Rediscover your keyword search: Expand, Enrich and Rewrite

Praveen Mohan Prasad

Analytics Specialist TAM
AWS

Hajer Bouafif

Analytics Solutions Architect
AWS

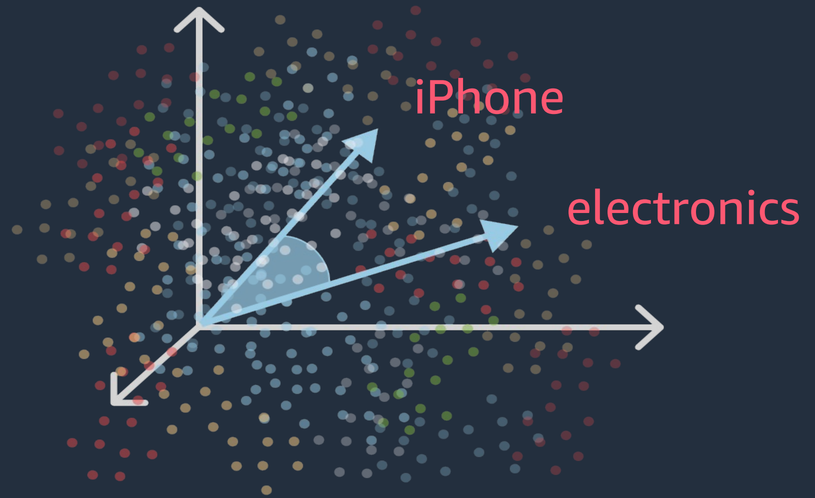
Information Retrieval

Sparse Lexical Search

	cat	dog	is	it	my	not	old	wolf
It is dog	0	0.25	0.22	0.25	0	0		0
my cat is old	0.3	0	0.22	0	0.3	0	0.3	0
It is not dog, it is wolf	0	0.11	0.19	0.22	0	0.13	0	0.13

Algorithm: TF-IDF, BM25
Statistic: Frequency

Dense Vector Search



Algorithm: kNN, ANN
Measure: Cosine, Euclidean distance

Strengths

Lexical Search

+ Exact Matching
+ Interpretability
+ Less memory and Fast retrieval

Vector Search

+ Context matching
+ Natural language understanding: RAG
+ Multimodal search

Benchmarking on Generalisation

Beir

Benchmarking IR

18 multi-domain datasets

Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. [arXiv preprint arXiv:2104.08663.]()

- *Vector Search poorly generalizes on Out Of Domain (OOD) Data"*
- *Lexical search > Vector Search @ OOD*

Fine-tuning Vector Search is complex!

1. Training dataset
2. Data Science Expertise
3. retrain model and re-indexing

Can we do semantic search with lexical search?

Lexical search: limited semantic capabilities

Document: Exercising regularly makes $\boxed{\text{body}}$ and $\boxed{\text{mind}}$ $\boxed{\text{stronger}}$ and $\boxed{\text{healthy}}$

Query: How to $\boxed{\text{strengthen}}$ the $\boxed{\text{physical}}$ and $\boxed{\text{mental}}$ $\boxed{\text{wellness}}$?

→ ✗ No match

Challenges

Vocabulary Mismatch

Poor semantic understanding

Existing Solutions

- ✓ Synonyms
- ✓ Analyzers and Boosting techniques
- ✓ Learning features from User signals
- ✓ Re-ranking

Lexical Search + Text expansion = Semantic Search

Text expansion : Rewrite + Inject + Boost/Decay

Exercising	regularly	makes	body	and	mind	stronger
Exercise(s)	regular	make	muscle		brain	strong
workout	daily	help	Physique		mentally	Healthy
Training	often	strengthen	Fitness		meditation	Happiness
activity	routine	improve				Wellness
Gym	frequent					

Boost and decay the features

Document 1: "Apple Products are expensive"

```
{'apple': 3.09,
 'expensive': 2.58,
 'apples': 2.31,
 'cost': 2.06,
 'products': 1.78,
 'cheap': 1.61,
 'product': 1.57,
 'price': 1.28,
 'expense': 1.04,
 'best': 0.66,
 'brand': 0.46,
 'stock': 0.44,
 'chip': 0.44,
 'store': 0.25,
 'computer': 0.24,
 'offer': 0.22,
 'money': 0.21,
 'budget': 0.21,
 'good': 0.2,
 'buy': 0.19,
 'affordable': 0.19,
 'popular': 0.16,
 'gift': 0.14,
 'manufacturer': 0.12,
 'purchase': 0.09,
 'iphone': 0.09,
 'happiness': 0.06,
 'steve': 0.03,
 'amazon': 0.03,
 'hardware': 0.01}
```

Query: "apple headphones"

```
{'apple": 3.298149
"gift": 0.22470483,
"##phone": 1.6386933,
"electronics": 0.18978065,
"##phones": 2.0765076
"music": 0.725811,
"dj": 0.36222592,
"sound": 0.77559084}
```

$score(D1, Q)$= 7.4521604 > $score(D2, Q)$= 5.5605326

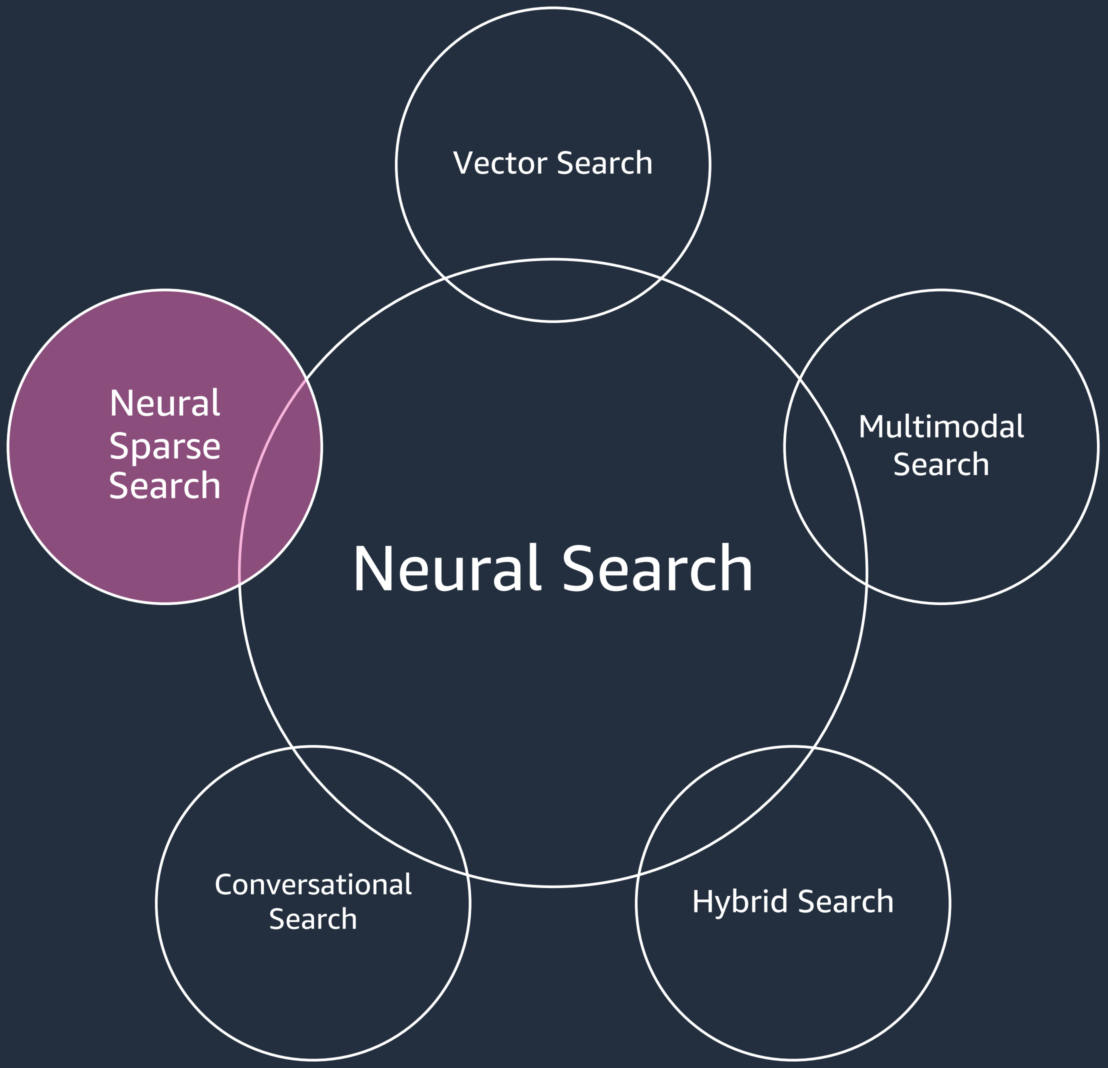
Scoring:

$$score(D, Q) = \sum w_j . w_i$$

Document 2: "An apple a day keeps doctor away"

```
{'apple': 2.45,
 'doctor': 2.26,
 'day': 2.08,
 'away': 2.05,
 'apples': 1.87,
 'doctors': 1.86,
 'daily': 1.83,
 'keep': 1.59,
 'medical': 1.19,
 'prevent': 0.97,
 'every': 0.95,
 'a': 0.94,
 'fruit': 0.91,
 'dr': 0.7,
 'keeping': 0.62,
 'help': 0.56,
 'stay': 0.46,
 'an': 0.42,
 'remove': 0.41,
 'remedy': 0.37,
 'drink': 0.35,
 'one': 0.31,
 'candy': 0.31,
 'pill': 0.31,
 'keeps': 0.26,
 'diet': 0.24,
 'eat': 0.23,
```

Neural Sparse Search
2.11



Open-source Sparse encoding models

DOCUMENT AND SEARCH QUERY SPARSE ENCODING

Sparse encoding model

opensearch-project/opensearch-neural-sparse-encoding-v1

naver/splade-v3

DOCUMENT ONLY SPARSE ENCODING

Sparse encoding model

opensearch-project/opensearch-neural-sparse-encoding-doc-v1

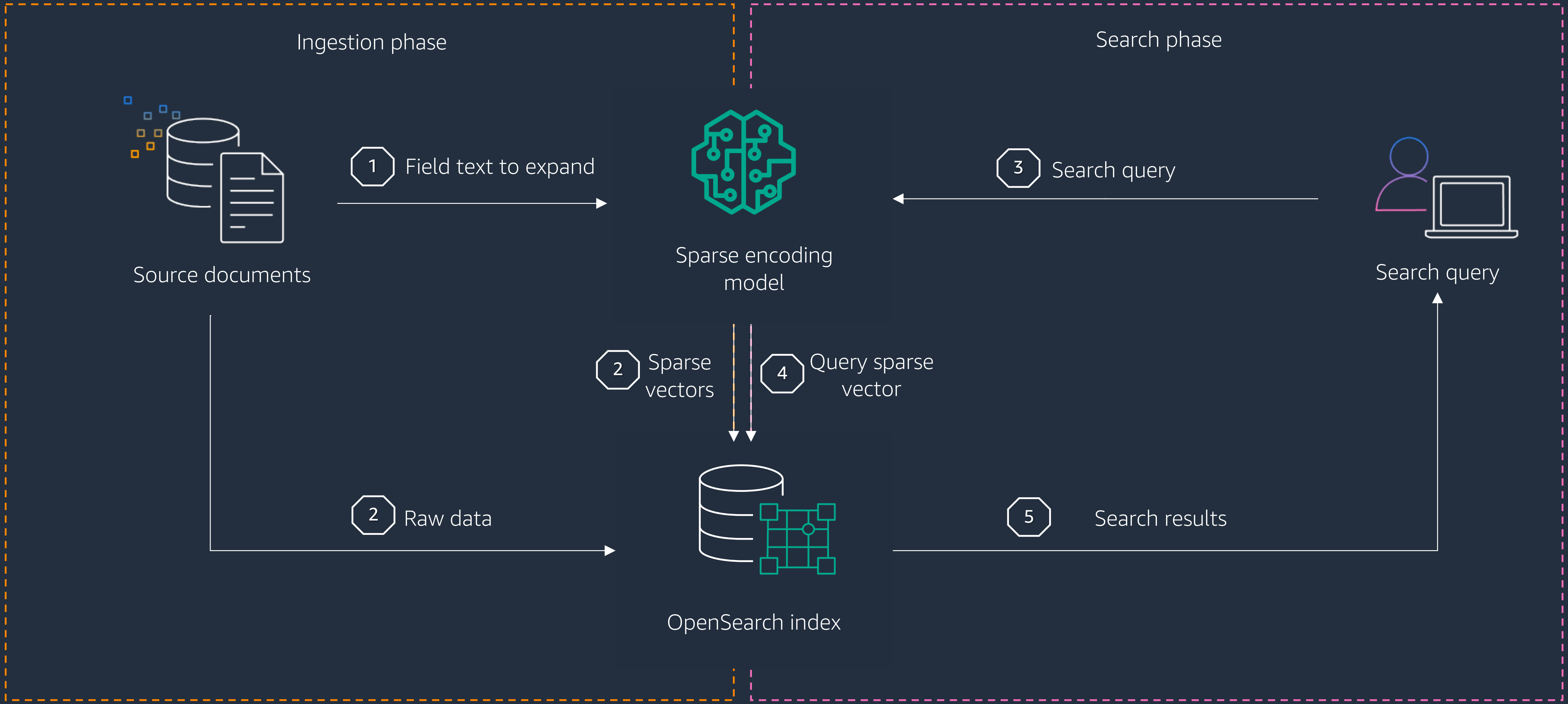
naver/splade-v3-doc

Tokenizer

amazon/neural-sparse/opensearch-neural-sparse-tokenizer-v1

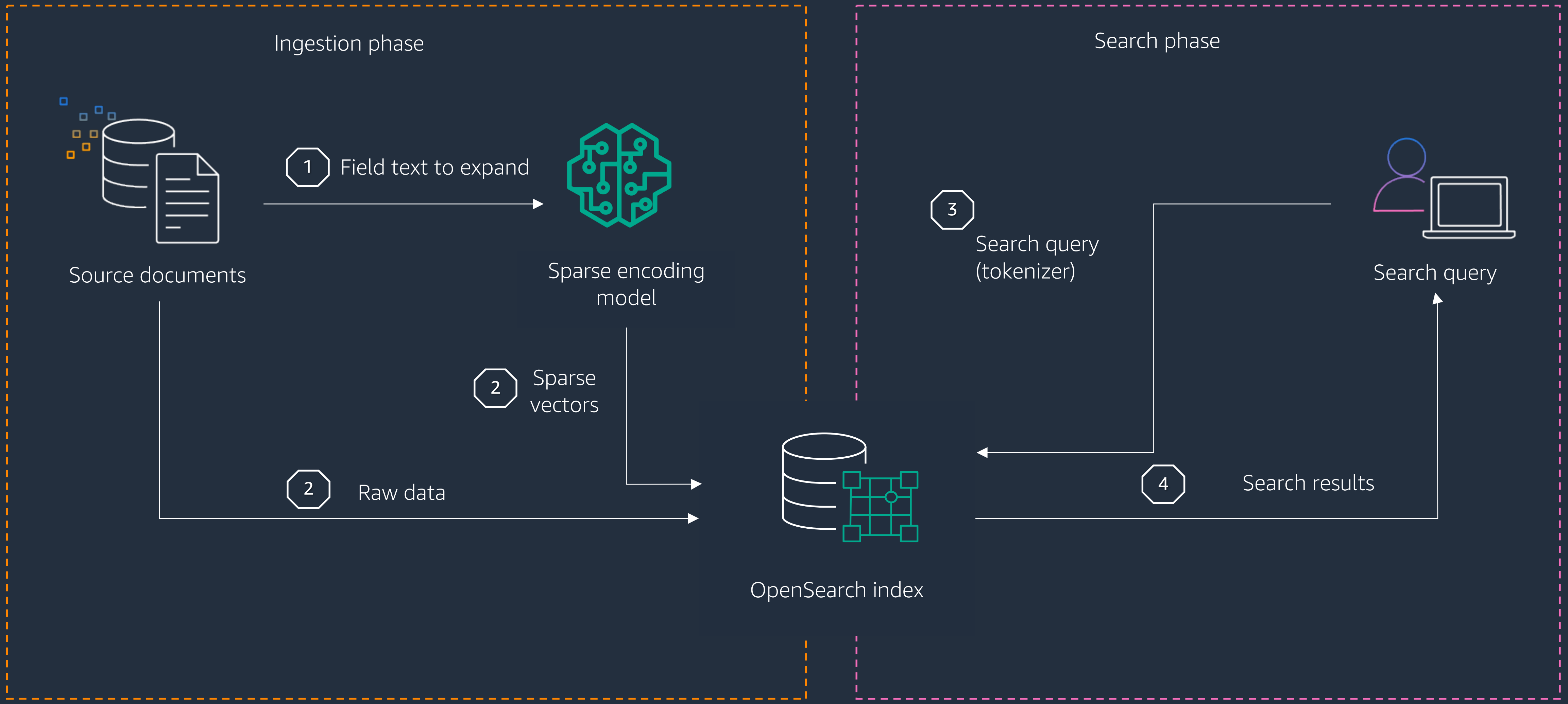
Sparse search – High level Search architecture

DOCUMENT AND SEARCH QUERY SPARSE ENCODING

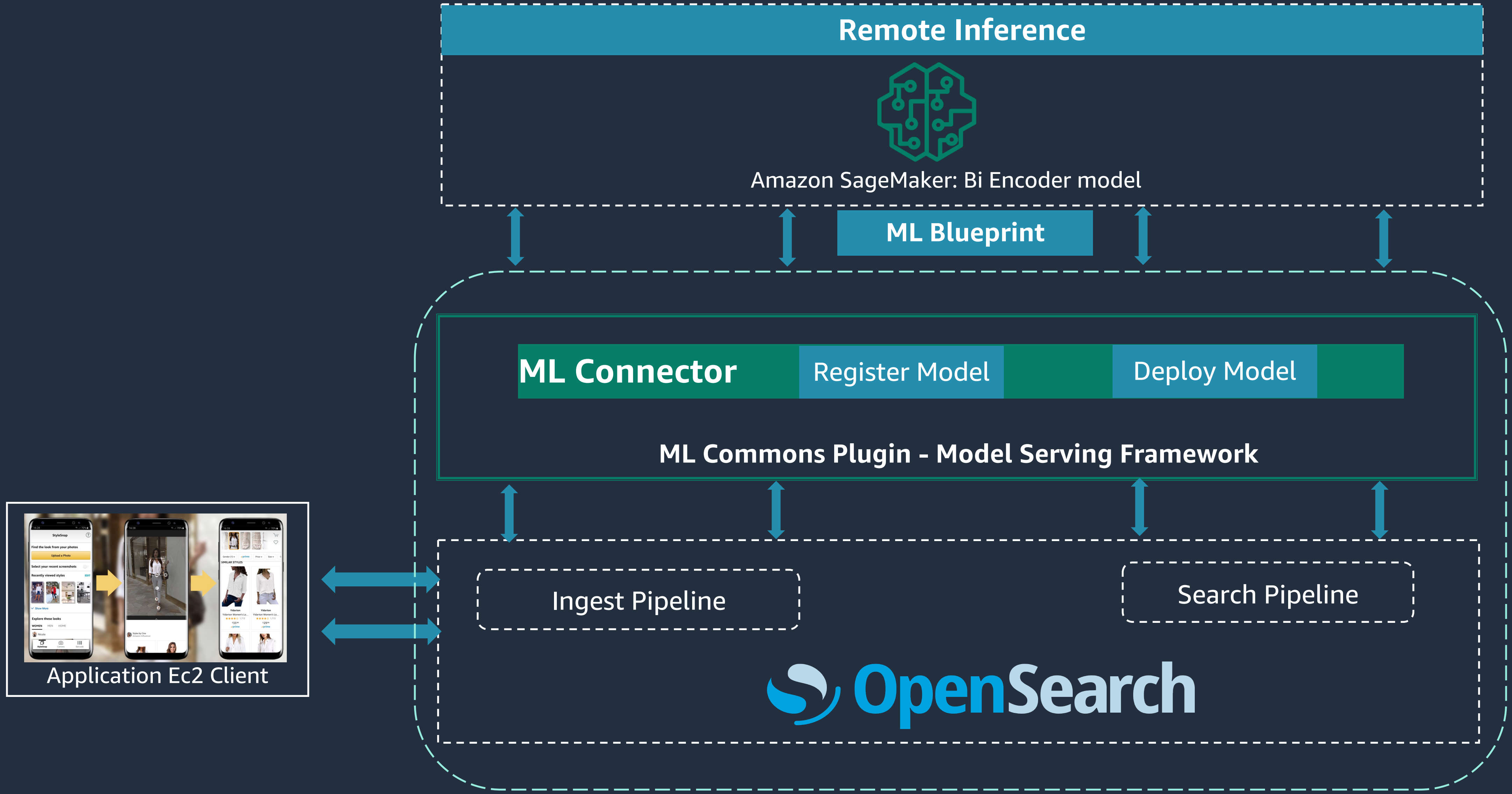


Sparse search – High level Search architecture

DOCUMENT ONLY SPARSE ENCODING



Neural Sparse Search with OpenSearch



Build Neural Sparse Search with OpenSearch

Create sparse ingest pipeline

```
PUT /ingest/pipeline/sparse-embedding-pipeline
{
  "description": "A sparse encoding ingest
pipeline",
  "processors": [
    {
      "sparse_encoding": {
        "model_id": "4ynZg4wB33bA6yQYW—",
        "field_map": {
          "caption": "caption_sv",
          "description": "desc_sv"
        }
      }
    }
  ]
}
```

Build the sparse index

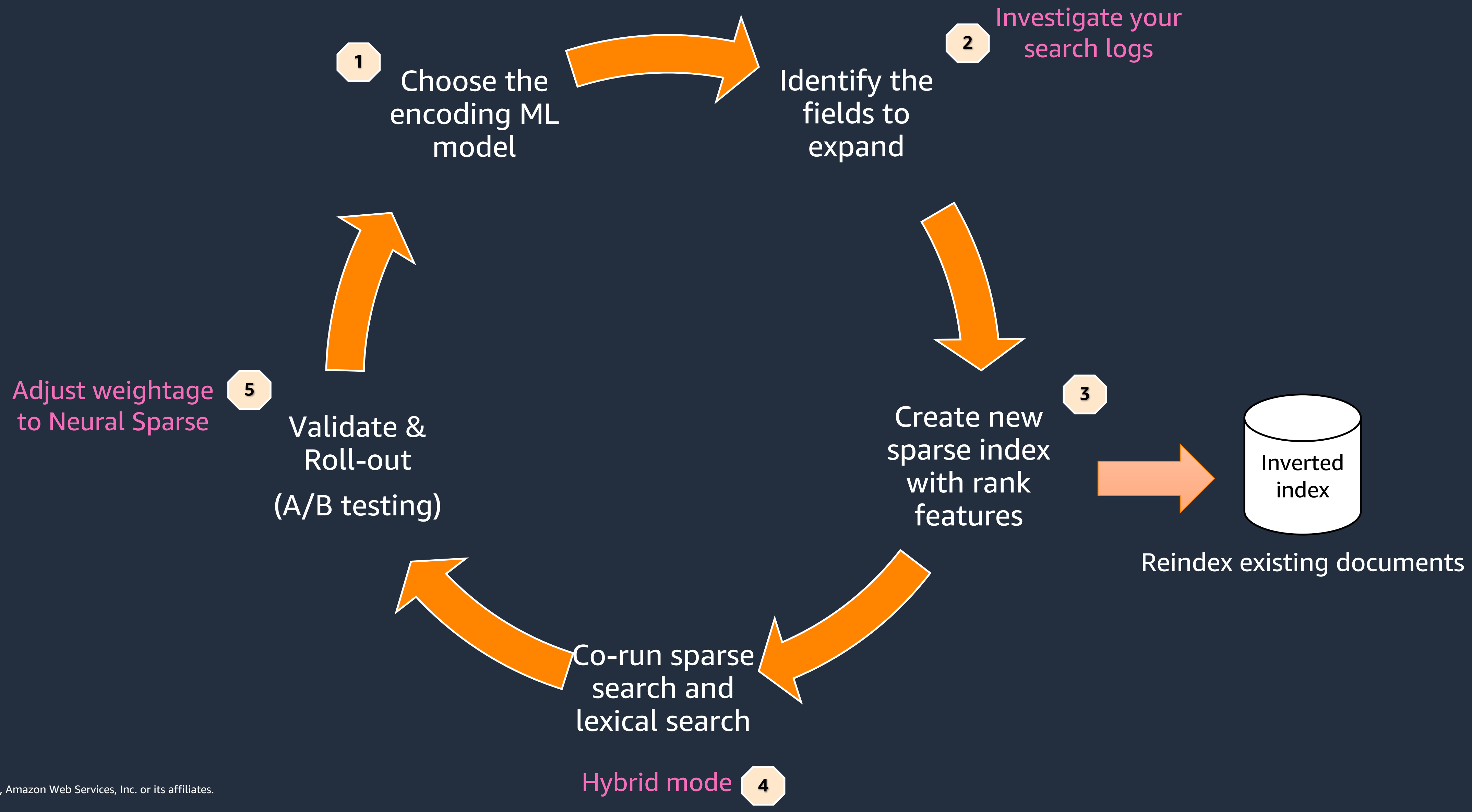
```
PUT /retail-sparse-index
{
  "settings": {
    "default_pipeline": "sparse-embedding-pipeline"
  },
  "mappings": {
    "properties": {
      "id": {
        "type": "text"
      },
      "caption": {
        "type": "text"
      },
      "caption_sv": {
        "type": "rank_features"
      },
      "description": {
        "type": "text"
      },
      "desc_sv": {
        "type": "rank_features"
      }
    }
  }
}
```

Run neural sparse search

```
GET retail-sparse-index/_search
{
  "query": {
    "neural_sparse": {
      "desc_sv": {
        "query_text": "pink backpack",
        "model_id": "4ynZg4wB33bA6yQYW—"
      }
    }
  }
}
```

Demo

Path to Neural Sparse search in OpenSearch



Neural Sparse Search Benchmarks

	BM25		Dense (with TAS-B model)		Hybrid (Dense + BM25)		Neural sparse search bi-encoder		Neural sparse search document-only	
Dataset	**NDCG**	**Rank**	**NDCG**	**Rank**	**NDCG**	**Rank**	**NDCG**	**Rank**	**NDCG**	**Rank**
Trec-Covid	0.688	4	0.481	5	0.698	3	0.771	1	0.707	2
NFCorpus	0.327	4	0.319	5	0.335	3	0.36	1	0.352	2
NQ	0.326	5	0.463	3	0.418	4	0.553	1	0.521	2
HotpotQA	0.602	4	0.579	5	0.636	3	0.697	1	0.677	2
FiQA	0.254	5	0.3	4	0.322	3	0.376	1	0.344	2
ArguAna	0.472	2	0.427	4	0.378	5	0.508	1	0.461	3
Touche	0.347	1	0.162	5	0.313	2	0.278	4	0.294	3
DBPedia	0.287	5	0.383	4	0.387	3	0.447	1	0.412	2
SciDocs	0.165	2	0.149	5	0.174	1	0.164	3	0.154	4
FEVER	0.649	5	0.697	4	0.77	2	0.821	1	0.743	3
Climate FEVER	0.186	5	0.228	3	0.251	2	0.263	1	0.202	4
SciFact	0.69	3	0.643	5	0.672	4	0.723	1	0.716	2
Quora	0.789	4	0.835	3	0.864	1	0.856	2	0.788	5
Amazon ESCI	0.081	3	0.071	5	0.086	2	0.077	4	0.095	1
Average	0.419	3.71	0.41	4.29	0.45	2.71	0.492	1.64	0.462	2.64

NDCG(Neural Sparse) > NDCG(lexical, vector)
By atleast 5 Points

No Fine-tuning of models

https://opensearch.org/blog/improving-document-retrieval-with-sparse-semantic-encoders/

Latency and Memory

Latency	BM25	Dense (with TAS-B model)	Neural sparse search bi-encoder	Neural sparse search document-only
P50 latency (ms)	8 ms	56.6 ms	176.3 ms	10.2ms
P90 latency (ms)	12.4 ms	71.12 ms	267.3 ms	15.2ms
P99 Latency (ms)	18.9 ms	86.8 ms	383.5 ms	22ms
Max throughput (op/s)	2215.8 op/s	318.5 op/s	107.4 op/s	1797.9 op/s
Mean throughput (op/s)	2214.6 op/s	298.2 op/s	106.3 op/s	1790.2 op/s

Memory	BM25	Dense (with TAS-B model)	Neural sparse search bi-encoder	Neural sparse search document-only
Index size	1 GB	65.4 GB	4.7 GB	6.8 GB
RAM usage	480.74 GB	675.36 GB	480.64 GB	494.25 GB
Runtime RAM delta	+0.01 GB	+53.34 GB	+0.06 GB	+0.03 GB

https://opensearch.org/blog/improving-document-retrieval-with-sparse-semantic-encoders/

Configuration:
3 data nodes – 256 GB memory, 32 vCPU
1 ML node – 384 GB memory, 48 vCPU
20 query clients
MS MARCO Dataset – 8.8M docs

- Sparse(Bi-encoder) > Dense
- Lexical Search ~ Sparse(Doc encoder)

- Dense > Sparse (Bi, doc encoder) by 7.9 %
- Lexical ~ Sparse(Bi, Doc encoder)

Enrich and Re-write

Enrich your documents

Images

{
"description": "Rugged Brown Leather Boots",
"price": "50"
}

Amazon Rekognition
or
Object detection models

{
"description": "Rugged Brown Leather Boots",
"price": "50
"color":"brown",
"category":"Apparel and Accessories",
"objects":"Footwear,Boot,Shoe,Clothing"
}

Text

{
"message": "Bonjour"
}

{
"Text": "Bob ordered two sandwiches and three ice cream cones today from a store in Seattle."
}

Amazon Comprehend
or
NER models

{
"language": "FR",
"message": "Bonjour"
}

{
{"Text": "Bob", "Type": "PERSON" },
{"Text": "two sandwiches", "Type": "QUANTITY" },
{"Text": "three ice cream cones", "Type": "QUANTITY" },
{"Text": "today", "Type": "DATE" },
{"Text": "Seattle", "Type": "LOCATION" }
}

Filters in the Query

Query: "Brown leather shoes for men under 50$"

```
GET /_search
{
  "query": {
    "match": {
      "description": "Brown leather shoes for men"
    }
```

```
GET _search
{
  "query": {
    "query_string": {
      "query": "Brown leather shoes for men under 50"
    }
  }
}
```

```
GET /_search
{
  "query": {
    "bool": {
      "must": [
        { "match":
          { "description":"shoes"}}
      ],
      "should": [
        { "term":  { "color":"brown" }},
        { "term":  { "gender":"male" }},
        { "term":   { "material":"leather"}},
        { "range": { "price":{ "lt": 50 }}}
      ]
    }
  }
}
```

DSL Query re-writing by applying filters

TAME LLM TO EXTRACT FILTERS

Query: Brown shoes for men under $50

Your goal is to structure the user's query to match the request schema provided below . . . .

<<query>>
<<Schema>>
<<examples>>

Structured Query

```
{
    "query": "shoes",

    "filter": "and
        (
            eq("category", "footwear"),
            lt("price", 50),
            eq("gender", "male")
            eq("color", "brown")
        )"
}
```

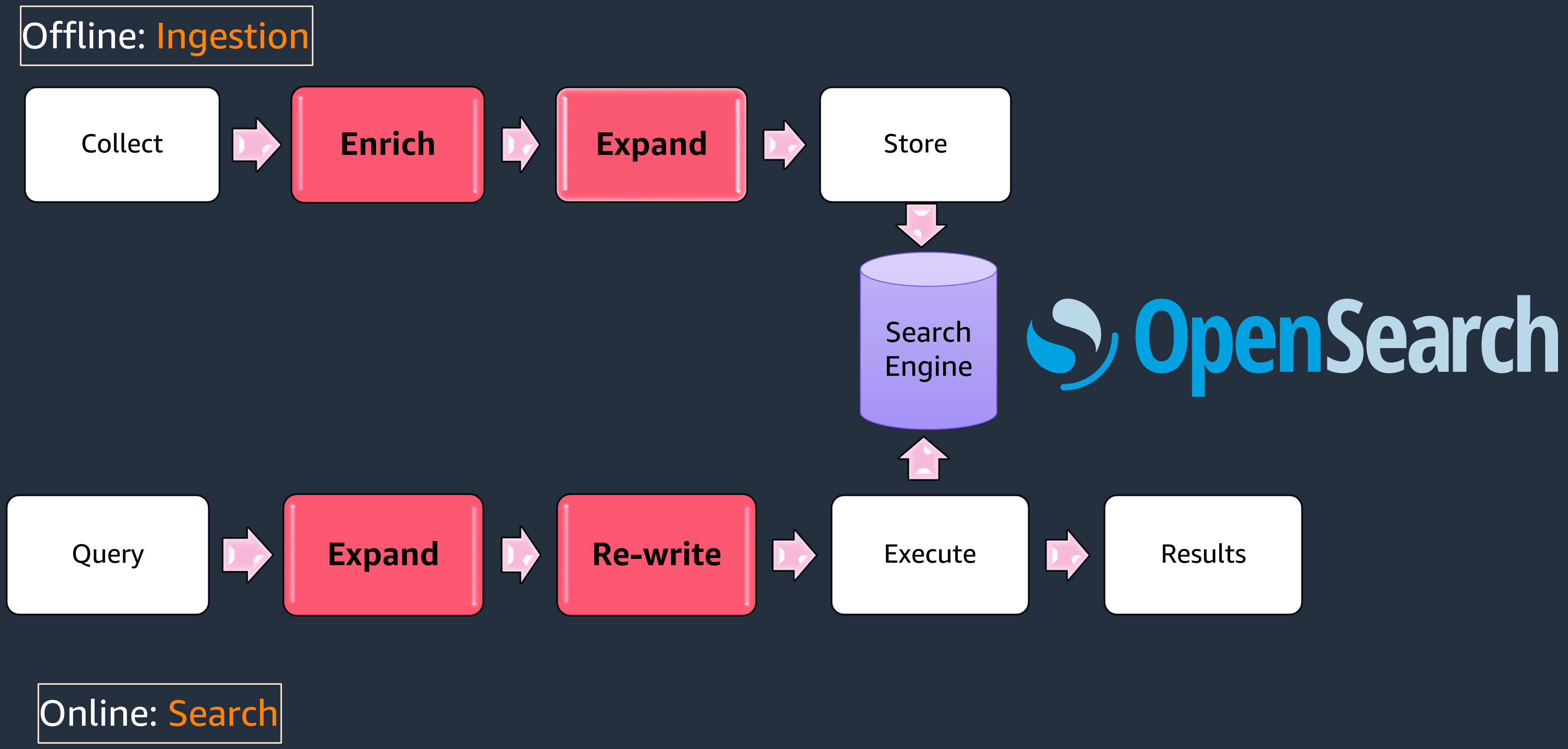
OpenSearch Query DSL

```
{
  "query": {
    "bool": {
      "must": [
        { "match":{ "description":"shoes"}}
            ],

      "filter": [
        { "term":  { "category": "footwear" }},
        { "range": { "price":{ "lt": 50 }},
        { "term":  { "gender":"male" }},
        { "term":  { "color": "brown"}}
      ]
    }
  }
}
```

Demo

(Semantic) lexical search lifecycle



Key takeaways

1. Provide semantic search capabilities with Neural Sparse Search
    - Bi-Encoder and Document only encoder models
2. Use AI/ML connectors of OpenSearch to reduce the heavy lifting
3. Enrich your image and text documents
4. Use LLM to rewrite your DSL queries
    - Use existing frameworks and improve upon (LangChain, Haystack)
    - Measure, Evaluate and Iterate

Thank you!

Hajer Bouafif
bouhajer@amazon.fr

Praveen Mohan
prasadnu@amazon.com