           was equally important to the success of the Manhattan
                           Project as scientific intellect was. The only cloud
                           hanging over the impressive achievement of the atomic
                           researchers and engineers is what their success truly
                           meant; hundreds of thousands of innocent lives
                           obliterated.'''
}])
```

```
presence communication scientific minds equally important to success manhattan
project intellect cloud impressive achievement atomic researchers engineers truly
meant lives obliterated
```

# DeepImpact

Università di Pisa | University of Glasgow

**Main idea: Compute a quantized score (impact) to store in posting list**

- Not just BM25 + learned TF as in DeepCT
- Add text to documents using a contextualized language model
- Address the vocabulary mismatch  problem
- Exploit BERT to compute a single score for each query-document pair
- Quantize linearly the scores into positive integers
- Store quantized scores as frequencies in postings, using simple sum as weighting model

*Mallia, Khattab, Tonellotto, Suel. Learning Passage Impacts for Inverted Indexes. SIGIR 2021*

# DeepImpact practical

- Sparse Indexing – Learned

```
vaswani = pt.get_dataset("vaswani")

pt_index_path = './terrier_di_vaswani'

parent = pt.index.IterDictIndexer(pt_index_path)
parent.setProperty("termpipelines", "")

indexer = DeepImpactIndexer(parent, batch_size=32)
indexer.index(vaswani.get_corpus_iter())

index_ref = pt.IndexRef.of(pt_index_path + "/data.properties")
index_di = pt.IndexFactory.of(index_ref)
```

```
pt_index_path = './terrier_vaswani'

indexer = pt.index.IterDictIndexer(pt_index_path)
indexer.setProperty("termpipelines", "")
index_ref = indexer.index(vaswani.get_corpus_iter())

index_ref = pt.IndexRef.of(pt_index_path + "/data.properties")
index = pt.IndexFactory.of(index_ref)
```

- End-to-end Retrieval

```
pt.Experiment([
        pt.BatchRetrieve(index, wmodel="BM25"),
        pt.BatchRetrieve(index_di, wmodel="Tf")
    ],
    vaswani.get_topics(), vaswani.get_qrels(),
    names=['bm25', "deep_impact"],
    eval_metrics=["map", "recip_rank", "ndcg_cut_10"]
)
```

| | name | map | recip_rank | ndcg_cut_10 |
|---|---|---|---|---|
| 0 | bm25 | 0.143471 | 0.516597 | 0.249405 |
| 1 | deep_impact | 0.186312 | 0.661398 | 0.332272 |

**Original Terms:** of the in origin atmosphere upper nitrogen ionization
**DeepImpact Terms:** of the a in and to is described are design suitable discussed characteristics probe positive small such experiments density measurement ionosphere type designed rocket spherical laboratory ion borne investigate adopted meshed

# Sparse Representation Learning

**Main idea: compute scores and query/document expansions at the same time (cf. EPIC)**

- BERT models compute a term language model for each output embeddings
  - A language model is a probability distribution over the vocabulary
- These language models capture the semantic correlations between the input term and all other terms in the vocabulary
- They can be used to expand documents with new terms or compress them by removing existing terms

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

# SPLADE

**Main idea: compute scores and query/document expansions at the same time (cfr. EPIC)**

- Compute a (un-normalized) LM for each token in the document
- Filter the weights with logarithm to saturate high values
- Sum everything up to get a document-level language model
- Use the highest-valued components to add new terms to the documents
  - Special training to boost sparsity (FLOPS regularisation)

*Formal, Piwowarski, Clinchant. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. SIGIR 2021.*

# SPLADE practical

- Sparse Indexing – Learned

```
1  import pandas as pd
2  import pyterrier as pt ; pt.init()
3  import pyt_splade
```

```
1  splade = pyt_splade.SpladeFactory(agg='max')
```

```
1  doc_pipeline = splade.indexing()
```

```
1  doc_pipeline(pd.DataFrame([
2    {'docno': '0', 'text': 'The presence of communica
3    {'docno': '100', 'text': "Antonín Dvorák (1841–19
4    {'docno': '1000', 'text': 'QuickFacts Matanuska-S
5  ]))
```


the presence of communication amid scientific minds was equally important to the success of the manhattan project as scientific intellect was . the only cloud hanging over the impressive achievement of the atomic researchers and engineers is what their success truly meant ; hundreds of thousands of innocent lives ob ##lite ##rated .

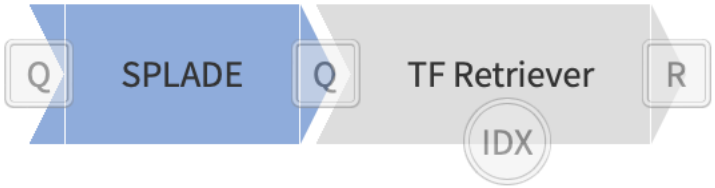
**Expansion Tokens:** communications importance successful intellectual intelligence mind clouds projects achievements york amongst engineer bomb among engineering atom scientist helped nuclear amidst thinking impact thought destroyed scientists experiment existence telecommunications many failed because research thoughts hang message researcher science millions life death die contributed brains mean enlightenment murder minded significance ##rate survival hiroshima radio significant leader tunnel threat were clouded goal role really invented ##athic accomplishments controversial successes ended resulted purpose defeated casualties interesting physicist amazing communicate roosevelt people intent discovery started franklin telegraph within einstein journalist merit lack army tech hoped mental truman ny ##bility rocket led story great ##bm went study suspended respect comet radios collaboration alla destruction kennedy perception died committee studies society also said radioactive relativity conducted spirit results miracle suicide necessary crash result thinkers affect war ##rut absence contribution mission forensic news destroy knowledge wanted involved

# SPLADE practical

- Indexing & End-to-end Retrieval on MSMarco v1

```
import pyterrier as pt
pt.init(version='snapshot')
import pyt_splade

dataset = pt.get_dataset('irds:msmarco-passage')
splade = pyt_splade.SpladeFactory()

indexer = pt.IterDictIndexer('./msmarco_psg', pretokenized=True)

indxer_pipe = splade.indexing() >> indexer
indxer_pipe.index(dataset.get_corpus_iter())
```


```
bm25_retr = pt.BatchRetrieve.from_dataset('msmarco_passage', 'terrier_stemmed', wmodel='BM25')
splade_retr = splade.query() >> pt.BatchRetrieve('./msmarco_psg', wmodel='Tf')

pt.Experiment(
    [splade_retr, bm25_retr],
    pt.get_dataset('msmarco_passage').get_topics('test-2019'),
    pt.get_dataset('msmarco_passage').get_qrels('test-2019'),
    batch_size=200,
    filter_by_qrels=True,
    eval_metrics=[RR(rel=2), nDCG@10, nDCG@100, AP(rel=2)],
    names=['splade','bm25_stemmed'])
```




| name | RR(rel=2) | nDCG@10 | nDCG@100 | AP(rel=2) |
|---|---|---|---|---|
| splade | 0.918605 | 0.730178 | 0.671292 | 0.503885 |
| bm25_stemmed | 0.641565 | 0.47954 | 0.487416 | 0.286448 |

# Other Approaches

| Method | Query | Passage | Reg. | Supervision | | |
|---|---|---|---|---|---|---|
| | | | | Level | Neg. | Type |
| DeepCT [3] | MLP | MLP | - | Term | - | - |
| uniCOIL [14] | MLP | MLP | - | Passage | BM25(s) | Human |
| $\text{uniCOIL}_{dT5q}$ [14] | MLP | expMLP | - | Passage | BM25(s) | Human |
| $\text{uniCOIL}_{tilde}$ [14] | MLP | expMLP | - | Passage | BM25(s) | Human |
| EPIC [17] | MLP | MLM | Top-k | Passage | BM25 | Human |
| DeepImpact [18] | MLP | expMLP | - | Passage | BM25 | Human |
| TILDE [30] | BINARY | clsMLM | - | Term | - | - |
| TILDEv2 [30] | BINARY | expMLP | - | Passage | BM25(s) | Human |
| Sparta [29] | BINARY | MLM | - | Passage | BM25 | Human |
| SPLADE-max [4] | MLM | MLM | FLOPs | Passage | BM25(s) | Human |
| DistilSPLADE-max [4] | MLM | MLM | FLOPs | Passage | Hard | Teacher |

**Table courtesy of Thong Nguyen.**

# Summary

- **DeepCT**: use BERT to learn in-document term frequencies, to use with the classical BM25 model
- **DeepImpact**: use BERT to learn in-document term scores, to store directly in the inverted index in quantised format
- **SPLADE**: use BERT to learn in-document term scores, to store directly in the inverted index, and to perform document expansion at indexing time and query expansion at query processing time
- Other approaches are variants of these approaches

University of Glasgow | Università di Pisa

SEARCH SOLUTIONS London, UK

Part 3C

# DENSE RETRIEVAL

# Dense Retrieval

- Dense Retrieval, particularly for passages is of great interest
  - At indexing time, represent passages by one of more embeddings
  - At retrieval time, encode the query using the same model, and identify passages with embedding(s) **similar** to the query
- Great improvements have been proposed in large-scale **similarity search systems**
  - **E.g. FAISS** is the Facebook large scale nearest neighbour (NN) search system
  - FAISS supports for **exact** and **approximate search**
  - The type of search depends on the **problem dimension**
- In the following, we review the main dense retrieval models, then dense retrieval NN search

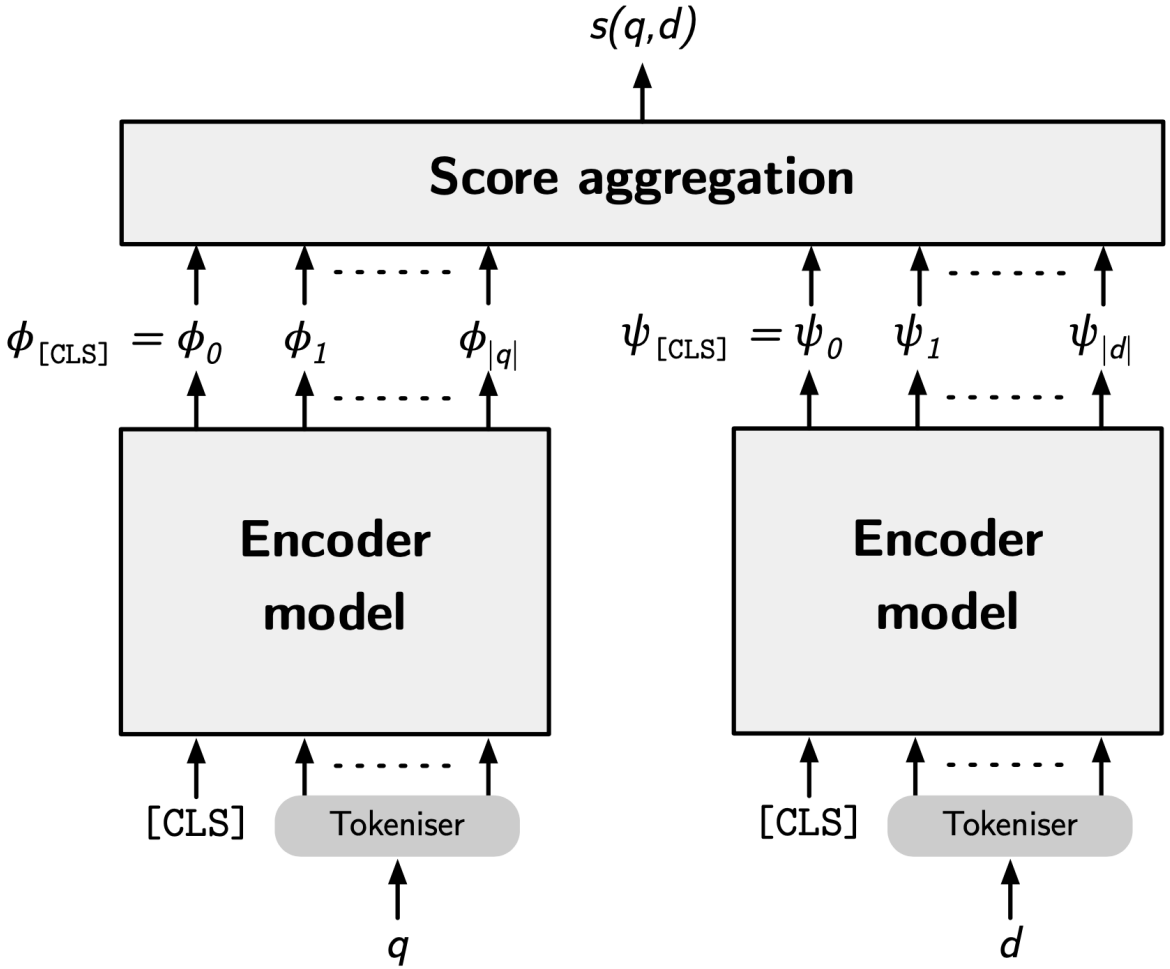
# Sparse vs Dense Retrieval (I)

- **Sparse Retrieval**
  - Every document is represented by a single score for each term in the lexicon $V$ (equal to 0 for terms not appearing in the document)
  - A document is a $|V|$-dimensional vector with many 0s, i.e., sparse
  - A query-document relevance score is computed as the sum of the score of the query terms in the documents
  - Only query terms appearing in the document contributes to the final score
  - A query is then a $|V|$-dimensional vector with almost all entries set to 0, only a few set to 1
- **Pros**
  - Efficient storage in inverted indexes
  - Easily scalable
  - 40+ years of query processing optimizations based on the inverted index
  - Widely adopted as first stage ranker in cascading architectures
- **Cons**
  - Very few relevance signals
  - Not considering semantics information (expect learned sparse models)
  - Limited support for proximity signals

# Sparse vs Dense Retrieval (II)

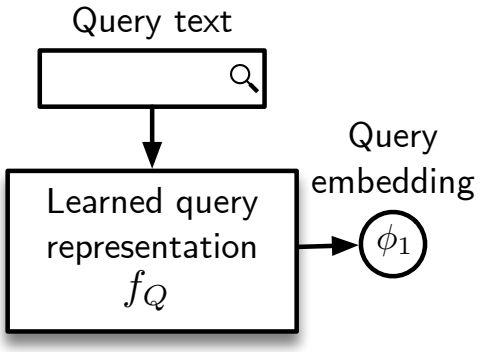
- **Dense Retrieval**
  - Every document and query are represented by a *D*-dimension vector(s) of scores with few 0s, i.e., dense (embeddings)
  - The query-document relevance score is computed as the L2 or cosine similarity between the query-document vectors
- **Pros**
  - Efficient storage in similarity indexes
  - 20+ years of query processing optimizations based on the nearest neighbour search
  - Rich set of relevance signals
  - Consider semantics
- **Cons**
  - Missing a clear interpretation of the embedding space
  - Missing a clear interpretation of the captured semantics

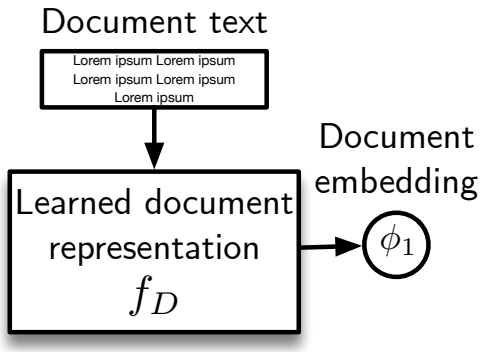
# Representation-focused System



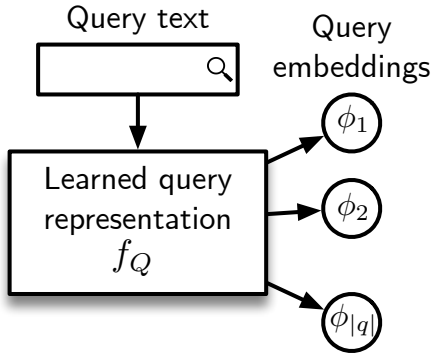
# Learning Representations

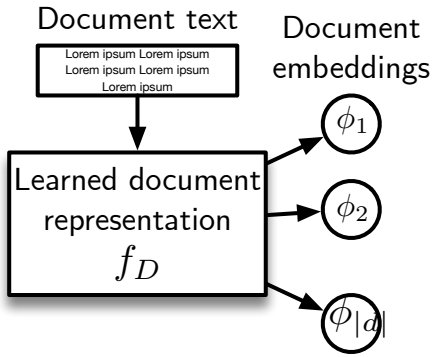
- **Single Representation (USE, DPR, ANCE, STAR/ADORE)**



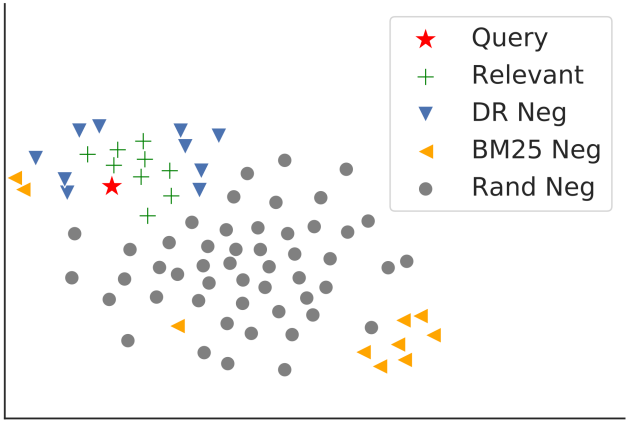

- **Multiple Representation (Poly-encoders, ME-BERT, COIL, ColBERT)**

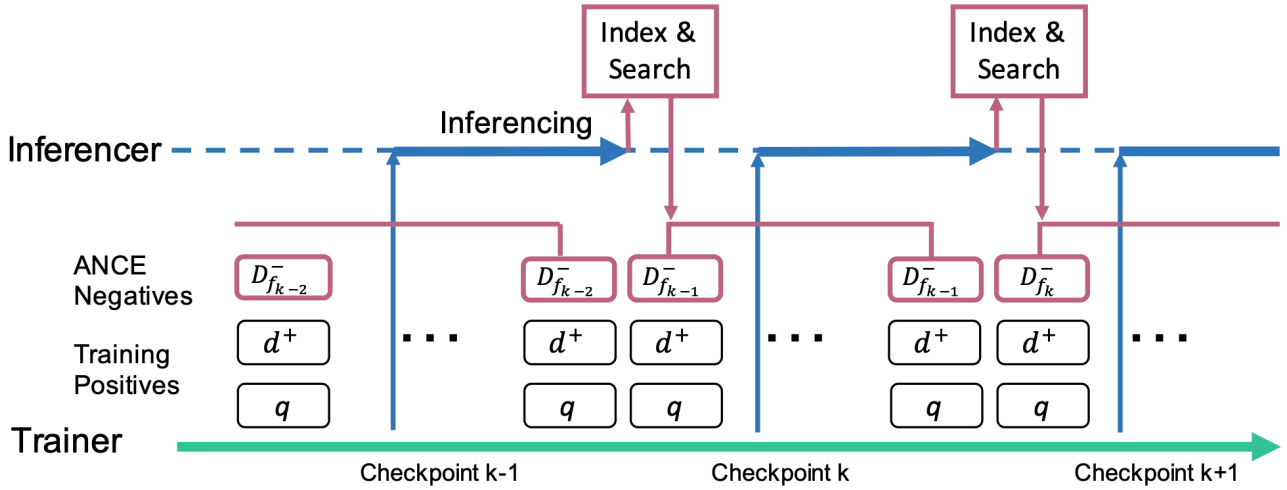


# ANCE: Approximate Negative Contrastive Learning

- Fully learnable representation
- Support for (approximate) NN search
- Problem: negative examples are usually drawn from a (classical) first stage
- How do we generate global negatives?
- Use the model learned so far → dynamic hard negatives



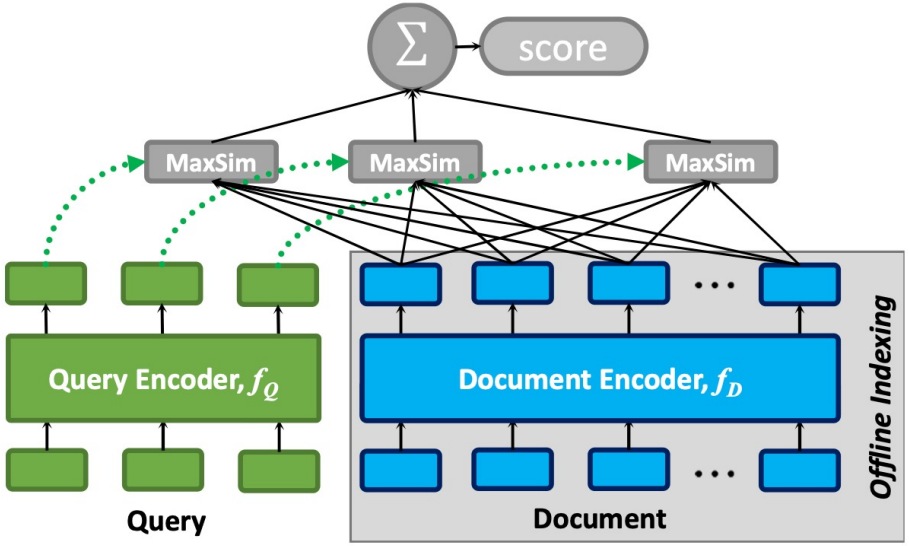

Images from: https://arxiv.org/abs/2007.00808

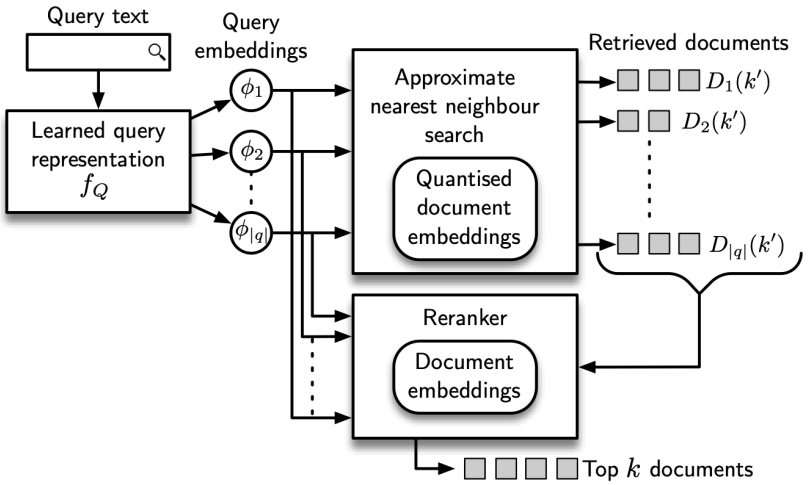
# Exact NN Dense Retrieval (DPR, ANCE)

| | MARCO Dev Passage Retrieval | | TREC DL Passage NDCG@10 | |
|---|---|---|---|---|
| | MRR@10 | Recall@1k | Rerank | Retrieval |
| **Sparse & Cascade IR** | | | | |
| BM25 | 0.240 | 0.814 | – | 0.506 |
| Best DeepCT | 0.243 | n.a. | – | n.a. |
| Best TREC Trad Retrieval | 0.240 | n.a. | – | 0.554 |
| BERT Reranker | – | – | **0.742** | – |
| **Dense Retrieval** | | | | |
| Rand Neg | 0.261 | 0.949 | 0.605 | 0.552 |
| NCE Neg | 0.256 | 0.943 | 0.602 | 0.539 |
| BM25 Neg | 0.299 | 0.928 | 0.664 | 0.591 |
| DPR (BM25 + Rand Neg) | 0.311 | 0.952 | 0.653 | 0.600 |
| BM25 → Rand | 0.280 | 0.948 | 0.609 | 0.576 |
| BM25 → NCE Neg | 0.279 | 0.942 | 0.608 | 0.571 |
| BM25 → BM25 + Rand | 0.306 | 0.939 | 0.648 | 0.591 |
| ANCE (FirstP) | **0.330** | **0.959** | 0.677 | **0.648** |

Images from: https://arxiv.org/abs/2007.00808

# ColBERT: Contextualised Late Interaction BERT



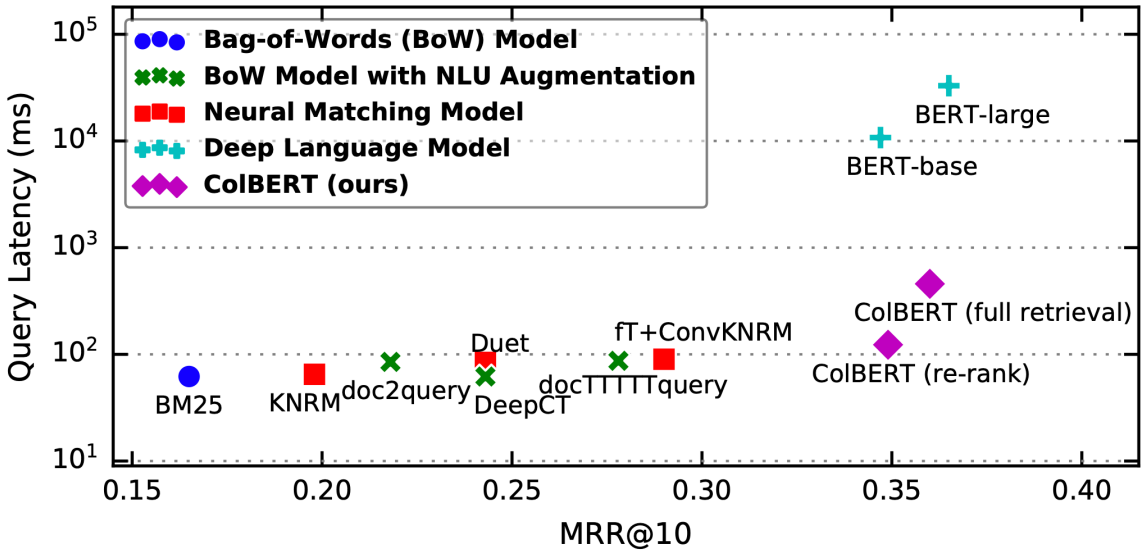




```python
def end_to_end(self) -> TransformerBase:
    """
    Returns a transformer composition that uses a ColBERT FAISS index to retrieve documents, followed by a ColBERT index
    to perform accurate scoring of the retrieved documents. Equivalent to `colbertfactory.set_retrieve() >> colbertfactory.index_scorer()`.
    """
    #input: qid, query,
    #output: qid, query, docno, score
    return self.set_retrieve() >> self.index_scorer(query_encoded=True)
```

Images from: https://arxiv.org/abs/2004.12832

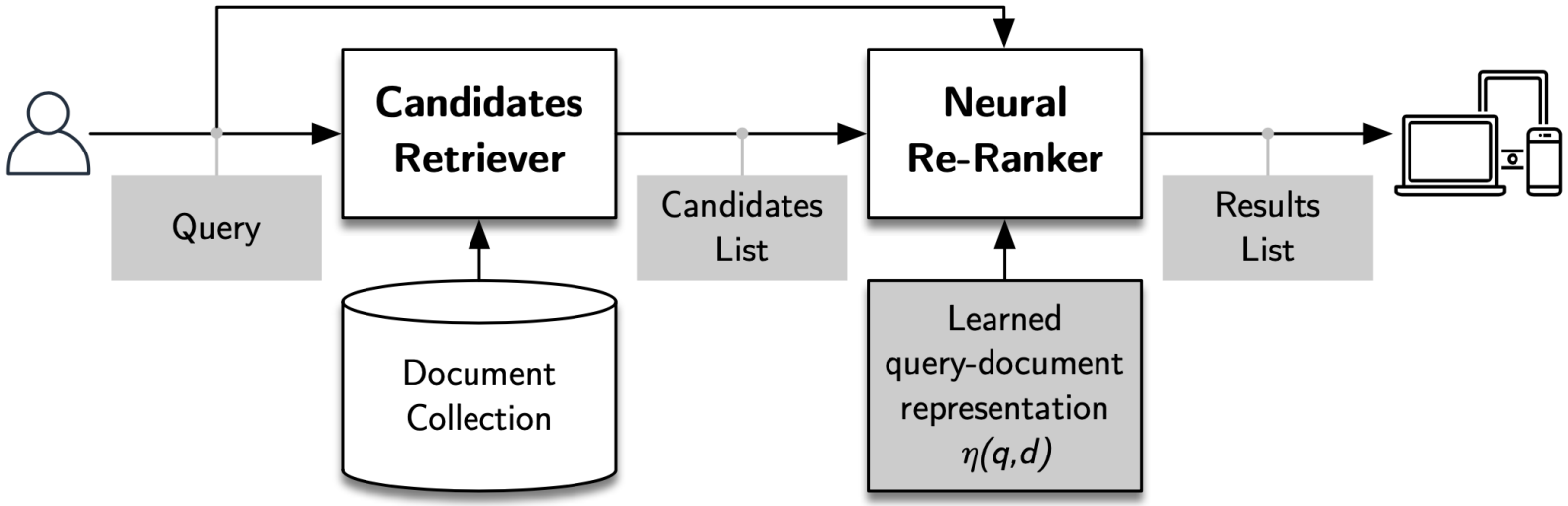
# Approximate NN Dense Retrieval ColBERT

| Method | MRR@10 (Dev) | MRR@10 (Local Eval) | Latency (ms) | Recall@50 | Recall@200 | Recall@1000 |
|---|---|---|---|---|---|---|
| BM25 (official) | 16.7 | - | - | - | - | 81.4 |
| BM25 (Anserini) | 18.7 | 19.5 | 62 | 59.2 | 73.8 | 85.7 |
| doc2query | 21.5 | 22.8 | 85 | 64.4 | 77.9 | 89.1 |
| DeepCT | 24.3 | - | 62 *(est.)* | 69 [2] | 82 [2] | 91 [2] |
| docTTTTTquery | 27.7 | 28.4 | 87 | 75.6 | 86.9 | 94.7 |
| ColBERT$_{L2}$ (re-rank) | 34.8 | 36.4 | - | 75.3 | 80.5 | 81.4 |
| ColBERT$_{L2}$ (end-to-end) | 36.0 | 36.7 | 458 | 82.9 | 92.3 | 96.8 |

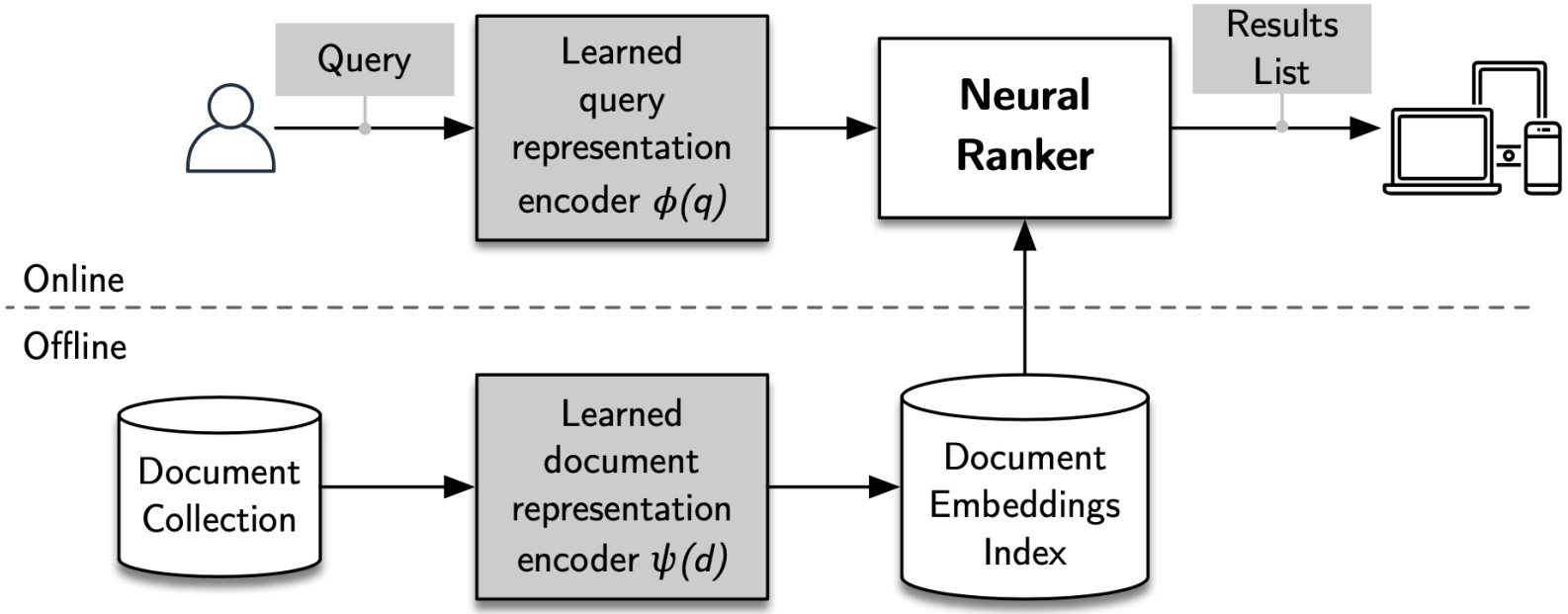



Images from: https://arxiv.org/abs/2004.12832

# Re-ranking Pipeline for Interaction-focused Systems



# Dense Retrieval for Representation-focused Systems




University of Glasgow


UNIVERSITÀ DI PISA

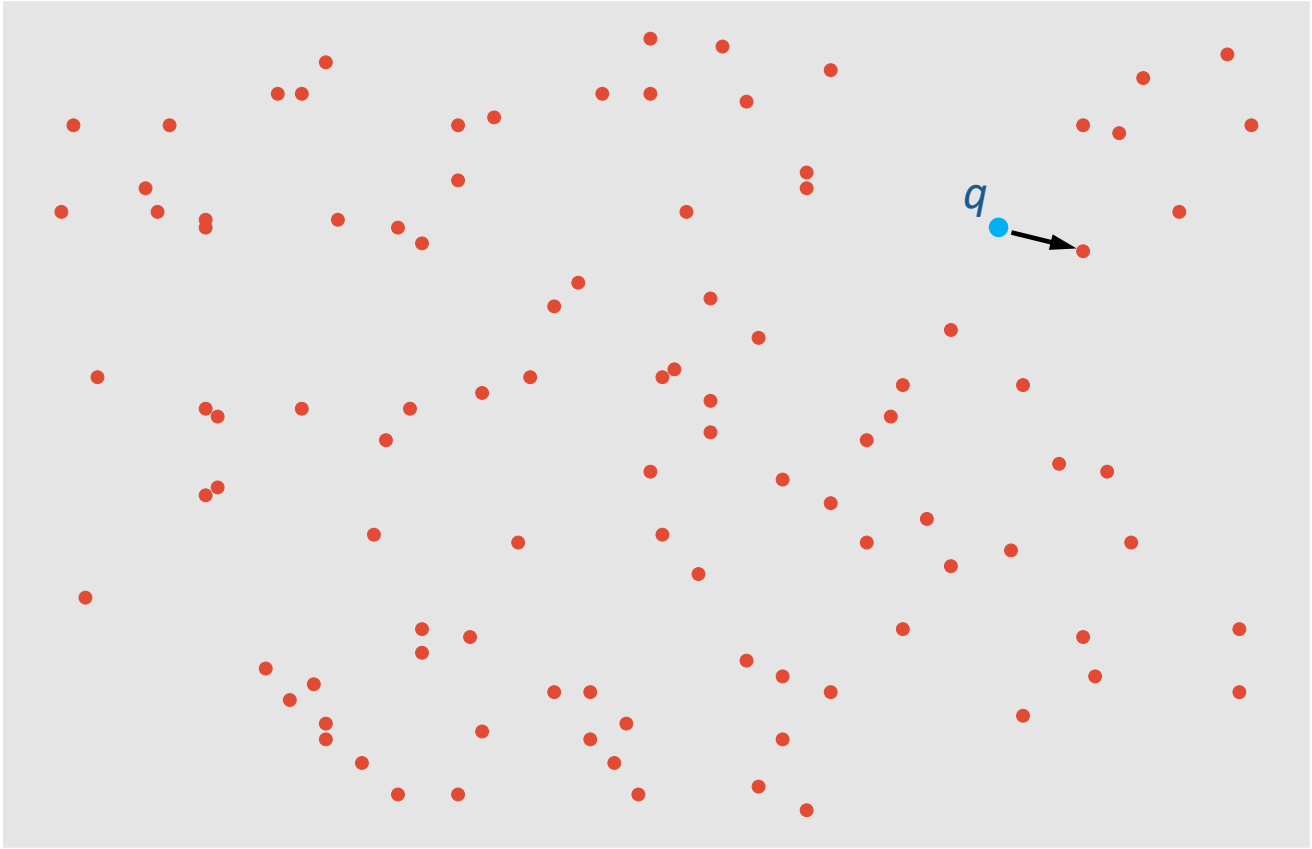

Part 3D

# EMBEDDING INDEXES & VECTOR SEARCH

# Nearest Neighbour

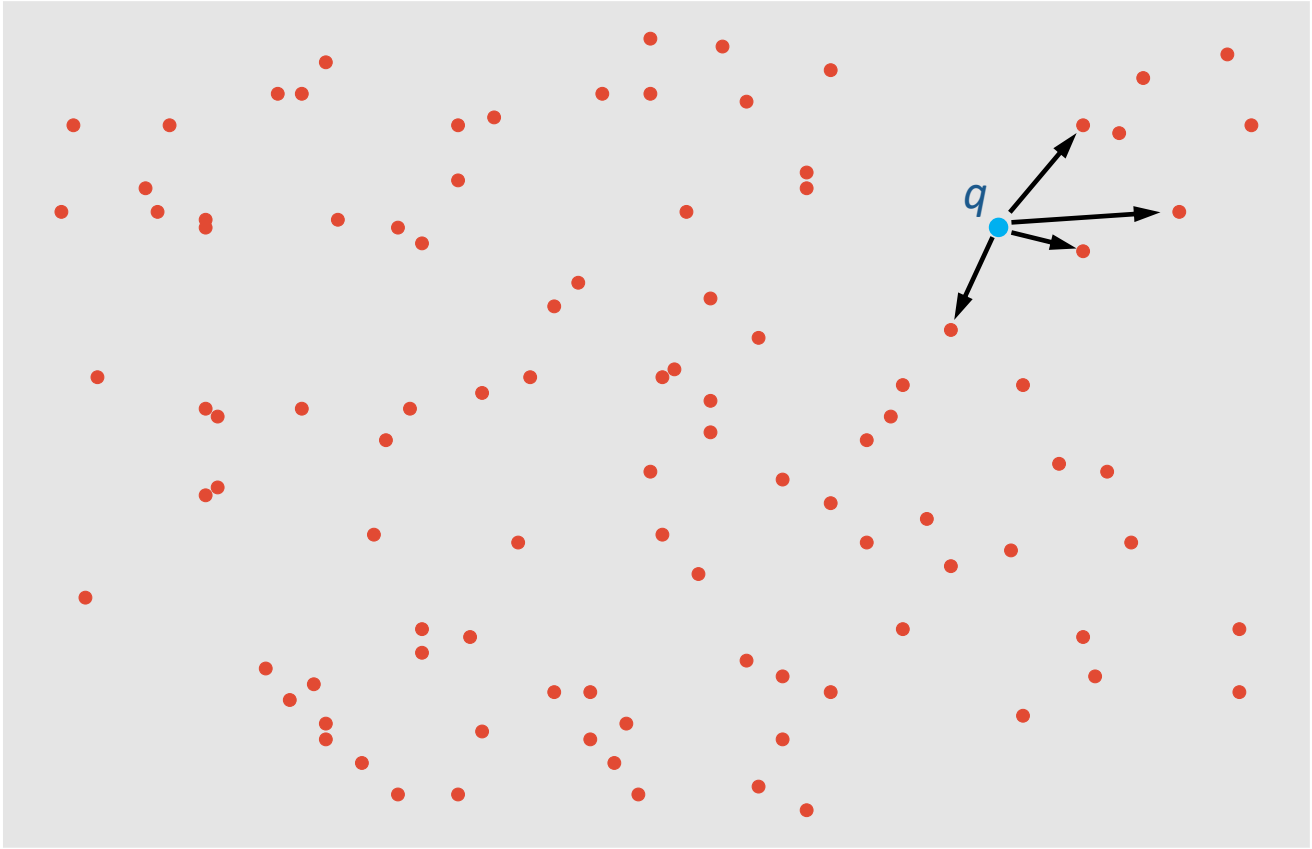
Università di Pisa | University of Glasgow

Complexity O(*n*)



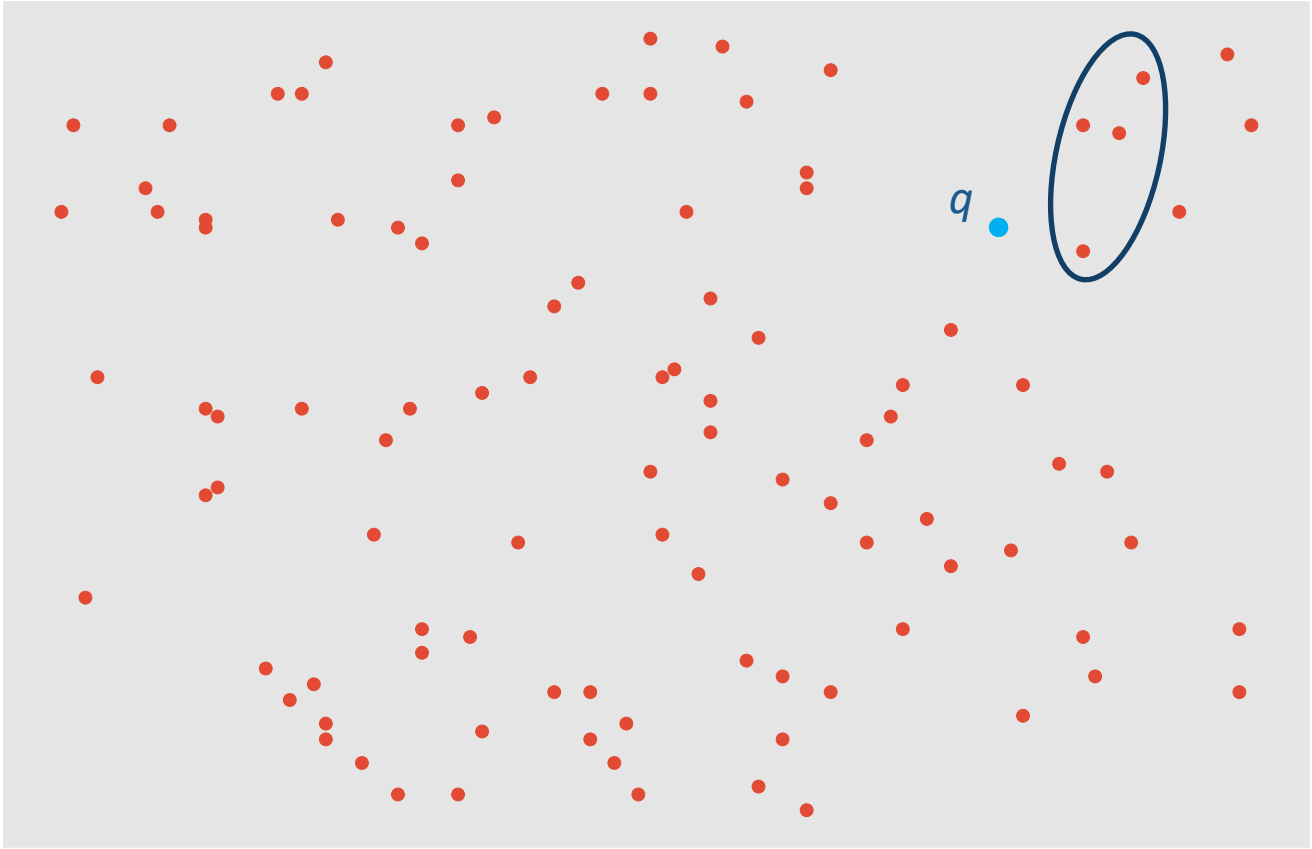
# *k* Nearest Neighbours

Complexity O(*n* log *k*)



# Approximate Nearest Neighbours

- To make it **scalable**, we instead turn to **approximation** techniques
- **Avoid computing** the distance to every reference vector
- Return points **close** to the nearest neighbors
- Non-Exhaustive
- Common accuracy metric: recall@k



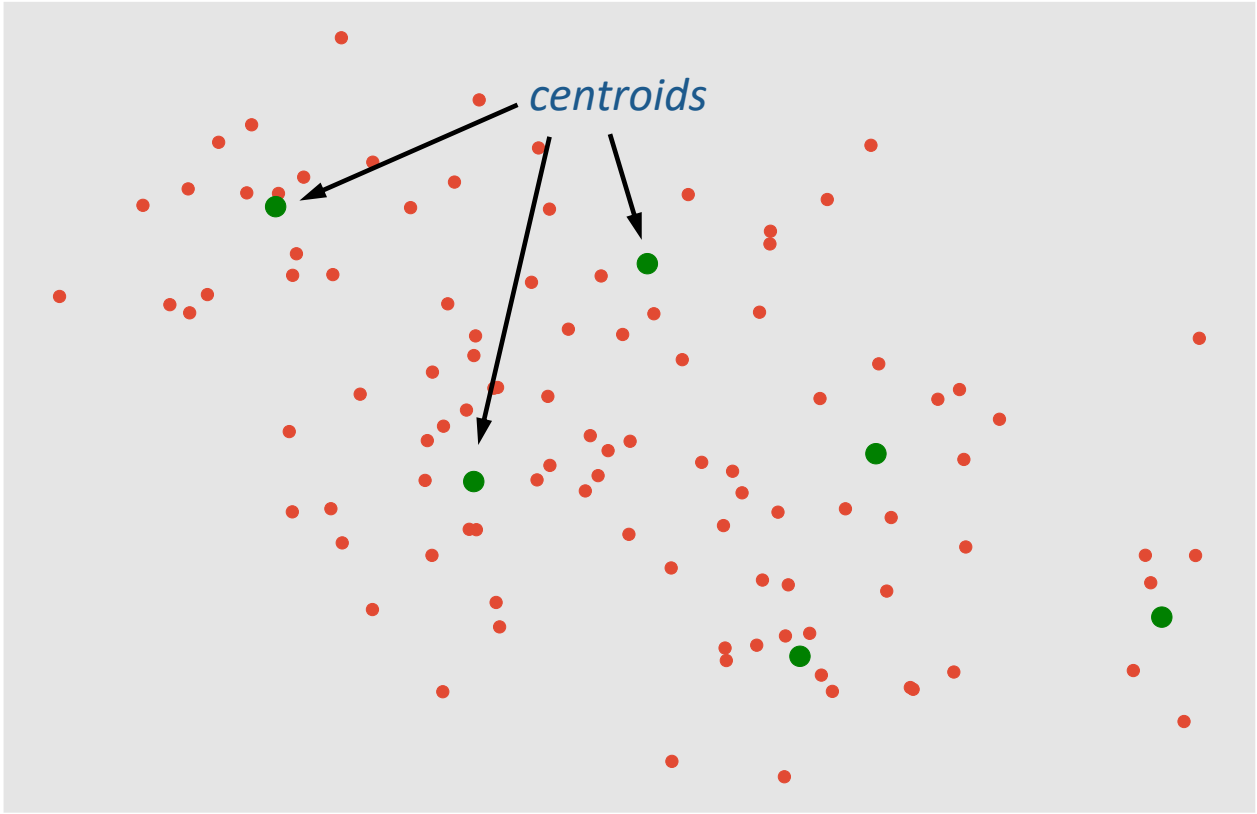
# ANN Strategies

- Trees and forests
- Locally Sensitive Hashing
- Quantization
- Neighborhood graphs

- Most strategies address the curse of dimensionality problem

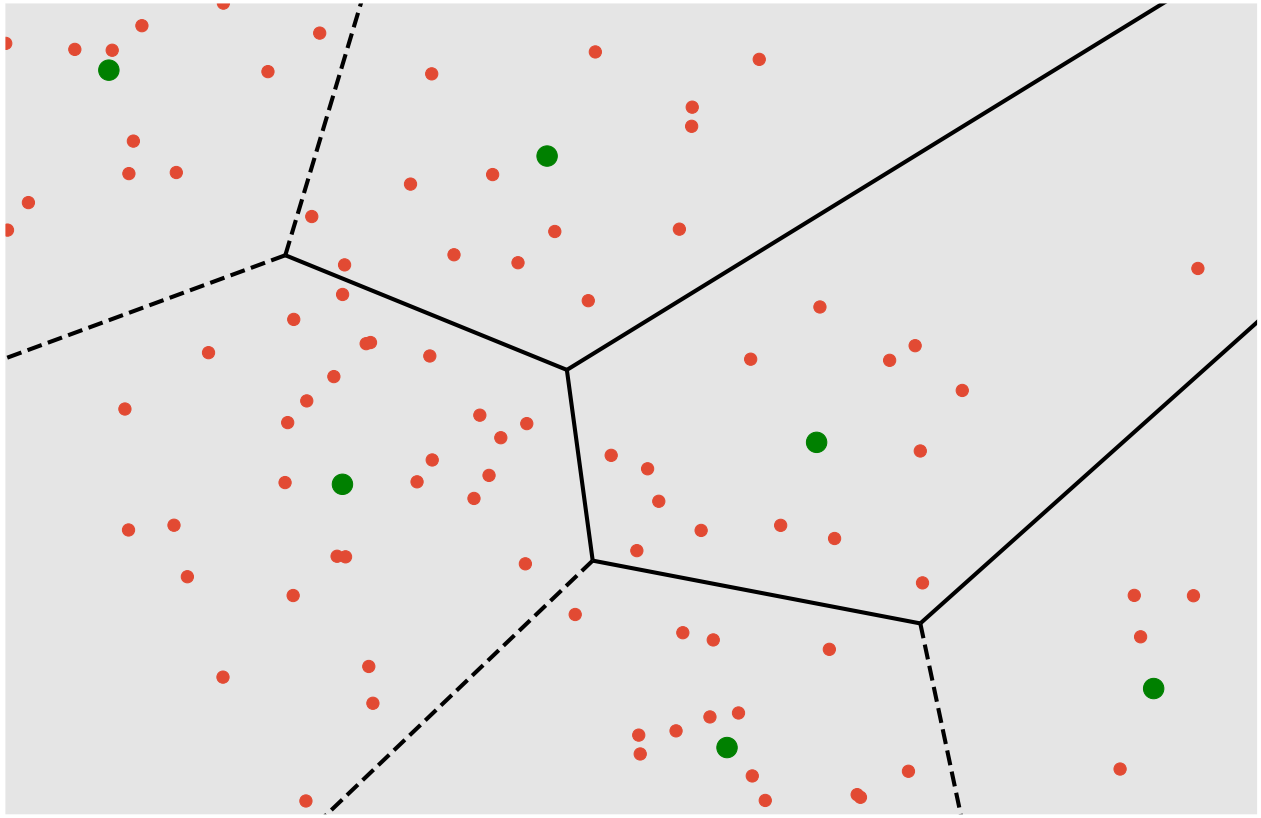
# Quantization (I)

From *n* points to *k* centroids

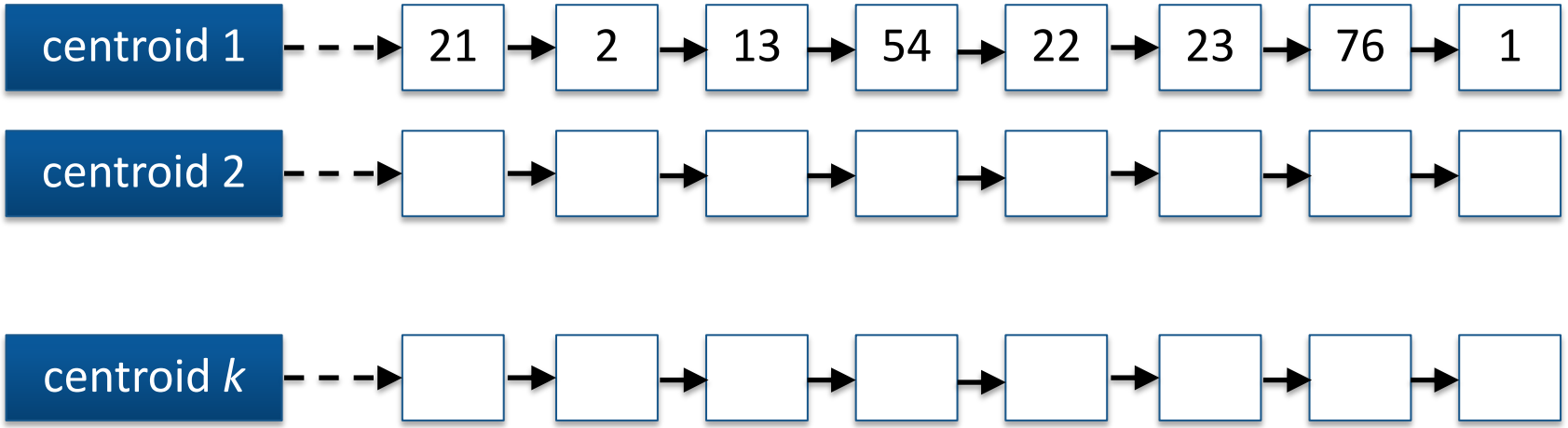


# Quantization (II)

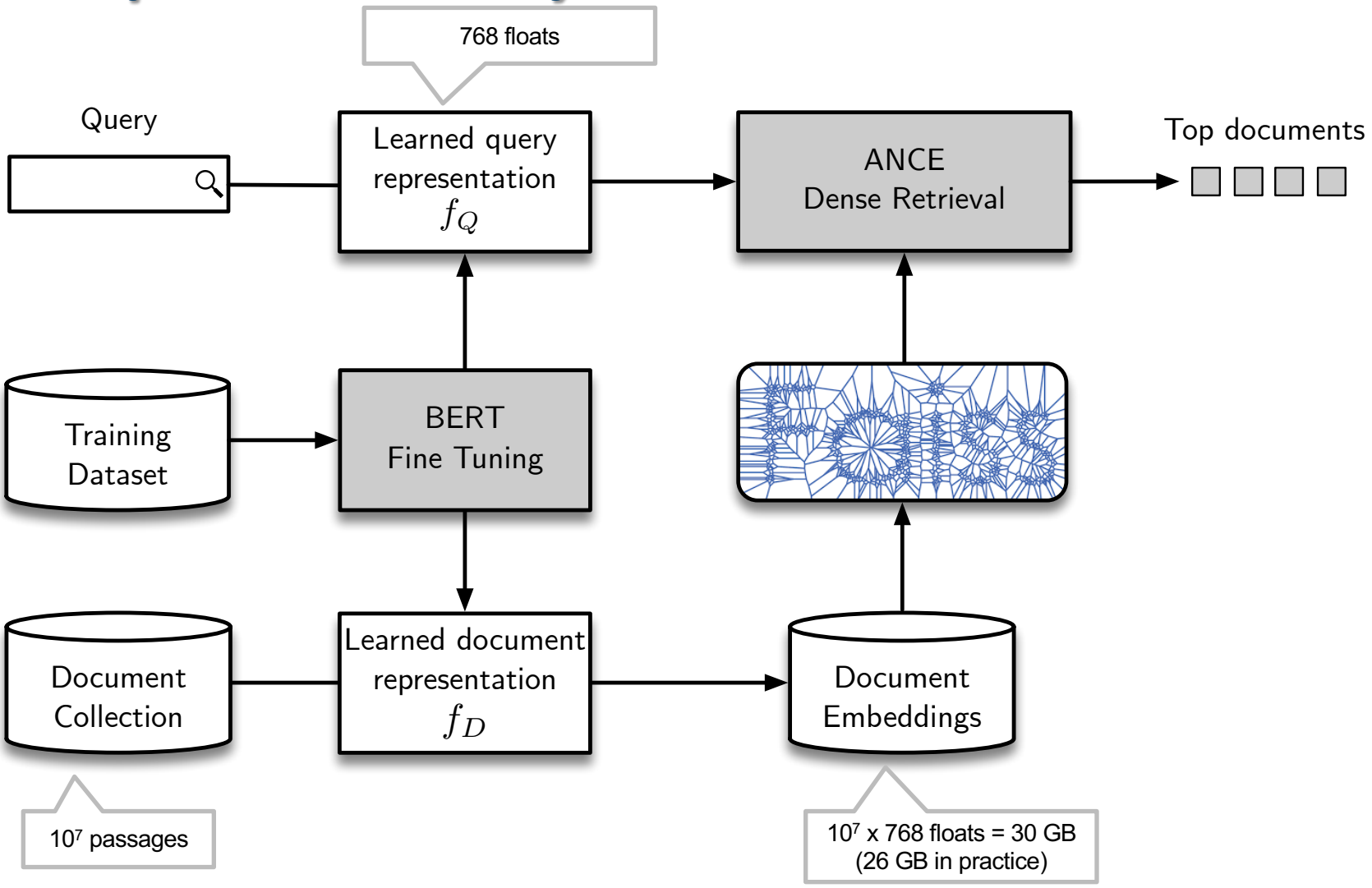
Use k-Means to select centroids



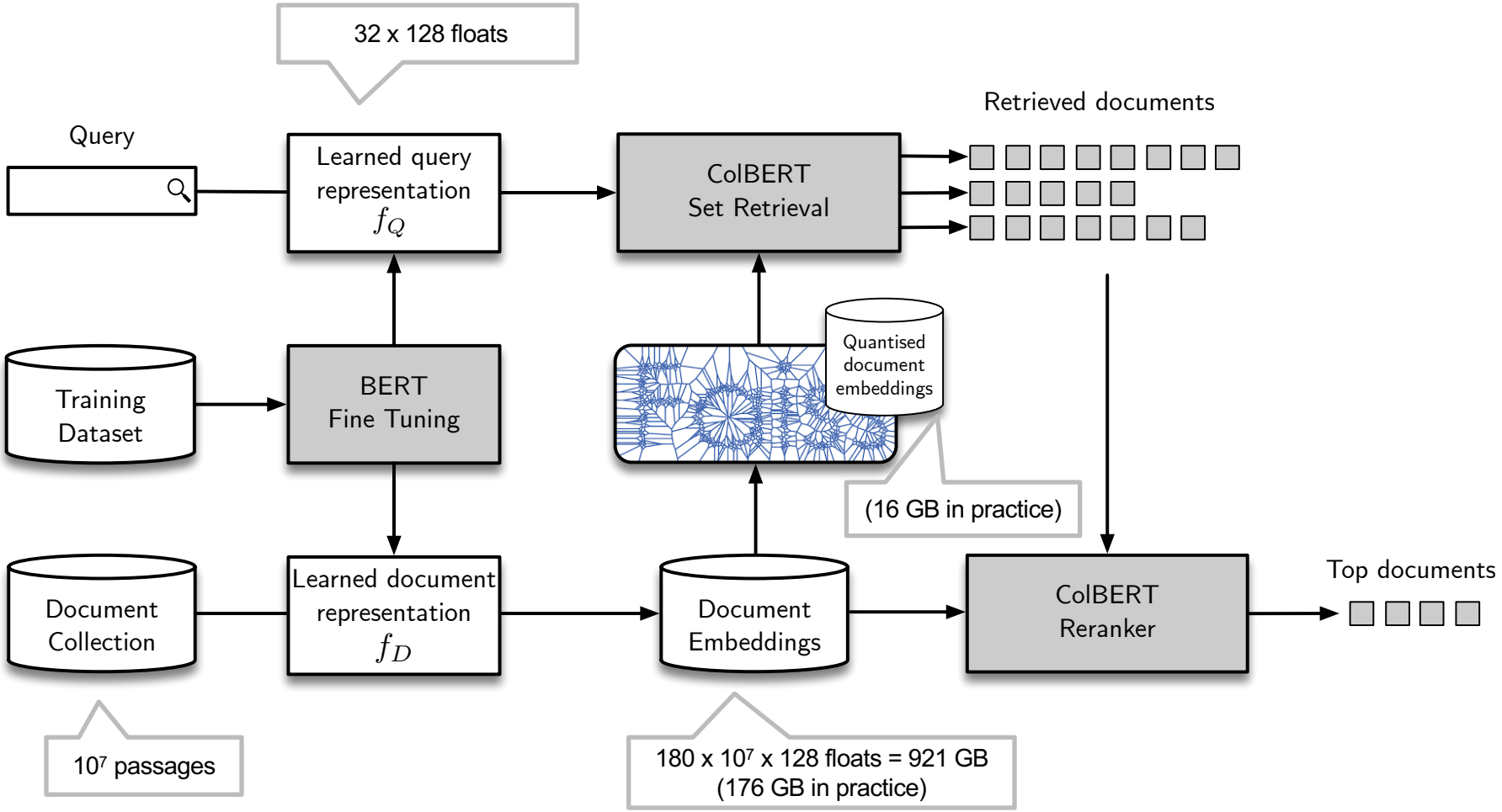
# Quantization (III)



# Dense Retrieval for Multiple Representation Systems



# Approximate NN Dense Retrieval (ColBERT)



# ColBERT PRF

- ColBERT's token embeddings can be used to
  - select representative embeddings in a set of feedback documents using clustering
  - select discriminative embeddings among the representative embeddings using corresponding token's IDF
  - select the top tokens corresponding to discriminative embeddings for query expansions
- This approach works for both ranking and re-ranking scenarios

```
dense_e2e = pytcolbert.set_retrieve() >> pytcolbert.index_scorer(query_encoded=True, add_ranks=True)
if rerank:
    prf_pipe = (
        dense_e2e
        >> ColbertPRF(pytcolbert, k=k, fb_docs=fb_docs, fb_embs=fb_embs, beta=beta, return_docs=True)
        >> (pytcolbert.index_scorer(query_encoded=True, add_ranks=True) %1000)
    )
else:
    prf_pipe = (
        dense_e2e
        >> ColbertPRF(pytcolbert, k=k, fb_docs=fb_docs, fb_embs=fb_embs, beta=beta, return_docs=False)
        >> pytcolbert.set_retrieve(query_encoded=True)
        >> (pytcolbert.index_scorer(query_encoded=True, add_ranks=True) % 1000)
    )
return prf_pipe
```

*Wang et al. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. ICTIR 2021*

# ColBERT Optimisations

- **Single** (ANCE) **vs Multiple** (ColBERT) **Representations**
  - ANCE is more efficient than ColBERT
  - ColBERT is more effective than ANCE for MAP and MRR@10
  - ColBERT obtains better improvements than ANCE for queries that are the hardest for BM25, as well as for definitional queries, and those with complex information needs

*Macdonald, Tonellotto, Ounis. On Single and Multiple Representations in Dense Passage Retrieval. IIR 2021*

- **Embedding Pruning**
  - ColBERT does not need all 32 query embeddings, just 3
  - Same effectiveness, 2.65x speedup in efficiency

*Tonellotto & Macdonald. Query Embedding Pruning for Dense Retrieval. CIKM 2021*

- **Approximate Scoring**
  - ColBERT can use approximate scores from ANN to decrease the number of candidate documents being fully scored, e.g. 200 documents only
  - Same effectiveness, 2x speedup in efficiency

*Macdonald & Tonellotto. On Approximate Nearest Neighbour Selection for Multi-Stage Dense Retrieval. CIKM 2021*

# From passages to documents

Università di Pisa | University of Glasgow

## ANCE and ColBERT are designed for passages

- e.g. BERT has limited number of tokens for inference

## For long documents, we need to break into passages

## PyTerrier's indexing pipelines (c.f. part 2) offer a solution

```
import pyterrier_ance

ance_indexer = pt.text.sliding() >> pyterrier_ance.ANCEIndexer(
    checkpoint_path="./ance_model_checkpoint",
    index_path="/content/anceindex",
    num_docs=192509,
    text_attr='abstract' # COVID
)

ance_indexer.index(dataset.get_corpus_iter())
```

```
from pyterrier_ance import ANCERetrieval

ance_retriever = ANCERetrieval.from_dataset('trec-covid', 'ance_msmarco_psg') >> pt.text.max_passage()
```

# ANCE Example

- Indexing

```
index = pt.datasets.get_dataset('trec-covid').get_index('terrier_stemmed')
ance_index = pt.datasets.get_dataset('trec-covid').get_index('ance_msmarco_psg')
```

- Retrieval

```
bm25_retriever = pt.BatchRetrieve(index, wmodel="BM25")
ance_retriever = ANCERetrieval.from_dataset('trec-covid', 'ance_msmarco_psg') >> pt.text.max_passage()
```

- Experiment

```
pt.Experiment(
    [bm25_retriever % 10, ance_retriever % 10],
    topics,
    qrels,
    eval_metrics=["map", "recip_rank", "P_10", "ndcg_cut_10", "mrt"],
    names=['BM25', 'ANCE'],
)
```

# ColBERT Example

- End-to-end Retrieval

```
from pyterrier_colbert.ranking import ColBERTFactory

bm25_terrier_stemmed = pt.BatchRetrieve.from_dataset('vaswani', 'terrier_stemmed', wmodel='BM25')

factory = ColBERTFactory.from_dataset('vaswani', 'colbert_uog44k')
colbert_e2e = factory.end_to_end()
```

- Experiment

```
pt.Experiment(
    [bm25_terrier_stemmed % 10, colbert_e2e % 10],
    topics,
    qrels,
    eval_metrics=["map", "recip_rank", "P_10", "ndcg_cut_10", "mrt"],
    names=['BM25', 'ColBERT'],
)
```

# Round Up

- **Index augmentation** can bring benefits from deep learning to traditional indices
  - Emphasize important terms, predict other terms that could match
- We can **use neural models** to directly learn scalar values to score terms in documents and store them in an inverted index
- We can use **dense retrieval systems** to **improve effectiveness** while **reducing the processing time** of queries
  - Represent **queries and documents as** one or more **embeddings** (single vs. multiple representation)
  - Embeddings are store in a system providing **efficient support for NN search**
  - Depending on the size and number of embeddings, NN search can be **exact** or **approximate**

# PyTerrier Github References

- Doc2Query plugin: https://github.com/terrierteam/pyterrier_doc2query
- DeepCT plugin: https://github.com/terrierteam/pyterrier_deepct
- DeepImpact plugin: https://github.com/terrierteam/pyterrier_deepimpact
- SPLADE plugin: https://github.com/cmacdonald/pyt_splade
- ANCE plugin: https://github.com/terrierteam/pyterrier_ance
- ColBERT plugin: https://github.com/terrierteam/pyterrier_colbert

# *Questions*

# Practical Time

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

- https://github.com/terrier-org/searchsolutions2022-tutorial
- Press the 

link for each notebook to start a Colab session

## Timings

- Practical: 15:45–16:15 GMT

# *Acknowledgements*

**The authors gratefully acknowledge other users & contributors to PyTerrier**

**Iadh Ounis and Xiao Wang provided input to this tutorial**