CBM-GUI: GUI for Concept Bottleneck Layer Integration in LLMs

Gabriel Cha
gcha@ucsd.edu

Steven Luong
sxluong@ucsd.edu

Tsui-Wei (Lily) Weng
lweng@ucsd.edu

Abstract

Large Language Models (LLMs) have shown remarkable capabilities, but their "black-box" nature often makes them difficult to interpret. Concept Bottleneck Layers (CBLs) address this by linking model outputs to human-understandable concepts. Our project introduces a user-friendly Graphical User Interface (GUI) that allows users, regardless of their technical background, to easily integrate CBLs into pre-trained LLMs (provided as .pt files), display top N highly activated concepts, and analyze and prune biased concepts. By identifying and unlearning biased concepts, these models can produce fairer, more reliable predictions. Our end goal is to provide users with the ability to scrutinize and adjust models to be explainable and fair. This work builds upon the research and framework developed by Weng et al. (2024).

Frontend Code: https://github.com/gabrielchasukjin/cbm-gui-frontend
Backend Code: https://github.com/sxluong/CBM-GUI-Backend

1 Background and Related Work

Large Language Models (LLMs) excel in various tasks but often lack transparency. Concept bottleneck layers (CBLs) allow for transparency by connecting model outputs to human-readable concepts. However, existing CBL implementations require technical expertise, restricting their accessibility. Our project builds on Weng et al. (2024) to create an easy-to-use GUI, allowing users of all skill levels to visualize concept activations, adjust weights, and unlearn biased concepts for greater interpretability, trustworthiness, and fairness.

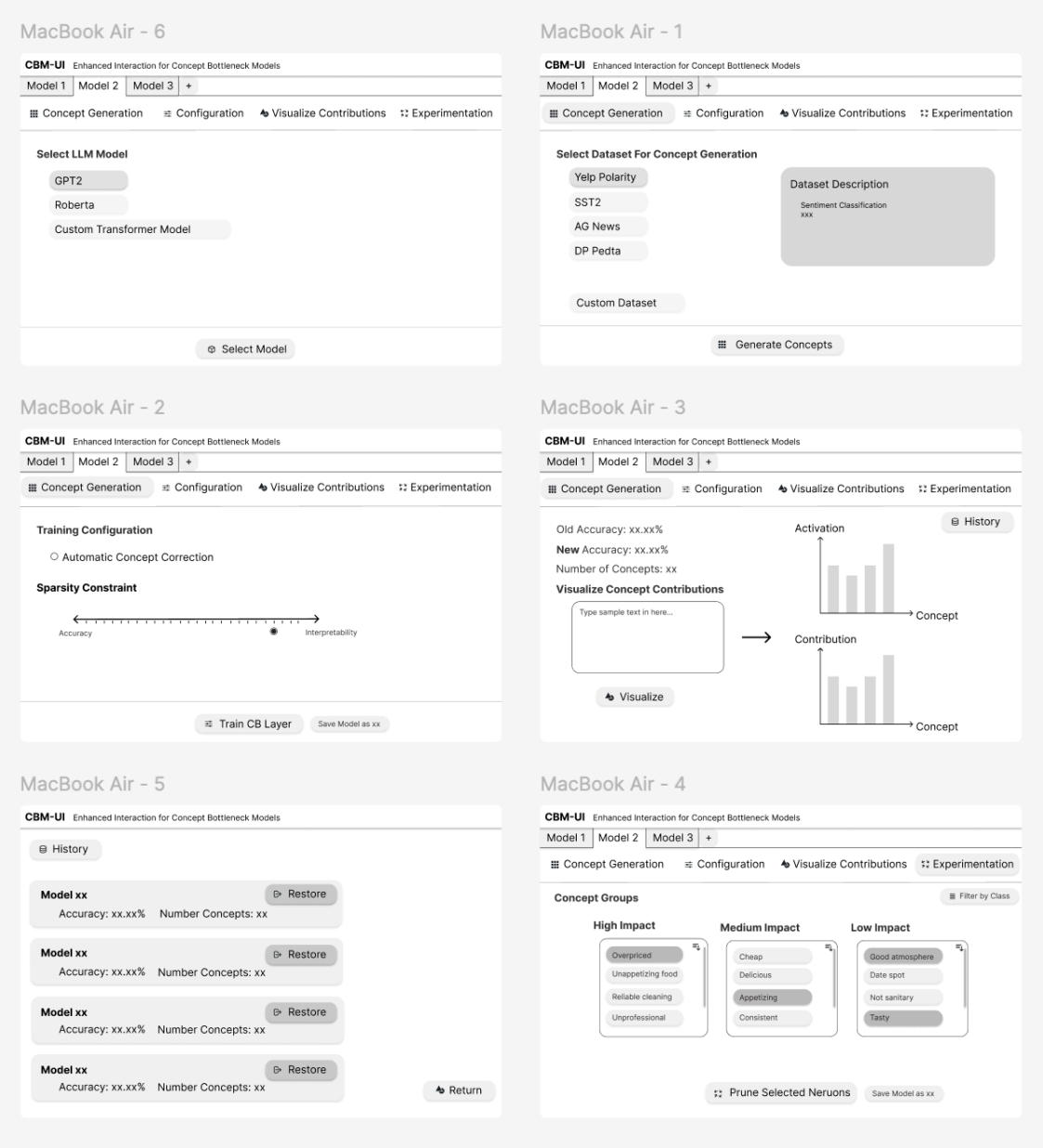

Figure 1: CBM-GUI Workflow: Simplified Interface for Model Transparency and Fairness

2 Application Architecture and Features

We have moved beyond the Minimal Viable Product (MVP) stage and developed a full application with microservices, database storage, and a structured backend-frontend system. The application features a Django backend, React frontend, and SQLite3 database for model storage.

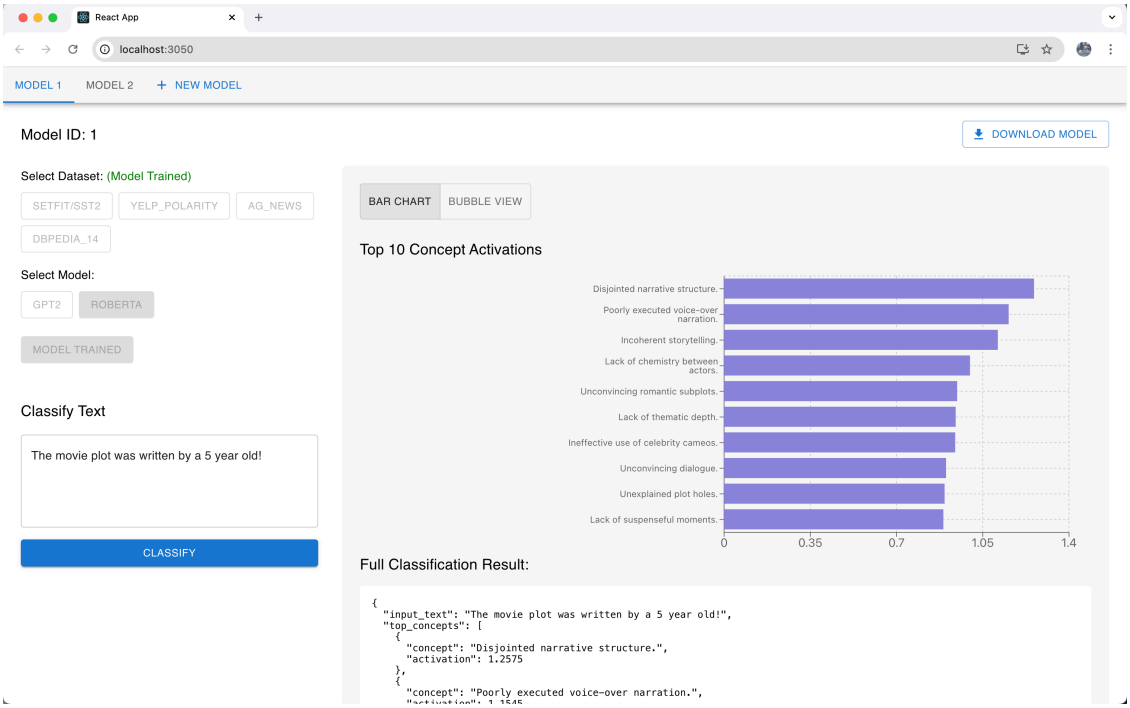

Figure 2: Intuitive experience for user to view top 10 concept activations

The CBM-GUI is designed to provide users with an intuitive and interactive experience for training, visualizing, and refining Concept Bottleneck Layers in LLMs. Users can:

- Train multiple models in separate tabs with different configurations.
- View real-time progress of training and see concept activation visualizations.
- Analyze the top contributing concepts for each model decision.
- Save trained models to the SQLite3 database for future use and comparisons.

The GUI simplifies the process of integrating CBLs into LLMs by providing clear workflows and real-time visual feedback, making advanced interpretability tools accessible to both researchers and practitioners.

3 Tech Stack

The CBM-GUI utilizes a Django-React structure:

- **Frontend:** React handles dynamic UI, user interactions, and state management.

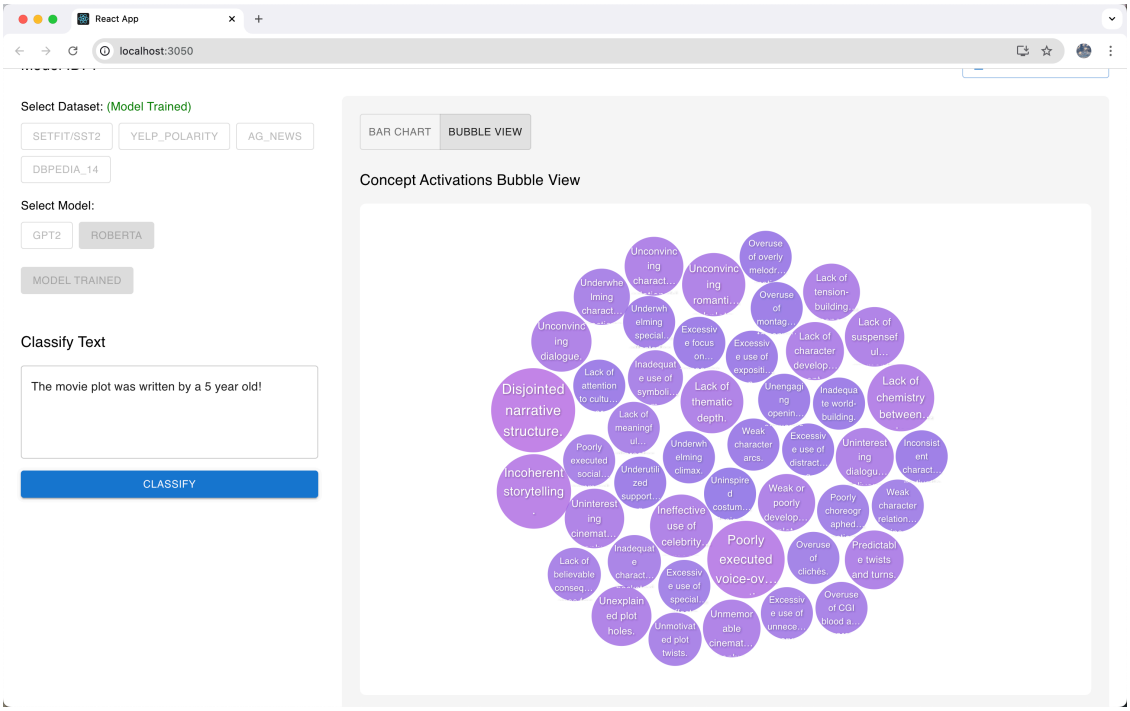

Figure 3: An intuitive bubble view allows users to visualize concept activations effectively.

- **Backend:** Django processes API requests, manages model training scripts, and interacts with the database.
- **Microservices:** Each microservice performs specific tasks, such as running model training scripts and handling post-processing.
- **Database:** SQLite3 stores trained models, allowing users to switch between them and restore previous configurations.

When a user interacts with the GUI, React sends API requests to Django, which calls the appropriate microservice. The backend then retrieves or updates stored models, allowing real-time training updates and result visualization.

4 Database and Model Management

The system integrates an SQLite3 database to manage trained models. This enables:

- Users to train multiple models in separate tabs.
- Storage of trained models for future retrieval and comparison.
- Easy switching between models without retraining from scratch.

Users can restore past models, facilitating iterative experimentation and performance evaluation. Each model entry in the database includes metadata such as training parameters, dataset used, and timestamps.

5 Experimentation and Pruning

The pruning feature, which is currently in development, enables users to remove biased or low-impact concepts and retrain the model. Users can:

- Select concepts to prune based on their contribution.
- Observe how pruning affects model accuracy.
- Store updated models in the database for comparison.

This feature ensures users can systematically refine models and improve fairness and robustness.

6 History and Restoration

The history tab now includes:

- A record of trained models with metadata.
- A ”Restore” button to load previous configurations.
- Easy model switching for comparative analysis.

This supports reproducibility and iterative model development.

7 Conclusion

The CBM-GUI extends LLM interpretability by offering an intuitive, accessible interface with integrated model training, microservices, and database storage. Users can train, compare, and refine models efficiently, ensuring transparency and fairness. Future work will explore additional features such as cloud-based training and real-time collaborative editing.

8 Acknowledgment of Design Tools

The following drawings were created using Figma. You can view the designs at this link: CBM-UI Design in Figma.

9 References

Sun, C.-E., Oikarinen, T., and Weng, T.-W. Crafting large language models for enhanced interpretability. In ICLR, 2024.