Designing a Robust Forecasting System for Wikipedia Web Traffic: A Systems Engineering Approach

Johan Sebastián Beltrán Merchán
Code: 20222020019
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: jsbeltranm@udistrital.edu.co

Edison David Álvarez Varela
Code: 20222020043
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: edalvarezv@udistrital.edu.co

Yader Ibraldo Quiroga Torres
Code: 20222020034
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: yiquirogat@udistrital.edu.co

Julián David Celis Giraldo
Code: 20222020041
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas
Email: jdcelisg@udistrital.edu.co

***Abstract*—Wikipedia receives diarily visits around the worlds from severals devices types, from humans or agents, with people from different culture and language. The proposed architecture follows systems engineering principles to address the challenge of predicting daily pageviews for Wikipedia articles across multiple languages, access types, and temporal dimensions. The design emphasizes modularity, scalability, and maintainability while handling large-scale temporal data (2.5GB+) and complex page identifier structures. The system transforms raw historical pageview data into competition-ready predictions through a carefully orchestrated pipeline of data processing, feature engineering, machine learning modeling, and validation components.**

***Index Terms*—Systems Engineering, Time Series Forecasting, Hierarchical Ensemble, Design Patterns, SMAPE, Scalability.**

I. Introduction

The challenge of predicting Wikipedia web traffic views transcends a simple algorithmic exercise; it is a complex Systems Engineering problem. The competition involves forecasting daily views for approximately 145,000 articles, generating a dataset characterized by immense scale and inherent nonlinearity. This environment is highly susceptible to **chaos factors**, such as unpredictable viral events, sudden shifts in search engine algorithms, and complex, multi-scale seasonality [1]. Addressing this requires a robust architecture capable of processing large volumes of data while dynamically adapting to volatile patterns.

Previous research in time series forecasting has demonstrated the utility of various computational techniques. Traditional statistical models, such as **ARIMA** (AutoRegressive Integrated Moving Average) models, are well-established for stationary series [2]. More recent approaches utilize specialized libraries like **Prophet**, which is optimized for business time series data exhibiting strong seasonality and holiday effects [3]. For highly non-linear or long-term dependencies, deep learning techniques, particularly **Long Short-Term Memory (LSTM) Neural Networks**, have shown superior performance in capturing complex temporal dynamics [4]. Our initial analysis (Workshop 1) confirmed that no single technique is optimal for all 145,000 series due to the data's profound heterogeneity, leading us to favor an ensemble approach.

Our primary constraints identified were: (i) the **massive data scale** demanding high computational efficiency; (ii) a **fixed temporal window** (550 days) limiting the capture of long-term trends; and (iii) the **ambiguity of zero-values**, which are critical for an accurate SMAPE evaluation. The metric used for performance evaluation, the **Symmetric Mean Absolute Percentage Error (SMAPE)**, penalizes prediction errors across all magnitudes, demanding a balanced forecasting strategy.

$$SMAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

This paper details the design of our system architecture, which is built upon core Systems Engineering principles to manage complexity and ensure scalability, thereby setting the stage for a high-performing final submission.

II. Methods and Materials: System Design Decisions

Our design philosophy prioritizes a balance between maintainability and scalability, which led us to adopt a **Modular Monolith** architecture. This structure avoids the overhead of microservices while enforcing clear separation of concerns, crucial for a project of this complexity managed by a small student team.

A. Architectural Blueprint and Data Flow

The entire system operates as a sequential data pipeline, rigorously defined by the **Chain of Responsibility** pattern. In this architecture, each of the nine modules processes the data and passes it to the next, simplifying error tracing and enforcing data integrity standards. The workflow is not only linear but self-correcting, as depicted abstractly in Figure 1

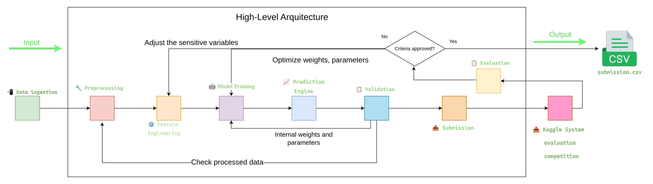

Fig. 1. Design system

The flow begins with the **Data Ingestion** module, which efficiently loads the large datasets using the `Pandas` library. This is immediately followed by the **Preprocessing** module, where critical tasks like handling missing values and resolving the zero-value ambiguity are performed through conservative imputation techniques. The processed data then feeds into the **Feature Engineering** module, where the raw time series is augmented with features such as lag values, temporal components (e.g., day of the week, holidays), and metadata features (language, access type). This rigorous feature generation is key to tackling the data's chaotic nature.

The central part of the system involves the **Model Training** and **Prediction Engine** modules, which implement our forecasting strategy. After prediction, the output is passed to the **Validation** module to ensure format consistency for the competition, and then to the **Submission** module. Finally, the **Evaluation** module calculates the SMAPE. This result drives the **Feedback and Iterative Cycle** module, which sends signals back to the *Feature Engineering* and *Model Training* modules for necessary adjustments, thereby closing the loop and making the system dynamically self-improving.

B. Hierarchical Ensemble and Strategy Pattern

A single prediction model cannot effectively forecast all 145,000 heterogeneous time series. Our most critical design decision was the implementation of a **Hierarchical Ensemble** driven by the **Strategy Pattern**. This pattern allows the system to change its prediction algorithm (strategy) based on the contextual characteristics of the specific article being forecasted.

At the base level (Level 1), we implement multiple prediction models—such as specialized classes for ARIMA, Prophet, and LSTM—each encapsulated as a concrete strategy. This approach is highly flexible and allows for the seamless integration of more advanced models (like the Random Forest or Neural Network model required for the final project) without modifying the high-level control logic.

At the meta-level (Level 2), a controller acts as a decision-maker. This **Meta-Model** analyzes features derived from the article's metadata (language, project type, historic volatility, traffic level) and determines which Level 1 strategy is most likely to yield the lowest SMAPE for that particular series. For instance, a highly stable, low-traffic article might be assigned the ARIMA strategy, whereas an article with complex, periodic seasonal spikes might be assigned the Prophet strategy. This design is the technical realization of the **Equifinality** principle, achieving the goal (low SMAPE) through multiple valid pathways.

C. Scalability and Implementation Assumptions

The most challenging non-functional requirement is **scalability**. To efficiently train and predict across 145,000 distinct time series, the *Model Training* and *Prediction Engine* modules are designed to use **parallel processing**. We will leverage the `Joblib` library in Python to distribute the training of individual time series models across multiple CPU cores, drastically reducing the overall execution time.

Our primary technical stack is **Python**, chosen for its rich ecosystem of data science libraries including `Pandas` (data manipulation), `scikit-learn` (utility functions), and specialized forecasting libraries. The system assumes a multi-core environment for efficient operation. A key limitation inherent to the project is the **fixed temporal window**, which means the system's performance is intrinsically limited by its inability to learn from events outside the 550-day window provided in the training data.

III. Results Projected and Discussion

Given that this paper precedes the final implementation, this section outlines our comprehensive testing philosophy and the expected mechanisms for validating the system's design robustness. Our approach covers all three major testing stages: **Unit Tests**, **Integration Tests**, and **Acceptance Testing**.

A. Unit and Integration Testing Philosophy

Unit Tests are essential for verifying the integrity of our core classes. We will implement extensive unit tests focusing on the deterministic components of the system. For instance, the **FeatureEngine** class (which handles lag calculation and temporal features) will be rigorously tested to ensure it prevents **data leakage** and consistently generates expected values from small sample series. Similarly, the individual **Strategy Pattern** implementations (ARIMAStrategy, ProphetStrategy, etc.) will be tested in isolation to confirm correct initialization and prediction output on synthetic datasets.

Integration Tests will address the correct functioning of the **Chain of Responsibility** pattern. These tests will simulate the entire data flow using a small subset of the training data (e.g., the 500 samples provided in the initial dataset). An integration test will verify that data successfully moves from *Data Ingestion* to *Prediction Engine*, ensuring all intermediate data transformations (preprocessing, feature engineering) are correctly applied and that no information is lost or corrupted between modules. This is crucial for verifying the system's modularity and decomposition.

B. Acceptance Testing and Granular Analysis

The final quality and user expectation are addressed via **Acceptance Testing**, which is intrinsically tied to the Kaggle submission and the **SMAPE** metric. The system's success will be determined by its ranking in the competition.

However, the *Evaluation Module* (#8) does not stop at a single SMAPE score. We have designed a **Stratified Post-Prediction Analysis** capability to perform granular decomposition of the SMAPE across multiple dimensions (see Table 1).

TABLE I
PLANNED SMAPE DECOMPOSITION CATEGORIES

Category	**Purpose**
Traffic Level	Differentiate low, medium, and high-traffic pages.
Language	Identify bias/poor performance in specific cultural data (e.g., *ja*, *zh*).
Access Type	Compare mobile vs. desktop prediction accuracy.
Volatility Score	Correlate SMAPE with the historical variance of the series.

This granular analysis is vital for the **Feedback** loop. For example, if the evaluation reveals that Japanese articles accessed via mobile devices consistently show a SMAPE greater than 50%, the discussion moves from general system failure to a specific problem within the *Feature Engineering* or *Meta-Model* logic for that subgroup. This approach allows us to pinpoint weaknesses, investigate specific hypotheses (e.g., uncaptured local events or unique seasonality), and direct iterative improvements effectively.

IV. Conclusions

Acknowledgments

The team extends its sincere gratitude to Professor Carlos Andrés Sierra Virgüez for his guidance and invaluable technical insights throughout the Systems Analysis and Design phase of this project.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.