SRS

See also: checklist, this document revision, and problem statement

The following Software Requirements Specification for `tinymorph` *is using Volere Requirements Specification template*

1. Purpose of the Project

> `tinymorph` aims to explore new interfaces and workflow for glossary > auto-regressive model to help writers craft better writing artifacts.

1.1 User Business

UB-1

Offer service for creative writing.

Rationale: `tinymorph` is a web-based text editor, thus, its main goal is to provide interfaces for users to write.

UB-2

Help stakeholders to create writing artifacts

Rationale: `tinymorph` will provide environment for stakeholders to do creative writing. That means writing artifacts are self-contained, owned by users, forever.

1.2 Goals of the Project

comprehensive lists

G-1

A *file-over-app* what-you-see-is-what-you-get (WYSIWYG) editor.

Rationale:

> *The files you create are more important than the tools you use to create them. Apps are ephemeral, but your files have a chance to last.* -- Steph Ango

`tinymorph` aims to amplify users' ability to write through planning and suggestion interfaces, done over files that owned by users.

G-2

Planning interfaces for ideas formulation

Rationale: `tinymorph` will propose certain directions using settings configured by users for idea generations as well as suggestions to help users *get over writers block* or *expand on current ideas*.

G-3

Efficient attention caching for feature steering.

Rationale: Generating similarly composed features vectors every forward pass might not be the best use of available resources. Thus, designing a caching mechanism for glossary > KV block generated as a result for feature steering via SAEs would help overall throughput and reduce latency.

G-4

SAEs suitable for planning and suggestion for creative writing.

Rationale: Recent works show glossary > sparse autoencoders can be trained to minimise certain bias scope [@rajamanoharan2024improvingdictionarylearninggated]. This is prevalent in `tinymorph` use-cases as it needs to ablate harmful features and amplify features related to creative writing to improve steering direction. OI-2 will also investigate current open-issues for activation steering.

G-5

OpenAI-compatible API for LLM server.

Rationale: OpenAI's API definition has been widely adopted in the industry for deploying LLMs. OpenAI-compatible endpoint is required for future integrations with upstream tools.

G-6

> Amplify users creativity

external motivation

Rationale: There are currently two incentives emerging within the industry as we approach the intelligence age: entities that seek to build systems capable of outperforming humans with tasks at a greater degree of efficiency and accuracy (OpenAI, Anthropic, SSI, AI21, etc.), and companies aims to utilise these systems to amplify our cognitive self (Runway, Cohere, etc.).

Our interest lies firmly with the second group, as we believe that tools should fundamentally improve our lifes, not replacing it. `tinymorph` is the manifestation of this belief system.

2. Stakeholders

2.1 Client

Description: The client base would include design studios focusing on producing novel writing artifacts and independent creative labs exploring malleable software.

Role: They help defining project's scope and deliverables, ensuring certain qualities and requirements to be met while enabling `tinymorph`'s innovations on top of existing interfaces paradigm.

2.2 Customer

Description: Primary customers for `tinymorph` include writers and engineers who seek tool for thoughts, where they can formulate ideas and amplifies their agencies. Writers would value features that guide them through a writers block, while engineers seek functionalities that help articulate complex ideas clearly.

Role: They help with users' adoption and influence project's roadmap. Their feedbacks are proven to be vital for continous development of `tinymorph`.

2.3 Other Stakeholders

Description: Open source developers who shares interests in building local-first software; venture capitalists and angel investors who are looking for investment opportunities in this area.

Role: Developers in the open-source community will help with maintaining `tinymorph` and its features development. This is vital for the project's sustainability and growth. Venture capitalists and angel investors might provide financial backing for the team to scale and spend our time on building out the product.

2.4 Hands-On Users of the Project

Description: comprises of individuals who will interact with `tinymorph` directly through its interface to use for their creative writing endavour.

Rationale: They are essential for the initial adoption, specifically benefited from planning and suggestion interfaces that `tinymorph` offers for their creative writing.

Role: Users will provide critical usability feedback and actively participate in iterative beta rollout phases. Their feedback will give the team a lense into different use-cases and requirements from writers' process and workflow.

Categorical breakdown:

Users	Description	Target Audience
Writers	groups that utilize `tinymorph`'s planning features for creating ideas and get over blocks	novelists, writers, bloggers
Engineers	groups that pursue creative writing as a hobby, but want to experiment with SOTA language models to aid with their writing process	software engineers, computers scientists

Experience:

Type	Description	Range	Reasons
Subject Matter	Refers to how sufficient a person is with their writing	engineers exploring creative writing to seasoned creative professionals	Utilize `tinymorph`'s interface to improve the quality of their work
Technological	Refers to how comfortable one is to use tools for text manupulation	designed with varying levels of technical proficiency	ensure both tech-savy individuals and the unversed to benefit with what `tinymorph` can offer

Notable mentions for other characteristics:

- Physical location: `tinymorph` should be reliable regardless of users' geological location, and should be accessible by users' worldwide with different backgrounds and ethnicity.

- Education levels: `tinymorph` must cater to a spectrum of different educational background
- Attitude Toward Technology: generally positive, with a preference for user-friendly technologies that require minimal learning curves. Tools for thoughts should be intuitive without imposing complex technical challenges.

2.5 Personas

Description: Representative user profiles based on extensive user research. These personas are constructed from typical characteristics, behaviors, and needs observed among potential users and serve to bring user stories and requirements to life during the development process.

Role: Serve as a focal point during design and development. It helps to tailor features, functionality, and user interfaces to meet the specific needs and behaviors of different user groups. `tinymorph` is designed to assist in planning writing projects and overcoming writer's block, rather than simply rewriting or analyzing sentiment.

The following personas are constructed based on `tinymorph`'s stakeholders profile:

Persona 1

- Name: Emily Brown
- Age: 28
- Occupation: Freelance Fiction Writer
- Interests: Novel writing, short stories, fantasy and science fiction
- Goals: To find new ways to build complex characters and intricate plots that captivate her readers.
- Technology Proficiency: Intermediate, utilizes various digital tools for research and manuscript editing.

Narrative: Emily looks for a tool to help her plan and overcome writer's block and formulate ideas to finish her novel. She uses `tinymorph` to generate suggestions and outline her story structure, helping her develop her characters and plot more effectively. The planning features of the platform guide her through the creative process, enhancing her writing workflow and productivity.

Persona 2

- Name: Michael Liu
- Age: 35
- Occupation: Software Engineer

- Interests: Technical documentation, project schematics, professional development articles
- Goals: To integrate creativity into his writing to make complex concepts accessible and engaging.
- Technology Proficiency: Advanced, highly skilled in technical design software and documentation tools.

Narrative: Michael utilizes `tinymorph` to finish his current writing piece on social anxiety. He utilizes suggestion feature to curate and search potential arugments points that have similar tone to Dr. Ana Lembke. He then uses this to improve his current writing, in which it helps him to become a better writer.

Persona 3

- Name: Sarah Johnson
- Age: 40
- Occupation: Professor and Personal Blogger
- Interests: Academic publishing, lifestyle blogging, engaging a broad audience online
- Goals: To distill complex academic concepts into engaging blog posts that appeal to a general audience.
- Technology Proficiency: Intermediate, adept with blogging platforms and social media engagement tools.

Narrative: Sarah utilizes `tinymorph` to infuse her academic works into technical blog. By using certain tonality offered by `tinymorph` that is tuned to her academic writing style and her favorite author Raymond Carver, she is able to produce engaging blog posts while maintaining a certain degree of technical depth without losing in translation.

2.6 Priorities Assigned to Users

Description: Involves categorizing users based on their usage patterns and the criticality of their needs. For example, professional writers might need advanced editing tools more than casual users.

Role: This categorization helps in prioritizing development tasks such as feature enhancements, bug fixes, and custom integrations to align product capabilities with the most valuable user demands. By focusing resources on the most impactful areas, the development team can more effectively address the core needs of primary users while still be virtuosic and open-minded on feature development.

Key Users:

- Creative Writers and Engineers: They are vital to the product's success as `tinymorph` is designed to help with this group's issues in mind. Their requirements are given the highest priority, and their feedback directly influences major product decisions.

Secondary Users:

- Academic Researchers and Educators: While important, these user's needs are secondary to those of key users. Their feedback influences product enhancements but is prioritized after the essential needs of the primary user group.

Unimportant Users:

- Casual Content Creators: This group includes users who occasionally use the platform for non-professional writing. Their needs are considered, but they have the lowest priority and minimal impact on the core functionality and strategic direction of the product.

2.7 User Participation

Description: Active involvement of professional writers and engineers in the development process through mechanisms such as such as targeted workshops, specialized feedback forms, and direct interviews

Role: Crucial for gathering qualitative and quantitative data on user satisfaction, system performance, and potential improvements. It will guide the agile development process and feature prioritization.

2.8 Maintenance Users and Service Technicians

Description: Includes the technical team responsible for the deployment, maintenance, and troubleshooting of tinymorph. It ensures that the application remains operational and secure.

Role: Handle regular updates, patch deployments, system monitoring, and troubleshooting. Their work is critical to maintaining the high availability and reliability of the service and responding to emerging security threats and technical issues.

3. Mandated Constraints

3.1 Solution Constraints

MC-S1

Base language models used for generations must be open weights (Gemma 2, Llama 3.2, etc.)

Rationale: Having open-weight models allows for training custom SAEs (e.g., tonality, writing styles) and ability to build experimental features such as, entropy sampler, features composition, gated steering, etc.

MC-S2

asteraceae inference server must implement OpenAI-compatible endpoints

Rationale: OpenAI-compatible endpoints are widely adopted in the industry. An inference server providing an OpenAI-compatible endpoint would make it easier for API integration with upstream tools.

MC-S3

The Minimum Viable Product must be web-based, and accessible from standard web browsers (Chrome, Firefox, Safari) without requiring installation or browser-specific extensions.

Rationale: Starting with a web-based solution allows faster prototyping and wide accessibility, ensuring that the application can reach a broad audience quickly while allowing room for future iteration.

MC-S4

`tinymorph` file system should adhere to "file-over-app" philosophy, avoiding server-side databases for minimal system dependencies.

Rationale: By following a file-over-app architecture, there is a reduction in the overhead associated with traditional database transaction, making the tool more lean and give back files' ownership to the users.

MC-S5

Personalization features, such as user preferences for tone and style, must remain within predefined limits to ensure compatibility with the model's underlying architecture.

Rationale: Limiting personalization features to predefined parameters ensures that the system's core architecture remains stable and compatible with the underlying language models. While users can still select tone and style preferences through features like dropdown menus, these constraints prevent excessive complexity in the model's behavior, maintaining a balance between user customization and technical feasibility.

MC-S6

There shall be no explicit storage of user-specific content on external servers

Rationale: Protecting user privacy and ensures that their data remains secure. By not storing user-specific content on external servers, the application reduces the risk of data breaches and aligns with privacy-conscious practices. This reassures users that their content is handled locally or securely on their own devices, maintaining trust in the platform.

3.2 Implementation Environment of the Current System

MC-I1

The implementation environment must follow modern web application best practices.

Rationale: Following modern web best practices ensures compatibility, and ease of maintenance across various browsers and devices.

MC-I2

Server-side components must support deployment on scalable infrastructure, including compatibility with GPU usage and Kubernetes clusters.

Rationale: This ensures that the server-side architecture can efficiently handle the computational demands of the models (e.g., GPU support) while remaining flexible for deployment in various cloud environments, promoting scalability and ease of management.

MC-I3

The system must support cloud-based inference via APIs access, while managing request queues and concurrency limits.

Rationale: This ensures the system can handle high-performance inference tasks both locally and in the cloud, while maintaining efficient resource usage and managing multiple concurrent requests to prevent bottlenecks.

MC-I4

The implementation must accommodate scalable infrastructure that manages increased load during high-traffic periods.

Rationale: This ensures the system can dynamically scale to meet user demand, maintaining performance and stability by efficiently managing multiple requests and preventing overload during periods of peak traffic.

3.3 Partner or Collaborative Applications

MC-P1

The system must support integration with tools like Notion, Obsidian, and text editors such as Neovim and VSCode.

Rationale: This allows users to work with their preferred tools, enhancing productivity and collaboration by enabling them to transfer and synchronize content seamlessly between the application and other commonly used applications in the same space.

MC-P2

The system must provide export options in multiple formats (e.g., Markdown, PDF).

Rationale: Providing multiple export formats allows users to share and collaborate across a variety of tools and systems, supporting flexible workflows and interoperability with a wide range of applications.

3.4 Off-the-Shelf Software

MC-O1

off-the-shelf writing assistance tools (e.g., Jasper, Copywrite) can be used for initial comparison and benchmarking purposes during POC development.

Rationale: Benchmarking against existing tools helps identify areas where the application can offer more personalization and control compared to standard solutions.

3.5 Anticipated Workplace Environment

MC-A1

The team must support remote collaboration, using GitHub for version control and Microsoft Teams for communication.

Rationale: This ensures that development cycle can proceed asynchronously, facilitating effective collaboration across distributed teams, with GitHub managing code changes and Teams handling communication.

MC-A2

All developers must set up their environments according to the contribution guidelines for `tinymorph`.

Rationale: Ensuring consistent environments across all developers minimizes integration issues, enhances collaboration, and maintains uniformity in development practices.

MC-A3

Developers are encouraged to add unit and integration testing to ensure fault tolerance and workflow stability, using tools like GitHub Actions for continuous integration.

Rationale: Adding testing early in the development cycle promotes code stability and helps identify issues across environments, supporting smooth development workflows and reliable user experiences.

3.6 Schedule Constraints

MC-S1

The team must meet the Revision 1 deadline of March 24, 2025 as per the capstone project timeline, allowing additional buffer time for debugging, user testing, and revisions before the final presentation.

Rationale: Meeting the Revision 1 deadline ensures that there is sufficient time for testing and improvements, which are crucial for addressing issues and ensuring a polished final product by the end of the capstone project.

MC-S2

Preliminary research, including design thinking and proof of concept development, must be stabilized within the first two to three months of the project.

Rationale: Completing early-stage work promptly allows for more time to focus on complex engineering tasks like model integration, ensuring that key functionalities are implemented effectively within the project's timeline.

MC-S3

The time required for training and validating SAEs must not exceed 5 days of GPU time, including any hyperparameter tuning, to keep the project on schedule.

Rationale: Limiting the training time for SAEs ensures that the system stays within development timelines, preventing bottlenecks and allowing time for other critical tasks.

3.7 Budget Constraints

MC-B1

The project has a maximum budget of 200 dollars in credits on for online inference, and access to available services
such as GitHub.

Rationale: This constraint ensures that the system operates within the available budget, focusing on efficient resource use and cost-effective solutions for cloud-based serving.

3.8 Enterprise Constraints

MC-E1

All software dependencies must follow the Apache 2.0 license or a compatible subset.

Rationale: Ensuring that all dependencies align with the project's open-source licensing reduces legal risks and maintains consistency with the project's license requirements. If any dependencies use a more strict license, the team must address license and references accordingly.

MC-E2

Open-weight models used for inference must adhere to their respective community licenses and be used only for research purposes.

Rationale: This ensures that any models integrated into tinymorph comply with their community usage terms, preventing misuse and maintaining alignment with ethical research standards.

4. Naming Conventions and Terminology

glossary

agency

The ability and freedom for an individual to *act* based on their immediate context and interests.

Ivan Illich [@illich1973tools] claimed that through proper use of technology, one can reclaim agency and practical knowledge for your everyday Joe.

> Tools for conviviality (conviviality means 'alive with') suppress other ideas and systems of knowledge and concentrate control of knowledge and power in the few and the elite [...] - Tools for Conviviality

The idea of agency for machine learning is that models have the ability to enact on their own without human intervention. Given the emergent properties of "intelligence" in these systems, it is crucial for us to understand their world view such that we can make informed decisions for building interfaces that will amplify our own cognitive abilities.

See also Alex Obenauer's work on personal computing, Self-Determination Theory

inlay hints

Special markers that appear in your editor to provide additional context about context of the code[1]

In a context of a text editor, inlay hints can work as a suggestion from a providers based on current context.

auto-regressive model

A statistical model is autoregressive if it predicts future values based on past values. For example, an autoregressive model might seek to predict a stock's future prices based on its past performance.

In context of LLMs, generative pre-trained transformers (GPTs) are derivations of auto-regressive models where it takes an input sequence of tokens length n and predicting the next token at index $n + 1$.

Auto-regressive models are often considered a more correct terminology when describing text-generation models.

transformers

A multi-layer perception (MLP) archiecture built on top of a multi-head attention mechanism [@vaswani2023attentionneed] to signal high entropy tokens to be amplified and less important tokens to be diminished.

low-rank adapters

Paper: "LoRA: Low-Rank Adaptation of Large Language Models" [@hu2021loralowrankadaptationlarge], GitHub

> ELI5: Imagine you have a big complex toy robot. Now you want to teach this robot some new tricks. With LoRA, you are giving this robot a small backpack. This backpack won't change how the robot function, but will give it some new cool tricks. Now with SAEs, you are adding enhancement directly into the robot, which makes it a lot better at some certain tricks.

The idea is to freeze a majority of the network weights, and inject trainable rank decomposition matrices to influence the models' outputs.

each LoRA layer can then be merged with the main models, in which create specialised models on given tasks. The main benefit of LoRA is to reduce costs for fine-tuning tasks.

In a sense, LoRA is a different comparing sparse autoencoders.

- For LoRA, we are controlling the outputs of a models by training additional "parameters" to add into the models

- With SAEs, we are directly editing features activations within the neural net, which means we don't have to worry about fine-tuning the model. We observe this through Claude's Golden Gate Bridge.

mechanistic interpretability

alias: mech interp

The subfield of alignment that delves into reverse engineering of a neural network.

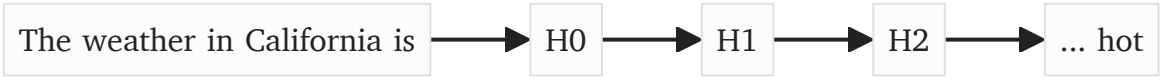
To attack the *curse of dimensionality*, the question remains: How do we hope to understand a function over such a large space, without an exponential amount of time?

manual steering

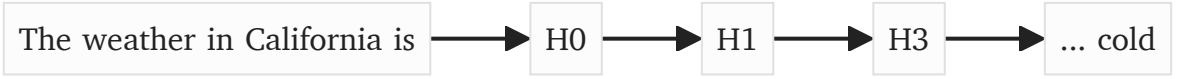
also known as glossary > features steering

refers to the process of manually modifying certain activations and hidden state of the neural net to influence its outputs

For example, the following is a toy example of how GPT2 generate text given the prompt "The weather in California is"



To steer to model, we modify H_2 layers with certain features amplifier with scale 20 (called it H_3)[2]



One usually use techniques such as glossary > sparse autoencoders to decompose model activations into a set of interpretable features.

For feature glossary > ablation, we observe that manipulation of features activation can be strengthened or weakened to directly influence the model's outputs

superposition hypothesis

> Linear representation of neurons can represent more features than dimensions. As sparsity increases, model use superposition to represent more glossary > features than dimensions.

When features are sparsed, superposition allows compression beyond what linear model can do, at a cost of interference that requires non-linear filtering.

features

When we talk about features [@elhage2022superposition{see "Empirical Phenomena"}], the theory building around several observed empirical phenomena:

1. Word Embeddings: have direction which coresponding to semantic properties [@mikolov-etal-2013-linguistic]. For example:

   ```
   V(king) - V(man) = V(monarch)
   ```

2. Latent space: similar vector arithmetics and interpretable directions have also been found in generative adversarial network.

We can define features as properties of inputs which a sufficiently large neural network will reliably dedicate a neuron to represent [@elhage2022superposition{see "Features as Direction"}]

hyperparameter tuning

Refers to the process of optimizing the hyperparameters of a model to improve its performance on a given task.

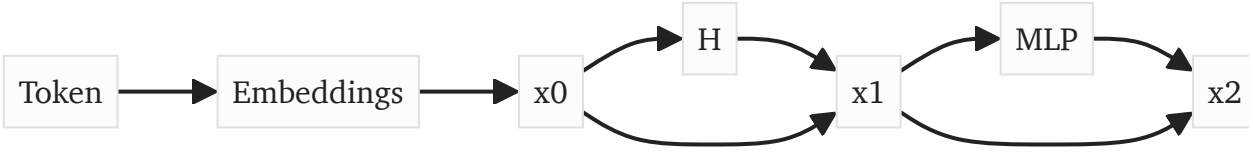
In the context of mech interp, we refer to adjusting given scale and Scratch > entropy of given feature vectors.

ablation

In machine learning, ablation refers to the process of removing a subset of a model's parameters to evaluate its predictions outcome.

Often also referred as feature pruning, but they have some slightly different meaning.

residual stream



residual stream x_0 has dimension (C, E) where

- C: the number of tokens in context windows and
- E: embedding dimension.

attention mechanism H process given residual stream x_0 as the result is added back to x_1:

$$x_1 = H(x_0) + x_0$$

logits

the logit function is the quantile function associated with the standard logistic distribution

inference

Refers to the process of running the model based on real world inputs to generate text completions.

next-token prediction is commonly used in the context of LLMs.

time-to-first-tokens

Denotes the latency between request arrivate and the first output token generated by system for the request.

minimise TTFT will help with UX for users.

sparse autoencoders

abbreviation: SAE

Often contains one layers of MLP with few linear ReLU that is trained on a subset of datasets the main LLMs is trained on.

For example, if we wish to interpret all features related to the author Camus, we might want to train an SAEs based on all given text of Camus that is also found in Llama-3.1

retrieval augmented generation

First introduced by [@lewis2021retrievalaugmentedgenerationknowledgeintensivenlp] which introduces a pipeline that includes a retriever models p_n queried from existing knowledge base to improve correctness and reduce hallucinations in LLM generations.

How would this work with SAEs specifically?

- Run an embedding models => SAEs to interpret features from relevant documents. => Search related features.
- Added said documents embedded in input tensors => better planning for contextual embeddings.

See also Contextual Document Embeddings

data

Representation of information in a formalised manner suitable for communication, interpretation, or processing by humans or by automatic means.

connectionism

Initially conceptualized to represent neural circuitry through mathematical approach. [@rosenblatt1958perceptron]

Second wave blossomed in late 1980s, followed by Parallel Distributed Processing group [@10.7551/mitpress/5236.001.0001], where it introduced intermediate processors within the network (often known as "hidden layers") alongside with inputs and outputs. Notable figures include John Hopfield, Terence Horgan.

Third waves (the current meta we are in) are marked by the rise in deep learning, notable contributions include the rise to fame of large language models. This era focuses artifical neural networks, focusing on designing efficient architecture to utilize available computes.

KV cache block

While generating tokens in auto-regressive models, previously generated tokens are fed into the network again while generating a new tokens. As input sequence becomes longer, inference glossary > FLOPs will grow exponentially.

KV cache solves this problem by storing hidden representations or previously computed key-value pairs while generating a new tokens.

the KV-cache will then be prefilled during forward propagation.

See also source

FLOPs

Also known as floating point operations. Used as a common metric to measure the the computer performance.

ref

---

1. Introduction from JetBrains, but this is implemented widely in other IDEs as well (VSCode, Neovim, Emacs, etc.) ↩
2. This is a toy representation of hidden layers in MLP, in practice, these models contain ~ 96 layers of MLP or more. ↩

5. Relevant Facts And Assumptions

5.1 Relevant Facts

RFA-RF1

Using open models such as Gemma or Llama as base language model

Rationale: Google's Gemma is a language model family supporting long context generation, with GemmaScope as pre-trained SAEs for feature interpolation. `tinymorph` will utilize Gemma for planning and guiding users' writing.

RFA-RF2

offers web-based interface

Rationale: provide an universal access for all platforms and operating systems.

RFA-RF3

`tinymorph` will utilise online inference for planning suggestion.

Rationale: Running model locally poses challenges for users to setup both base model with specific SAEs for different tasks. While `tinymorph` roadmap is to release a packaged binary that can be run everywhere, a online inference server will be used for web-based interface to ensure the best user experience.

5.2 Business Rules

RFA-BR1

Data locality

Rationale: Users' configuration will be stored within a vault-like directory, locally on users' machines. No data will be stored on the cloud.

RFA-BR2

Suggestions and planning steps must adhere to safety guidelines

Rationale: usage of SAEs to reduce hallucinations, as well as improve general safety of text quality.

5.3 Assumptions

> **RFA-A1**
>
> User knows how to use the browser

Rationale: `tinymorph` will offer a web-based interface, thus users must know the basic navigation of the existing environment (in this case, the browser of choice).

> **RFA-A2**
>
> Network connection

Rationale: `tinymorph` will require network connections to run inference for suggestion and planning UI.

> **RFA-A3**
>
> Interests in writing

Rationale: The assumption for `tinymorph` relies on users who have interests in writing.

6. The Scope of the Work

6.1 The Current Situation

The problem with writers block is that it is cyclical by nature. Julia Cameron's The Artist Way suggests people to commit writing a certain numbers of words everyday, regardless of productivity and word counts. Yet, it is not as simple as "just put words down on paper". For some, inspirations are stream of consciousness that flows in and outs, yet for others there would be times that we would just sit, and stare at a blank Google docs. Frustration starts to creep in: "*come on, just write something, damn it!*"

We then proceed to overload our prefrontal cortex with surges of dopamine in a quest to find inspiration: one might spend a 200 dollars on that Club Monaco tees, others might turn into 400 pages of Nietzsche's The Gay Science to look for lines that resonate with their state of mind. In a sense, the hardest problem about writers block is so detached from the physical act of putting down words on papers, rather a layer beyond this interface.

Engineers who are looking to expand their writing toolset might turn to conversational AI tools such as ChatGPT, Claude, etc. to synthesize their ideas and expand on their writing. Yet, collectively we have phased out a sort of "robotic" and "gpt-esque" tonality from these synthesized essays. Additionally, conversational tools often take a linear approach that doesn't cater well to non-linear nature of creative writing process, where retrieval of information are often non-deterministic. And users end up "optimizing prompt" to guide the model to generate in a certain direction.

In the physical world, tools such as blackboards or planning books provide a canvas for users to physically interact with such artifacts. A certain planner workflow would help writers to get over certain mental block. With software, most tools aiming to help improve writing are often lack this planning workflow, while emphasizing on more cookie-cutter generations templates through prompt engineering.

`tinymorph` aims to address these shortcomings. It intends to provide a more dynamic, interactive writing environment that not only offers suggestions but also provide spatial interfaces for non-linear idea exploration. This "How Now" view aims to replace the rigidity of current LLM-powered text editors by providing certain planning workflow, allowing users to manually steer and direct generations based on their tonality that is closed to their true writing style, creating a more enjoyable and cohesive writing experience.

6.2 The Context of the Work

`tinymorph`'s work context identifies the environment, systems, and users that the tool interacts with, defining the boundaries of its operation. Understanding the following milestones would help `tinymorph` better help writers with their creative writing experience.

Adjacent Systems

encapsulate related environment that provides as a base for tinymorph to build upon.

`tinymorph` will be built on top of:

- **Local file storage system**: a vault-like directory will be implemented and used for `tinymorph` to provide a workspace for users to start writing. This vault will act as ground truth for storing both users files as well as their configuration for `tinymorph`. By doing so, `tinymorph` stays true with its *file-over-app* philosophy.
- ***asteraceae* inference server**: `tinymorph`' inference server will be responsible for running LLM for generating suggestion and planning step. Note that no users data are stored on this given server, preserving true users' data privacy

  ("asteraceae" is the scientific name for "daisy")

- **user devices**: `tinymorph` will offer a web-based interface, ensuring it to work on multiple platforms, operating systems that can sufficient run a modern browser.

Information Flow

Below is a table representing the key interactions in the context of the project with adjacent systems:

Adjacent System	Interaction Type	Input/Output	Purpose
Local file storage system	Configuration Management	Input/Output	Stores user preferences locally, ensuring settings are retained without the need for cloud storage.
Inference Server	Planning	Input	Processes user inputs to generate planning steps, without storing any data to maintain privacy.
Inference Server	Suggestions Generation	Output	Generates and returns suggestions to users based on configuration settings, based on provided inputs.
User Devices	Planning Interface	Output	Displays writing suggestions, planning workflows, based on user input

6.3 Work Partitioning

Key business events represent the actions and scenarios that tinymorph responds to during typical usage. These events encompass user interactions, system activities, and the flow of information across `tinymorph`'s components. It offers a detailed view of the operational flow.

Understanding the key business events for `tinymorph` is essential for partitioning the work into manageable sections, ensuring each business use case (BUC) is clearly defined and independently understood. By breaking the work into logical segments it enables the process to help support better design decisions, validate workflows, and manage requirements effectively to ultimately maintain a user-centric focus throughout development.

The business event list is presented in a tabular format. Each event includes:

- **Event Number**: Identifies the specific business event.
- **Event Name**: Describes the nature of the action or scenario.
- **Input and Output**: Specifies whether the interaction is an input or an output.

- **Summary of the Business Event (BUC)**: Provides a description of the expected result of the business event.

Event Number	Event Name	Input/Output	Summary of BUC
1	User Uploads Document	Input	User uploads or begins editing a document. The file is processed locally and formatted into a request to the inference server for suggestions, but no data is stored on the server.
2	Generate Writing Suggestions	Input	Generates writing suggestions based on the user's text, considering style preferences.
3	Planning Interfaces	Output	Generates planning steps based on users inputs and configuration settings.
4	Save Configuration Settings	Input/Output	Saves user’s preferences, including tone, writing style, and personal configurations locally to a vault directory.
5	Inference Request Sent	Input	User's text content is formatted and sent to the inference server for text generation or suggestions. This is a stateless transaction and no user data is stored.
6	Display Suggestions	Output	Displays generated suggestions inline within the user’s document for easier adoption or rejection.
7	Manual Edits to Document	Input/Output	User manually edits the document, accepting or rejecting suggestions made by tinymorph.
8	Save Document Locally	Input	User chooses to save their document locally, and the content is stored on the user's device.
9	View Writing Analytics	Output	Tinymorph provides analytical insights to the user such as structure, readability, and suggested improvements.
10	User Changes Theme	Input/Output	User changes between light or dark mode for enhanced visual comfort. Configuration is saved locally.

6.4 Specifying a Business Use Case (BUC)

The Business Use Cases (BUC) detail how `tinymorph` responds to specific business events by providing a comprehensive description of each interaction. These descriptions ensure

that the requirements for system actions are fully understood and documented, enhancing clarity during the implementation phase.

> Each BUC is carefully articulated to capture how `tinymorph` behaves in response to user actions and how each event impacts the system's workflow.
>
> The purpose of defining detailed Business Use Cases is to understand how tinymorph responds during different user scenarios. This understanding helps identify the necessary requirements and ensures the solution meets the expected functionalities without ambiguity. By examining each BUC, we can ensure that all events are accounted for, creating a robust system that addresses user needs comprehensively. These scenarios build on the events specified in section 6.3, providing a full account of system behavior.

Below are detailed BUC scenarios, specifying how `tinymorph` handles each event:

1. User Uploads Document

When the user uploads or begins editing a document, `tinymorph` processes the file locally. The document is formatted into a string, which is sent to the inference server for suggestions. Importantly, no user data is stored on the server to enusre in preserving user privacy. The interaction is designed to maintain a stateless transaction while providing suggestions based on user inputs.

2. Generate Writing Suggestions

When a user inputs their text, tinymorph generates writing suggestions that align with the user's style preferences such as tone and clarity. These suggestions are then provided in a way that facilitates easy integration into the user's writing process. This use case emphasizes `tinymorph`'s ability to assist users in enhancing their writing creatively while staying true to their personal style.

3. Planning Interfaces

When users receive suggestions, they are presented with a planning tab that acts as stages and potential idea flow one can use to incorporate into their writing. This use-case demonstrates `tinymorph`'s spatial interfaces to help curate new ideas and solve writers block

4. Save Configuration Settings

Users have the ability to save their preferences including tone, writing style, and custom configurations locally to a vault directory. This ensures that each time a user interacts with

tinymorph, the tool aligns with their personalized needs, without relying on cloud-based storage. The local storage approach gives users the control and flexibility they need to maintain their preferred settings.

5. Inference Request Sent

User text content is formatted and sent to the inference server for generating suggestions. The request is processed on the server without saving any data, ensuring a stateless transaction that respects user privacy. This use case ensures that while the model provides sophisticated writing assistance, it does so in a privacy conscious manner.

6. Display Suggestions

Once the inference server processes the request, tinymorph displays the generated writing suggestions directly within the user's document. This helps the user see the potential improvements in real-time and decide whether to accept or reject each suggestion. This approach is designed to seamlessly integrate AI assistance into the user's creative process.

7. Manual Edits to Document

Users are encouraged to make manual edits to their document, either accepting or rejecting the suggestions made by tinymorph. The flexibility provided by this ensures users have complete creative control over the text. The manual editing process is integral to enhancing the accuracy of the content and ensuring that the suggestions align with the writer's intent.

8. Save Document Locally

After editing, the user may choose to save their document locally. Tinymorph ensures that the content is securely stored on the user's device. The emphasis on local storage enhances user control over their documents, fostering a sense of security and convenience.

9. View Writing Analytics

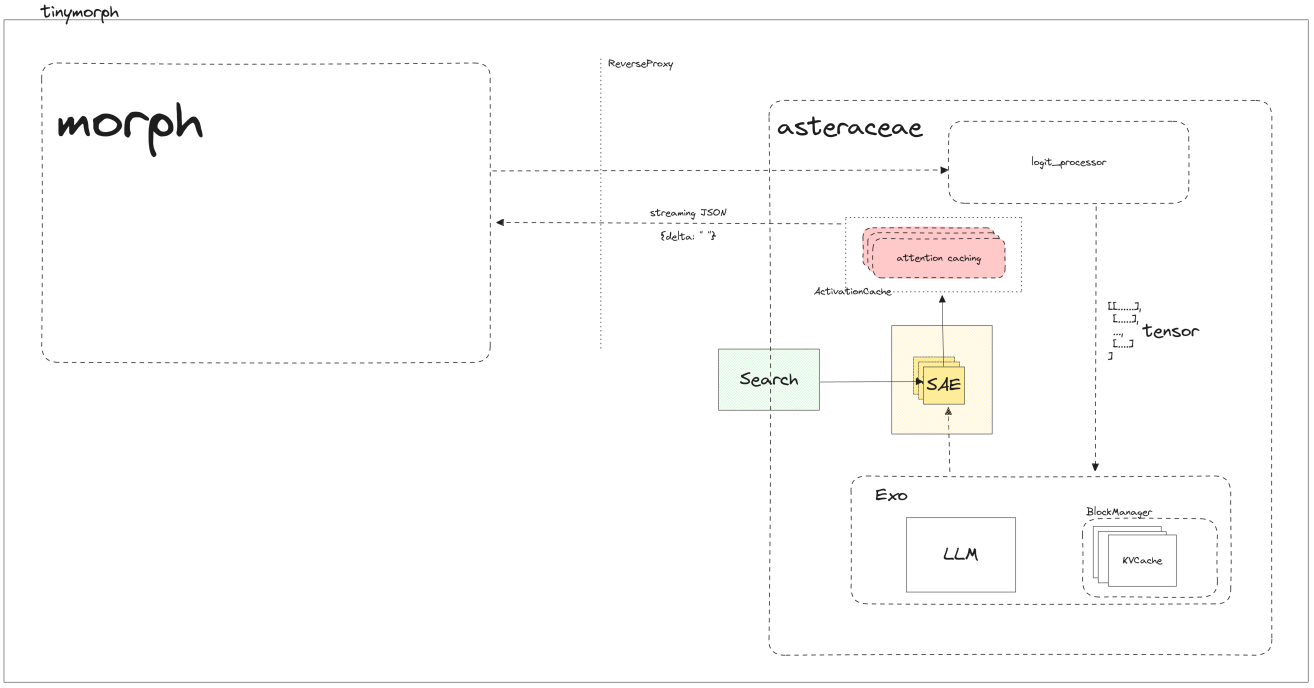
`tinymorph` provides the user with analytical insights into their writing. These analytics might include metrics such as structure, readability, and suggested improvements. The analysis helps writers to better understand their strengths and areas for improvement, ultimately enhancing the quality of their content. The insights also encourage users to make thoughtful changes to their work.

10. User Changes Theme

tinymorph offers users the ability to switch between light and dark modes to enhance visual comfort. The change is made on the user's device and stored locally, ensuring that the theme aligns with user preferences each time they use the application. This feature is meant to make the writing experience visually comfortable, catering to different working environments and times of day.

7. Business Data Model and Data Dictionary

7.1 Business Data Model



7.2 Data Dictionary

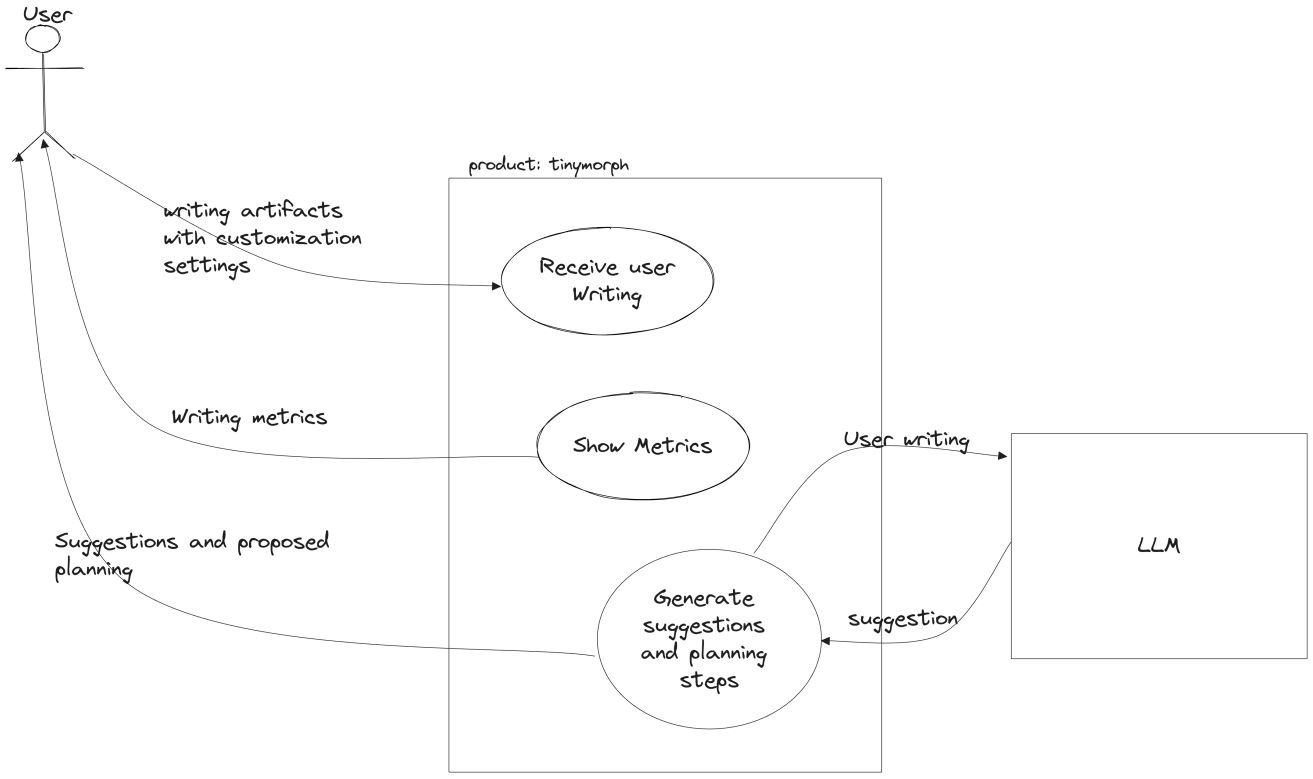
Name	Content	Type
`tinymorph`	`morph` + `ReverseProxy` + `asteraceae`	package
`morph`	Web interface for `tinymorph`	package
`ReverseProxy`	A middleware for reverse proxy with load balancer	module
`asteraceae`	`Search` + `ActivationCache` + `SAEs` + `logit_processor` + `Exo`	package
`logit_processor`	A function to convert incoming requests to logits	function
`exo`	Inference engine to run given LLM	package
LLM	open-weights models to be run for inference	package
`BlockManager`	Help manage KV-cache during inference	class
SAEs	a set of sparse autoencoders trained against given LLM to steer activations generation	package

Name	Content	Type
Search	Additional search tool to improve correctness	module
`ActivationCache`	Store said generations activations for performance purposes	class
streaming JSON	Outputs from `asteraceae` will be streamed back to `morph`	text
tensor	$n \times m$ matrix represented inputs processed by `logit_processor`	text

see also glossary > logits and glossary > sparse autoencoders

8. The Scope of the Product

8.1 Product Boundary



8.2 Product Use Case Table

PUC #	Name	Actor(s)	Inputs & Outputs
1	Receive user writings	User	Original writing artefacts (in)
2	Generate suggestions and planning steps	LLM	Original writing artefacts (in), suggestions (out)
3	Show metrics	User	Writing metrics (out)

8.3 Individual Product Use Cases (PUC's)

PUC-1: Receive user writings

Trigger: User press "Plan" button on the interface

Precondition:

- User has a document already loaded into `morph`
- The inference server is ready from cold-start

Actor: User

Outcome: Original writing artefacts are received by `tinymorph`

Input: Original writing artefacts

Output: Internal representation of the document in tensor format (internal)

PUC-2: Generate suggestions and planning steps

Trigger: Inference server received an incoming inflight request

Precondition:

- KV-cache is already warmed up and ready for generations

Actor: LLM

Outcome: Generate suggestions and planning steps from given suggestion

Input: User writing artefacts

Output: Suggestion that will be streamed back to `morph`

PUC-3: Show metrics

Trigger: After planning and suggestions has been finished.

Precondition:

- Inflight requests for suggestions has been completed

Outcome: Show writing metrics to users, including tonality, score, similarity

Output: Writing metrics

9. Functional Requirements

9.1 Functional Requirements

FR-1

`tinymorph` shall provide suggestions during the planning phase of creative writing, offering assistance in structuring ideas, topics, or themes based on user input, with a focus on brainstorming and outlining rather than real-time text generation.
The system should generate relevant planning suggestions within 10 seconds of receiving a prompt, producing outlines or idea guidance based on user input.

Rationale: Traditional text suggestions may not suit creative writing, which is highly personal and abstract. Instead, guiding users through the planning phase helps them build a solid foundation for their content.

FR-2

`tinymorph` shall provide users with manual control over text generation, enabling them to select advanced steering options such as tone, style, or creativity level. For beginners, a simplified option shall allow tone suggestions based on well-known authors. Additionally, users shall have the option to input their previous writing samples, allowing the model to adapt to their tone and style within a maximum of 30 seconds.

Rationale: Providing manual control over the model's output allows both beginners and advanced users to customize the text generation to their preferences. Offering tone suggestions and the option to input past writing samples enhances personalization, making `tinymorph` more adaptable to individual writing styles.

FR-3

Users shall be able to interact with generated text through a left-to-right (LTR) feedback panel, allowing them to provide real-time feedback on model-generated suggestions. The panel should present previews that are easy to modify without causing cognitive overload, particularly for planning-related tasks.

Rationale: The LTR panel offers an intuitive way to refine model behavior, but it must be designed to reduce cognitive overload, enabling users to focus on effective interaction without being overwhelmed.

FR-4

Users shall be able to set preferences for tone, style, voice, and formality, which `tinymorph` will apply to all future generations of text.

Rationale: Allowing users to customize the tone, style, voice, and formality ensures `tinymorph` adapts to individual preferences, making the generated text more personalized and relevant to the user's needs.

FR-5

Users shall be able to save their preferred configurations as profiles, enabling them to switch between different writing styles or goals. For the initial version, the focus shall be on creative writing, with the potential to expand to other writing types (e.g., academic, casual blog posts) as additional datasets are integrated.

Rationale: Allowing users to save profiles for different writing styles supports personalization, and initially focusing on creative writing ensures that `tinymorph` can refine its functionality before expanding to other types of content that may require specialized datasets.

FR-6

`tinymorph` shall allow users to navigate through their text non-linearly by providing a visual map or tree view that displays key points, topics, sections, and revision history. This tree-based exploration shall support both text structure and revision tracking, allowing users to explore and edit different versions or sections easily.

Rationale: Providing a tree-based view enhances the user's ability to navigate their content efficiently, particularly during revision or structural changes. This visual aid allows for more intuitive exploration of key points, sections, and revisions, offering greater control over the writing process.

FR-7

`tinymorph` shall present possible planning steps that users can choose, modify, or combine to guide the structure and flow of their writing.

Rationale: Focusing on planning steps rather than text alternatives aligns with the creative writing process, helping users organize and structure their ideas more effectively. This

approach avoids the limitations of traditional generation methods like beam search, which may not perform well in this context.

FR-8

`tinymorph` shall offer version control features that allow users to navigate through previous edits, revert to earlier document versions, and compare different drafts visually.

Rationale: Providing a clear representation of revision history helps users track changes, make informed decisions about reverting to previous states, and compare different versions of their work with ease.

FR-9

`tinymorph` shall support an offline mode where users can continue to write and interact with the editor without internet access, using pre-downloaded resources.

Rationale: While offline mode may not be a priority in the current development phase, providing the ability to work offline in future versions ensures greater accessibility and flexibility for users in varying environments.

FR-10

`tinymorph` shall allow users to continue writing and saving files locally during offline sessions. However, certain features, such as planning-related suggestions, will only be available when the user is connected to the internet.

Rationale: By allowing users to write and save locally during offline sessions, `tinymorph` ensures uninterrupted productivity. However, advanced features requiring internet access, like planning assistance, will only function when a connection is restored.

FR-11

Users shall be able to set and track specific writing goals (e.g., word count, tone consistency, argument development) through customizable progress tracking features, such as word count meters, tone analysis, and style consistency checks. `tinymorph` will offer suggestions to help users meet these goals, ensuring alignment with their intended writing objectives.
Rationale: Combining goal-setting with progress tracking allows users to monitor their progress and receive actionable feedback, making the writing process more structured

and goal-oriented. This unified approach supports personalized goal management, helping users stay on track.

FR-12

`tinymorph` shall allow users to categorize and tag different sections of their text (e.g., introduction, argument, conclusion), and automatically generate an outline based on the document's structure.

Rationale: Combining tagging with automatic outline generation streamlines document organization and navigation. This will provide a high-level overview of the content and enable quick access to specific sections for better organization and navigation within large documents. This feature also helps users maintain clarity and easily navigate large documents, providing both a detailed view of the content.

FR-13

`tinymorph` shall allow users to export their documents in .pdf, .md (Markdown), and plain text formats, ensuring compatibility with external platforms. Support for .docx format will be considered for future development as better libraries become available.

Rationale: Exporting documents in widely compatible formats like Markdown, PDF, and plain text ensures flexibility for users without adding unnecessary complexity. Focusing on these formats simplifies implementation while covering most use cases.

FR-14

`tinymorph` shall allow users to customize the visual appearance of the editor by choosing from different themes, such as dark mode, light mode, and high-contrast options.
Rationale: Providing customization of the visual appearance enhances user experience and accessibility, allowing users to choose themes that suit their preferences or visual needs.

10. Look and Feel Requirements

10.1 Appearance Requirements

LF-A1

`tinymorph` shall adopt a unified, non-intrusive, and uncluttered visual design.

Rationale: A consistent and non-intrusive design ensures brand recognition and provides a visually pleasing experience for users, fostering greater engagement and trust in the platform. This minimizes distractions to allow users in focusing more effectively on their creative tasks.

LF-A2

`tinymorph` must implement a consistent design system across all user interfaces, involving standardized typography and color palette.

Rationale: A consistent design system enhances user experience by ensuring visual coherence in typography and colors. This uniformity aids readability, reduces user distraction, and contributes to a seamless intuitive interface.

10.2 Style Requirements

LF-S1

The design of the application will be minimalist, utilizing clean lines and a monotonic color palette.

Rationale: A minimalist design with a monotonic color palette emphasizes functionality and content, enhancing usability by directing the user's focus to essential elements without distraction. This approach ensures that the interface remains uncluttered and the features more accessible, supporting a focused and efficient user experience.

LF-S2

The application must be responsive, adapting seamlessly to various device screens and orientations.

Rationale: As users may access the application from different devices with varying screen sizes, responsiveness is essential to provide a consistent experience across all platforms.

LF-S3

Interactive elements such as buttons and links must contrast significantly with the background to ensure visibility and accessibility.

Rationale: High contrast between elements and backgrounds enhances the visibility of interactive features, making navigation intuitive and preventing user frustration.

LF-S4

The user interface should enable smooth transitions and intuitive animations across various sections and features.

Rationale: Smooth transitions and intuitive animations contribute significantly to a seamless user experience. By enhancing user engagement, these visual cues help guide users smoothly through the application's features. Implementing animations effectively can indicate activity or the completion of tasks, reducing user confusion and improving clarity during navigation.

LF-S5

The application should include visual cues and feedback for user interactions to reinforce usability.

Rationale: Providing immediate visual feedback for user actions confirms the system's responsiveness, which will help users understand the application's behavior and reduce errors.

11. Usability and Humanity Requirements

11.1 Ease of Use Requirements

UH-EOU1

`tinymorph` shall include a session history feature that records and displays the user's most recent editing activities such as document accesses and text modifications.

Rationale: This functionality streamlines user workflow by providing quick access to recent actions, which reduces the time needed for navigation and increases efficiency.

UH-EOU2

`tinymorph` must allow users to interactively review and manually accept or reject changes suggested by the system after their inputs are submitted.

Rationale: Providing users with the option to manually accept or reject suggested changes allows them with greater control over their content. This functionality would enhance user engagement by making the editing process more interactive and ensures that the final output aligns precisely with their preferences and intentions.

UH-EOU3

The application shall include a planning interface to assist users in organizing and debugging their creative writing steps.

Rationale: The interface will improve user efficiency by supporting the iterative refinement of writing tasks and planning steps. It enables users to easily adjust and debug their creative outlines, enhancing the overall usability and functionality of the application.

11.2 Personalization and Internationalization Requirements

UH-PI1

`tinymorph` interface must include multilingual support to cater to an international audience.

Rationale: Multilingual support enhances the application's global accessibility and user engagement, ensuring that users can interact with the platform in their preferred language.

UH-PI2

The application shall provide options for users to select between light or dark mode based on their system settings or preference.

Rationale: These theme customization improves visual comfort and personalization, enabling users to adapt the interface to their visual preferences and working environments.

11.3 Learning Requirements

UH-L1

New users should be able to understand basic functionalities and start creating or editing content within 10 minutes of initial use.

Rationale: A straightforward and intuitive onboarding process is critical to ensuring that users can quickly become proficient with the application, leading to higher satisfaction and continued use.

11.4 Understandability and Politeness Requirements

UH-UP1

The application should utilize clear and concise language for all instructions, feedback, and user interface elements.

Rationale: Simple and direct language helps to avoid misunderstandings and ensures that the platform is user-friendly, making it accessible to a wide audience regardless of their background.

11.5 Accessibility Requirements

UH-A1

`tinymorph` should support text resizing without loss of content or functionality.

Rationale: Allowing text resizing helps accommodate users with visual impairments who require larger text to read effectively. Ensuring that the application remains functional and has content accessible at different text sizes guarantees a more inclusive user experience.

UH-A2

`tinymorph` should ensure that all user interface components are accessible via keyboard navigation.

Rationale: Keyboard navigability is essential for users who cannot use a mouse, including those using screen readers or other assistive technologies. Providing comprehensive keyboard access enhances the functionality and inclusivity of the platform, ensuring all users can efficiently use all features.

UH-A3

Implement ARIA (Accessible Rich Internet Applications) attributes throughout the application.

Rationale: ARIA attributes help provide essential information about the element's role, state, and property, which is crucial for users who interact with the application via assistive technologies. This ensures that all functionalities are conveyed and usable through these technologies.

12. Performance Requirements

12.1 Speed and Latency Requirements

PR-SLR1

TTFT should be minimum, around 200-500ms

Rationale: Suggestion and planning should feel smooth, and fast. Therefore, time-to-first-token is important.

PR-SLR2

Throughput should be approximate 300 tokens/sec for a batch size of 4

Rationale: `tinymorph` inference server should be able to handle incoming requests in batches, ideally handling a decent amount of throughput. Note that we will have to sacrifice some throughput for higher TTFT.

12.2 Safety-Critical Requirements

PR-SCR1

Suggestions must not be **harmful**

Rationale: SAEs must ablate activations that represent offensive language or inappropriate text.

PR-SCR2

The interface must not contain harmful images or NSFW content.

Rationale: All contents and icons from web-based interfaces must be safe for work.

12.3 Precision or Accuracy Requirements

PR-PAR1

The generated text should match users' steering direction

Rationale: `tinymorph`'s SAEs should activate specific attentions based on users inputs. Additionally, it must take into account all users' feedback.

12.4 Robustness or Fault-Tolerance Requirements

PR-RFR1

A notification toast must be sent to users in case inflight requests fail to complete.

Rationale: If any current requests fail to finish, a toast must be surfaced to users. This helps notify users to either resubmit specific plans or revert to previous planning steps.

PR-RFR2

`tinymorph` must implement a recreate deployment strategy

Rationale: In case certain replica and nodes failed to start, given Kubernetes cluster that run said inference server should be able to recreate the deployment.

12.5 Capacity Requirements

PR-CR1

Suggestions would be run asynchronously on request.

Rationale: `tinymorph` will support multiple users running suggestions at once. Users will be able to submit given requests and said inference server should be able to handle

multiple requests at once.

PR-CR2

Input should not show any certain delay

Rationale: `tinymorph` must ensure text manipulation on users' content to be as smooth as possible.

12.6 Scalability or Extensibility Requirements

PR-SER1

`tinymorph` inference server must include scale-to-zero and concurrency-based autoscaling.

Rationale: During high traffic, the inference servers must be able to scale up based on incoming requests. Additionally, in lower traffic, the server should be able to scale to zero to save on costs and resources.

12.7 Longevity Requirements

PR-LR1

Future integration with other language model architecture

Rationale: `tinymorph` should be able to extend to different model architectures with variety of SAEs.

PR-LR2

Support different distribution platforms.

Rationale: `tinymorph` will first ship a web interface. It should then reserve the ability to be packaged into a standalone universal binary that can be run on different operating systems and architectures.

13. Operational and Environmental Requirements

13.1 Expected Physical Environment

OER-EPE1

`tinymorph` will be able to run on different hardware environment, given it can run modern browser.

Rationale: `tinymorph` will ship a web interface through browsers. Therefore, it should support any hardware environment that can run a browser.

OER-EPE2

`tinymorph` should have minimal increase in power consumption

Rationale: `tinymorph` should avoid a huge increase in RAM for a browser tab.

13.2 Wider Environment Requirements

13.3 Requirements for Interfacing with Adjacent Systems

OER-RIAS1

`tinymorph` inference server should provide an OpenAI-compatible endpoints.

Rationale: The inference server must offer an OpenAI-compatible endpoint to ensure a handshake with the web interface. This server can also be accessed with any other tools that accept OpenAI-compatible endpoints.

13.4 Productization Requirements

OER-PR1

Secrets must be configured with certain Role-based access control (RBAC) rules

Rationale: To ensure all production environment variables are safe from bad actors and adversarial parties.

OER-PR2

Relevant documentation should be accessible by users.

Rationale: Usage manuals and technical-related details should be easily accessible from `tinymorph`'s interface.

OER-PR3

Feedback should also be included within the interface

Rationale: Enable user-feedback to improve the product.

13.5 Release Requirements

OER-RR1

Release cycle must utilize current GitHub CD workflow.

Rationale: Version control and release cycle should follow semantic versioning and utilize GitHub's CI for automation.

OER-RR2

End-to-end tests should pass before deploying to production.

Rationale: end-to-end workflow must be the minimum for all feature development to ensure `tinymorph` is functional within a production environment.

14. Maintainability and Support Requirements

14.1 Maintenance Requirements

OER-MR1

Security updates must be done periodically

Rationale: Regular security updates to adjacent dependencies must be done quickly to avoid certain CVE exploits if they exist.

OER-MR2

Feature integrations must pass existing tests

Rationale: Given features works must not fail existing testing infrastructure.

14.2 Supportability Requirements

OER-SR1

User feedback loop must be present.

Rationale: For further development and UX improvement, a user-feedback loop is required.

14.3 Adaptability Requirements

OER-AR1

`tinymorph` must be able to run with existing users' environment

Rationale: For web interface, `tinymorph` should be able to run on all existing modern browser. For packaged binary, it must support major architectures and operating system.

15. Security Requirements

15.1 Access Requirements

Not applicable given the application is open source, and inference server are exposed over a HTTPS endpoints.

15.2 Integrity Requirements

SR-INT1

All communication between the client UI, backend services, and external APIs must be encrypted using HTTPS.

Rationale: HTTPS encryption secures data in transit, preventing interception or tampering. It also ensures the confidentiality and integrity of user data and commands.

SR-INT2

Implement DNS security measures to ensure that DNS queries and responses are protected against tampering and spoofing.

Rationale: Securing DNS interactions prevents attackers from manipulating or rerouting network traffic. This is critical for maintaining the integrity of application data.

SR-INT3

The application will use content security policies to mitigate the risk of XSS attacks.

Rationale: Content Security Policies (CSP) are an effective security control to prevent XSS attacks by restricting the sources from which scripts can be loaded and executed in the application. CSP will help in safeguarding against data theft and maintain the integrity of the content delivered to users.

SR-INT4

Implement JWT and short-lived tokens to secure session communications.

Rationale: Utilizing JWT and short-lived tokens for session management enhances security by ensuring that session data remains protected against unauthorized access. This approach helps prevent bad actors from intercepting or tampering with session data, ensuring that user content and session details remain confidential and intact.

15.3 Privacy Requirements

SR-P1

The application must ensure that it does not collect or store personal information, adhering strictly to privacy by design principles.

Rationale: By not collecting personal information, the application minimizes privacy risks and complies with privacy laws and regulations. Avoiding personal data storage also reduces the need for complex data security measures, allowing the project to focus more on enhancing user experience and functionality.

15.4 Audit Requirements

SR-AU1

Implement monitoring of interactions with external service providers to ensure their use complies with security policies and performance expectations.

Rationale: Monitoring interactions with external service providers is essential to ensure they are used within the bounds of security protocols and that their performance aligns with the application's requirements. This helps in detecting any deviations that might compromise security or functionality, allowing for quick mitigation actions to maintain the integrity and reliability of the application services.

15.5 Immunity Requirements

SR-IM1

Employ up to date security measures to protect against known threats and vulnerabilities, including regular updates and patches to the software components.

Rationale: Keeping software updated ensures that known vulnerabilities are addressed, which will protect the application and its data from emerging threats.

SR-IM2

Configure the application to minimize the surface area for attacks by disabling unused services and endpoints.

Rationale: Minimizing the attack surface reduces the number of potential entry points for attackers, enhancing the overall security of the application. This proactive measure significantly lowers the risk of exploitations and helps maintain system integrity.

16. Cultural Requirements

16.1 Cultural Requirements

CulR-CR1

English supports

Rationale: English will be supported for alpha release of `tinymorph`. This is due to the limited capabilities of models when dealing with multilingual inputs.

CulR-CR2

Cultural reference must be factual

Rationale: If given cultural references are generated, it must utilize tools to fact-check given suggestions (using web-search).

CulR-CR3

Support left-to-right (LTR) reading flow

Rationale: Panels will be presented in LTR manner. RTL will be supported once multilingual are added.

17. Compliance Requirements

17.1 Legal Requirements

CompR-LR1

Suggestion must follow strict US copyright law.

Rationale: Certain suggestions, if any, uses additional contents, references must be made to ensure copyright law compliance.

CompR-LR2

SOC2 compliance

Rationale: `tinymorph` should follow SOC-2 attestation for its inference server.

CompR-LR2

Users permission to run inference against their content

Rationale: `tinymorph` will require users' inputs to make corresponding suggestions. In other words, existing user content will be sent during inference requests.

17.2 Standards Compliance Requirements

CompR-SCR1

follows standard HTTP protocol for client-server communication

Rationale: `tinymorph` will adhere to Hypertext Transfer Protocol (HTTP/1.1) standards as defined by the Internet Engineering Task Force (IETF) in RFC 2616 (for HTTP/1.1).

18. Open Issues

OI-1

How should we compose correct features matrix to ensure correct steering?

Rationale: We can train intepreter networks to extract human-readable activations layers (referred as "features") [@cunningham2023interim] [@templeton2024scaling], but features alone won't offer too much value for end users (engineers, writers).

`tinymorph` then must be able to compose multiple activations that represents certain tonality, in which a auto-interp pipeline [@juang2024autointerp] should be implemented to guide base models to generate in certain direction.

OI-2

What datasets one should use to train SAEs?

Rationale: [@goodfire2024research] demonstrated LMSys-1M chat datasets are great fit to train SAE for chat application specifically. For the interface for planning ideas a more general datasets that contains more essays, paragraphs might be more beneficial.

OI-3

For a planning interface, what if we add tool use (such as web-search) to enhance correctness in generations?

Rationale: RAG-infused pipeline [@lewis2021retrievalaugmentedgenerationknowledgeintensivenlp] has been widely adopted in industry-related workflows to reduce LLM hallucination. For steering specifically, this

might be useful given the additional context for online blog posts to influence certain direction writers want to formulate their ideas.

OI-3

Effectiveness against fine-tuned models?

Rationale: Fine-tuned models are a distilled version of the base models that is trained to perform generations in a given format/text. Methods such as LoRA has been proven to be useful for steering generations purely through prompt. The questions remains whether intuitively having SAEs to steer generations at the activation level would prove to be more useful than specifically fine-tuned models.

OI-4

file-over-app philosophy for building a text editor?

Rationale: The end goal is to build a text editor, which means we are building on top of the notion of "files". We rarely have to think about files nowadays in our daily tasks, yet we are still operating with them on a daily basis: Photos stored on your iPhone., music catalog saved in Spotify, knowledge pages in Notion etc. The industry seemingly to replace this primitive with something stored "on the cloud". I do think there are arguments to be made to give back this heuristic back to the users, if we are thinking about building digital artifacts that will last long after we are gone. Additionally, it will greatly simplify any internal logics.

OI-5

Inference performance for server deployment versus on-device?

Rationale: For the past year, the need for efficient inference to run these language models has been top priorities for companies to deploy these models in production. Framework such as vLLM [@kwon2023efficient], lmdeploy [@2023lmdeploy] offers different trade-off for running efficient inference on server. Given `tinymorph` will offer a web interface, how should we evaluate given frameworks to use in conjunction with trained SAEs. Additionally, for on-device inference, we must also investigate how one can run the models locally.

OI-6

Multilingual support?

Rationale: LLMs we are considering for `tinymorph` doesn't have good multilingual support. This has to do with a lack of datasets during pre-training of these foundation models. Therefore, to fully support analysing essays with language other than English, we must use base models that support multilingual, with the likes of aya, followed with a set of SAEs trained against this model.

19. Off-the-Shelf Solutions

19.1 Ready-Made Products

The following encapsulate a few existing products that may fits for `tinymorph`:

Text editor framework

CodeMirror:

- an open-source, browser-based text editor that supports rich editing features.
- can be integrated as the core text-editing interface for `tinymorph`, enabling basic editing functionality while allowing custom enhancements to support user steering and model behavior.

Language Models

Llama 3:

- an open-source large language model optimized for maintaining coherence in long-form writing.
- Has support for long context windows, and variable model-size makes it possible for future iterations of `tinymorph` to run local inference.

Gemma 2:

- an open-weights language model family optimized for safety with long context windows. Its ability to filter out harmful features would make it ideal to use with `tinymorph`, as it fits `tinymorph`'s cultural and safety requirements.
- Google also ship pre-trained SAEs trained on Gemma features, make it easier for `tinymorph` to prototype with its MVP.

Writing Assistants

Quill:

- AI-driven writing assistant that offers real-time feedback and suggestions to improve writing quality.
- advanced grammar and style checks and collaborative features, Quill can help users refine their text.
- A benchmark for `tinymorph` to compare its suggestions and planning features.

Jasper:

- AI-powered writing assistant focused on generating content for diverse formats, including blogs, essays, and marketing materials.
- With personalization options for tone and style
- Jasper can be used as comparison against `tinymorph` as SaaS versus Open-source offering.

End-to-end platform

Goodfire:

- recently releases preview demonstrate the usage of SAEs to steer Llama in conversational settings
- can be use as reference for `tinymorph` UX design for feature steering

19.2 Reusable Components

> ### sparse autoencoders
>
> abbreviation: SAE
>
> Often contains one layers of MLP with few linear ReLU that is trained on a subset of datasets the main LLMs is trained on.
>
> For example, if we wish to interpret all features related to the author Camus, we might want to train an SAEs based on all given text of Camus that is also found in Llama-3.1

19.3 Products That Can Be Copied

GitHub Copilot (Interaction Paradigms):

- Copilot integrations with different IDEs for inlay hint suggestion can be adapted for tinymorph's text generation.

- `tinymorph` can borrow this interaction style by offering inlay suggestions for text, allowing users to manually steer and adjust the output based on their writing goals.

Google Docs' Suggestion Mode (Inlay Suggestions):

- "Suggestion Mode" allows users to propose edits without making permanent changes.
- Tinymorph could adopt a similar mechanism for showing planning steps.

Grammarly's Tone Detector (Tone Adjustment Feature):

- tone detection system analyzes writing to give feedback on the mood and tone of the content.
- `tinymorph` could replicate this feature via SAEs to train on the user's writing style and suggesting tone adjustments, allowing users to fine-tune the emotional or stylistic qualities of their text.

20. New Problems

20.1 Effects on the Current Environment

EoCE-1

Workflow updates for writers

Rationale: `tinymorph` will introduce an alternative way to plan and write essays.

EoCE-2

Real-time collaboration

Rationale: `tinymorph` can provide real-time feedback on certain planning steps, which could influence how users approach one's writing.

20.2 Effects on the Installed Systems

EoIS-1

Performance impacts

Rationale: `tinymorph` will introduce additional computation for steering generations, which could uses additional resources from users' local machine. This means it might

requires more modern computers to run the application efficiently.

EoIS-2

Storage considerations

Rationale: `tinymorph` follows "file-over-app" philosophy, meaning certain folders structures for users files must be adhere to in order for applications to function correctly.

20.3 Potential User Problems

PUP-1

Learning curve

Rationale: For text editors, users might have a higher learning curve to setup their vault and file structure accordingly.

PUP-2

Integration from existing tools

Rationale: Changing ones' behaviour is hard, which means users might find a hard time to integrate `tinymorph` into their existing writing workflow.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

LAIETMINP-1

Browser support and accessibility

Rationale: Given we will ship the web-based version of `tinymorph` first, all version of chromium might not have support for certain file API. Additionally, different browser engine have different accessibility support, which might interfere with usability.

LAIETMINP-2

On-device inference

Rationale: If users wish to run models on-device, they might not have the sufficient hardware to perform given tasks. Additionally, setting up local inference might be proven to be challenging for the unversed.

20.5 Follow-Up Problems

FUP-1

Over-reliance on suggestion

Rationale: Advanced agents workflow for planning might increase a risk of homogenization in writing styles. as Ted Chiang commented on ChatGPT:

> The more that text generated by large language models gets published on the Web, the more the Web becomes a blurrier version of itself [...] Repeatedly resaving a jpeg creates more compression artifacts, because more information is lost every time.

FUP-2

Disruption in flow

Rationale: constant suggestion prompt might prove to be annoying to writers' flow and concentration.

FUP-3

Feature overloading

Rationale: The setup/UX might be too complex, potentially too intimidating for users who might prefer the simplicity of CUIs.

21. Tasks

See also Development Plan for an up-to-date development cycle as well as project planning.

For a more unstructured brain dump for potential exploration avenue see ideas.

22. Migration to the New Product

22.1 Requirements for Migration to the New Product

MNP-RMNP1

Minimal downtime during migration process

Rationale: When the inference server is updating or maintaining, users should be aware of the downtime, given it shouldn't affect users' workflow.

22.2 Data That Has to be Modified or Translated for the New System

MNP-DMTNS1

Migration in future config format should ensure backward compatibility for one-time transition

Rationale: When configuration or certain features require breaking change, `tinymorph` must be able to migrate existing configuration to the new format without breaking change.

23. Costs

C1

Running inference server on cloud provider (BentoCloud)

Rationale: We will use BentoCloud as our cloud provider to run the inference server.

Assumption: Using Gemma 2 7B requires at least a L4 for efficient KV cache management and with acceptable throughput.

Result: This costs around 1.0143 dollar/h

Assumption: The service will receive around 8 hours of traffic everyday, 5 days a week. For all of the other time period, we assume there are no traffic, or the inference server scales to zero.

The costs for running the inference server for a month will be around

$$1.0143 \times 8 \times 5 \times 4 = 162.288 \ \$/\text{month}$$

note: we exclude the calculation for hosting the sites, given we will be using GitHub pages to serve the web interface

24. User Documentation and Training

24.1 User Documentation Requirements

UDT-D1

Create a user's guide that details operational procedures, user interfaces, and troubleshooting tips.

Rationale: This guide will act as the primary resource for users to fully understand and effectively utilize all features of the application, enabling them to solve common problems independently. Keeping the guide updated will reflect ongoing changes and user feedback, enhancing user satisfaction and self sufficiency.

UDT-D2

Develop an installation manual with detailed instructions for the software installation, configuration, and initial setup processes.

Rationale: The manual ensures that all users regardless of their technical expertise can easily set up the application. This minimizes setup errors and facilitates smoother adoption of the software.

UDT-D3

Compile a design document describing the software architecture, API specifications, data flow diagrams, and code structure.

Rationale: This document is crucial for developers and technical stakeholders to understand the inner workings of the application. It wil support maintenance tasks, future development efforts, and integration with other systems.

UDT-D4

Create a Verification and Validation (V&V) plan and report that documents testing strategies, methodologies, results, and compliance checks.

Rationale: This documentation confirms that the application adheres to all technical specifications and user requirements, enhancing transparency and providing a basis for regulatory compliance and quality control.

UDT-D5

Develop a comprehensive Software Requirements Specification (SRS) that outlines functional and non-functional requirements, interfaces, data models, system behaviors, user interactions, and compliance obligations.

Rationale: Serves as a foundational document that guides the entire development process, ensuring that every feature and functionality aligns with user expectations and business objectives. It provides a detailed blueprint for developers and stakeholders, facilitating better planning and consistency in implementation. This approach ensures the development process remains focused on user and business needs, preventing deviations and ensuring the end product is robust, compliant, and aligned to the user's needs.

UDT-D6

Produce an emergency procedure manual outlining critical steps for handling urgent software issues and unexpected operational scenarios.

Rationale: Essential for enabling users to quickly respond to and resolve emergencies. This manual reduces downtime and ensures continuous operational efficiency, bolstering user confidence and system reliability.

24.2 Training Requirements

UDT-T1

Conduct interactive online workshops that demonstrate the core functionalities and creative writing tools of the web-based text editor.

Rationale: Interactive workshops provide hands-on learning experiences that allow users to see practical applications of the editor's features in real-time. This direct interaction enables users to ask questions and enhances their understanding and proficiency.

UDT-T2

Prepare video tutorials covering detailed use cases that highlight how to leverage advanced editing tools for various writing styles and genres.

Rationale: Video tutorials allow users to visually follow processes at their own pace, which is crucial for understanding complex features that enhance creative writing. It ensures users can fully exploit the text editor's capabilities.

UDT-T3

Offer a scheduled Q&A session following the release of new features to address user queries and ensure clarity on the latest updates.

Rationale: Scheduled Q&A sessions after updates ensure that users fully understand new functionalities and can discuss any issues or suggestions. This helps in maintaining high user satisfaction and engagement with the platform.

25. Waiting Room

25.1 Future Functional Requirements

FR-15

Users shall be able to click on a specific part of the text and select from a dropdown of alternative phrasings or sentence structures generated by the model.

Rationale: Providing users with quick access to alternative phrasings or structures allows for easier refinement of text, enhancing the user's ability to improve clarity, tone, or style without needing to manually rewrite sections. This feature offers flexibility and improves the writing experience by allowing users to explore multiple options efficiently.

FR-16

The system shall support an offline locally hosted language models for text generation through on-device inference.

Rationale: Offline functionality would rely on on-device inference to ensure users can generate text without internet connectivity. While this requires more local computational resources, it allows for uninterrupted productivity in environments where internet access is

limited or unavailable. However, this feature might be deferred to future iterations due to the complexity of integrating efficient on-device inference for large language models.

26. Ideas for Solution

includes a few solutions for `tinymorph`

IS-1

Provide Canvas for Planning

Rationale: A canvas interface allows users to visualize and organize their writing ideas, plans, and structures in a flexible and intuitive manner. Users can create, move, and connect elements to represent their writing process visually, enhancing planning and organization.

IS-2

Suggestion Panels

Rationale: Suggestion panels display alternative phrasings, sentence structures, or word choices generated by the model based on users input steering. Users can select and apply these suggestions to their text, improving writing quality and efficiency. Users can also preselect pre-trained SAEs to guide suggestion based on a certain author styles

---

Appendix

ref

Reflection

Aaron

The requirements helps me to structure the project and write down tasks needed to be done to build `tinymorph`.

I think requirements documentations are two-edged swords. Thinking from a perspectives of a person who write codes, requirements documents helps me to write down what needs to be done. However, some sections are some what too

vague and often share the same terminology with other sections, which leads to some duplications and sometimes premature optimization.

Most of the functional features were inspired from the chat I have with one of my friends in NYC, and the ideas stems from [@goodfire2024research] work on steerable Llama.

Well, I think Concurrency course 3BB4 would help with building inference server.

Training models, building sparse autoencoders, designing new UX for interacting with text, implementation activation caching for frequency used paged KV blocks.

I usually send the author of those papers and email and they usually respond back with answers with a lot of details. A lot of experimentation with existing infrastructure (meaning writing models and implementing papers in PyTorch), which I have been doing during my free time.

Nebras

One of the positive aspects of working on the SRS deliverable was the level of detail we were able to capture for each of the sections. Taking the time to explore the user needs thoroughly made it easier to articulate requirements that are meaningful and aligned with the project objectives. The iterative discussions for each section especially when working on the “Scope of Work” and “Stakeholders,” helped ensure that all aspects of tinymorph's functionality and purpose were effectively covered, which contributing to a more cohesive document.

A significant pain point was maintaining precision without overcomplicating the requirements or stepping outside the project's scope. This was especially challenging in sections like "Usability and Humanity Requirements" where striking the right balance between thoroughness and clarity required revisiting multiple drafts. I tackled this by actively rewriting areas that felt unclear and incorporating peer feedback to refine my approach. Another challenge was determining the relevance of certain sections like access requirements, which we ultimately decided to remove as they did not apply to our stateless system.

Our requirements were primarily based on our team's understanding of the tinymorph project's objectives and the target audience's needs. Insights from brainstorming sessions within the team and individual expertise shaped critical areas such as the "Look and Feel Requirements" and "Security Requirements". This internal feedback loop helped refine requirements to align with the vision for `tinymorph` as a creative writing tool that emphasizes user privacy and control.

Courses such as "Human-Computer Interfaces" have been invaluable for shaping our approach to creating an intuitive user experience, which is key for `tinymorph`. Additionally, "Software Requirements and Security Considerations" has helped me better understand how to structure the SRS and ensure consistency throughout the document, while the "Application of Machine Learning" course has given insights into how to effectively integrate ML models to enhance creative writing.

To complete this capstone project successfully, we need to expand our domain knowledge in creative writing tools and user psychology to better tailor the experience for writers. We also need backend expertise such as in deploying cloud-based inference servers and securely processing requests. UI/UX design skills are essential for creating an engaging and intuitive interface, and we also need to enhance our team management and communication skills to ensure smooth collaboration throughout the development process.

For domain knowledge, we could either study literature on effective writing tools and user workflows or conduct interviews with potential users to understand their needs better. I plan to engage directly with users for more practical insight. Backend expertise can be developed through online courses or by building hands on cloud based projects. We will try to have our backend focused member pursue hands on projects to gain more practical experience. For UI/UX, a combination of online courses and prototyping in tools like Figma will be pursued by designers like me, providing both theoretical knowledge and practical skills. To improve team management and communication, we will focus on applying best practices during our regular meetings to enhance team coordination.

Waleed

1. Something that went well in this deliverable was that it allowed me to clear up my past confusion as to what we are actually building. I focused on constraints and functional requirements, as the main sections and thus I really got a chance to understand more about how the product will actually function.
2. One of the main pain points I experienced was clarifying the functional scope of the project. Initially, it was challenging to align the technical requirements with the vision of the product, particularly when defining specific features like offline mode and version control. To resolve this, I had several discussions with the team to ensure that we were all on the same page regarding the project's goals and limitations, particularly for the first release.
3. Several requirements related to model tuning and steering were heavily influenced by discussions with supervisor and fellow peers. Features like manual control over text generation, tone and style customization, and

planning-related suggestions all were brought up in meetings and lead to related requirements. Conversations also highlighted the importance of allowing users to fine-tune model behavior to meet their writing goals, leading to the inclusion of advanced options like steering hyperparameters and inputting past writing samples for better model adaptation.

4. Courses like Real-Time Systems and Introduction to Machine Learning will be particularly valuable for this capstone. Real-Time Systems helps with understanding performance constraints and system responsiveness, while the Machine Learning course provides the theoretical foundation necessary to implement language models and customization features within tinymorph.
5. As a team, we will need to acquire expertise in model fine-tuning and language model behavior, especially related to LoRA and customizing language models for specific use cases. Additionally, we need to strengthen skills in front-end web development to create an intuitive and user-friendly editor interface, and cloud infrastructure management to efficiently scale and deploy the system.
6. Model fine-tuning and language model behavior (Aron and Waleed): Approach 1: Take specialized courses or tutorials on fine-tuning and language model behavior to understand the technicalities of adapting models for specific writing tasks. Approach 2: Experiment with open-source models, working on personal or side projects that involve fine-tuning models for specific applications.

Front-end web development (Nebras):

- Approach 1: Follow a structured curriculum in JavaScript, HTML, and CSS on platforms like Udemy.
- Approach 2: Collaborate on open-source front-end projects to get hands-on experience and immediate feedback from the developer community.

Cloud infrastructure management (Nebras and Lucas):

- Approach 1: Use platforms like AWS or Google Cloud to follow tutorials and labs related to setting up scalable cloud infrastructure.
- Approach 2: Participate in cloud computing hackathons to quickly learn cloud architecture and deployment strategies in a competitive setting.

Lucas

1. The part of task division went quick, we choose to use communication in group to resolve the confusions. The feedback in teams is quick in response.
2. I am still not fully catch up the background knowledge about this project, and for some of the decisions I'm still not on the same page with my peers. Self-demand

learning and communications are definately needed for following procedures.
3. Most of the requirements are from peers as this is also a texting tool we can use for ourselves. We discussed the requiremnts from the user's perspective of view.
4. COMPSCI 4ML3, the course about machine learning gave me some theoretical knowledge background for the project. The SFWRENG 3X03 is also useful as guidance to have some overview of developing a project.
5. SAE training is the skill I need for this project. It is a feature extraction phase on user input to get token for larger model to run. Some knowledge on applied machine laerning is also important to bring this project to practice.
6. We have gathered related website about the SAE training, with the github project as tutorial to have a look. Some paper from google scholar is also greatly helpful for self-demand learning needed in this project. These two approches are recommanded because they allow self-paced learning asynchronously, matching to individual time slot.

Revision

Date	Version	Notes
Sept. 16 2024	0.0	Initial skafolding
Sept. 25 2024	0.1	Migration to Volere template
Oct 10 2024	0.2	SRS first draft