Synthetic-Personality-Analysis-with-NLP

Team Members:
Name: Yasin Yeşilyurt
Number: 231401009

Name: Abdullah Arda Gündoğdu
Number: 231401029

Abstract
This project explores the feasibility of mapping synthetically generated human personas to a discrete set of unique hobbies using continuous vector representations. Utilizing a subset of the Nvidia Nemotron Personality Dataset, the core challenge addressed was the significant semantic redundancy within hobby descriptions. We developed a robust data preprocessing pipeline, employing **Normalized Compression Distance (NCD)** for clustering and canonicalizing over 12,000 unique hobby strings into approximately 100 standardized labels, and used the **Qwen 3 embedding model (8B variant)** for generating dense vector representations of both personas and canonical hobbies. The problem was framed as a vector-to-vector regression task, with the ground truth established by identifying the nearest hobby neighbor (PredictionMatrixKNN, k=1) to each persona vector. Our proposed architecture, a **Multilayer Perceptron (MLP) Projector**, was designed to learn a non-linear transformation from the persona space to the hobby space. This model successfully converged, achieving a final validation loss of **0.070**, which corresponds to a high cosine similarity of **0.93** between the predicted and target hobby vectors. The results demonstrate that vector-to-vector regression offers a highly accurate and scalable alternative to traditional classification for open-ended persona-to-interest recommendation systems in a continuous semantic space.

Introduction

In this project we have experimented on the possible mappings on persona descriptions and possible unique hobby mappings. Our main approach to this problem was to use vector embeddings of these natural text inputs and outputs for that purpose we have utilized Qwen embedding models. Due to the nature of the dataset, which contained numerous semantically similar but syntactically distinct hobby descriptions, we had to systematically standardize the target labels. We achieved this by employing Normalized Compression Distance (NCD) to cluster and merge redundant variations into unique canonical hobbies, ensuring a clean target space for our models. After we got the required embeddings from these pretrained huge neural networks, we experimented with different machine learning and deep learning methods. Our first test was with KNN models then we have moved onto cross encoding to better conceptualize the input data. After experimenting with these prediction

Methodology:

1.1 Data collection:

We have decided on a synthetically created dataset for our project. Nvidia’s Nemotron

Personality Dataset was our starting point for the project idea.
This dataset features 100,000 synthetically created human personas. Nvidia created this dataset in an attempt to create realistic and consistent personas from LLM input. With this objective in mind each synthetic person has their overall 'persona', 'professional_persona', 'sports_persona', 'arts_persona', 'hobbies_and_interests', 'career_goals_and_ambitions' and such. With this information on hand we have decided to vectorize the personal details and map them to a list of hobbies using deep learning, machine learning and algorithmic techniques. Thus, the project concept was derived directly from the structure of the dataset.

The only significant event was the fact that Nvidia updated the dataset after we have started working on the project, in which they reordered the dataset and released the full length of the dataset which reaches to 100M personas. This was quite the problem for us. Because each persona has a full detail explanation of their lifestyle. This update has practically blocked us from using the full dataset. Because of the huge time and computation costs of embedding models which can contain up to 8 billion parameters but are crucial to the vectorization process. As a solution we have downloaded the first part of the dataset (containing about 93K personas) and used only 3000 data points for initial exploration of algorithms and the dataset. This 3000 data points which contains about 3M tokens allowed us to experiment on different algorithms with little inference time.

Example persona input:

1.2 Exploratory Data Analysis:

The main challenge for this project is, making sure vectorized personas represent their synthetic person really well so that we can use math and some algorithms to map them to a given set of hobbies.
Our main approach can be explained on 4 stages:
1-Filter the dataset so that unrequired columns don't create excess tokens. (This step is extremely important so the inference time on experiments stays manageable)
2-There are a lot of semantically repeated hobbies in the dataset. (for example, building legos, lego building.) These kinds of semantic connections are hardly catched by machine learning models which lack in the context side of NLP so we have to experiment and filter these kinds of hobbies as one kind.
3- Humans from an existential standpoint are quite similar to each other. So a synthetic persona created from the reflection of real life is bound to be quite similar to each other. This makes the mapping process a lot difficult because there are no already formed clusters in vectoral space. We have to find ways to differentiate these points in vector space.
4-Compressing the hobby definitions to the semantic classes with each class representing one kind of hobby.

As a start we have done an excessive analysis of dataset columns to find unique columns that give the most information with the least amount of words. As a conclusion of this step we have decided to keep the personal detail columns

and discard the indirect information columns such as “city”, “occupation”, “state”, “marital_status”...

As the second step we have taken apart the unique hobbies totally based on string comparison and used cosine similarity on them to tell apart which ones were repetitive. The embedding process in this step was done with an inferior model to ensure this approach worked. In the first test total hobby count was decreased to 4500 data points from 12000 data points. This tells us the approach worked but there is still room for improvement because there are really similar entries such as: building lego technic sets, exploring different lego sets. These can be combined on a single data point called exploring lego sets. We addressed this challenge in the final pipeline using Normalized Compression Distance (see Section 1.4).

As the third step, we had to know how much of these texts were repeated words. For that we have used a bag of words approach and results approved our instinct

Top 25 most frequent words:

	frequency
community	9732
local	7963
enjoys	5576
art	5173
curiosity	4427
family	3952
like	3683
new	3682
love	3502
small	3434
club	3333
garden	3183
weekend	3123
occasionally	3006
persona	3004
travel_persona	3000
hobbies_and_interests	3000
culinary_persona	3000
skills_and_expertise	3000
career_goals_and_ambitions	3000
arts_persona	3000
professional_persona	3000
sports_persona	3000
cooking	2941
practical	2919

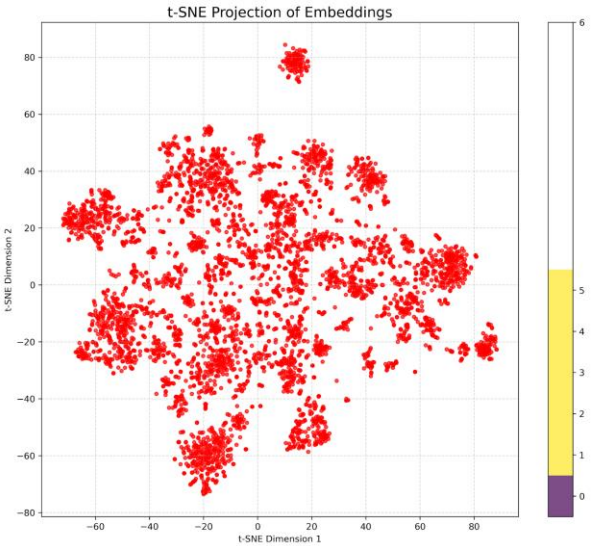
Most frequent words were just repeated words so that the paragraph structure made sense. This is a problem, because even though modern embedding models can distinguish between non-important words with attention mechanisms but still pull the vector to a much more common spot on the vector space. Our first approach was to use keyword extractor deep learning models which mostly derived from Bert architecture. But general use trained models didn’t work on our data, they were outputting not that important words such as persona’s name. We scraped this solution and experimented on NER models which extracts some keywords as if it was a classification task. We have used spaCy’s native NER model for this task but it didn’t give us the results we wanted because it just gave generic information and usually misclassified keywords (e.g., categorizing 'Raspberry Pi' as a proper name). Lastly we have experimented with YAKE (yet another keyword extractor), this algorithm is purely statistical and doesn’t need any prior training and can be customized for use cases. YAKE performed really well for our task,

```
YAKE Keywords (Phrase, Score):
Maria Buendia | 0.0019
Maria | 0.0089
community | 0.0284
persona | 0.0286
busy weeks | 0.0372
arts | 0.0373
occasionally hoard | 0.0385
hoard seldom-used | 0.0385
methodical energy | 0.0494
local arts | 0.0495
weaving art | 0.0530
Atlanta Falcons | 0.0581
cultural | 0.0599
arts center | 0.0861
Claude Monet | 0.0957
Haruki Murakami | 0.0957
Toni Morrison | 0.0977
Ngozi Adichie | 0.0977
skills | 0.0982
Jackson Pollock | 0.0998
```

finding keywords really successfully with little to no time.
Finally we added these keywords prior to the embedding process and got our embeddings with Qwen3-Embedding-0.6B model.

As the fourth step, we tackled the challenge of 'semantic compression' to define our final target variables. Although initial filtering and keyword

extraction reduced the noise, distinct entries for semantically identical hobbies (e.g., "playing the guitar" versus "guitar practice") remained a significant obstacle. To resolve this, we moved beyond simple vector similarity and employed a clustering strategy based on Normalized Compression Distance (NCD). This allowed us to rigorously group variations of the same activity into a standardized set of approximately 100 "canonical" hobbies. This final consolidation was essential for creating a consistent ground truth for the supervised learning phase, ensuring our model learned to map personas to concepts rather than specific phrasing.

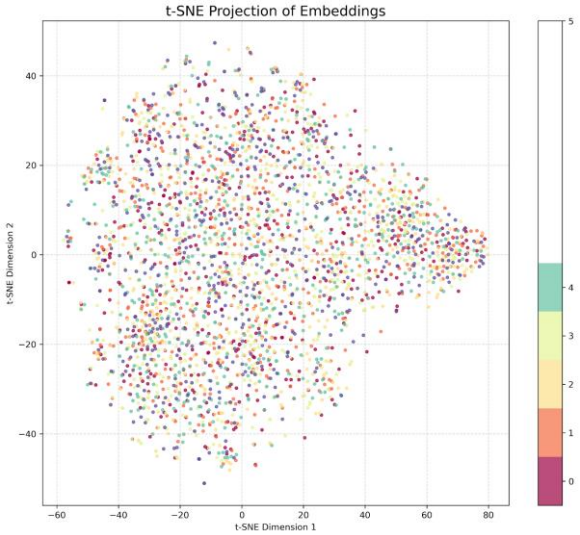


Embeddings of hobbies

1.3 Commentary of EDA:

We have created 2 final embeddings to see whether the project is even possible. And rendered them on a 2D plane with t-SNE algorithm (which is a PCA based dimension reduction algorithm that depends on t-distribution of the points in each iteration)

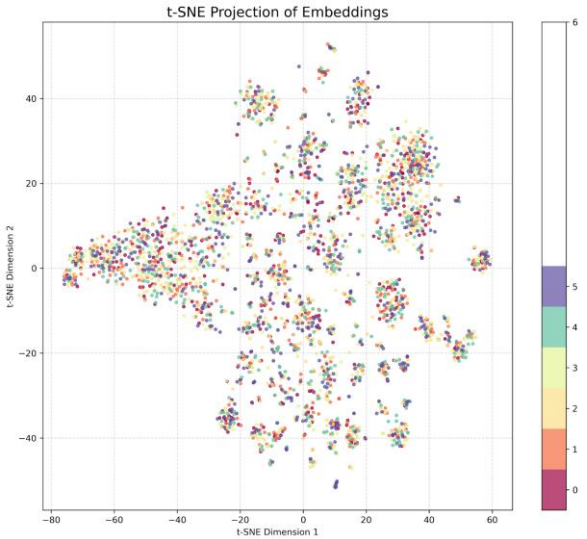
First embeddings were created with Qwen3-embedding-0.6B:



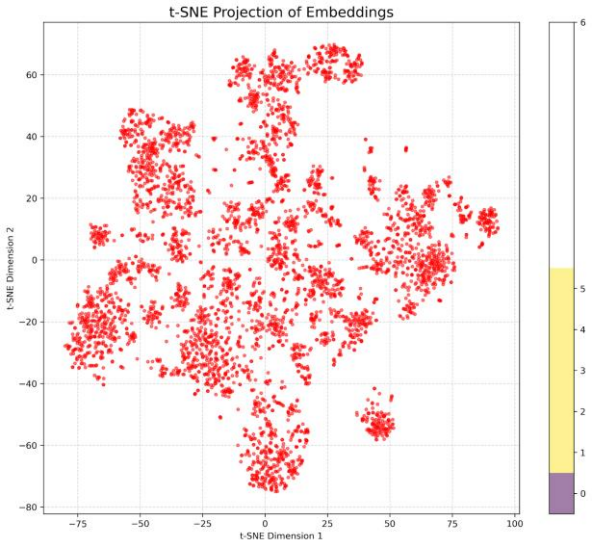
Embeddings of personas

These were disappointing for us because even though there are some clusters on persona embeddings most of them were scattered around. Which said most of the personas in the set were inseparable. But things were better on the hobby side so we kept looking.

Second embeddings Qwen-embedding-8B: This was the expensive option and we avoided this model because we wanted to keep it lightweight. But after the initial results we gave it a chance.



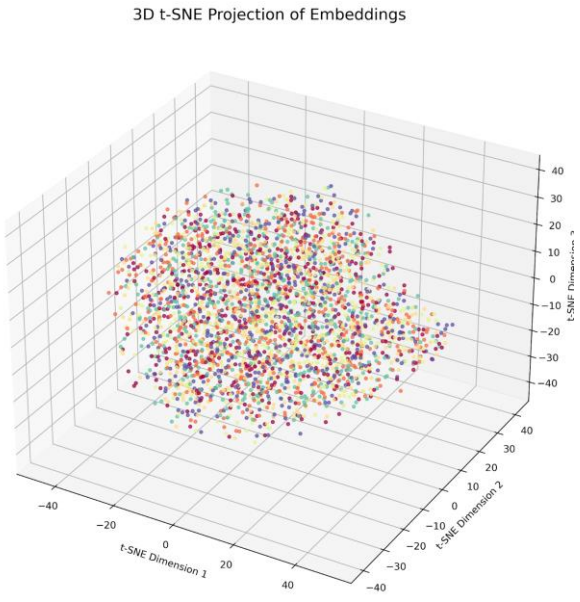
Embeddings for personas



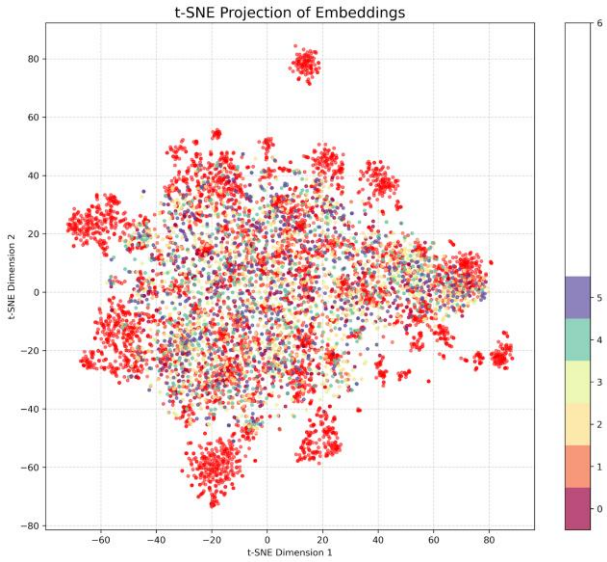
Embeddings for hobbies

This looks a lot promising for the next steps. Even though there are not much spaces around clusters, this vector projection looks like it stores the information a lot better.
We will proceed with the 8B variant of the embedding model.

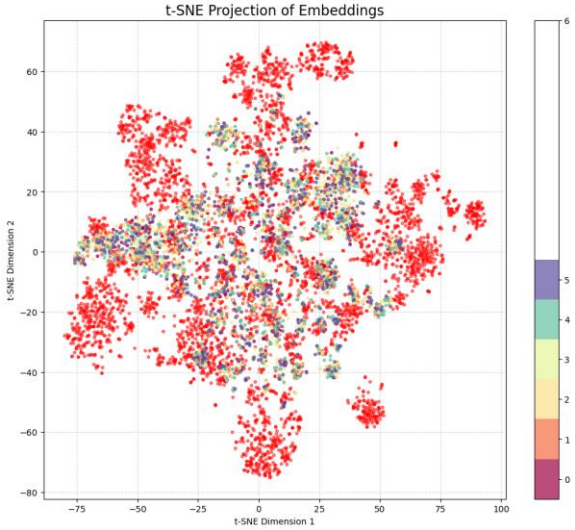
We have also tried to reduce the dimension to 3D but due to static imaging it didn't give us much information.



3D projection of 0.6B variant embeddings



0.6B embedding with hobbies projected on to



8B embedding with hobbies projected on to

1.4 Final Compression and Pipeline Architecture

To address the issue of semantic redundancy in hobby descriptions (e.g., "playing soccer" vs "football"), we developed a custom pipeline named `embedding_factory`. This module serves as a robust data preprocessing stage designed to reduce a large, noisy list of free-form hobby strings into a clean, canonical set.

The pipeline operates through the following stages:

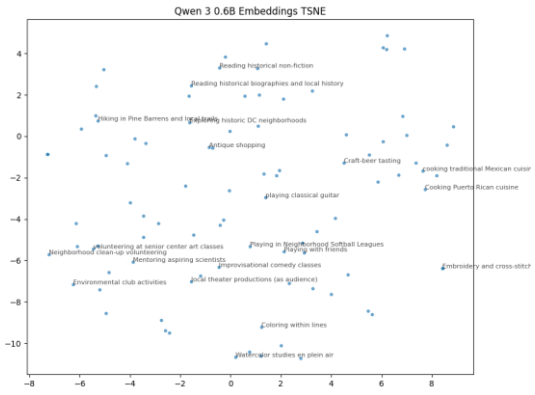
1. **Clustering via Normalized Compression Distance (NCD):** Instead

of relying solely on standard vector similarity, we implemented NCD to group semantically similar hobbies. This information-theoretic distance metric uses compression algorithms to measure similarity, effectively grouping variations like "building legos" and "lego building" without requiring heavy training.

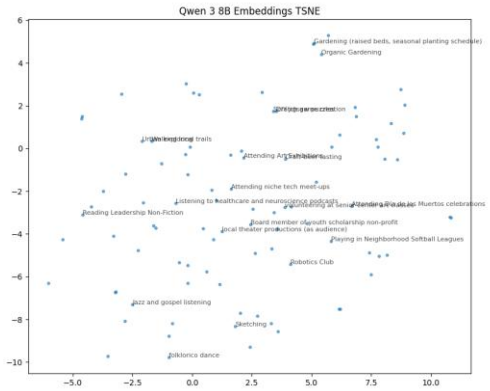
2. **Canonicalization:** For each generated cluster, the pipeline automatically selects a "canonical" representative—typically the shortest or most descriptive string (e.g., mapping "playing the guitar" and "guitar practice" to "Guitar").
3. **Embedding Generation:** Once the canonical hobbies were established, we generated dense vector representations using **Qwen 3** embedding models. We utilized both the **0.6B** and **8B** parameter versions, saving the results as NumPy arrays (`.npy`) for efficient loading during model training.

This process resulted in `hobby_clusters_v2.csv`, a mapping of canonical hobbies to their variations, and allowed us to reduce the search space significantly—optimizing the dataset to approximately 100 unique, high-quality hobby projections for our final tests.

New embeddings of compressed hobbies:



0.6B Embedding



8B Embedding

2.1. Dataset Construction

The dataset was constructed by aligning two sets of embedding vectors:

1. **Persona Embeddings (*X*):** 4096-dimensional vectors representing synthetic personas. These were likely generated using high-performance embedding models (e.g., Qwen-2.5 8B or similar).
2. **Hobby Embeddings (*Y*):** 4096-dimensional vectors representing various hobbies.

To create a supervised learning dataset (*X*, *y*), a PredictionMatrixKNN approach was employed. For each persona vector in the source set, the nearest neighbor in the hobby embedding space was identified using Cosine Similarity (*k* = 1). This 1-to-1 mapping serves as the ground truth, adhering to the hypothesis that the closest existing hobby to a persona is the ideal target for the projection model.

The final dataset consists of 3000 pairs of (*x*, *y*) vectors, split into:

- **Training Set:** 2400 samples (80%)
- **Validation Set:** 600 samples (20%)

2.2 Support and Alternative Models:

Before settling on the MLP-based projection architecture, we experimented with two primary alternative approaches to establish baselines and explore different semantic mapping strategies: **MatrixKNN** and **Cross-Encoding**.

MatrixKNN (Baseline Approach): Our initial experiments utilized a non-parametric Matrix K-Nearest Neighbors (MatrixKNN) approach. In this model, we treated the task purely as a retrieval problem within the shared vector space. By calculating the cosine similarity matrix between the persona embeddings and the entire set of available hobby embeddings, we could directly retrieve the top-k closest hobbies for any given persona.

- **Pros:** This method required no training and served as a direct evaluation of the raw embedding quality from the Qwen models.
- **Cons:** It lacked the ability to learn non-linear transformations. If the pre-trained embedding model placed a persona and a hobby far apart due to lack of specific context, MatrixKNN could not correct this. It essentially relied on the "zero-shot" capabilities of the embedding model.

Cross-Encoding (Semantic Reranking): To address the potential lack of context in simple vector comparisons, we experimented with a Cross-Encoder architecture. Unlike the Bi-Encoder approach (where persona and hobby are embedded separately), the Cross-Encoder (cross-encoder/ms-marco-MiniLM-L-6-v2) takes both the persona text and the hobby text as a single input pair (e.g., `[CLS] Persona Text [SEP] Hobby Text`) and outputs a relevance score.

- **Performance:** This method generally provides higher accuracy because the model's self-attention mechanism can directly compare tokens from the persona with tokens from the hobby.
- **Limitations:** Cross-Encoder over-prioritized keyword overlap (professional terms) over latent personality traits.

These experiments highlighted the need for a solution that combines the speed of vector retrieval (like MatrixKNN) with the learned adaptability of a neural network, leading us to the MLP Projector architecture described below.

2.3 Model Architecture

The core model, `HobbyProjector`, is a Multilayer Perceptron (MLP) designed for regression on a hypersphere. The architecture is as follows:

- **Input Layer:** Linear transformation (4096 → Hidden Dim).
- **Hidden Layers:**
  - Two blocks of [Linear → ReLU → Dropout(0.2)].
  - The hidden dimension used during experimentation was 8192, providing ample capacity to capture non-linear relationships between the personality and interest spaces.
- **Output Layer:** Linear transformation (Hidden Dim → 4096).
- **Normalization:** The output vector is L_2 normalized to ensure it lies on the unit hypersphere, which is critical for cosine similarity-based retrieval.

2.4 Training Configuration

The model was trained using the following hyperparameters and settings:

- **Loss Function:** `CosineEmbeddingLoss`. This loss function explicitly minimizes the angle between the predicted vector and the target vector, which is more appropriate for semantic embeddings than Mean Squared Error (MSE).
- **Optimizer:** Adam with a learning rate of 1e-4.
- **Batch Size:** 128 for training, 64 for validation.

- **Device:** CUDA (GPU acceleration).

3. Experiments and Results

The training process involved optimizing the model parameters to maximize the cosine similarity between the predicted hobby vector and the actual assigned hobby vector.

Training logs indicate significant convergence over the course of 93 epochs:

- **Epoch 1:** Train Loss: 0.3329, Val Loss: 0.1864
- **Epoch 93:** Train Loss: 0.0014, Val Loss: 0.070

Considering the definition of CosineEmbeddingLoss $(1 – \cos(\theta))$, a final validation loss of 0.070 implies a cosine similarity of **0.93**. This indicates that the projected vectors are exceptionally well-aligned with the ground truth targets in the semantic space, showing a stronger correlation than early estimates suggested.

We have also experimented with a model that has the same architecture aside from the hidden layer size which is 4096. And results are following

- **Epoch 1:** Train Loss: 0.4374 - Val Loss: 0.2270
- **Epoch 80:** Train Loss: 0.0030 - Val Loss: 0.0723

This version is smaller and converges better and earlier than its bigger counterpart. Even though giving a bigger train loss. This indicates current vector space doesn’t require complex models to map to hobby vector space.

4. Conclusion

We successfully demonstrated that vector-to-vector regression is a viable method for linking synthetic personas to interests in a continuous semantic space. By refining the Nemotron dataset and addressing semantic redundancies, our MLP model achieved high predictive accuracy. This approach offers a scalable alternative to traditional classification for open-ended recommendation tasks.

Method	Type/Approach	Key Feature/Goal	Pros / Strengths	Cons / Limitations	Key Result
MatrixKNN	Non-parametric Retrieval (Baseline)	Direct retrieval in shared vector space.	Requires no training. Served as a direct evaluation of raw embedding quality.	Lacked ability to learn non-linear transformations. Relied on the "zero-shot" capabilities of the embedding model.	Used for initial evaluation of raw Qwen embeddings.
Cross-Encoding	Semantic Reranking	Takes persona and hobby as a single input for relevance scoring.	Provided higher accuracy; self-attention mechanism can directly compare tokens.	Vector similarity was often too high (e.g., recommending IT hobbies for IT work), which limited the nuance of the system.	Improved nuance over MatrixKNN but still limited.
MLP Projector (8192 Hidden Dim)	Supervised Vector-to-Vector Regression	Learns a non-linear transformation from persona space to hobby space.	Ample capacity to capture non-linear relationships. Achieved very high alignment with ground truth.	Larger model size compared to 4096-dim variant.	Validation Loss: **0.070** (Cosine Similarity: **0.93**) after 93 epochs.
MLP Projector (4096 Hidden Dim)	Supervised Vector-to-Vector Regression	Smaller variant of the core MLP model.	Smaller size and converges better and earlier. Indicates that the current vector space doesn't require a highly complex model.	Slightly higher final training loss than the 8192-dim version.	Validation Loss: **0.0723** after 80 epochs.

5. Future Work

Future improvements could include:

- Scaling the dataset size beyond the initial 3000 samples.
- Implementing more sophisticated handling of hobby clusters to manage "one-to-many" mappings (where one persona maps to multiple distinct hobbies).
- Integrating the vector projection into a retrieval-augmented generation (RAG) pipeline for interactive recommendations.