# I'm Feeling Lucky

Published on June 9, 2022



(Source: Pixabay)

**Yogesh Kulkarni**
Principal Architect (CTO Office, Icertis) | PhD in Geometric Modeling | Google Developer Expert (Machine Learning)
12 articles

You don't have to be lucky anymore if you wish to have a precise search result.

We all are familiar with keyword based search. It works well for structured data, like, if you want to know actors names from a particular movie, such information is typically stored against the movie

different now, like, searching within videos, finding similar shapes, etc. That's Neural Search. E.g. it would let you search through YouTube videos describing the scene you remember, making it more flexible and convenient than traditional keyword-based Search.

Such neural search is being made available by Jina AI. It is a cloud-native, open source, cross/multi-modal, data type-agnostic neural search engine which offers anything-to-anything search, ie. Text-to-text, image-to-image, video-to-video, or whatever else you can feed it.

## Concepts

Lets understand some basic concepts/components in Jina AI.

Modality:

- Single Modality: Types of input and output are same. e.g. find similar images by providing a reference image.
- Cross Modality: Find relevant documents of modality A (let's say —“image”) by querying with documents from modality B (let's say —“text”).
- Multi-modality: A query with combining different modalities (text + image) to get the output.

Jina AI ecosystem consists of:

- DocArray - A standard data structure for all kinds of data
- Jina - Core neural search framework with plug and play search pipeline

- Finetuner - Fine-tuning any deep learning model

Jina consists of following components:

- Document: Base/primitive data type. Both inputs and outputs are documents.

- Executor: Processes the documents. Processes/algorithms such as encoding images into vectors, storing vectors on the disk, ranking results, etc. Examples:

1. Crafter: Pre-processes documents into chunks.

2. Encoder: Takes the input pre-processed chuck of documents from the crafter and encodes them into embedding vectors.

3. Indexer: Takes the encoded vectors as input and indexes and stores the vectors in a key-value fashion.

4. Ranker: Ranker runs on the indexed storage and sorts the results based on a certain ranking.

- Flow: Streamlines and distributes the Executors. It allows you to chain together the DocumentArray and Executors to build and serve an application out of it. It consists of pods (segmenting, encoding, and ranking., etc), a Context manager and abstraction of high-level tasks, e.g. indexing, searching, training. Jina core typically comprises of two main flows, which are the heart and soul of the semantic search engine:

1. Indexing Flow: Makes the whole corpus search-able by sentence. The input documents are fed in, processed, and output at the other end is stored as search-able indexes.

on the similarity score within the word embeddings.

We can understand these concepts better with an example.

## Example

Let's implement a simple Neural Search service to demonstrate searching a similar image, based on human perception (Source:[4]). Dataset used is `Totally-Looks-Like Dataset` (Source: [5]). Here are a couple of pairs out of 6016 image pairs.



Lets define a few Executors and then compose them into a sequential pipeline called, Flow.

Executor to pre-process input images :

```
from docarray import Document, DocumentArra
from jina import Executor, Flow, requests

class PreprocImg(Executor):
  @requests
  async def foo(self, docs: DocumentArray, **kwargs):
    for d in docs:
      (
        d.load_uri_to_image_tensor(200, 200) # load
```

```
        .set_image_tensor_channel_axis(
          -1, 0
        ) # switch color axis for the PyTorch model later
      )
```

Executor to embed preprocessed images via Resnet.

```
class EmbedImg(Executor)
  def __init__(self, **kwargs):
    super().__init__(**kwargs)
    import torchvision
    self.model = torchvision.models.resnet50(pretrained=True)

  @requests
  async def foo(self, docs: DocumentArray, **kwargs):
    docs.embed(self.model):
```

Executor to perform matching:

```
class MatchImg(Executor)
  _da = DocumentArray()

  @requests(on='/index')
  async def index(self, docs: DocumentArray, **kwargs):
    self._da.extend(docs)
    docs.clear() # clear content to save bandwidth

  @requests(on='/search')
  async def foo(self, docs: DocumentArray, **kwargs):
    docs.match(self._da, limit=9)
    del docs[...][:, ('embedding', 'tensor')] # save bandwidth
```

Use the Flow to connect all the Executors and use `plot` to visualize it.

```
f = Flow(port_expose=12345).add(uses=PreprocImg).add(uses=Embe
f.plot('flow.svg'))
```

Get the data, download image dataset and pass them to the Flow defined above:

```
index_data = DocumentArray.pull('demo-leftda', show_progress=T
```

Using Flow `f` index the documents

```
with f
  f.post(
    '/index',
    index_data,
    show_progress=True,
    request_size=8,
  )
  f.block():
```

Search-able data and corresponding service is ready now. Use a Python client to access the service.

```
from jina import Clien

c = Client(port=12345) # connect to localhost:12345
print(c.post('/search', index_data[0])['@m']) # '@m' is the ma
```

Straight-forward, isn't it?

More code-level examples can be seen at github.

- Video Search (https://github.com/jina-ai/example-video-search)
- Multi-modal Search (https://github.com/jina-ai/example-multimodal-fashion-search)
- Meme Search (https://github.com/jina-ai/example-meme-search)

Fascinating!!

## Conclusion

Humans think multi-modal. We have description, visuals, sound (and may be, smell and taste also) etc in our minds while looking for something specific. Can AI (Artificial Intelligence) be as effectives as humans? As more and more data-types get merged like demonstrated above, though for a narrow task of `search`, we are getting more closer to Human Intelligence, ie towards AGI (Artificial General Intelligence), isn't it?

Give Jina AI a try and feel free to share your comments...

## References

1. Getting started with Jina AI (https://kunalkushwaha.com/getting-started-with-jina-ai)
2. Getting started documentation (https://docs.jina.ai/get-started/create-app/)

4. An Overview of Jina AI (https://www.section.io/engineering-education/an-overview-of-jina-ai/)

5. Totally-Looks-Like Dataset (https://sites.google.com/view/totally-looks-like-dataset)

6. Audio to Audio example (https://github.com/jina-ai/examples/blob/master/audio-to-audio-search/.github/demo.png)

7. Official website (https://jina.ai/)

---

Published by

**Yogesh Kulkarni** 12 articles

Principal Architect (CTO Office, Icertis) | PhD in Geometric Modeling | Google Developer Expert (Machine Learning)

Published • 1mo

Jina AI is a next-gen open source platform for searching with/within text, images, videos. Here is my introductory article giving an overview.

Kunal Kushwaha Han Xiao (remember 'bert-as-a-service'?) Pratik Bhavsar Pradeep Sharma Bing He Nan Wang

#ml #machinelearning #nlp #naturallanguageprocessing #ai #jinaai #search #searchengine #artificialintelligence #datascience #deeplearning #neuralnetworks #data #neuralsearch #neuralnetworks #tech #Innovation #technology #opensource

---

Like Comment Share

Pravin Shinde and 40 others 2 comments

Reactions

+29

2 Comments

Most relevant