D2.1 Requirements Analysis & Design of the Themis Platform

Christos Karanikolopoulos[1], **Panagiotis Papadakos**[1, 2], **Panayiotis Tsaparas**[1]

[1] Department of Computer Science and Engineering, University of Ioannina, Greece
[2]Institute of Computer Science (ICS) - Foundation for Research and Technology - Hellas (FORTH), Greece

1. Introduction

The purpose of this document is to provide a comprehensive analysis of the requirements, the design of the architecture, and the implementation details of the open-source bias measuring platform Themis. Themis aims to become a reference benchmarking platform for measuring bias in data that are extracted from online information platforms (OIPs). This document specifically focus on bias measurement over text data. However, the proposed architecture could be adapted for other kind of data like graph-data (e.g., social data) in the future.

The platform will host publicly available data collected from various OIPs and other available relevant datasets, and will provide benchmarks and tools for analyzing metrics for various aspects of human and algorithmic bias. Themis will use state-of-the-art software engineering practices, software frameworks, and efficient and effective machine learning (ML) and deep learning (DL) pipelines and models, offering a simple, easily maintainable, robust and evolvable platform. The platform will consider the current state-of-the-art approaches for evaluating large language models (LLMs) and DL models [4] and open-source libraries and frameworks[1].

In the following, the document covers the functional and non-functional requirements of the platform, the system architecture, and various design and technical considerations. Further, it describes the key components and their interactions, in order to achieve the desired bias measuring benchmarking functionality, that can be easily deployed and used by interested parties, through an intuitive User Interface (UI).

2. Requirements

In this section we discuss the requirements of the Themis platform. Initially, we describe the functional requirements and the non-functional ones, since both of them are important for defining the scope and behavior of the platform. The aim is to develop a platform that will be used as a benchmark for measuring the bias of datasets, models, configurations, and mitigation algorithms across a number of metrics, in a concise, transparent, reproducible, and easy to publish way.

2.1. Functional Requirements

The functional requirements describe the behaviors, features, and functionalities that the Themis platform must have to meet the needs of its users.

2.1.1. Definition of Bias Measuring Tasks

The platform will provide a cohesive and standardized approach for defining and deploying bias measuring tasks. Specifically, through configuration files the users of the platform will be able

[1] https://github.com/EleutherAI/lm-evaluation-harness/

to define the whole evaluation pipeline, affecting the deployed and compatible datasets, models, debias algorithms, and bias metrics.

At the core of the above is the bias measuring task, that is described through a configuration file. Such tasks cover measuring the bias in QA, text generation, masked tokens completions, and counterfactual inputs. The tasks are model-agnostic and connected to one or more categories of applicable datasets, debias algorithms and bias metrics. The inputs of the tasks also include utility functions for data pre-processing, that are again implemented through the configuration file.

Prompts, are the text containing the set of instructions that will trigger specific responses from the LLMs. Except from already created datasets of prompts, THEMIS will also facilitate the generation of new prompt collections. he supported prompts will be designed on the specific bias measuring task formulation. Tokenizers will be abstract, allowing prompts to be prepended with role messages and special tokens, depending on the chosen model. This will enhance reproducibility and ensure that measurements are consistent and comparable across different runs and models.

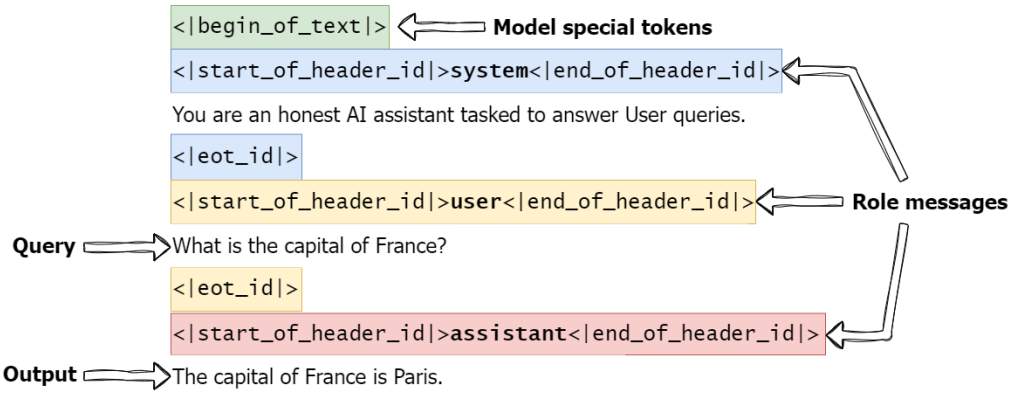

Figure 1. Prompt design using Llama3 special tokens

2.1.2. Support Different Types of Bias Related Datasets

There is a plethora of publicly available datasets for bias measurement [2]. For example, with regard to LLMs, bias can be measured through counterfactual inputs or prompts. In the first case, datasets contain masked token sentences, and the LLM is asked to predict the token, or pairs of sentences with a disadvantaged / advantaged social group, and the model is asked to predict which sentence is more probable. The prompt-based datasets use LLMs to complete sentences or for QA. The platform should be sufficiently general to support the variety of datasets that evaluate the various bias measurement tasks. The users of the platform should be able to load already published datasets or upload and import custom-made datasets. For prompt-based datasets, approaches that would allow the generation of model-independent prompts and their translation to model-based prompts (if possible) should be explored. The platform should also support retrieval augmented generation (RAG) [3] for LLMs, where the prompt is fed into a document store and the retrieved documents are then fed along with the prompt to the models.

2.1.3. Variety of DL and LLM Models and Configurations

The domain of LLM and DL models is rapidly evolving. THEMIS should allow the easy deployment and configuration of state-of-the-art LLM and DL models by exploiting already developed open-source libraries like the transformers[2], vllm[3], LMDeploy[4], and LangChain[5]. If possible, support for commercial models where THEMIS could interact through APIs should be provided.

[2] https://huggingface.co/docs/transformers/index
[3] https://github.com/vllm-project/vllm
[4] https://github.com/InternLM/lmdeploy
[5] https://python.langchain.com/v0.2/docs/introduction/

2.1.4. Bias Metrics

As analyzed in deliverable D1.1, there are different types of metrics for bias evaluation. For example, for LLMs there are metrics based on embeddings, probabilities, and generated text-based ones. The platform should provide general interfaces that will allow the easy implementation, deployment, and evaluation of any kind of aforementioned metrics.

2.1.5. Debias Algorithms

Mitigation algorithms are also an important component of the platform. Debias algorithms can be deployed in different stages in the LLM/DL workflows. Specifically, they can be deployed as a *Pre-processing* step that can change the inputs (e.g., training data and prompts) fed into the model, in the *Training* step (e.g., updating parameters, loss-function modifications), in the *Processing* step with no further training (e.g., modify model probabilities), and finally in the *Post-process* step, where algorithms are run over the generated results (e.g., replace biased words). Ideally, THEMIS should allow the easy implementation and deployment of any kind of the previous four categories of mitigation algorithms. However, due to the variety of mitigation algorithms in each step, the complexity of implementing each algorithm, and the fact that they can be used in different parts of the pipeline, we will focus on supporting the most efficient ones based on the bibliography [2].

2.1.6. Bias Reporting and Leaderboard

Finally, the platform will support a consistent and concise way of reporting the evaluation results of the various bias measuring tasks. Specifically, for each benchmark run, the platform will generate a detailed json file that will report all the metadata relevant to the benchmark task (e.g., model, parameteres, dataset, metrics, values, etc.). The platform will also provide a leaderboard where these files can be uploaded for future reference and comparison. In this way, THEMIS aims at becoming the reference for bias benchmarking.

2.2. Non-functional Requirements

The non-functional requirements focus on the quality attributes and global characteristics of the THEMIS platform.

2.2.1. Modularity

A key aspect of the platform design is modularity. The separation of concerns in different components in a layered architecture, ensures that the platform will be easy to understand, maintain, extend and troubleshoot. For each component, abstract interfaces will be defined, in order to allow for different implementations. This will make it easier to replace or upgrade components without affecting the rest of the system.

2.2.2. Scalability

The platform should allow the easy exploitation of new hardware resources when scaling vertically. Specifically, the platform should utilize additional resources like CPUs, GPUs, and/or memory when they become available. The management of the datasets should be done efficiently, so that the number and size of the datasets do not become a bottleneck. Finally, the bias detection algorithms should be optimized and implemented efficiently using efficient data-structures.

2.2.3. Transparency

Since THEMIS plans to become a hub for bias benchmarking, it is important to provide all provenance metadata about the data sources, the collection methods, and any preprocessing and transformation steps, as well as versioning for their evolution. In addition, regarding the implemented algorithms, detailed documentation about each algorithm, including mathematical descriptions,

assumptions, and limitations should be provided. Since the platform will be open-source, this will allow anyone with code skills, to review, audit and understand the inner workings of the algorithms. The entire benchmarking pipeline should also be documented along with the reported results. Detailed instructions, scripts and configuration files for running a benchmark independently should be provided for independently replicating the results and comparative assessments.

2.2.4. Usability

Finally, the platform should be user-friendly, with a clean and intuitive interfaces. Specifically, the configuration files for defining the evaluation tasks should be clean and concise. The involved user interfaces (UIs), such as the leaderboard, should provide the results in a concise and user-friendly way, using (if applicable) diagrams and graphs for more intuitive representation of the results.

3. System Architecture

3.1. Overview

In order to build a reliable and easy-to-maintain system, we divide THEMIS into six components. Each component is implemented separately, allowing for a low-coupled and highly cohesive system. This modular approach improves reliability, scalability, simplifies maintenance, and provides the corresponding interfaces for defining, loading, implementing, deploying, measuring, and publishing the various bias benchmarking tasks, datasets, algorithms, and metrics.

3.2. Components

Below we describe the six components, namely the *Controller*, the *Data Management*, the *LLMs/DL Model Management*, the *Mitigation*, the *Evaluation*, and the *Front-end* components. The arcitecture is depicted in Fig. 2.

3.2.1. Controller

The controller is the central component of the proposed THEMIS architecture. It is responsible for interacting and coordinating the other components, validating the pipeline submitted by the user of the architecture, and finally deploying it. In more detail, the validation component is responsible for checking that the submitted evaluation pipeline, task, metrics, mitigation algorithms, models, and datasets involved are compatible. If the validation is successful, the controller is then responsible for deploying the pipeline by interacting and coordinating the rest of the components.

3.2.2. Data Management

This unit is responsible for the storage, access, and processing (e.g., simple preprocessing transformatioons) of all the data that can be used in deployed bias measurement pipelines. This component also manages the properties of the corresponding artifacts along with the data/metadata describing bias measuring tasks and the compatible pipelines, along with any instances (runs) of it and its results. This data can then be presented in the leaderboard. Specifically, this includes datasets of prompts, prompt templates, vector stores with embeddings or collections of documents that can be used as context in RAG approaches, lexicons with biased or stereotypical words, along with bias measurement tasks metadata and metric values, that are used by the THEMIS platform. For example, for a given task, the module will provide a collection of prompts to the deployed LLM models, which will generate the outputs that will be evaluated. In addition, this component can also enrich the models with a context provided by external collections for which the models were not trained, to examine any bias present in some corpora. This component is also responsible for feeding the controller's validator with the required metadata.

3.2.3. LLMs/DL Model Management

This module is responsible for managing the LLM and DL models that will be deployed by the platform. It provides the controller with the different model variants, hyperparameter configurations, and adapter plugins for quantization, which improves efficiency, and low rank adaptations for fine-tuning. It then feeds the output of the models to the next stages of the pipelines. The module can also provide access to external, commercial or free models, where the access is provided through APIs. Finally, this component also offers the ability to fine-tune the provided models over specific datasets provided by the *Data Management* component.

3.2.4. Mitigation

This component is responsible for providing implementations of various mitigation algorithms along with interfaces for implementing new ones. As already mentioned mitigation algorithms can be deployed in various stages of the pipeline, and THEMIS will provide the interfaces and representative implementations for the most effective ones. This component communicates with the *Controller* for applying the mitigation algorithms to the appropriate stage of the pipeline.

3.2.5. Evaluation

The evaluation component is responsible for the definition and implementation of the various bias metrics for the various supported bias evaluation task. It will provide interfaces for each supported type of metric which can then be implemented by the various metrics under consideration. The component is responsible for logging all the metadata of every run, such as the input and output of a prompt, hyperparameters, and model / dataset versions. There metadata are then stored in a convenient format in the *Data Management* component.

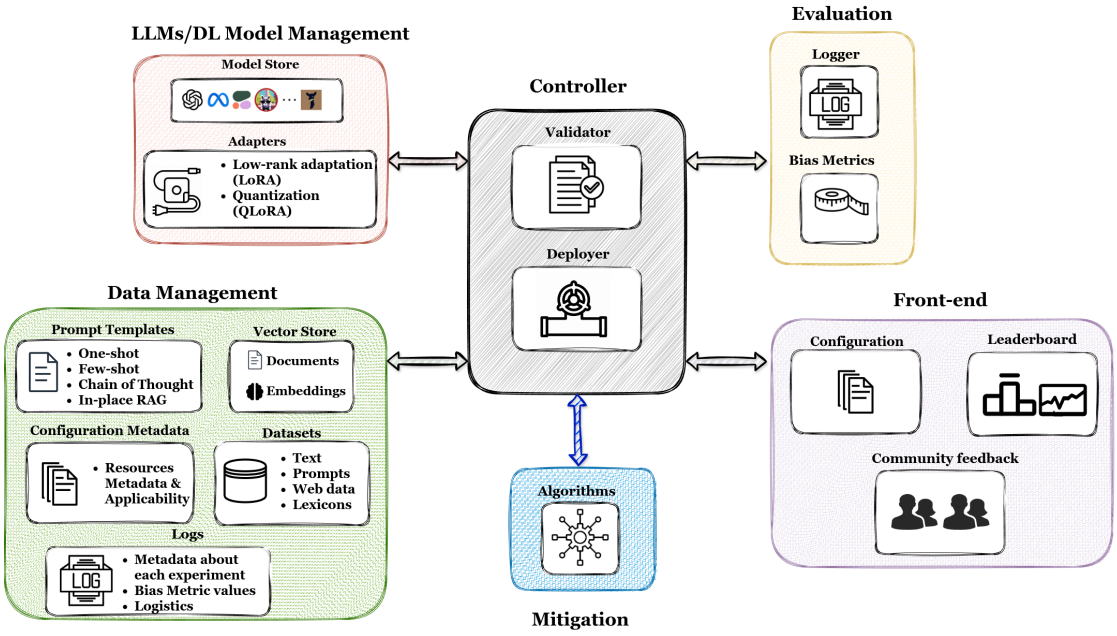

Figure 2. THEMIS Architecture

3.3. Front-end

This component provides the User Interfaces (UI)s with which any user of the platform can inspect the various artifacts supported by the platform (.e.g, datasets, models, tasks, metrics, etc.), setup experiments that should be deployed to measure bias and inject their results to the leaderbooard, explore the leaderboard for different types of tasks, retrieve the corresponding configurations for reproducing the experiments, and allow feedback and comments.

4. Implementation

In the proposed architecture, the *Controller* is a core component that interacts with all the other components. The implementation of the platform should provide appropriate abstractions, interfaces and APIs that can encapsulate the specifics of the various implementations, allowing the easy integration of new functionality/implementations (e.g., metrics).

4.1. Software Requirements

The platform will be developed in Python using the conda and poetry[6], two well-established tools in the ecosystem, as the environment and package manager respectively. The combination of both, provides a conflict-free, collaborative and easy to deploy system.

Currently, the norm of interacting with DL and LLM models is through the HuggingFace[7] ecosystem. Themis will be based on HuggingFace, through which access to public models and datasets will be gained. Additionally, the ability to load custom models and datasets is also supported. Moreover, inference and service LLM libraries like vLLM[8], which is known for its efficient memory usage [5], will be explored. Other extensions that will allow useful functionality such as abstract templates for easy prompt design, as well as in-context learning for RAG applications can be provided by LangChain[9], that can be integrated with vLLM.

The evaluation of the pipeline will be developed on top of libraries and tools build by EleutherAI[10] and Weights & Biases[11]. EleutherAI implements LM Eval Harness[1], a framework used for testing LLMs on existing or custom tasks. Tools provided by Weights & Biases will be used to systematically monitor the LLM experiments, their configurations, hyperparameters, outputs and artifact versioning.

4.2. Hardware Requirements

The platform will be deployed using resources available in the Distributed Management of Data Group lab of the Computer Science and Engineering Department of the University of Ioannina. The lab offers access to an NVIDIA RTX 6000 with 48GB of VRAM. This GPU offers a good starting point for our experiments, since it has enough memory to support inference for most medium range parameter LLMs. In order to scale our experiments to even larger models, the options are to either scale vertically or use external APIs for running inference. Data storage infrastructure will be extended with more capacity by acquiring a NAS server through project resources.

5. Conclusion

This document provides a detailed analysis of the requirements, design architecture, and implementation details for the open-source bias measuring platform Themis. The platform aims to serve as a benchmark for measuring bias in text data from online information platforms (OIPs). Themis will host publicly available datasets from various OIPs and other relevant sources, and will offer tasks and metrics for measuring and analyzing human and algorithmic bias. It will deploy state-of-the-art LLMs and DL models in a robust, evolvable, and easy to maintain platform. It will also integrate the latest approaches for evaluating large language models and deep learning models, leveraging open-source libraries and frameworks. Finally, by offering tools for exploring and reproducing the results of the various models and configurations, users of the platform will be able to get various insights related to the different stages that bias can emerge and how it can be mitigated.

[6] https://python-poetry.org
[7] https://huggingface.co/
[8] https://github.com/vllm-project/vllm
[9] https://python.langchain.com/v0.2/docs/introduction/
[10] https://www.eleuther.ai/
[11] https://wandb.ai

References

[1] Stella Biderman et al. "Lessons from the Trenches on Reproducible Evaluation of Language Models". In: *arXiv preprint arXiv:2405.14782* (2024).

[2] Isabel O Gallegos et al. "Bias and fairness in large language models: A survey". In: *arXiv preprint arXiv:2309.00770* (2023).

[3] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* (2023).

[4] Zishan Guo et al. "Evaluating large language models: A comprehensive survey". In: *arXiv preprint arXiv:2310.19736* (2023).

[5] Woosuk Kwon et al. "Efficient Memory Management for Large Language Model Serving with PagedAttention". In: *arXiv preprint arXiv:2309.06180* (2023).