WHAT IS JARVIS

JARVIS is a system that allows the user to upload files to it and have them evaluated with all the best malware detection software. It is being created as an alternative to expensive third-party virus detection software like VirusTotal [1].

JARVIS will utilize all the most prominent and accurate malware detection methods from respected sources to assess a file.

Joining during production of the project meant that a significant amount of the foundation had been completed and more is yet to be done at the end of the semester. However, this team played a key role with respect to many aspects of the design and implementation stages.

In the proposal, it was outlined that it would be crucial to understand the pre-existing codebases for the team to coherently extend it. JARVIS includes a frontend interface (Cyberdeck), web services based on socket.io, vector store for similarity search (Homer), and a database for malware code and distributed executor (Vision). The primary focus of this team was on the vector store and the database work.

Through a mix of independent work and various meetings with the project lead (Professor S. Ding), the team studied how the program files interacted with each other and what dependencies were necessary for the program to run. Notable important requirements include having the proper version of python installed and setting up a working local development environment.

TESTING ANNS

One of the most important tasks undertaken by the team was the research and implementation of multiple artificial neural networks (ANNs) for use in JARVIS. They are needed because the system heavily relies on the performance and reliability of vector-to-vector search algorithms when a large amount of data is passed into the system.

The main benefit of Faiss is that it is optimized for search speed and memory usage. It also supports multiple methods of similarity search, and support GPU approach for these methods.

These features allow for incremental indexing of vectors, which allows system to change the index directly instead of having to reload the entire index. The main disadvantage of this approach is its memory usage. Compared to other solutions we looked at, HNSWlib's memory usage is much larger. Due to the limitations of the testing machines, we were only able to estimate the memory required to store the size of indexes that the system will be using during deployment.

FAISS

Facebook AI Similarity Search is a library for efficient similarity search and clustering of dense vectors. The main benefit of Faiss is that it is optimized for search speed and memory usage. It also support multiple method of similarity search, and support GPU approach for these methods.

The task of migrating the vector database included migrating and verifying code to implement HNSWlib and Faiss into the JARVIS Version 3 codebase. This task also included writing test cases to ensure the functions performed correctly.

The creation of the database was an essential part of the project for storing results. The team created a design schema to handle the attributes of files and then using a tool called peewee imported a testing dataset to the SQLite database.

Lastly, to better understand the result of database query. The team chose to use Apache Superset, a data exploration and visualization platform to display the result from Result Database (Task #2).

1. https://www.virus Total.com/gpu/home/upload
2. https://github.com/erikbern/ann-benchmarks
3. https://github.com/nmslib/hsnowlib

Figure 1: What is vector similarity search?

Figure 2: ANN Performance Comparison. [2]

Based on the ANN performance comparison it was determined that three candidates would be selected for research based on performance and memory capabilities.

HNSWlib

An open-source library for the implementation of hierarchical navigable small world graphs [3]. HNSW is a very popular algorithm for approximate nearest neighbours searches. It was chosen because it is one of the top-performing indexes for vector similarity searches. The advantages of using HNSWlib are its scalable indexing and support for element deletion.

Figure 3: Comparison of ANN Outputs

Figure 4: Workflow of JVD3

DEPLOYMENT ENHANCEMENT

JARVIS is set to be deployed this summer and as a result there are many things that must be done to deploy the software successfully. The team was tasked with three main objectives for this component. First, implement/verify/migrate vector db. Second the creation of the results database. The third is the testing of Apache Superset for deployment.

The task of migrating the vector database included migrating and verifying code to implement HNSWlib and Faiss into the JARVIS Version 3 codebase. This task also included writing test cases to ensure the functions performed correctly.

The creation of the database was an essential part of the project for storing results. The team created a design schema to handle the attributes of files and then using a tool called peewee imported a testing dataset to the SQLite database.