Contents

Introduction

Terraphim is a privacy-first AI assistant which works for you under your complete control. It starts as a local search engine, which can be configured to search for different types of content, such as Stackoverflow, Github, and local filesystem with a pre-defined folder, including Markdown Files. We use modern algorithms for AI/ML, data fusion, and distributed communication techniques to run AI assistants on the user's hardware, including not used mobile devices.

Methodological aspects

To personalise the search relevance, Terraphim uses knowledge graph, either local, in Obsidian or Logseq, or shared, in Atomic Server. We follow the methodology to create and update such a knowledge graph. First, each Terraphim role has its own separate knowledge graph that contains relevant concepts, with all synonyms. Usually, at the start a couple of key industry standards, reference process models, or handbooks are identified, and then such concepts and synonyms are extracted from them, using named entities recognition and expert opinions. Terraphim also can import curated industry taxonomies and produce a knowledge graph from it. The typical structure of a knowledge graph follows the SIPOC pattern – concepts at the input and output of some process are identified, and the process and its activity names are listed. After that, for each additional piece of data, for example, blog post, article, book, or video recording transcript, Terraphim identifies either role concepts from the knowledge graph or synonyms, extending the applicability of the knowledge graph that represents the domain, that is relevant for a selected Terraphim role. Using such textbook-quality datasets for semantic search enables delivery very accurate and precise search results. This methodology has been validated within the INCOSE community for the systems engineering handbook v.4 and the systems engineering digital process model v.1 and was recognised as a valid low-effort substitution of the formal model-based systems engineering, especially for brownfield systems engineering, reverse-engineering, and professional certified systems engineering certification, as well as for engineering education. Connecting the Terraphim role to the widely accepted industry standard helps to activate the accumulated wisdom of generations, augments communication by referring to common and recognised concepts and terms used in the standard, and also helps to formulate and reuse lessons learned, pass audits, following organisational guidelines, and match communication to the documentation and standard terms, used in this documentation and user interfaces of the professional software tools, that often exploit the standard terminology.

Scientific innovation and history

At the core of Terraphim - Unique knowledge graph embeddings: Terraphim AI is designed to be determinicstic precise and have 100% precision and recall: in an engineering context, the more complex the system is being built, the more important to be able to specify not only "the battery" but be able to differentiate between battery cr2365 and AA battery. Hence, Terraphim embeddings maintain the sequence of words in a sentence. Terraphim Graph Embedding maintain the position of the term in a

sentence and doesn't need traditional training techniques like "attention". At the same time, by allowing users to specify required synonyms manually and then re-build graph embeddings for the role within 20 milliseconds, Terraphim AI allows to match terms in different languages to the same concept, thus no need to run language detection in the pipeline. By giving users full and precise control over term matching within the selected context, there is no need for a stop-words dictionary: "The Pattern" is the specific name of the project, which was Terraphim AI's predecessor. "The" is normally considered a stop-word and will be filtered pre-tokenisation, and "The pattern" will not match the project's name using traditional techniques. The Pattern used more traditional techniques for graph embeddings, BERT QA and T5 summarization. To highlight the leanings and the major groundbreaking innovation for Terraphim AI: "The Patern" project was Platinum winner https://redis.com/blog/build-on-redis-hackathon-winners/ for Redis Hackathon challenge and at the time will outperform Nvidia ML pipeline for BERT QA inference on CPU and most likely in training. The prize demonstrated that external experts accepted the approach as innovative, and results were presented and discussed on the public lecture at Oxford University at Green Templeton College. The Pattern grew out of participation in two Kaggle data science competitions, where the original ML pipeline will not finish processing data in 6 days, The Pattern will process data for training in 6 hours and have under 2 millisecond inference. By rethinking Terraphim AI from the ground up we can achieve pipeline processing time in hundreds of milliseconds and query - knowledge graph-based inference in 5 to 10 nanoseconds. Our approach to Terraphim AI allows us and our users way faster and more practical AI applications, and we will engage the scientific community via an existing arrangement with Alan Turing UK NLP group. All projects are open-sourced and learnings are shared on https://terraphim.ai, https://reference-architecture.ai or https://thepattern.digital

Deployment innovation

Terraphim private cloud is designed with minimal shared of infrastructure between tenants, being close to hardware for high performance, effective use of resources and encryption in transit: Each user has it's own dedicated virtual machine, based on AWS Firecracker Virtual Machine (microVM). Firecracker Virtual Machine is an open source lightweight VM by AWS, which is the same technology behind AWS Lambda and AWS Firegate services. Use of Firecracker VM instead of docker or Kubernetes allows us:

1. Create a new environment for users under 1 second: 600-800 ms, including copying a new root partition for the user
2. Provide high performance and effective use of resources with response time in milliseconds and memory usage measured in megabytes - even for Machine Learning/AI
3. Maintain security and encryption in transit: each user has their own valid TLS certificate from Cloudflare to certify HTTPS entry point and encrypt traffic until Firecracker VM: traffic flow is via HTTPS server/load balancer (caddy) which direct traffic to corresponding tun/tap interface (kernel level, no docker/containerd) and VM attached directly to tun/tap interface. In our previous experiments with ML models training for The Pattern we noticed substantial performance degradation in network flow when using Kubernetes or Docker networks compared to standard

linux kernel based networks.. Terraphim Private cloud designed specifically with high throughput requirements and as close to the network as possible.

4. Firecracker VM allows us to embed additional services - like Redis for caching for the users into their VMs, with high performance (no need to spin additional containers, unlike with Docker), but user's data won't leave VM boundaries.
5. Terraphim AI private cloud has additional capabilities designed but not enabled by default - for example, the use of Zerotier VPN network to optimise routing between nodes and provide additional security and encryption layer In summary, Terraphim Private cloud is an evaluation environment for Terraphim AI users, which allows them to test Terraphim AI capabilities while maintaining privacy and ownership of their data. It's a substantial technological innovation in favour of customers and against current tech

Contributing

Guide for Terraphim AI contributors

You can run the full stack using Earthly. From the project root, execute the following command:

```
earthly ls
```

This will list all the available targets. You can then run the full stack using the following command:

```
earthly +pipeline
```

This will build the full stack using Earthly.

Local Development

If you want to develop without using Earthly, you need a local node.js, Yarn, and Rust environment.

Then you can run the following commands:

Install the sample data for `system_operator`

```
git clone https://github.com/terraphim/INCOSE-Systems-Engineering-Handbook.git /tmp/system_operator/
```

Run the backend

```
cargo run
```

Run the frontend in development mode:

```
yarn run dev
```

from ./desktop folder

to compile tauri in dev mode run:

```
yarn run tauri dev
```

Architecture

Use mermaid to draw the architecture of the system.

```
graph TD;
    A-->B;
    A-->C;
    B-->D;
    C-->D;
```