1. Introduction

This design document outlines the architecture, objectives, requirements, and components of a comprehensive Large Language Model (LLM) based Abstract Classification Framework. An LLM is a type of generative AI that has been specifically architected to help generate text-based content. The system's primary goal is to seamlessly integrate a versatile LLM for abstract classification while providing users with a flexible interface to configure hyperparameters, override classifications, and evaluate model accuracy. Hyperparameters are tunable settings influencing a machine learning model's behavior and performance. The system aims to enable the efficient management of classified abstracts and empower users with a secure and intuitive platform for abstract processing.

2. Objectives & Requirements

- Develop a flexible and modular architecture that allows the seamless integration of a Large Language Model (LLM) for abstract classification.
- Provide an interface for users to select a LLM and configure the hyperparameters to optimize its classification performance.
- Allow users to evaluate a model’s accuracy.
- Enable the use of classified abstracts to fine-tune a model.
- The system should allow for manual overrides if the automatic LLM classification for a category appears inaccurate.
- Create a user management system including registration, authentication, security, roles and permissions.
- Create an online database to store existing abstracts, fetch new abstracts from credible sources such as PubMed and Google Scholar, assign those abstracts to a user and store the results of classified abstracts in the database.

3. Architecture

The overall architecture consists of several interconnected components, as shown in Figure 1.

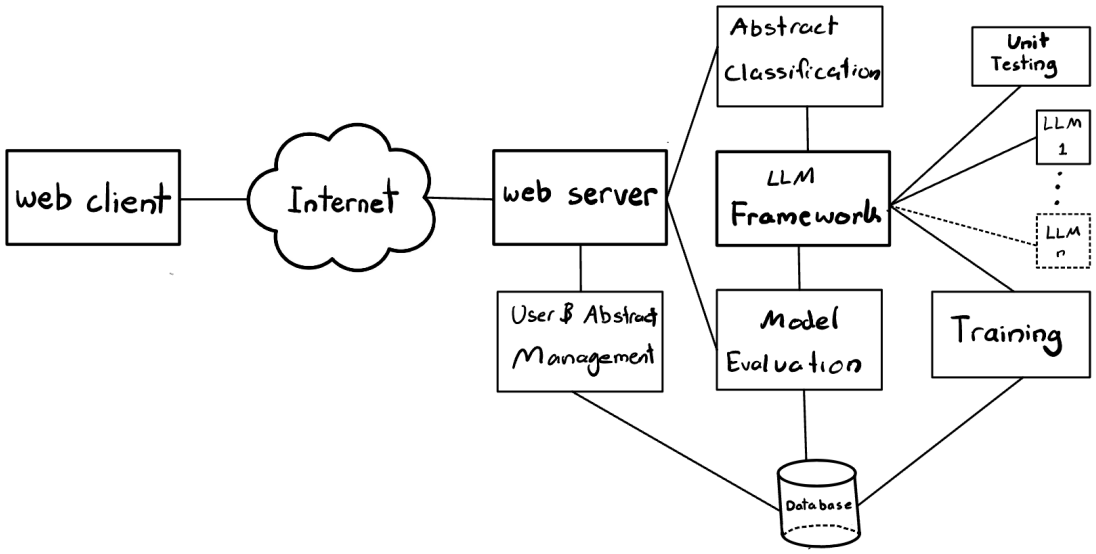

Figure 1 - Abstract Classification Architecture

The user interacts with the classification system through the ‘web client’. The requests are handled by the ‘web server’, which routes them to either of the following four components:

- Abstract classification: Responsible for classifying abstracts using selected LLMs and configurable hyperparameters.
- Model evaluation: Evaluates the accuracy of the LLM and provides visual representations of the results.
- User management: Manages user registration, authentication, security, roles, and permissions.
- Abstract management: Administers the assignment of abstracts to users, allowing users to fetch, review, and upload results.

The requests for ‘abstract classification’ and ‘model evaluation’ are processed by the ‘LLM framework’, which in turn communicates and gets responses from third party LLM providers (LLM-1, LLM-2, … LLM-n). A cloud database is used to store user and abstract information, which can be used for assigning the unclassified abstracts to a set of users, as well as using the classified abstracts to train the LLM.

3.1 Abstract Classification

The abstract classification component is responsible for processing classification requests and interacting with the selected LLMs. The following sections describe the user interface (web client) and the logic (web server) to process the classification requests.

3.1.1 Web Client

The web client exposes the following two user interfaces:

- Classification input form
    - Configurable LLM and hyperparameter settings

    - Text boxes to enter title and abstract
    - Option to fetch assigned abstract, edit abstract and title content, and classify abstract
- Classification result
    - Display title and abstract
    - Summary of abstract
    - Classification per category (such as publication type, data type, etc.) and LLM generated justification for the choice
    - Help documentation specific to each category, describing its options
    - Manual override options for each category classification. This results in a follow up request to generate a justification for the user-specified option. Any dependent categories will automatically be regenerated with the user-specified option as context.
    - Option to upload results to the database

3.1.2 Web Server

The web server handles classification logic through the following endpoints. HTTP endpoints are specific URLs that serve as entry points to a web server's functionalities, allowing clients to interact with the server by making requests to these URLs:

- Get Abstract (HTTP-GET): Handles the user request to fetch the assigned abstract.
    - Input: None
    - Output: Title and Abstract
    - Fetches unclassified abstract from database, and sends to the web client, which uses it to fill the title and abstract fields
- Classify Abstract (HTTP-GET): Sends the classification input form to the web client to get Title, Abstract, Model (GPT-3.5, GPT-4, Davinci, etc.), Hyperparameters (temperature, etc).
    - Input: None
    - Output: Classification input form
- Classify Abstract (HTTP-POST): Handles the request for classification.
    - Input: Title, Abstract, Model, Hyperparameters
    - Output: Classification result page, with Title, Abstract, and a placeholder for Summary and Category classifications
    - Redirects user from classification input form to classification result page
- Answer (HTTP-GET): Fetches the Category and Justification from the LLM, and populates the placeholders on the classification results page.
    - Input: Category (e.g. Publication type, Subpopulation, etc), Model, Title, Abstract, Temperature, Parent category (such as Population to Subpopulation), Manual override (if any)
    - Output: Classification result and Justification
    - Generates a prompt based on the Category, Parent category, and Manual override (used to override the LLM generated answer). Interacts with the

framework using the above prompt, and specified model/hyperparameters to generate the result
- Upload (HTTP-POST): Uploads the classified abstract to the database
    - Input: Title, Abstract, Summary, Category results and justifications
    - Output: None
    - The implementation will use a unique identifier for an abstract while interacting with the database

3.2 Model Evaluation

The model evaluation component assesses the accuracy of the LLM by comparing its classifications with pre-classified abstracts. The following sections describe the user interface (web client) and the logic (web server) to process the model evaluation requests.

3.2.1 Web Client

The web client exposes the following two interfaces:
- Evaluation input form
    - Configurable LLM and hyperparameter settings
- Evaluation result
    - Display progress bar for each category indicating the number of abstract results compared and evaluated
    - Accuracy metric for each category
    - Overall accuracy across all categories
    - Category-Accuracy graphs for key combinations of models and hyperparameters

3.2.2 Web Server

The evaluation logic is organized as the following endpoints:
- Evaluate Abstract (HTTP-GET): Sends the evaluation input form to the web client to get Model (GPT-3.5, GPT-4, Davinci, etc.), Hyperparameters (temperature, etc).
    - Input: None
    - Output: Evaluation input form
- Evaluate Abstract (HTTP-POST): Handles the request for evaluation.
    - Input: Model, Hyperparameters
    - Output: Evaluation result page, with a placeholder for per-category accuracy, overall accuracy, number of pre classified abstracts (for index), and visual representations
    - Display accuracy vs category graphs for key combinations of models and hyperparameters
- Compare (HTTP-GET): Fetches the comparison results for each combination of category and classified abstract.
    - Input: Model, Temperature, Category, Index (abstract to process, updated by client), Match

    - Output: Model, Temperature, Category, Index, Match (whether category result matches pre classified abstract result, updated by server)
    - Compares each category result with pre-classified abstract category and updates the match count. The client uses the index and match count to calculate and update the accuracy metric.

3.3 Model Tuning

The model tuning component is designated to explore methods for fine-tuning LLMs to enhance classification accuracy.

TBD - Needs further research.

3.4 User Management

The user management component covers user registration, authentication, security, roles, and permissions.

- Registration
    - The module allows admins to create new user accounts by providing necessary information such as username, email, and password.
    - It should validate and store this information securely in a database, ensuring that passwords are hashed (using a strong encryption algorithm) before being stored.
- Authentication
    - Once registered, users need a way to log in. The authentication process verifies the user's identity before granting access to the system.
    - The module should handle authentication securely, preferably using industry-standard protocols like OAuth, OpenID, or token-based authentication.
- Security (Do we need this?)
    - The module needs to protect sensitive user data, both during transmission (e.g., over HTTPS) and at rest (when stored in the database).
    - It should handle security features like password reset mechanisms, account lockouts after multiple failed login attempts, and session management to prevent unauthorized access.
- Roles and Permissions
    - In many systems, users have different roles with varying levels of access (e.g., regular user, admin). The module should support role-based access control (RBAC) to manage permissions.
    - Admin users should have the ability to manage other user accounts, like updating information or deactivating accounts.

3.5 Abstract Management

- Admin can assign abstracts to any of the registered users.
- Enable users to fetch assigned abstracts.

- Allows users to flag an abstract for review.
- Once users are done with an assigned abstract, allow them to upload the results.

4. Data Flow

The dataflow between the front end and the back end for the following operations is captured below:

- Fetch abstract
- Classify abstract
- Upload abstract
- Evaluate model

4.1 Fetch Abstract

- User requests an assigned abstract.
- Web server retrieves the abstract from the database.
- Web client displays the title and abstract.

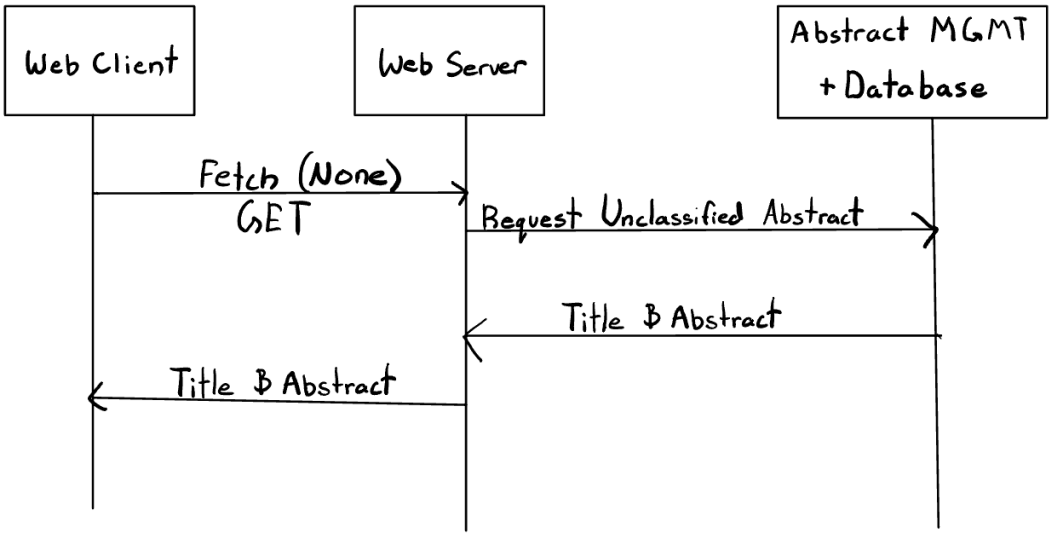

Figure 2 - Fetch Abstract Data Flow

4.2 Classify Abstract

- User submits classification request (Title, Abstract, Model, Hyperparameters).
- Web server sends the request to the selected LLM.
- LLM generates category and justification.
- Web client displays the classification result.

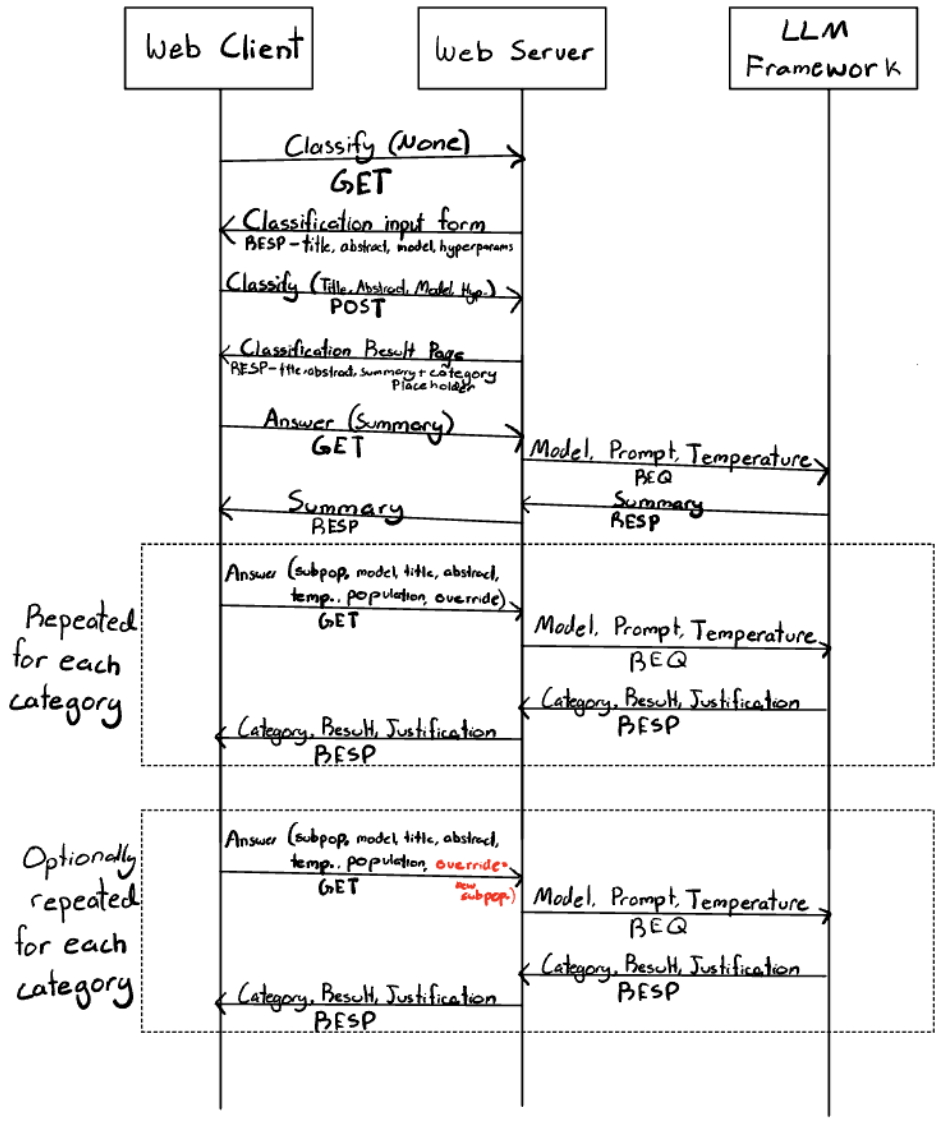

Figure 3 - Classify Abstract Data Flow

4.3 Upload Abstract

- User submits classified abstract and results.
- Web server processes the submission.
- Data is uploaded to the database.

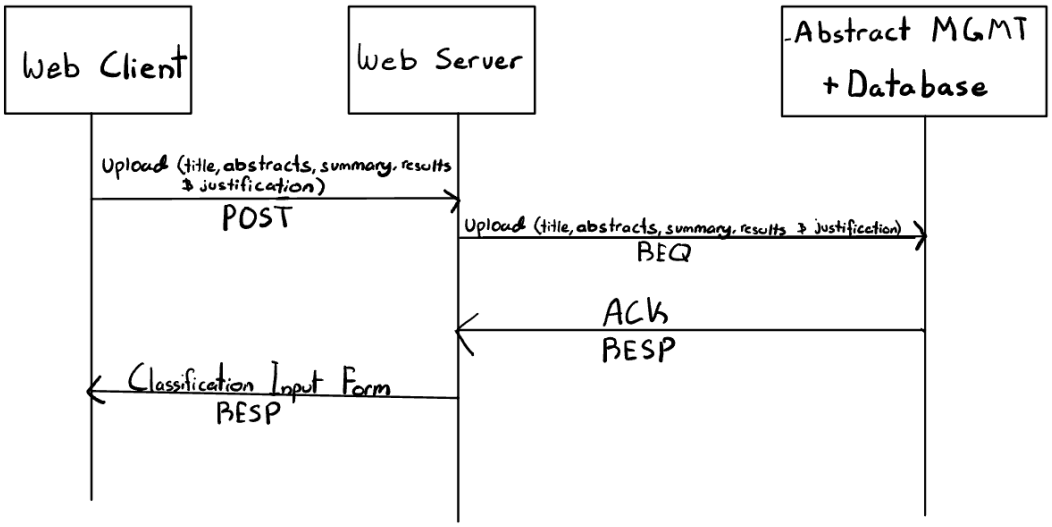

Figure 4 - Upload Abstract Data Flow

4.4 Evaluate Model

- User requests model evaluation (Model, Hyperparameters).
- Web server initiates the evaluation process.
- Comparison with pre-classified abstracts is conducted.
- Web server displays evaluation results.

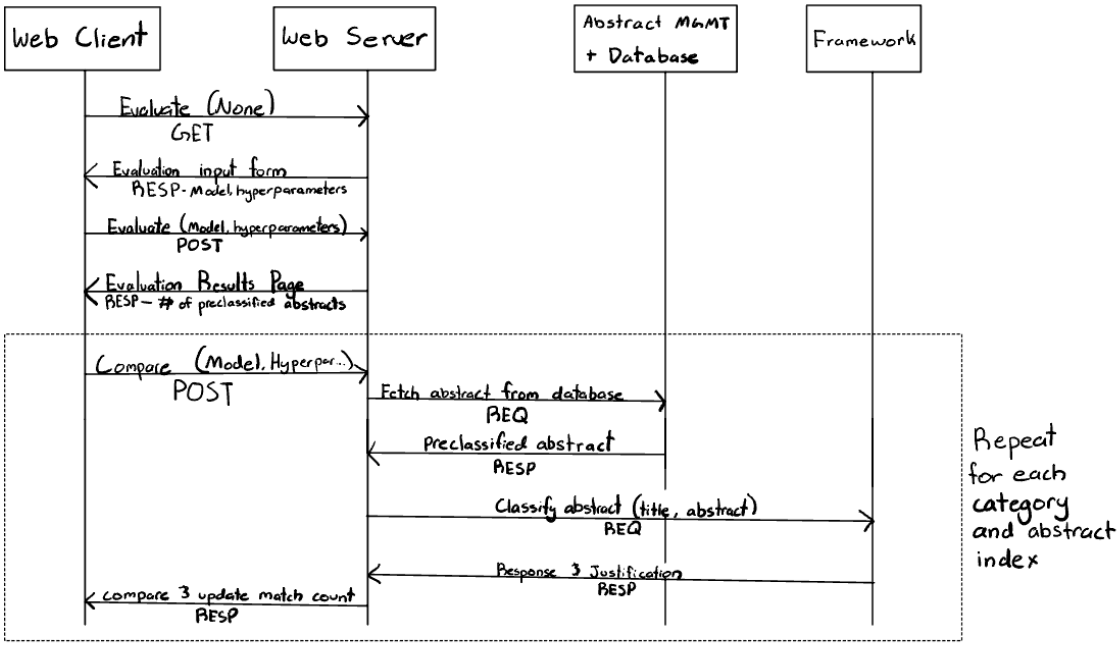

Figure 5 - Evaluate Model Data Flow

5. Future Work

- Explore additional ways to evaluate model accuracy, and use the optimal settings as default settings for classification.
- Expose more options to tune the LLM, such as custom prompts.
- Allow unsupervised learning techniques, such as clustering and topic modeling, to discover new categories or questions.
- Extend the system's capabilities to allow classifications of abstracts from various fields, such as neuromarketing and neurobiology.

6. Conclusion

This design document outlines the comprehensive Large Language Model (LLM) Abstract Classification Framework, encompassing user interfaces, backend logic, model evaluation, user management, and abstract management. The system's flexible architecture and modular components aim to provide users with a powerful platform for efficient and accurate abstract processing, while also paving the way for future advancements in the field of language modeling and classification.