## INTRODUCTION

Systematic LLM experimentation at scale presents a significant engineering challenge. The goal is often to compare numerous models across various configurations (different prompts, parameters, datasets) to find the optimal setup for a given task. This immediately leads to a combinatorial explosion.

For instance, the foundational work leading to Kernel 29 – the dxgpt-testing-main project –, aimed to assess how effectively different LLMs could propose differential diagnoses based purely on simulated plain-text non-standarized clinical cases. The goal was to test various LLMs and prompts to see how well they could produce a ranked list of 5 potential diagnoses, supported by reasoning that compared the patient’s symptoms to those of the suggested conditions(a differential diagnosis).

This required extensive testing across different setups to find the most effective combinations. Specifically:

* 18 distinct model configurations (Table 1) accessed via multiple APIs.
* 5 different prompting strategies (Table 2) tailored for diagnosis.
* 5 clinical case datasets (Table 3).

From a technical standpoint, this setup translates to managing **18 models × 5 prompts × 5 datasets = 450 unique experimental runs**, even before considering variations in parameters (like temperature) or incorporating evaluation steps (e.g., using LLM-as-judges).

**Table 1: Model Configurations**

| Alias | Source API / Service | Base Model |
| --- | --- | --- |
| c3opus | Anthropic API | claude-3-opus-20240229 |
| c35sonnet | Anthropic API | claude-3-5-sonnet-20240620 |
| c3sonnet | AWS Bedrock | anthropic.claude-3-sonnet-v1:0 |
| mistralmoe | AWS Bedrock | mistral.mixtral-8x7b-instruct-v0:1 |
| mistral7b | AWS Bedrock | mistral.mistral-7b-instruct-v0:2 |
| llama2\_7b | Azure ML Endpoint | Llama-2-7b-chat-dxgpt |
| llama3\_8b | Azure ML Endpoint | llama-3-8b-chat-dxgpt |
| llama3\_70b | Azure ML Endpoint | llama-3-70b-chat-dxgpt |
| cohere\_cplus | Azure ML Endpoint | Cohere-command-r-plus-dxgpt |
| geminipro | GCP Vertex AI | gemini-1.5-pro-preview-0409 |
| gpt4\_0613azure | Azure OpenAI Service | gpt-4 (0613 deployment) |
| gpt4turboazure | Azure OpenAI Service | gpt-4-turbo (1106 deployment) |
| gpt4turbo1106 | OpenAI API | gpt-4-1106-preview |
| gpt4turbo0409 | OpenAI API | gpt-4-turbo-2024-04-09 |
| gpt4o | OpenAI API | gpt-4o |
| o1\_mini | OpenAI API | o1-mini |
| o1\_preview | OpenAI API | o1-preview |
| mistralmoebig | Mistral API | open-mixtral-8x22b |

**Table 2: Prompt Strategies**

| Alias | Description |
| --- | --- |
| dxgpt\_standard | Standard diagnosis prompt format. |
| dxgpt\_rare | Diagnosis prompt focused on rare diseases. |
| dxgpt\_improved | Diagnosis prompt with thinking step & XML. |
| dxgpt\_json | Diagnosis prompt requesting JSON output. |
| dxgpt\_json\_risk | Diagnosis prompt requesting JSON with risk handling. |

**Table 3: Datasets**

| Name | Description |
| --- | --- |
| RAMEDIS\_200 | RAMEDIS clinical cases, truncated to 200 examples. |
| PUMCH\_ADM | PUMCH admission notes dataset. |
| URG\_200 | 6500 URG Emergency room reports, truncated to 200 examples. |
| URG\_1000 | same, truncated to 1000 examples |
| v2 | 200 Synthetically generated rare disease cases (v2). |

##ENGINEERING CHALLENGES

Attempting to manage this combinatorial complexity using the approach adopted in the dxgpt-testing-main project quickly exposed significant structural and operational flaws. The ad-hoc methods, while functional for small initial tests, proved unsustainable and inefficient at scale. The key problems encountered are summarized below (Table 4):

**Table 4: Key Flaws Identified in the dxgpt-testing-main Project**

| Problem | Description | Consequences (Concise Scientific Impact) |
| --- | --- | --- |
| **Cumbersome Workflow & Tracking** | Reliance on intermediate files for data transfer and manual code changes between runs. | Hinders accurate result reproducibility. |
| **Difficult Scalability** | Adding new models, prompts, or datasets required significant code duplication and manual modification. | Limits scope of investigation; prevents thorough comparison. |
| **Inflexible Configuration** | Settings hardcoded or scattered across code, comments, and env variables, lacking a central system. | Impairs systematic exploration; risks inconsistent parameters. |
| **Poor Code Structure** | Logic tightly coupled, duplicated across files, and mixed concerns within single scripts. | Increases risk of bugs affecting results; hinders verification. |
| **Lack of Standardization** | Absence of common architectural patterns, coding conventions, or workflow structure across scripts. | Reduces clarity of methods; hinders collaboration/verification. |

### DATA & KNOWLEDGE SCIENCE CONSIDERATIONS

Evaluating large language models for differential diagnosis generation from biomedical cases with gold-standard annotations requires structured data preparation. Several conceptual treatments (summarized in table 5) are necessary to ensure that the evaluation measures true clinical reasoning rather than artefacts introduced by data variability.

**Table 5: Standard treatments in data science**

| Concept | Definition (with example) | Avoids (English) |
| --- | --- | --- |
| Concept Normalization | Different clinical terms that refer to the same underlying condition are mapped to a single standardized concept using controlled vocabularies like HPO or ICD-10. For example, “heart attack,” and “AMI” would all be mapped to the same standarized concept “acute myocardial infarction”. This prevents the model from being penalized when it uses a correct synonym. | False mismatches due to synonyms or phrasing differences. |
| Standardization | Data formats, coding conventions, and naming practices are unified across the dataset to prevent technical inconsistencies. For example, diagnoses from different hospitals might use slightly different code versions or structures; standardization ensures they are made uniform before evaluation. This allows fair comparison between model outputs and reference diagnoses. | Errors caused by inconsistent representations. |
| Named Entity Disambiguation - NED | Clinical terms that have multiple meanings are clarified based on context to ensure the correct interpretation. For instance, “MI” might mean “myocardial infarction” or “mitral insufficiency” depending on the case; disambiguation explicitly selects the correct intended meaning. This ensures that correct predictions are not wrongly evaluated as errors. | Incorrect scoring due to ambiguous terms. |
| Ontology Hierarchical Expansion | The evaluation allows partial matches when the model predicts a broader but clinically related concept. For example, if the gold-standard diagnosis is “hepatitis E” and the model predicts “viral hepatitis,” the prediction can still be considered partially correct. This better reflects clinical reasoning, where general categories are sometimes acceptable. | Penalizing clinically reasonable broader predictions. |
| Class Imbalance | Rare diagnoses are proportionally represented or weighted during evaluation to prevent metrics from being dominated by common diseases. Without this, a model that only predicts frequent conditions could appear highly accurate despite poor performance on rare but critical diagnoses. For example, rare genetic disorders should be properly considered alongside common infections. | Rewarding models only for predicting common conditions. |

Unfortunately, these data treatments were not applied to the datasets listed in Table 3 during previous evaluations. This omission impacts the validity of the measured differential diagnosis accuracy for the LLMs. For instance, a preliminary analysis of the PUMCH\_ADM dataset indicates that this lack of treatment could introduce an error rate of approximately 10%. Specifically, out of 79 cases analyzed: \* 3 cases lacking a gold-standard diagnosis due to non-existent mappings were incorrectly labeled as incorrect diagnoses (false negatives). \* All 5 cases of POEMs syndrome were systematically failed by all models, likely due to incorrect entity disambiguation and synonym handling.

In the URG\_200 and URG\_1000 datasets, the error rate was not numerically measured. However, it is likely similar or even higher, as these datasets contain cases where the gold-standard diagnosis uses medical jargon unfamiliar to LLMs (e.g., ITU, IRAVA, IRV, CRU), leading to systematic failures.

The described LLM evaluation process also presents data handling considerations. The initial experiments utilized 5 datasets, totaling approximately 1,500 clinical cases. With 18 models generating 5 potential diseases per case, this phase produced **135,000 suggested diagnoses** (1,500 × 18 × 5) (Estimated). Assuming each diagnosis justification references approximately 10 symptoms or features, this corresponds to **1.35 million potential disease-symptom relationships** generated during experimentation (135,000 × 10). In clinical application, the dxGPT tool has been used for approximately 500,000 differential diagnoses. This usage equates to a potential dataset of up to **25 million disease-symptom associations** derived from LLM outputs (500,000 × 5 × 10). Currently, the data generated by LLMs during both experimentation and application are not systematically captured or structured. Existing infrastructure does not facilitate the collection and organization of these outputs for further analysis. Storing and structuring this generated data would enable several technical applications: \* **Synthetic Data Generation:** Extract disease-symptom relationships from LLM outputs to construct synthetic clinical cases. This data could serve as input for testing LLM responses under varied conditions (e.g., simulating missing or added symptoms). \* **Hybrid System Development:** Analyze symptom frequency and context within the LLM reasoning traces of differential diagnoses. This analysis could inform the development of knowledge graphs or weighting mechanisms to potentially re-rank LLM-generated diagnoses within a hybrid system architecture. \* **Model Distillation and Reasoning Analysis:** Utilize the corpus of LLM-generated diagnostic text for knowledge distillation into smaller models. Alternatively, specific diagnostic reasoning chains could be extracted for analysis, optimization or reinforcement learning inputs.

### KERNEL29

Addressing these data handling aspects, alongside the experimental orchestration needs, is a focus of the **Kernel29** project. It aims to provide a framework for LLM experimentation at scale that includes capabilities for capturing and structuring generated outputs. Within the medical domain, Kernel29 is designed to provide infrastructure for evaluate LLM capabilities and utilizing their outputs in a systematic manner. To manage experimental scale and structure data handling, Kernel29 incorporates the following **design principles**: \* **High-Quality Data:** Standardizing clinical cases into a common format and normalizing diagnoses using the widespread ICD10 taxonomy. \* **Robust Analysis:** Defining new metrics to correctly analyze and compare LLM outputs against ground truth or expert evaluations. \* **Modularity & Standardization:** Ensuring all modules interacting with LLMs share a consistent structure and naming convention, based on a clear separation of concerns (API connections, model configs, prompts, parsers, database interactions). \* **Data-Centric Approach:** Emphasizing structured input and, crucially, the structured capture of LLM outputs into databases to enable analysis and downstream use of the generated knowledge. \* **Abstraction:** Hiding the specific implementation details of different LLMs behind a standard interface, simplifying interactions regardless of the provider (OpenAI, Anthropic, Azure, etc.). \* **Scalability & Dynamic Loading:** Utilizing aliases and runtime configuration to load necessary components (prompts, models, parsers), allowing easy experimentation with numerous combinations without hardcoding. \* **Programmatic Prompting:** Implementing a dedicated module for dynamic and systematic prompt construction, facilitating sophisticated prompt engineering experiments.