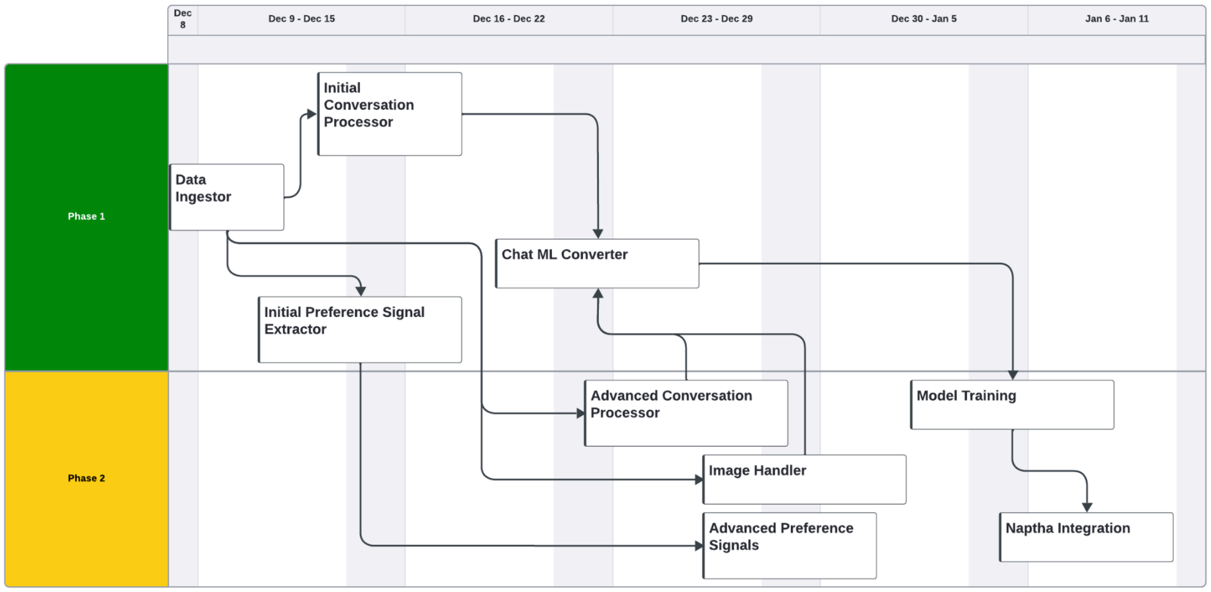
George Walker
Peter Graziano
Nathanael Hoyle
Claude

# Twitter/X Data to Naptha Personas: Implementation Plan

## Introduction

The TPOT (This Part of Twitter) community represents a unique nexus of machine learning thought leadership, including influential figures like Emmett Shear and Zvi Moskowitz. Their discussions, preserved in Community Archive's 12.4M (and growing) tweet dataset, contain valuable insights about machine learning, AI safety, and technological progress. This implementation plan details how we'll transform this corpus into fine-tuning datasets, preference pair datasets, at least one language model, and at least one AI Agent or Persona in the Naptha.ai Hub, while building reusable infrastructure for processing Twitter/X data.



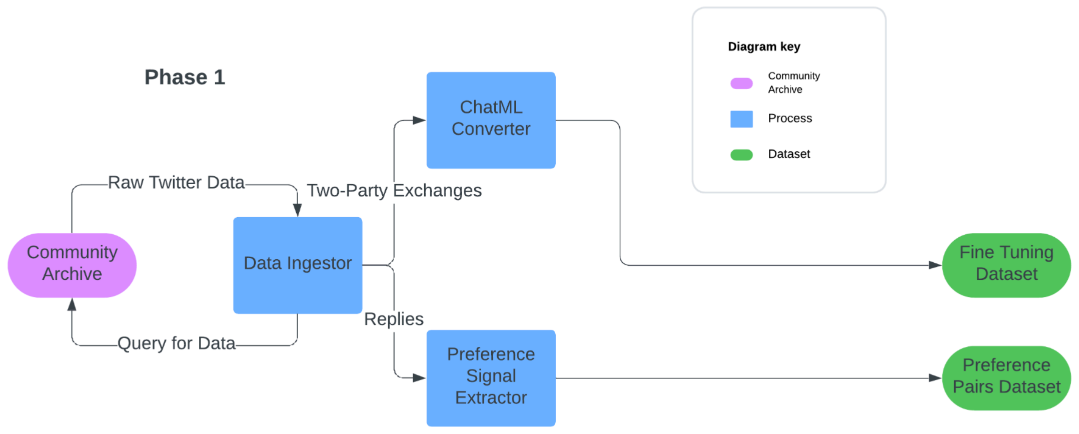
The implementation follows a three-phase strategy:

Phase 1 establishes the core functionality for the complete processing pipeline: automatic data ingesting, filtering and processing two-party conversations into ChatML format, mining basic preference signals, scraping linked content, and initial dataset publication. This creates immediate value while laying groundwork for more sophisticated features.

Phase 2 will enhance the pipeline with multi-party conversation handling, advanced preference signals, training a language model, and publishing at least one Agent or Persona to the Naptha.ai hub. Phase 2 concludes the features promised in the scope of the current Naptha.ai grant.

Phase 3 (subject to remaining hours left in the grant) includes exploring stretch goals and opportunities for future work: features may include producing and publishing datasets for training multimodal models, back-translation to create prompts tuned for low loss on top-level tweets, empirical evaluation of different filtering techniques for fine-tuning and preference pairs, and integration of reward models into model training via reinforcement learning. All of these are deeply interesting research directions, but again: they fall outside of the existing scope of the grant, and will only be pursued by this team if we have allocated hours remaining after delivering Phase 2.

## Phase 1: Core Pipeline Implementation



### Data Ingestion Service

The ingestion service polls Community Archive's user directory, downloading new or updated user archives. Implementation is straightforward: a Python script using the requests library fetches JSONs and push the raw archives to HuggingFace datasets prior to further processing.

Raw files are stored with a simple naming convention: {username}_{ISO_8601_upload_date}.json. The 'notes' section passes through a conversion stage to format them as tweets. This creates a clean, uniform data flow while preserving all original content.

### Conversation Processing

The conversation processor reconstructs two-party exchanges.

Core processing flow:
1. Follow reply_to_id chains to build conversation trees
2. Identify two-party exchanges within these trees
3. Concatenate consecutive replies from the same user with " ||| " delimiter
4. Integrate quote tweets with <qt> </qt> tags for downstream processing.

The processor outputs conversations ready for ChatML conversion, maintaining clear thread structure while stripping unnecessary metadata.

### ChatML Conversion

The conversion system transforms Twitter exchanges into ChatML format suitable for model training. System messages follow a deliberately simple template:
"User is {username}. Assistant is {username}. Date is {ISO_8601_date}."

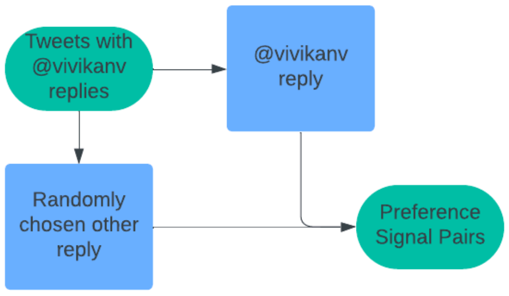
URL handling merges external content directly into messages:
- Extract text from all URLs in conversation.
- Maximum 50k words per URL
- Placeholder for images in Phase 1
- Simple string concatenation into message body (delimited with “<ref> </ref>” tags to separate it from the body of the tweet that references it).

To enable flexible downstream usage, we alternate between display names and handles based on conversation index. This teaches models the equivalence of these identifiers while maintaining clean training data.

### Preference Signal Dataset

Initial preference signal extraction uses @visakanv's replies as a gold standard, because @visakanv famously wrote the book on twitter ‘reply game’ for TPOT. For each @visakanv reply, we:
1. Identify the original tweet
2. Find all other replies to that tweet
3. Randomly select one of the replies (not @viskanv’s) as the rejected example
4. Create (accepted, rejected) training pairs

This creates a foundational preference dataset without complex metrics or filtering. The output is filtered to original tweets, accepted replies, rejected replies, and basic metadata, suitable for training a reward model.



### Dataset Publication

All processed datasets publish directly to HuggingFace:
- Raw archive JSONs
- Processed ChatML conversations
- Preference pairs dataset

Documentation will follow the HuggingFace Dataset Card format, providing basic statistics and format specifications. The processing pipeline publishes these datasets to HuggingFace, linked from the dataset documentation on GitHub.

### Error Handling and Logging

While we're not building mission-critical infrastructure, basic logging helps track pipeline operation:
- Failed downloads logged but not retried
- Malformed conversations simply skipped
- Basic counts of processed items maintained

### Initial Dataset Statistics

The pipeline tracks basic metrics:
- Number of two-party conversations extracted
- Average conversation length
- URL extraction success rate
- Preference pair counts

These metrics help users understand dataset characteristics without overcomplicating the infrastructure.

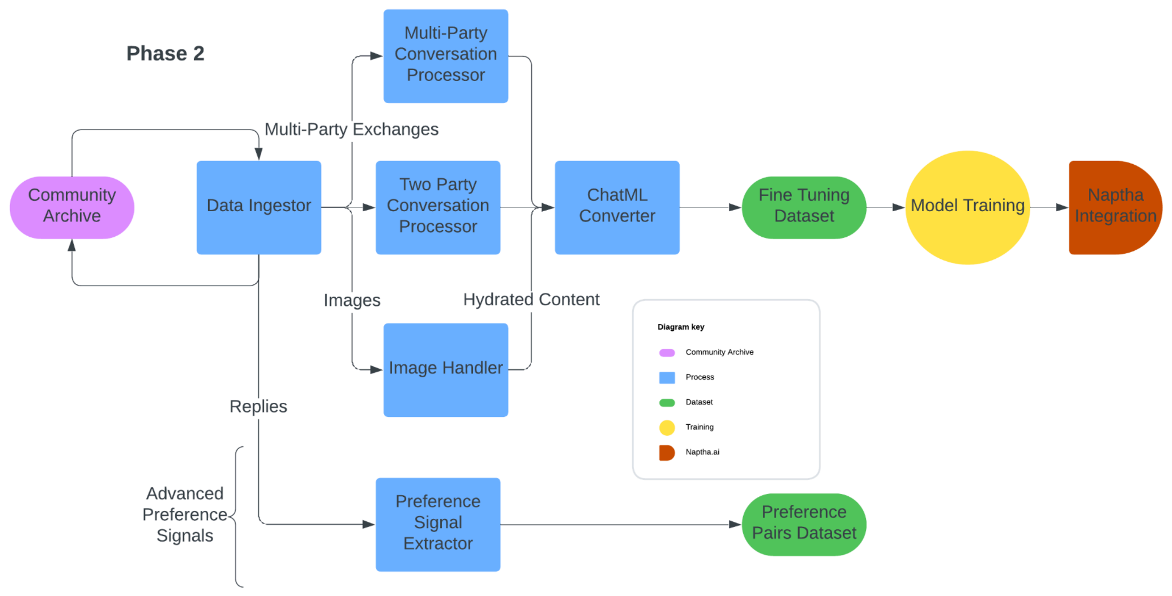
### Extensibility for Phases 2 and 3

The Phase 1 implementation deliberately leaves hooks for future enhancements:
- Multi-party conversation handling
- Advanced preference signals
- Image and media processing
- Naptha platform integration

Each component uses clear interfaces that can be extended without rewriting core functionality.

This Phase 1 implementation creates immediate value through usable datasets and tooling while establishing infrastructure for more sophisticated features in later phases. The focus remains on creating functional, extensible infrastructure rather than perfect systems. Each component is designed to be easily modified as we learn from usage and prepare for Phases 2 and 3. The focus on simplicity and extensibility ensures we can iterate quickly based on user feedback and emerging requirements.

## Phase 2: Improving Data, Training a Model, Integrate with Naptha.ai



### Multi-Party Conversation Processing

The multi-party conversation processor transforms complex Twitter discussions into structured training data while preserving the richness of multi-participant discourse. At its core, a streaming quality metric system uses t-digest to calculate running percentiles for at least four key metrics: like counts, retweet counts, follower counts, and message length. This chronological processing naturally handles the evolution of engagement metrics over time, creating normalized composite quality scores for our fine-tuning and preference datasets.

Conversation branches receive special attention through a three-part system. First, the processor tracks tweet frequency across branches, identifying repeated content. Second, it implements intelligent masking to ensure each tweet appears only once unmasked in the dataset, allowing retention of all conversational branches while permitting multi-epoch finetuning. Third, it preserves quote tweets by wrapping them in distinctive tags (<qt></qt>), allowing downstream models to understand these as specialized context rather than direct responses. These <qt></qt> tags may then be mapped to special tokens for training.

The system automatically splits processed conversations into training, development, and test sets, maintaining conversation integrity within each split. This creates a valuable benchmark for researchers fine-tuning or applying reinforcement learning methods with the dataset. All processed data maintains original tweet IDs and timestamps metadata, enabling easy reference back to source material.

### Dataset Hydration: Scraping Images and Articles

The image pipeline creates a systematic approach to handling Twitter's visual content and linked articles.
Images are downloaded and stored with a consistent naming convention: {tweet_id}_{image_index}.{extension}. Each image maintains metadata linking it to its tweet context, including timestamp, author, and conversation thread ID. While we may not get to fine-tuning a multimodal model under this grant (see Phase 3), it is important to 'hydrate' the dataset with these data to permit this to be done in future work.

The pipeline handles failed downloads gracefully, logging failures without interrupting processing and preserving text content when images are unavailable. URL content from tweets is extracted and stored with a 50k word limit per URL, maintaining a balance between comprehensive context and practical storage constraints.

Textual content will be integrated into the fine-tuning datasets we push to the Naphtha.ai Huggingface. To begin with, we intend to include this content in the first user message in the ChatML format within distinctive tags (which may be mapped to special tokens for training).

### Advanced Preference Signal Extraction
The preference signal system leverages our streaming architecture to create sophisticated training pairs for reward modeling. We will be using t-digest to calculate running percentiles of engagement metrics to normalize those values over time. This enables identification of high-quality later replies as accepted examples, matched with earlier responses of similar length as rejected examples.

Preference pairs are stored in a simple but comprehensive format containing:
- Original tweet content and metadata
- Accepted reply (higher quality, later in time)
- Rejected reply (lower quality, earlier, length-matched)
- Associated tweet IDs and timestamps
- Normalized quality scores for each component

This standardized format makes it easy for researchers to filter and adapt the preference pairs for their specific needs while maintaining provenance information.

### Model Training

We will train an "instruct" model with our fine tuning dataset on Prime Intellect's GPU infrastructure in partnership with Naptha.AI. After a "sanity check" training on a small model (most likely Qwen-2.5-1.5B), we will train Qwen-2.5-32B on our full dataset. These members of the Qwen-2.5 model family recommend themselves because they are highly overtrained (18T tokens), performance leading in their respective size classes, permissively licensed (Apache 2.0), and amenable to local inference on affordable hardware many developers already own.

### Naptha Platform Integration

Integration with Naptha's platform happens through their hub API, with our pipeline creating deployable agent or persona modules. At time of writing, it is still unclear whether Agents or Personas is the appropriate abstraction for this, and we will be in touch with the team on the Naptha.ai discord to clarify the means of integration that best serves the Naptha.ai community.

A proof-of-concept multi-agent chat environment demonstrates practical usage, while comprehensive documentation covers integration patterns and best practices. This includes example code for common usage scenarios and guidelines for adapting the pipeline components for custom applications.

The entire pipeline maintains consistent logging throughout, tracking processing statistics and enabling easy monitoring of dataset creation. This includes counts of processed conversations, image download success rates, preference pair generation statistics, and quality score distributions. These metrics help researchers understand dataset characteristics and adjust processing parameters as needed.

# Phase 3: Future Research Directions and Extensions

The following capabilities represent ambitious extensions beyond our core requirements and current project scope.

While these features are outside the scope of the current grant, and will only be pursued during it, time permitting after completion of Phase 2 milestones, we believe there is value in documenting them here as possible future directions for research, study, and further development. These proposals represent exciting possibilities for advancing the field of social media-based AI personas.

### Creating More Datasets
The ChatML fine tuning dataset we create in Phase 2 will be guided by our prior understanding of best practices from the literature in consultation with our contacts in the model fine-tuning community. Time permitting, we intend to create more fine tuning datasets, of different sizes, based on a variety of filtering criteria, and different formatting (i.e., different mappings to the ChatML format). The objective here would be to increase model performance, and reduce dataset size.

There is also potential for creating “Meme” datasets with virality potential (i.e. to drive user and community engagement). For instance, one may subset the data based on particularly large or famous accounts within TPOT, such as @visakanv. An alternate example: a “Wifejack” dataset comprising interactions between @selentelechia’s and @eigenrobot, her husband.

### Advanced Multimodal Processing
Future work could significantly expand our handling of visual content beyond basic image storage. A comprehensive multimodal processing system would support multiple emerging vision-language models, starting with Pixtral but designed to adapt as new models emerge. The

system would pre-tokenize images using model-specific vision encoders, maintaining metadata that enables future reprocessing as models improve.

A particularly innovative direction involves detecting and special-casing screenshots of tweets. These could be processed through OCR and converted into quote-tweet format, making them more accessible to language models. This capability would be especially valuable for technical discussions where screenshots often capture complex threads or code snippets.

Another avenue we are considering in future work is using a vision model to describe images for language models, to maintain the context of a tweet in the training set for language models which by definition cannot see them. An exciting research direction is an optimization loop in which a vision model iteratively describes an image, calls out to an image generation model to generate an image, compares the result with the original, and then iterates on the prompt.

### Back-Translation: Prompting Top-level Tweets
Back-translation is a synthetic data generation technique for generating prompts that would elicit specific responses – or, more precisely, maximize the likelihood of generating a specific response. The proposed system would use generation loss as an optimization metric: lower loss on target tweets indicates more accurate prompting.

This process would likely use DSPy to enable the automation of optimizing these meta-prompts for back-translation across models. Beyond immediate applications, this infrastructure could enable sophisticated tasks like in-filling plausible substitutions for deleted tweets for context reconstruction. More importantly, it would create valuable training data for fine-tuning specialized back-translation models, potentially advancing the broader field of synthetic data augmentation.

### Empirical Filtering Framework
Future work might develop systematic approaches to evaluating and improving filtering strategies. The proposed framework could integrate with reward models and provide A/B testing infrastructure, enabling quantitative comparison of different filtering approaches. This could include experiments with online learning through RLHFlow and the OpenInstruct pipeline, potentially using PPO for online reinforcement learning from twitter feedback.

This framework would help answer crucial questions about dataset quality and model training efficiency. While beyond our current scope, such tools could significantly impact how the field approaches dataset creation and filtering.

### Twitter Deployment
Direct Twitter deployment represents perhaps the most ambitious extension. This would involve creating a Naptha "Environment" for X. The system would need sophisticated context management to maintain coherent conversations while respecting Twitter's API constraints.

While this capability extends well beyond our current project scope, successful Twitter deployment would demonstrate the practical value of our infrastructure and provide valuable insights for future development of social media agents.

### Implementation Considerations
These Phase 3 features fall outside the scope of the current proposal and grant work which has been funded.

By documenting these possibilities, we hope to inspire and guide future work in this space. Researchers building on our Phase 1 and 2 infrastructure can use these proposals as starting points for their own investigations, potentially realizing capabilities we can only envision at this stage.