# From Files to Semantics: Data Services for AI practitioners

Maria Gutierrez – Field Technical Director, EMEA
Jan Heichler - Director of AI & HPC PreSales, EMEA


THE FILE
IS NOT
ENOUGH
FILE

# The Project: Create a Medical Assistant SLM

The Task: Implement an end-to-end pipeline to help diagnose rear disease from unstructured Health Records

**Domain:** Natural Language Processing (NLP) for Clinical Informatics.

**Focus:** Fine-tuning a Small Language Model (SLM) optimized for Rare Disease identification

**Dr. Elena Vance,**
Senior Researcher in Computational Medicine.

Model

Data

FineTune

Inference

### Take a pre-trained SLM
**(Gemma-3-4B e.g)**

- OpenWeights by Google – reduce resources
- MultiModal: text and images (it can process handwritten notes as well as electronic)

### Collect and Curate Data

- For that task we need data – that we have to collect and curate
- Data comes in different formats, from different places and with „noise“ – so can‘t be used in it‘s raw form.

### „Training“

- Can be run in different dimensions – using „tagged data“ or „supervised fine tuning“
- The above requires potentially different „quality“ of data – or with different highlights

### Production

- Put your model to work
- Feed it with new notes automatically annotated as being ingested
- Vectorize data, for other models to be used (RAG)
- Use our SLM to provide answers based on new notes

# Collecting your Data

**Collecting Data from various sources and multiple formats:**

- No consistent user management
- Different software platforms
- Varying experience in users providing data
- Different formats: EHR as DBs, Doctor notes raw text, semi-structured surveys PDFs,..


**Create Logins for external users**

- Login - and allow upload using scp, sftp, rsync…
- Significant administrative overhead

**Provide external services**

- WebDAV
- FTP
- …


**Create a bucket**

- Hand out the URL

# Build Deep Metadata – often a manual and distributed process

## Named Entity Recognition

*"Patient prescribed 50mg Sertraline for anxiety,"*

`[50mg] as` **`Dosage`**
`[Sertraline] as` **`Medication`**
`[Anxiety] as` **`Condition`**

## Relation Extraction (Mapping Logic)

*"Lisinopril was discontinued due to a persistent dry cough,"*

Entity A: Lisinopril
Entity B: Dry cough
Relationship: `caused_by` or `adverse_event`

## Attribute Assertion (Contextual Truth)

Medical notes often mention diseases the patient *doesn't* have it might incorrectly learn that the patient has chest pain.

*„Patient denies chest pain"*

`[Pneumonia]` **`Assertion Status:`** `Negative`

## Medical Concept Normalization

Doctors use shorthand, slang, and various synonyms (e.g., "heart attack" vs. "MI" vs. "Myocardial Infarction").

Map to ICD-10: I21

# Storing Deep Metadata

Filesystem

## Names and Location to identify data

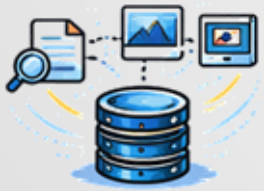
```
/Medical_Data/2026/Patient_001/record_01.docx
/Medical_Data/2026/Patient_061/record_06a.docx
/Medical_Data/2026/Patient_061/record_06a_v01.docx
```

## Create .json files with content

```
record_06a_v01.json
{
  "text": "Patient prescribed 50mg Sertraline.",
  "label": [
    {"start": 23, "end": 27, "label": "DOSAGE"},
    {"start": 28, "end": 38, "label": "MED.."}
}
```

## Issues

- Context in the tree structure: easily breaks!
- Versioning control: clunky and prone to error
- Querying data



Object Storage

## S3 Tags for data management

```
ProjectYear     2026
PatientID       061
CuratorID       Student_Smith_J
```

**-> Becomes part of the Object!**

## Object Versioning

Easy to keep older versions (data modified, reformatted)

## Flat bucket structure

.json for extensive MetaData

## Pros/Cons

- Makes data management easier
- Does not provide faster queries

Database

```
{
  "note_id": "001",
  "patient_id": "P-99",
  "annotations": [
    {"entity": "Chest pain", "type": "Symptom", "assertion": "..."},
    {"entity": "Metformin", "type": "Medication", "assertion": "..."}
  ]
}
```

| entity | type | assertion | note_id | patient_id |
|---|---|---|---|---|
| Chest pain | Symptom | Absent | 001 | P-99 |
| Metformin | Medication | Present | 001 | P-99 |

## Solve the Query issue

Easier to link back to ObjectID instead of fullPath

# Training


**Once the Data is selected you can read it from file or object.**

We see a trend in the market that software stacks move towards Object (Portability, ease-of-use, Cloud)

So users might come with that requirement if they use a stack ported from somewhere else!

# Put your model to work

New Note is uploaded → Create the DeepMetadata → Chunk the Note up → Create Vector Embeddings


**New Note lands in the filesystem**

Opt 1) – Batch job run every X h  Notes uploaded > set off process

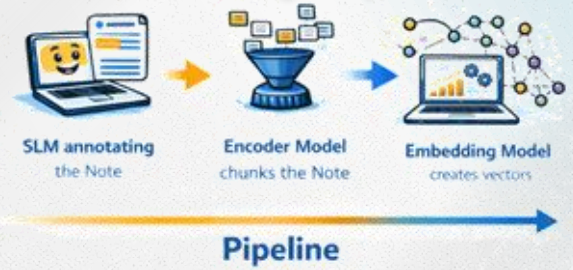
Opt 2) – Process constantly polling and scan for new files


**New Note lands in the Object Storage**

Opt 1) Scan your bucket (above)

Opt 2) Bucket Notification - event object is created -> Catch it and set off a batch job -> very coarse

**Running the job**



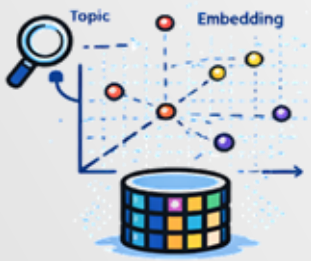

**Where do i store my vectors?**

- FILES?  – but then queries become impossible.
- Run own VectorDB with a file or block backend?  „pgvector“ (or similar) – but how could a user do this?

**Where do i store my vectors?**

- OBJECTS? – still queries are impossible at scale.
- Run your own VectorDB with a S3 backend: „Milvus“ .. – how could a user do this?

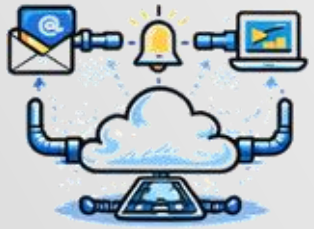
# Advanced Data Services




Vector Database

## Building a Knowledge Base for models with Vectors

- Find notes based on medical meaning
- RealTime insights – instead of re-training on every new note
- Privacy preserved inference: SLM can‘t be trained on personalized data – but you can do RAG on it
- Have a consistenly available service – rather than a workaround



Event Broker

## Building complex Data Driven Pipelines with Event Brokers

- Respond to internal and external events – like catching a bucket notification
- Start one step of the pipeline after the previous step has finished
- You can aggregate events – don‘t set off for one event but wait for 10 to line up
- Decouple your architecture
  - „Annotation“ and „Vectorization“ are independet of each other
  - add or remove processing steps without rewriting anything but by creating new consumers of event streams

# Continous improvement

New Note is uploaded

Create the DeepMetadata

Chunk the Note up

Create Vector Embeddings

New Model > Model Registry

Trigger fine-tune job

Expert validation


## New Model version

- New version (code, model weights & Lineage) into Model Registry

## Trigger Fine-tune job

- Gather expert input
- Trigger new fine-tune job with new output and corrections

## Expert evaluation

- Output with low confidence score sent to experts for evaluation
- Specialist Dashboard
- Good/bad rate & corrections

# Conclusion

## *AI adds multiple dimensions to the Data Model*

*Abstracting this complexity away from the user drives adoption up. Infrastructure must serve the data's "meaning," not just its "bits."*

## *Don't push "Platform Debt" up the stack*

*If you build "Storage," users have to build "Intelligence" making consumption hard – which leads to starved and idle compute.*

## *Minimize Data Movement to Maximize GPU ROI*

*Complex stacks that move data frequently create I/O bottlenecks that plummet effective GPU utilization.*

# The VAST AI Operating System

**DataEngine**
Scalable, Event-Driven Computing
Triggers, Functions, Containers

**SyncEngine**
Index & DataRouter
Massive-Scale

**InsightEngine**
Real-Time RAG
Embedding Creation

**AgentEngine**
Tools & Orchestration
Agents, Tools, Observability

**DataStore**
Universal Storage Infrastructure
NFS, SMB, S3, NVMe/TCP

**DataBase**
Transactional & Analytical Database
Kafka, Python, Parquet, SQL, Similarity (Vectors)

**DataSpace** Globally-Distributed Data Computing

Hardware Platforms: Hewlett Packard Enterprise, DELL Technologies, SUPERMICRO, cisco

Traditional CSPs: aws, Google Cloud, EQUINIX

GPU Clouds: CoreWeave, core42, Lambda

and more…

# Thank You!

Maria Gutierrez – maria.gutierrez@vastdata.com
Jan Heichler – jan.heichler@vastdata.com