Professional Training Program in Large Language Models

Day 4

Hamza Farooq
Founder & CEO
Traversaal.ai

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

Building LLM Powered Solutions

Week 2: Let’s build a Search Engine

Hamza Farooq

Recap: What we have covered so far?

1. What is TF-IDF?
2. What are Sentence Transformers
3. The Science Behind Embeddings
4. What is Semantic Search?
5. Sparse vs. Dense Vectors
6. What is Euclidean Distance?
7. Cosine Similarity

Learning Outcomes

We will be covering topics on:

- Faiss
- ANN
- Why is search so important?
- Understand the building blocks of search, again
- Sentence Transformers
- BM25, Bi-encoder, Cross-Encoders
- Cosine Similarity, again!
- Fitting all of this into a ML System
- Coding, lots of it, with API
- Evaluation of Models
- Query intent model

Introducing F.A.I.S.S.

Facebook Artificial Intelligence Similarity Search

FAISS

FAISS is a library that enables efficient similarity search.

FAISS

Just as we saw in the previous module, we ***index*** a set of vectors.

We then convert our own query to a vector using the same model as the indexed ones, and search the most similar ones within the index.

FAISS

But that’s **standard**, so what is so special?

FAISS increases search times, leading to *amazing* search performances.

FAISS

But that’s **standard**, so what is so special?

FAISS increases search times, leading to *amazing* search performances.

FAISS

It’s optimized for both CPU and GPU usage, and it is a really efficient method when dealing with **large** datasets, which conventional methods would take too long to search through.

FAISS - Its speed

But why is FAISS so fast?

The **indexing** plays a key part in it.

An index is not just a label you assign to different vectors (i.e., it’s not just a jersey you give to any team member).

An index represents a specific way to store and distribute elements (i.e., it’s the player setup as the match starts).

FAISS

FAISS has 6 main types of indexes:

- **Flat Index**
- **IVF (Inverted File) Index**
- **Hierarchical Navigable Small World (HNSW) Index**
- **Quantization-based Indexes**
- **Composite Indexes**
- **GPU-specific Indexes**

FAISS

FAISS has 6 main types of indexes:

- **Flat Index**

Can either be of type IndexFlatL2, where it performs an exhaustive search with L2 distance between the distance, or IndexFlatIP, by calculating the dot product between the vectors (especially when they're normalized, since then it's equal to cosine similarity).

- **IVF (Inverted File) Index**
- **Hierarchical Navigable Small World (HNSW) Index**
- **Quantization-based Indexes**
- **Composite Indexes**
- **GPU-specific Indexes**

FAISS

FAISS has 6 main types of indexes:

- **Flat Index**
- **IVF (Inverted File) Index**

Includes three types: IndexIVFFlat, which partitions vectors into coarse quantizers to speed up searches; IndexIVFPQ, which combines an inverted file system with product quantization for optimized memory usage and accuracy; and another variant of IndexIVFFlat, which uses flat encoding for the residuals.

- **Hierarchical Navigable Small World (HNSW) Index**
- **Quantization-based Indexes**
- **Composite Indexes**
- **GPU-specific Indexes**

FAISS

FAISS has 6 main types of indexes:

- **Flat Index**
- **IVF (Inverted File) Index**
- **Hierarchical Navigable Small World (HNSW) Index**

Includes HNSWFlat, a graph-based approach, maintaining multiple layers of interconnected graphs. Searches start from the upper layers and go deeper, making it efficient for both building and querying the index.

- **Quantization-based Indexes**
- **Composite Indexes**
- **GPU-specific Indexes**

FAISS

FAISS has 6 main types of indexes:

- Flat Index
- IVF (Inverted File) Index
- Hierarchical Navigable Small World (HNSW) Index
- **Quantization-based Indexes**

Composed of two types: IndexPQ, using a technique called product quantization to compress vectors and reduce the storage they require., and IndexSQ, reducing dimensions to smaller, scalar values (easier to process).

- Composite Indexes
- GPU-specific Indexes

FAISS

FAISS has 6 main types of indexes:

- Flat Index
- IVF (Inverted File) Index
- Hierarchical Navigable Small World (HNSW) Index
- Quantization-based Indexes
- **Composite Indexes**

Made from the IndexIVFScalarQuantizer combines inverted file indexing with scalar quantization for balanced performance and accuracy, and the MultiIndexQuantizer uses multiple quantizers at different levels to enhance vector quantization granularity

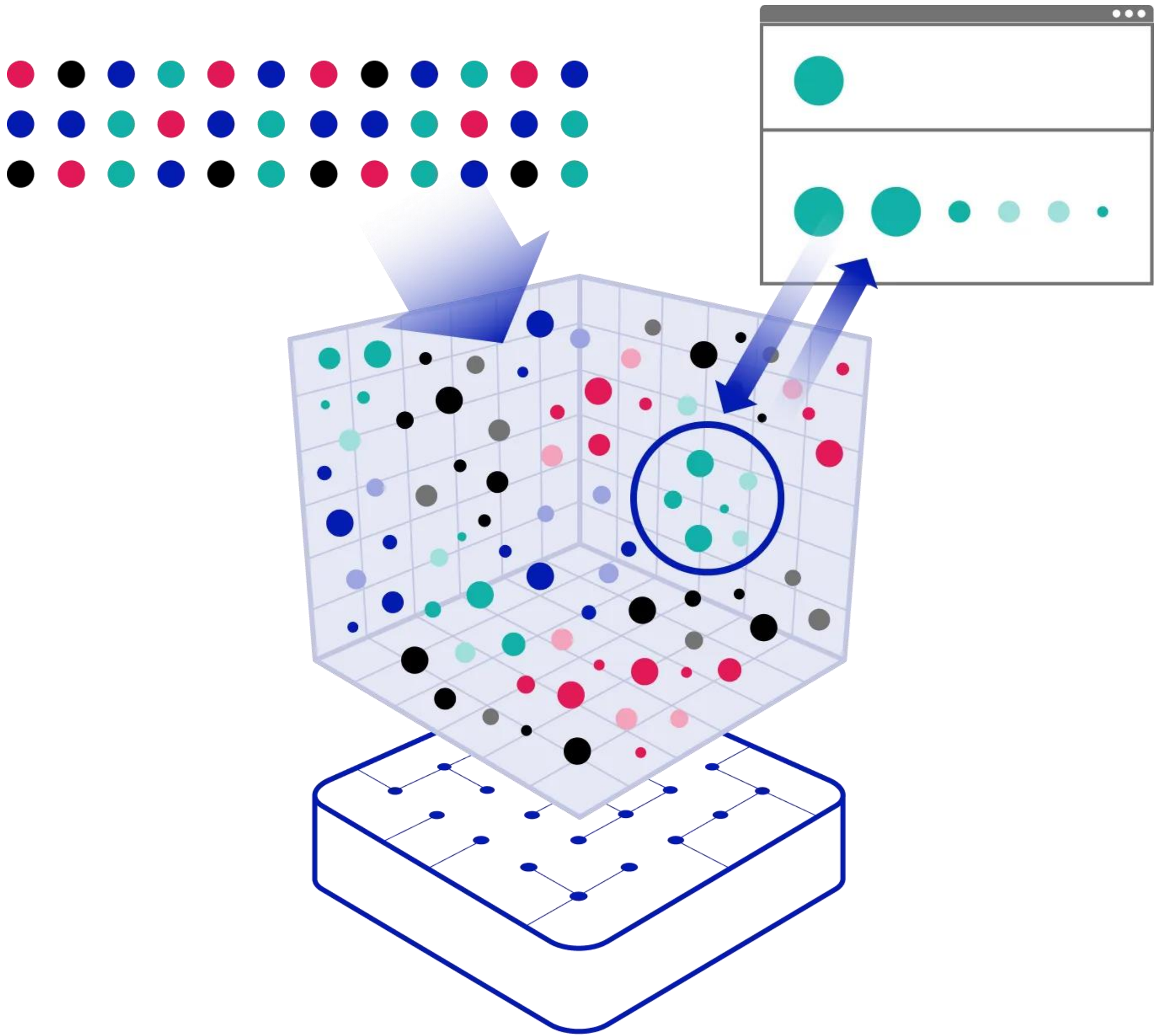
- GPU-specific Indexes

FAISS

FAISS has 6 main types of indexes:

- Flat Index
- IVF (Inverted File) Index
- Hierarchical Navigable Small World (HNSW) Index
- Quantization-based Indexes
- Composite Indexes
- **GPU-specific Indexes**

Comprised by the GpuIndexFlat, which performs brute force searches directly on GPUs for faster query processing, while the GpuIndexIVFFlat optimizes the IVFFlat index for GPU architectures to speed up searches and improve efficiency.



What else can be done when the dataset is too large?

The Approximate Nearest Neighbors (ANN) algorithms

Approximate Nearest Neighbors

Most Nearest Neighbors algorithms tend to find the closest neighbors to a given data point, i.e. the data points that are most similar to it, or share properties with very close values

However, in a very high density data space, this practice can be quite slow and expensive.

Approximate Nearest Neighbors

That's where Approximate Nearest Neighbors come into play: instead of ensuring the absolutely closest neighbors are found (as in exact search), they aim for high speed and efficiency at the expense of some accuracy.

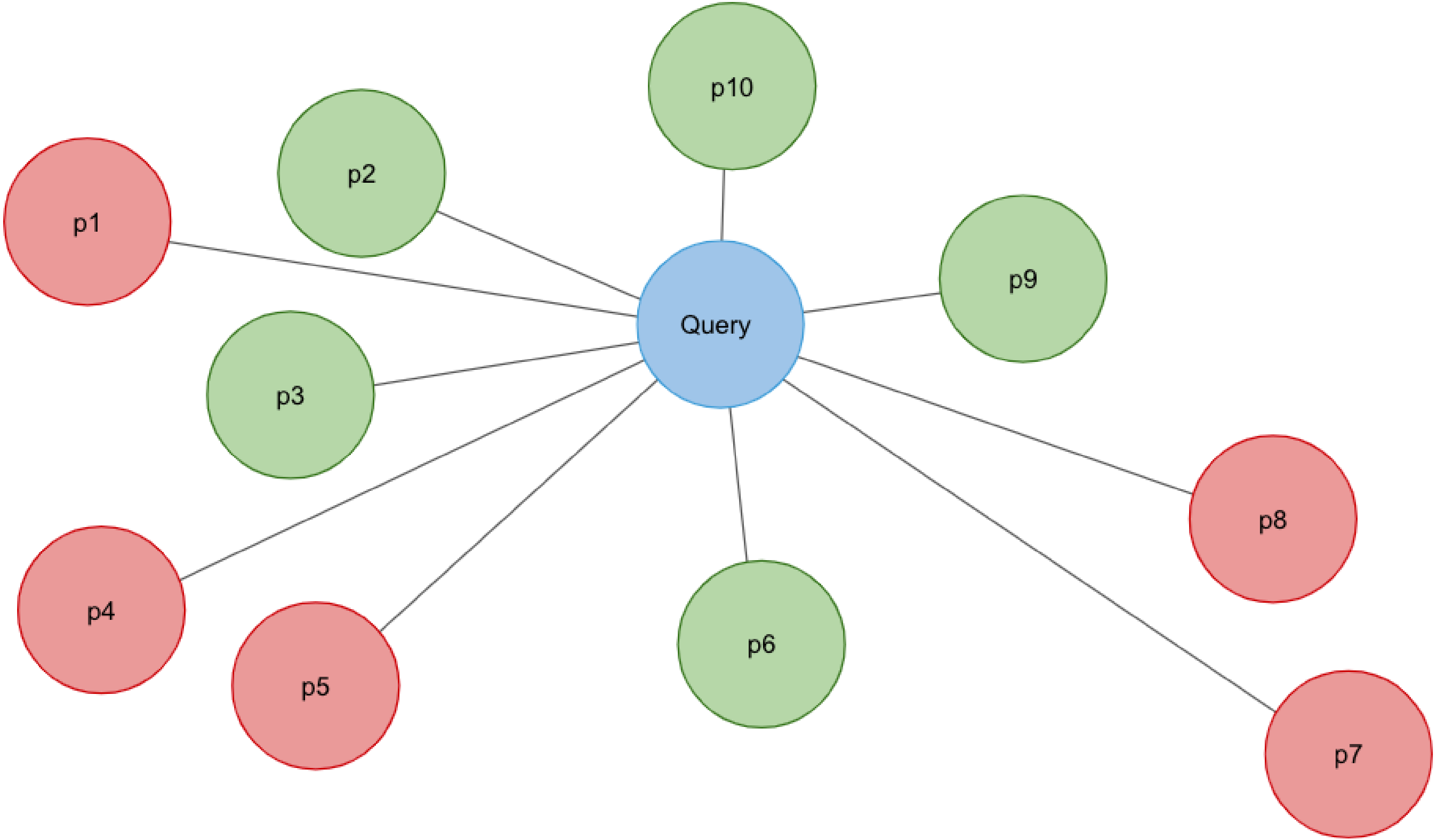
Approximate Nearest Neighbors

ANN algorithms are particularly useful when dealing with very large datasets where exact searches would be computationally expensive and time-consuming.

So how do FAISS and ANN work together?

FAISS and ANN

FAISS is specifically tailored for vector similarity search, making it an excellent tool for ANN applications. It utilizes a combination of quantization and partitioning strategies to efficiently handle billions of vectors. FAISS is particularly powerful when used with GPUs, providing significant acceleration in processing times for ANN searches.


p10
p2
p1
p9
Query
p3
p8
p4
p6
p5
p7

ANN - Key Characteristics

- **Speed** → ANN algorithms are designed to return results quickly, even for large datasets, by making acceptable compromises on accuracy.
- **Scalability** → They handle high-dimensional data and scale efficiently with the size of the dataset, making them suitable for applications like real-time recommendation systems and image or video retrieval.
- **Resource Efficiency** → By using memory-efficient data structures and reducing computational demands, ANN methods minimize resource usage.

ANN - Applications

- Image and Video Retrieval
- Recommendation Systems
- Natural Language Processing
- Clustering and Classification

A good search experience is key to a successful user journey

An estimated [50% of queries contain four or more words] - search is no more just keyword based

62% of consumers will switch to a different brand or decide not to purchase from your brand at all after a bad customer experience — and poor site search is a bad customer experience.

Project Athena: Adding Semantic search to Hotel search

We want to build a hotel search using:

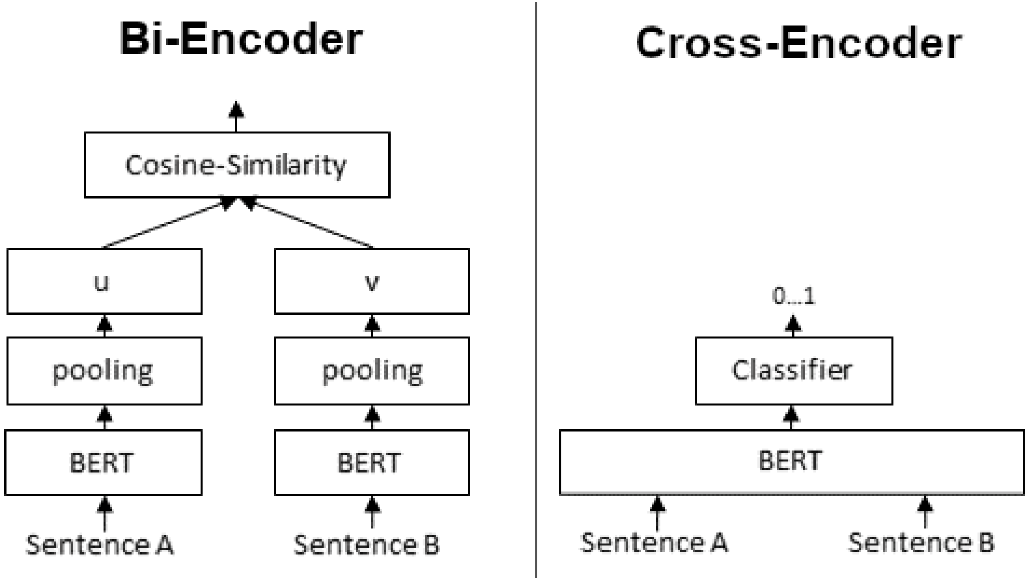
- Date check-in/check-out
- City
- Long text to get more granular choices such as:

Looking for a hotel in New York near Times Square with free breakfast and cheaper than $100 for 2nd June which is really kids friendly and has a swimming pool and I want to stay there for 8 days...

$$\sum_{i}^{n} IDF(q_i) \frac{f(q_i, D) * (k1 + 1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}$$

BM25

Numerous architectures are available for ranking: representation-focused, interaction-focused, all-to-all interaction(cross encoder), and late interaction.



Source:
https://www.sbert.net/examples/applications/cross-encoder/README.html

BM25

BM25 stands for "Best Match 25" and is an information retrieval algorithm used to rank documents based on their relevance to a given query.

BM25

It calculates a relevance score by considering the term frequency and document length in a collection of documents.

Bi-Encoder

A bi-encoder is a type of neural network architecture used in natural language processing (NLP) tasks.

Bi-Encoder

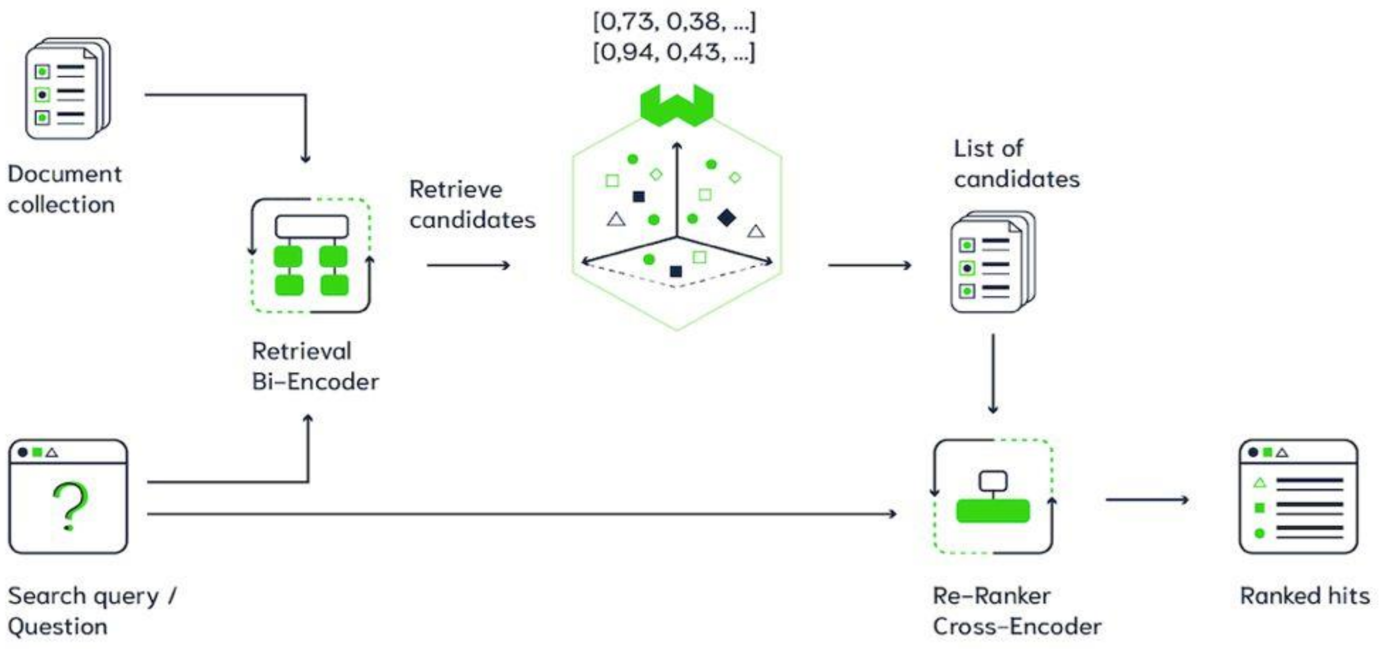
It consists of two encoders: one for encoding the input query and another for encoding the input document.

Bi-Encoder

Each encoder independently encodes the input into a fixed-length representation, often called an embedding.

Cross-Encoder

- A cross-encoder is another type of neural network architecture used in NLP tasks.
- It takes both the input query and document as a single input and encodes them into a fixed-length representation.
- Unlike the bi-encoder, the cross-encoder considers the interaction between the query and document when generating the representation.


[0,73, 0,38, ...]
[0,94, 0,43, ...]
Document
collection
Retrieve
candidates
Retrieval
Bi-Encoder
List of
candidates
Search query /
Question
Re-Ranker
Cross-Encoder
Ranked hits

The result…

Looking for a hotel in New York near Times Square with free breakfast and cheaper than $100 for 2nd June which is really kids friendly and has a swimming pool and I want to stay there for 8 days..

⇩

Looking for a hotel in New York GPE near Times Square FAC with free breakfast and cheaper than $100 MONEY for 2nd June DATE which is really kids friendly and has a swimming pool and I want to stay there for 8 days DATE

⇩

Top 5 most relevant hotels:

=====================

InterContinental New York Times Square

Relevancy: 0.4037

IBEROSTAR 70 Park Avenue Hotel

Relevancy: 0.3475

The Townhouse Inn of Chelsea

Relevancy: 0.3330

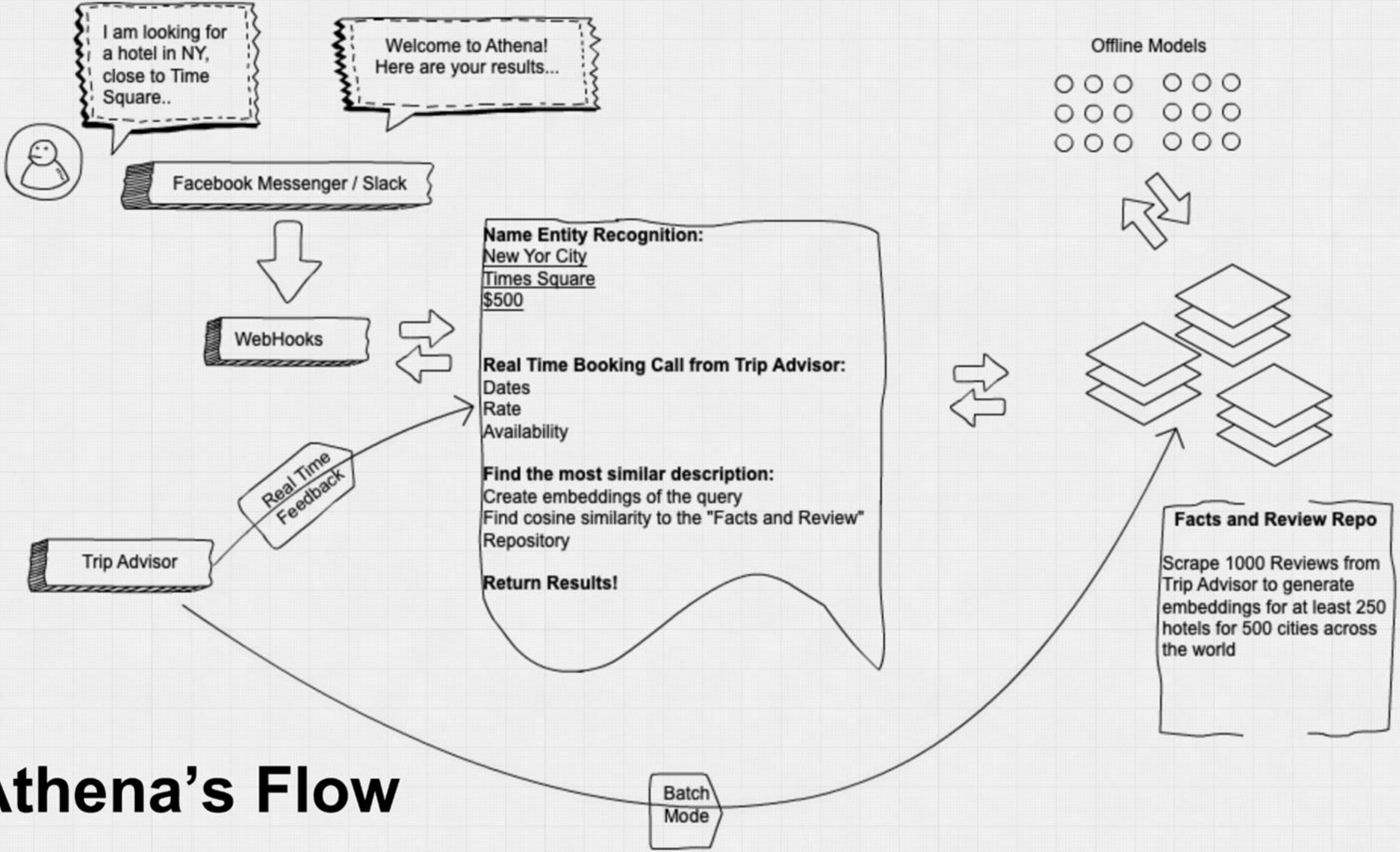
Pod 51 Hotel

Relevancy: 0.3162

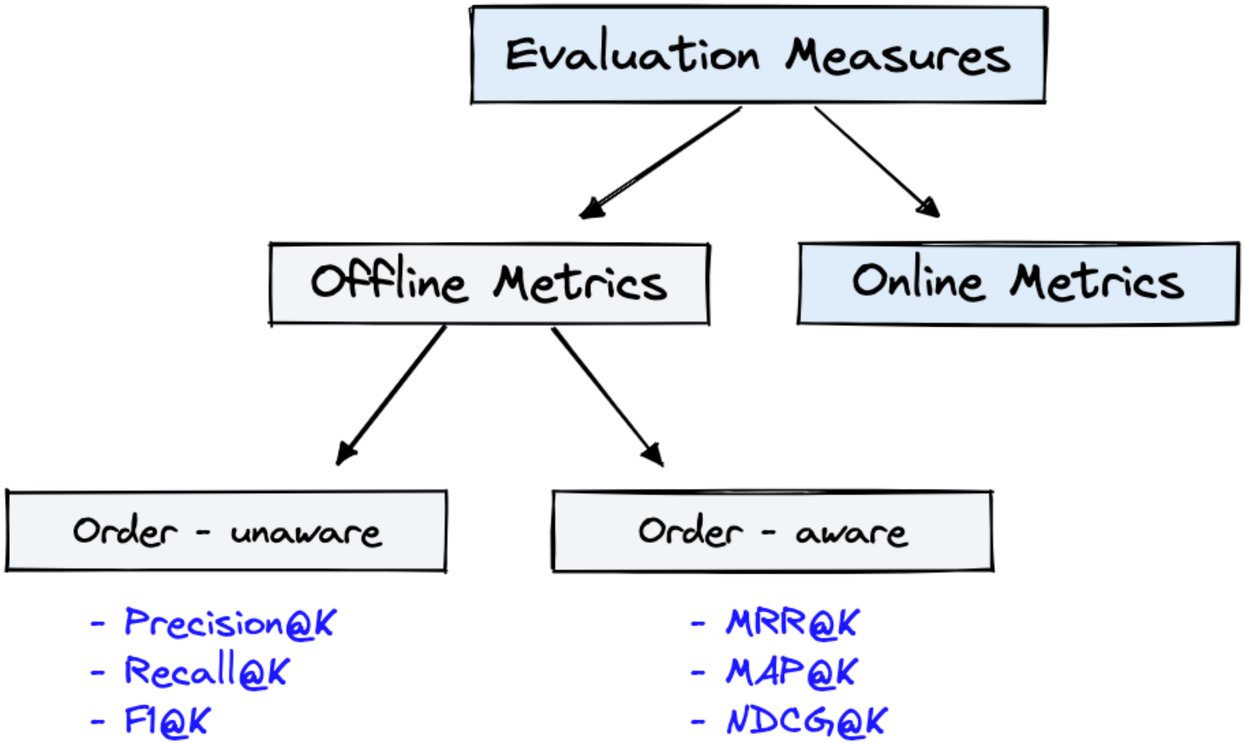
Soul Food Mont Morris

Relevancy: 0.2995

Athena’s Flow



Evaluation Criteria

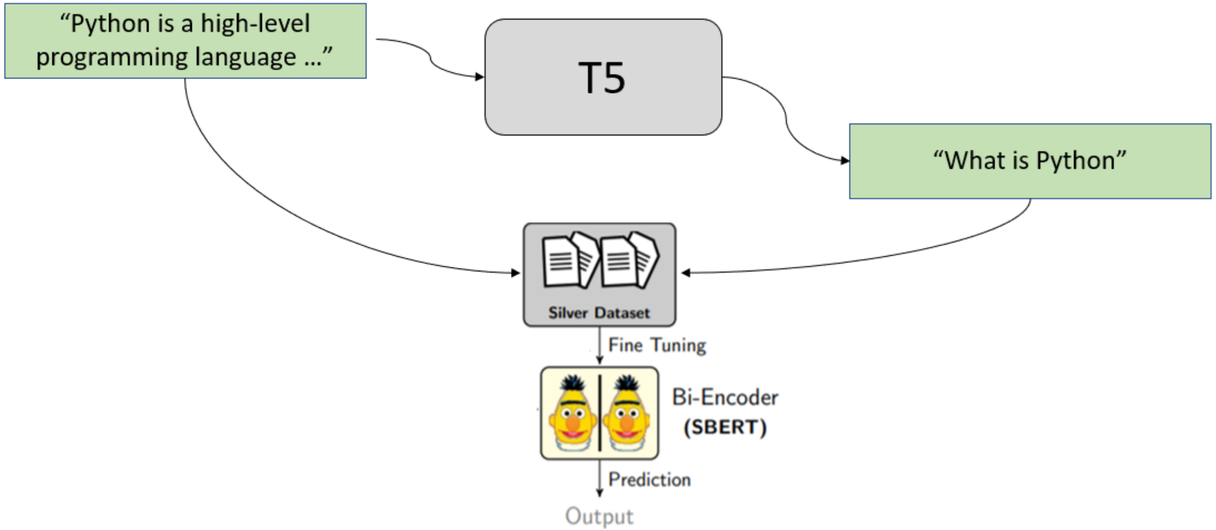


Source:

https://www.pinecone.io/learn/offline-evaluation/

What happens when we don’t have ground truth?

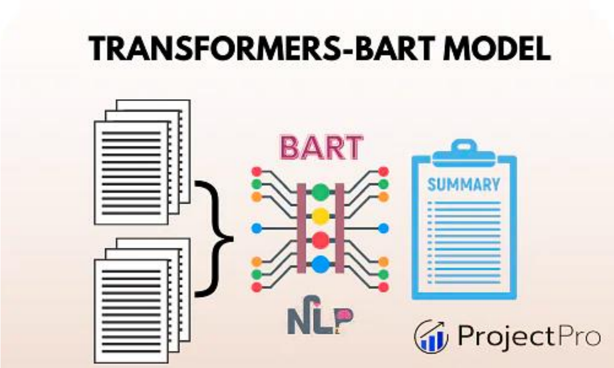
Creating ground-truth from scratch



BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, Iryna Gurevych

Summarization



Demo

- BART is a sequence-to-sequence model trained as a denoising autoencoder.
- A fine-tuned BART model can take a text sequence (for example, English) as input and produce a different text sequence at the output (for example, French).
- This type of model is relevant for machine translation question-answering, text summarization or sequence classification
- Also, given two or more sentences, evaluates whether the sentences are logical extensions or are logically related to a given statement.

Aspect-based Sentiment Analysis

Utilize transformers and other ML models to generate sentiment for various aspects of an entity


```
@misc{YangL2022,
  title = {PyABSA: Open Framework for Aspect-based Sentiment Analysis},
  author = {Yang, Heng and Li, Ke},
  doi = {10.48550/ARXIV.2208.01368},
  url = {https://arxiv.org/abs/2208.01368},
  keywords = {Computation and Language (cs.CL), FOS: Computer and information sciences, FOS: Computer and information sciences},
  publisher = {arXiv},
  year = {2022},
  copyright = {arXiv.org perpetual, non-exclusive license}
}
```

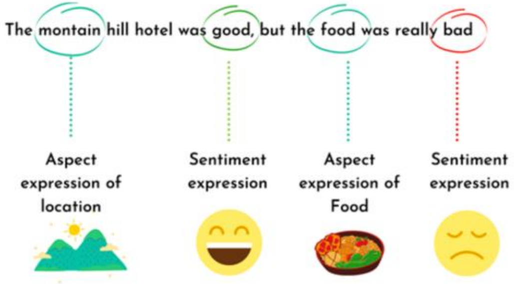

Consider these reviews:

Friendly and accommodating staff helpful with transportation, restaurants and directions. Great location for all activities. Easy walk to Louvre. Breakfasts exceeded expectations. Mattress was too soft to my liking.

The reception was friendly and professional and speedy. The room was ready and perfect. The bed was very comfortable and the air conditioning was silent and potent. The free afternoon tea was amazing and open until 2am. The breakfast was one of the very best you could find in Paris

The room was awesome

Stayed here for two nights after a work trip in the city. I made an error in my booking and the hotel were very gracious and sorted it out for me. Kindly offered breakfast on the morning of my arrival. Very good selection for breakfast. Excellent location and fab staff would recommend




```
{'staff': ['Positive', 'Positive'],
 'location': ['Positive', 'Positive'],
 'Breakfasts': ['Positive'],
 'Mattress': ['Negative'],
 'reception': ['Positive'],
 'room': ['Positive', 'Positive'],
 'bed': ['Positive'],
 'air conditioning': ['Positive'],
 'afternoon tea': ['Positive'],
 'breakfast': ['Positive', 'Positive']}
```

Keyword creation using Transformers

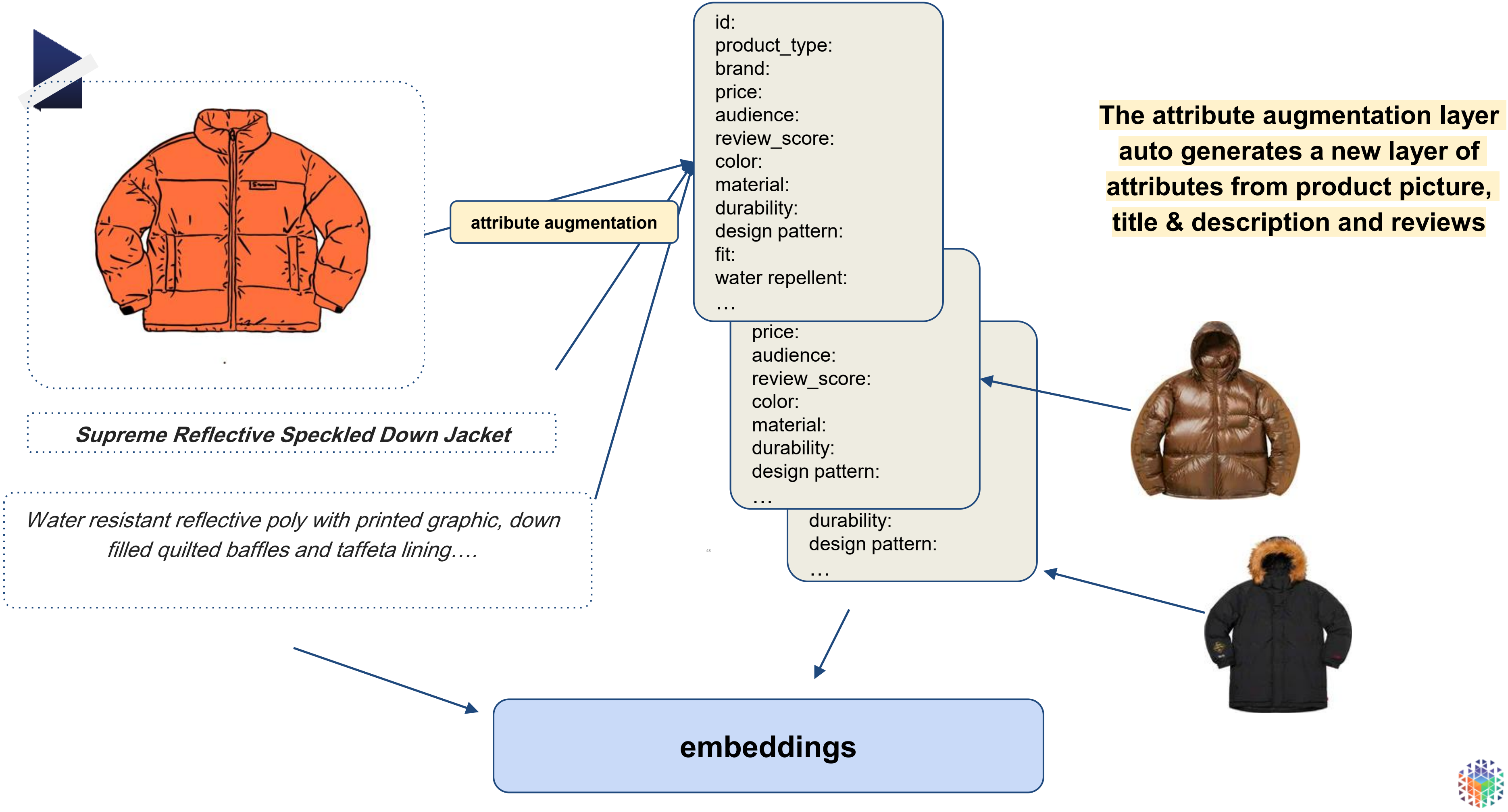
KeyPhraseTransformer is built on T5 Transformer architecture, trained on 500,000 training samples to extract important phrases/topics/themes from text of any length.

- *Hotel staff were very helpful and friendly.*
- *I was very happy with the room and bathroom.*
- *I was very happy with my stay at the hotel.*
- *I would highly recommend this hotel to anyone who is looking for a place to stay.*
- *Hotel staff is very friendly and helpful.*
- *I was so happy to stay at this hotel... it was amazing!*
- *Louvre and many other locations*
- *Hotel staff were very friendly and helpful.*
- *Breakfast and afternoon snacks*
- *I know where i will be staying on our next trip to paris*

Source:
https://github.com/Shivanandroy/KeyPhraseTransformer

Query Intent Models

Queries need special handling and interpretation due to their tendency to be short, and too often imply more than they state explicitly


Supreme Reflective Speckled Down Jacket
Water resistant reflective poly with printed graphic, down filled quilted baffles and taffeta lining….
attribute augmentation
id:
product_type:
brand:
price:
audience:
review_score:
color:
material:
durability:
design pattern:
fit:
water repellent:
…
price:
audience:
review_score:
color:
material:
durability:
design pattern:
…
durability:
design pattern:
…
The attribute augmentation layer auto generates a new layer of attributes from product picture, title & description and reviews
embeddings

water resistant orange down jacket supreme..|

Use a combination of BM25 with Bi-Encoder and Cross-Encoders

query intent

similarity measure

embeddings

Product Type: Jacket
Color: Orange
Brand: Supreme
Water Repellent: Yes

Use knn model to reduce the sample space

Assessment

- Implement similar hotel search engine for Miami hotels - feel free to apply any of the methods mentioned for retrieval and additional methods to improve your modeling
- Step 1:
  - Create a hotel summary/ encompasses a large amount of hotel info
- Step 2:
  - Create your search
  - Are the results similar to what you're searching for?
- Submission is a notion doc with the colab notebook and a writeup:
  - What you did?
  - How did you collate the data on the hotel
  - Simple feedback on your search

Appendix

Thank you!

الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority