Bachelor of Science in Economics, Management and Computer Science

LLM Forecasters: Harnessing Zero-Shot Learning for Advanced Time Series Forecasting with LLMs

Advisor:
Prof. Daniele Tonini

Bachelor of Science thesis by:
Vittorio Rossi

Academic Year 2023-2024

Contents

1 Introduction

The rapid development of large language models (LLMs), driven by advancements in transformer architectures [10] marks a significant leap forward in artificial intelligence. Particularly in natural language processing (NLP), these models have excelled, especially within generative applications. In contrast, the domain of time series forecasting did not experience similar groundbreaking developments until the recent M4 and M5 competitions, where traditional statistical methods began to be replaced by more sophisticated and accurate deep learning algorithms [16].

A novel approach to time series forecasting has emerged in light of these developments. Researchers are now exploring the potential of LLMs, known for their exceptional zero-shot learning abilities, to adapt swiftly to time series forecasting. This involves converting numerical data into a sequence of tokens, thus applying natural language processing techniques to numerical forecasting [9]. Given these promising results, this thesis seeks to address a gap in the field by establishing a comprehensive benchmark to evaluate the zero-shot forecasting abilities of LLMs on time series data.

Employing LLMs as pre-trained time series forecasters signifies a transformative shift in forecasting methods. By transforming numerical values into text, these models

utilize the general patterns they have learned from extensive textual data, which would typically necessitate intricate engineering and specialized architectural designs [3]. This pre-training endows the models with beneficial biases that improve their ability to recognize versatile, adaptable patterns, thereby enhancing their overall performance as core technologies progress. Moreover, because LLMs are inherently trained on language data, they acquire unique capabilities such as question answering. This not only expands their range of applications but also promotes a more integrated approach to machine learning, where a single model can perform effectively across different tasks and types of data. Crucially, the zero-shot learning capabilities of these models provide substantial practical benefits—they can achieve strong performance [9] without extensive computing resources, domain-specific expertise, or vast amounts of task-specific training data [7]. Hence, applying natural language processing techniques to time series forecasting could herald a new era of more effective, accessible, and powerful predictive analytics.

1.1 Thesis structure

This thesis is organized to ensure a comprehensive exploration and presentation of evaluating large language models (LLMs) for zero-shot forecasting capabilities in time series data. Following the introduction, the Literature Review examines existing research related to LLMs, zero-shot learning, and time series forecasting, identifying gaps that this thesis aims to address.

The Methodology section describes the selection of models and datasets, as well as the experimental design and metrics used for evaluation. The Results section presents the findings from the experiments, followed by a Discussion that interprets these results, discusses their implications, and acknowledges any limitations.

The thesis concludes with a section that summarizes the key findings and their potential impact on the field of AI and forecasting. References to all cited works are included in the References section.

2 Background

Relevance of time series forecasting Time series forecasting is a critical analytical tool used across various industries to anticipate future events and trends based on past data. This statistical technique is pivotal in numerous fields, including finance, where it predicts stock prices, meteorology, weather forecasting, supply chain management, inventory control, and demand forecasting. Despite its wide applications, the methodological complexity of time series forecasting poses significant challenges. Traditional models often require intricate tuning and robust historical data to capture seasonal patterns, trends, and irregularities effectively. The continual evolution of this domain reflects an ongoing quest to refine predictive accuracy and computational efficiency in handling time series data.

The COVID-19 pandemic dramatically illustrated the critical need for robust and adaptable time series forecasting models[7]. As the virus spread rapidly across the globe, it became apparent that traditional forecasting tools were often inadequate in handling such unprecedented conditions. These models, which were typically designed for stability and gradual changes, failed to accommodate the sudden shifts in data patterns caused by widespread lockdowns and economic disruptions. A survey by Makridakis et al. (2021)[4] specifically highlighted these shortcomings, emphasizing that the pandemic

revealed significant gaps in our forecasting capabilities, particularly in terms of speed and adaptability to new data. This has led to a surge in research focusing on developing methods that can dynamically integrate varying data streams and adapt more rapidly to changes, offering hope for more resilient forecasting tools in the face of future global challenges.

Deep learning for TS Despite the profound impact of deep learning in fields such as natural language processing (NLP) and computer vision, its integration into time series forecasting has been comparatively slow. One fundamental reason for this delay is the unique challenges that time series data presents. Unlike images or text, time series data often involves complex dependencies such as seasonality, cyclic behaviors, and abrupt shifts, which are not inherently captured by traditional deep learning architectures without extensive customization. Moreover, time series forecasting often requires not only predicting future values but also understanding uncertainty and variability, aspects that are less emphasized in other domains. According to a review by Benidis et al. (2023) [7], the specificity of these requirements has necessitated the development of specialized models like LSTM and GRU, which can handle such temporal dynamics but require significant tuning and domain expertise to deploy effectively. As highlighted in [16], while deep learning has been state of the art for domains like NLP and computer vision, its applicability to time series forecasting problems has been slower. For example, in the Makridakis (M) Competitions, a very famous time series forecasting competition, deep learning methods have entered the top performer leaderboard only recently with the M5 competition[4].

The landscape of time series forecasting has begun to shift with the more recent entries of deep learning models into the top tiers of forecasting competitions. Notably, during the M5 competition, as highlighted by Makridakis et al. (2022)[4], deep learning models demonstrated their potential by outperforming many traditional methods, especially in large-scale and complex datasets involving hierarchical and grouped time series. This success is attributed to their ability to model non-linear relationships and interact with a vast range of input variables simultaneously, a significant advantage over more traditional statistical methods. The M5 competition served as a pivotal moment, showcasing that with the right architecture and data handling, deep learning could indeed be adapted to meet the nuanced demands of time series data, thus paving the way for broader acceptance and application in this field.

LLM for time series The exploration of Large Language Models (LLMs) in time series forecasting represents an innovative shift from traditional and even recent deep learning approaches. As detailed in "A Survey of Deep Learning and Foundation Models for Time Series Forecasting," LLMs have begun to show promising results in this domain by leveraging their substantial pre-trained knowledge bases, originally developed for tasks in natural language processing [16]. These models, trained on vast amounts of diverse textual data, have inherent capabilities to recognize patterns and dependencies that can be abstractly similar across different types of data, including numerical time series. The primary advantage of using LLMs in forecasting is their ability to generalize from one domain to another without the need for extensive retraining. This characteristic is particularly valuable for time series forecasting, where acquiring large, domain-specific datasets can be challenging and costly.

Zero-shot learning Zero-shot learning, a concept that originates from the field of computer vision, refers to the ability of a model to correctly perform tasks it has not explicitly been trained to do. This approach has been adapted for use with LLMs in time series forecasting, where the models apply knowledge acquired during pre-training to generate forecasts without prior exposure to specific time series data. The application of zero-shot learning in this context is primarily facilitated through techniques such as prompt engineering, where numerical data is tokenized into a format understandable by LLMs, effectively transforming a forecasting problem into a language modeling task. This method allows LLMs to utilize their extensive pre-trained models to interpret and predict time series data, as they would with text, offering a novel and potentially more flexible approach to forecasting. Studies such as those conducted by Gruver et al. (2023)[9] have demonstrated that LLMs can indeed perform zero-shot forecasting effectively, underscoring their potential to revolutionize how forecasts are generated, particularly in scenarios where traditional models falter due to lack of data or sudden shifts in patterns.

Current reseraches Innovative applications of Large Language Models (LLMs) as zero-shot time series forecasters have been documented in various recent studies. A notable example is the "PromptCast" method, described in the work by Xue and colleagues (2023)[12], which leverages the natural language processing capabilities of LLMs by converting time series data into a sequence of prompts. This approach allows the model to interpret and forecast time series as if it were processing natural language, thus utilizing the robust predictive power of LLMs trained on extensive text data. Similarly, the study titled "Large Language Models Are Zero-Shot Time Series Forecasters"[9] explores the direct application of zero-shot learning principles to time series forecasting. Their research demonstrates that LLMs, without any specific tuning for time series data, can outperform traditional models in predicting economic and weather-related time series, thus highlighting the versatility and potential of LLMs in generalizing across vastly different data types.

Business use cases The ability of LLMs to perform zero-shot time series forecasting opens up numerous practical applications, especially in business contexts where rapid adaptability and minimal model training are crucial. In industries like retail and finance, where market conditions can change unpredictably, LLMs offer a significant advantage by quickly providing forecasts based on limited or no historical data specific to the new conditions. For startups and smaller companies that may not have the resources to collect extensive data or develop specialized models, zero-shot LLMs can level the playing field, allowing them to make data-driven decisions with the same confidence as larger entities. Additionally, in the realm of supply chain management, zero-shot forecasting can facilitate more agile responses to supply disruptions and demand fluctuations, which have become increasingly common in global trade environments. These applications not only demonstrate the versatility of LLMs but also suggest a shift towards more dynamic, cost-effective forecasting solutions in business operations.

Gap in current reserach Despite the promising developments in the use of Large Language Models (LLMs) for time series forecasting, a significant gap remains in the field: the lack of a unified benchmark for evaluating these models across varied domains and datasets. Current research often focuses on specific applications or datasets, which

limits the understanding of how these models perform under different conditions and in different sectors. This fragmentation hinders the ability to compare the effectiveness of LLMs directly with traditional forecasting models or even between different LLM approaches themselves. Addressing this gap, my research proposes the development of a comprehensive benchmark that includes a wide range of datasets—from economic indicators to environmental data—allowing for a systematic and comparative analysis of the zero-shot forecasting capabilities of LLMs. This benchmark will not only facilitate a deeper understanding of where LLMs excel or fall short but also aid in refining these models for more generalized use.

Research Necessity and Contribution The necessity of this research lies in its potential to transform the landscape of time series forecasting by leveraging the advanced capabilities of LLMs. By developing a standardized benchmark, this work aims to provide empirical evidence on the effectiveness of zero-shot learning in forecasting, offering insights that could lead to more robust, adaptable, and efficient forecasting models. Additionally, the findings from this study are expected to contribute to both academic knowledge and practical applications, bridging the gap between theoretical advancements and real-world needs. The ultimate goal is to demonstrate that LLMs can be a viable and superior alternative to traditional models, particularly in scenarios where quick adaptability and minimal data requirements are crucial. This research will not only try to validate the applicability of LLMs across various domains but also pave the way for future innovations in predictive analytics, enhancing decision-making processes in numerous fields.

3 Methodology

3.1 Data

In the first part of this study I did a very comprehensive survey of the time series data used in current research world. Among all the viable datasets, I decided to choose 11 different ones comprising around 500k time steps coming from 5 different sources. With granularity ranging from 15 minutes to quarterly, As described in the following paragraphs, each dataset comes from a different domain so to grant consistency of results across domains and different time series forecasting instances. In Table 1 a summary of the characteristics of the different datasets and their sources. Now let's dive deeper into the sources and domains this data spans.

PromptCast "PromptCast: A New Prompt-based Learning Paradigm for Time Series Forecasting"[12] introduces a novel forecasting approach that leverages the capabilities of natural language processing to handle time series data. The paper contrasts traditional numerical time series forecasting with this innovative prompt-based method. Unlike conventional models that directly interpret numerical data, PromptCast transforms these data into a sequence of textual prompts, allowing the use of powerful language models for forecasting tasks. This shift not only seeks to harness the inherent strengths of language models in learning from vast textual data but also proposes a potentially more flexible and robust framework for predicting future data points in time series.

The original paper utilizes the PISA dataset, specifically designed to benchmark the PromptCast method. The dataset encompasses three real-world forecasting scenarios with a total of 311,932 data instances across three distinct subsets:

Source	Dataset	Granularity	Series	Timesteps	Seasonality
PromptCast	CT	Daily	1	91,850	Yearly
	SG	Daily		96,552	Weekly
ETDataset	ETTh1	1h	1	17,400	Daily, Weekly[1]
	ETTh2			17,400	
	ETTm1	15 min		69,700	
	ETTm2			69,700	
Web Traffic	GWT	Daily	145,000	804	Weekly[Fig 1]
M4 competition	m4-M	Monthly	48,000	2,794	Yearly
	m4-Q	Quarterly	24,000	866	
	m4-W	Weekly	359	2,597	
M5 competition	m5	Daily	42,840	1,913	Weekly, Monthly [4]

Table 1: Detailed summary of the benchmark dataset for time series forecasting, including seasonality information

1. CT (City Temperature Forecasting): Involves predicting future temperature values for specific urban areas.
2. SG (Human Mobility Visitor Flow Forecasting): Aims to predict visitor flow dynamics, crucial for urban planning and management.
3. ECL (Electricity Consumption Load). This one was discarded for a matter of simplicity and because of the similarities in domain of the ETDataset

The selection of these two datasets is strategic, enabling a direct comparison between the results achieved with our methodology and those from existing research. By leveraging these established datasets, we can effectively benchmark our model against other models previously developed and tested under similar conditions. This approach is particularly advantageous given the constraints of our computational resources, as it allows for meaningful comparisons without the need for extensive and costly hardware setups typically required for testing more complex and resource-intensive architectures.

ETDataset The Electricity Transformer Dataset (ETDataset) [3] is a comprehensive dataset designed for the analysis and forecasting of long sequence time-series problems.

The ETDataset has become a popular choice in the time series forecasting community due to its real-world relevance and the complexity it presents, making it ideal for benchmarking advanced forecasting models. Its inclusion of long-term and short-term periodical patterns, offers a unique challenge for forecasting methodologies. Oil temperature is particularly significant as it reflects the operational condition of transformers and can indicate the need for preventive measures against potential failures, thus ensuring efficient energy management and equipment maintenance.

This dataset was specifically chosen because of it's large usage as benchmark for long-sequence time series forecasting [8]

Web Traffic I selected this dataset, which consists of around 145,000 time series, each representing the daily page views of a unique Wikipedia article from July 1, 2015, to September 1, 2017. This comprehensive dataset offers a rich source of information to

analyze and predict web traffic patterns, allowing for the development of robust forecasting models. By leveraging this dataset, we can understand how the LLMs behave in time series with weak weekly seasonality, where the prediction label is an integer and the ample number of data points allows for a long lookback window.

M4 competition The M4 Competition datasets are derived from the fourth iteration of the M Competitions [2], which have played a pivotal role in advancing the theory and practice of forecasting. The datasets from the M4 Competition include time series data categorized into various frequencies, specifically focusing on quarterly, monthly, and yearly datasets. These datasets were selected to cover a wide range of domains and provide a comprehensive platform for testing and improving forecasting models.

- Quarterly Dataset: Contains time series data collected at three-month intervals, useful for capturing seasonal trends and effects in various industries. Participants were asked to produce 8 forecasts beyond the available data, supporting budgeting and planning purposes for periods ranging from a few months to two years ahead.
- Monthly Dataset: Includes time series data recorded monthly, offering a more frequent glimpse into trends and patterns, ideal for detailed short-term forecasting and operational planning. Participants were asked to produce 18 forecasts beyond the available data, aiding in short to medium-term planning.
- Week Dataset: Comprises data points collected on a weekly basis, perfect for capturing long-term trends and strategic decision-making. Participants were asked to produce 13 forecasts beyond the available data, supporting strategic decisions for periods of 3 months and 1 week

The reason why this dataset is very useful to us is that it allows us to understand how the LLM (Large Language Model) acts in a setting where seasonality is not so straightforward, yet the asked prediction is very far in the future. This challenges the models to adapt and perform well across different temporal dynamics and forecasting horizons.

Since the dataset contained series from various domains, this approach enables the evaluation of forecasting models' effectiveness across a diverse array of domains such as micro, macro, finance, and demographics, ensuring broad applicability and relevance to various forecasting challenges.

M5 competition dataset The M5 Competition dataset follows the legacy of its predecessors by advancing the field of forecasting through competitive benchmarking. This dataset, specifically designed for the M5 Competition, includes detailed time series data focusing on retail sales forecasting. It is one of the most recent additions to the series of M Competitions, providing new challenges and opportunities for the development and testing of forecasting models. The M5 dataset is comprised of retail sales data from Walmart, covering product sales across various departments and stores. The dataset includes several features that influence sales patterns, such as promotional information, price changes, day of the week, and special events like holidays:

- Sales Data: Daily sales data for various products, providing a granular look at consumer purchasing behavior over time.

- Calendar Data: Information on promotional and event days which may affect sales, such as holidays and national events.
- Price Data: Weekly price data of products, allowing the exploration of price elasticity effects on sales volumes.

Choosing the M5 dataset for forecasting research is particularly beneficial because it reflects current challenges in the retail industry, such as demand forecasting and inventory management. The dataset's detailed attributes will enable us to test my pipelines on a real and relevant business use case, allowing us to finally evaluate if zero-shot LLMs can be proposed as a viable replacement for more complex statistical models.

3.2 Preprocessing

In order to assess the zero-shot capabilities of the model, and given that the primary objective of this benchmark is to evaluate the effectiveness of using large language models (LLMs) as forecasters with minimal preprocessing steps missing values will not be filled so that it can be tested how the model reacts to such inputs. Nonetheless there cannot be missing true label, so the observation with such values will be skipped. No normalization or standardization will be performed.

The majority of preprocessing occurs during the transformation of the time series data into a string format. To facilitate this, I employed a template system that inputs the time series data and any necessary metadata, incorporating it into a string constructed from predefined templates. This approach is highly flexible, allowing for easy modification of the templates and enabling experimentation with different prompts. The data was finally batched and passed to the model that performed tokenization with padding and truncation.

3.3 Post-processing

Working with Large Language Models (LLMs) means that the output generated by the model is typically a string. However, to evaluate these models effectively, the predicted values must be in a numerical format, specifically as floats. In order to ensure the accuracy and consistency of the data, I implemented a comprehensive post-processing strategy focused on cleaning and standardizing the output. My approach involved the extraction of numerical values from text strings, which was achieved through the use of regular expressions designed to identify sequences of digits, including those with optional decimal points. This method enabled the handling of both integer and floating-point numbers effectively. Once extracted, these numbers were stored in a list. To maintain consistency in data size, a mechanism to compare the length of the extracted numbers with a predefined target size was introduced. In cases where the number of extracted values fell short of this target, the value 'NaN' (Not a Number) is appended to to the list until the desired length is met. This ensures that our dataset remained uniform, facilitating subsequent analysis and processing steps. The implementation of this post-processing step was critical in mitigating issues related to variable data lengths and missing values, thereby enhancing the robustness of our analytical models.

3.4 Evaluation

To choose evaluation metrics a brief survey of the current state of reserach showed that the most commonly used metrics are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). Being extensively used across the time series forecasting domain I decided to use this three metrics so that we can compare results with other important papers in the field.

MAE This metric measures the average magnitude of the errors in a set of predictions, without considering their direction. It is given by the formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where y_i are the actual observed values and $\hat{y}_i$ are the predicted values.

RMSE The RMSE is a standard way to measure the error of a model in predicting quantitative data. It is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

MAPE The MAPE is also used as it expresses accuracy as a percentage, which makes it easily interpretable. This metric is especially useful when comparing forecasting errors on different data scales. It is defined as:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

However, MAPE cannot be computed when the true value (y_i) is zero, as this would result in division by zero. To address this, we report a modified MAPE that is masked to be computed only on non-zero true values. This adjustment is particularly important for datasets like Google Wiki Traffic and the M5 competition, where zero values are prevalent.

3.5 Baselines

Given that the aim of this benchmark is to explore whether the zero-shot capabilities of LLMs extend to the forecasting domain, I chose to compare the models in my study against three straightforward baseline models. These baseline models serve as a reference point, allowing us to better interpret the reported metrics.

Seasonal Naive model The Seasonal Naive model extends the traditional Naive model by incorporating seasonality into its predictions. Instead of using the most recent observation, it uses the observation from the same season in the previous cycle. If seasonality is specified as a list, i.e. the dataset has more than one seasonality, the model computes the mean of the last values based on the seasonal periods provided. This approach is particularly useful in time series data with strong seasonal patterns, where past values from the same season can be good indicators of future values. Despite its simplicity, the Seasonal Naive model often performs surprisingly well and serves as a critical benchmark for evaluating the predictive power of more complex models.

Mean model This is a simple yet effective baseline where the prediction for the next time step is the average of the previous time steps. For multi-step forecasting, the Mean model predicts the mean of the window size repeated for the target size. This model is computationally inexpensive and provides a straightforward benchmark to assess the performance of more complex models. By comparing the performance of our LLM-based models to the Mean model, we can gauge the improvements gained through advanced techniques. The simplicity of the Mean model ensures that any significant improvement in performance can be attributed to the complexity and capabilities of the more sophisticated models.

$$\hat{y}_{t+1} = \frac{1}{n} \sum_{i=0}^{n-1} y_{t-i} \tag{1}$$

Exponential Smoothing model Exponential Smoothing [1] (Exp. Smt) is a more sophisticated baseline that assigns exponentially decreasing weights to past observations. This method has been adapted to include a seasonality parameter, which allows it to handle data with both trends and seasonal patterns effectively. By smoothing out fluctuations and emphasizing recent observations, Exponential Smoothing can produce reliable forecasts. The seasonality parameter adjusts the model to account for repeating patterns at specified intervals, improving its forecasting accuracy. It provides a robust point of comparison for our study, helping us understand how well LLMs perform relative to a method known for its accuracy in various forecasting applications.

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t \tag{2}$$

where (α) is the smoothing factor learned by fitting the model using the MLE method. $(0 < \alpha < 1)$, (y_t) is the actual value at time t, and $(\hat{y}_t)$ is the forecasted value at time t. The reported results are derived from a model learned only on the values provided in the window, in the same way it was done with the LLMs.

3.6 Models

For this study, I selected models based on several key characteristics. Firstly, the models needed to be small enough to run on a Google Colab notebook, which led to the decision to use models with fewer than 8 billion parameters. Additionally, the models needed to be open source to facilitate research and ensure they were easily accessible. To find suitable models, I used the renowned platform Hugging Face and selected top models from the OpenLLM leaderboard [6].

Phi 3 The Phi-3 Mini-4K-Instruct [13] model, developed by Microsoft, is a cutting-edge language model designed for efficient and high-quality text generation tasks. With 3.8 billion parameters, this lightweight decoder-only Transformer model is part of the Phi-3 family, which includes various models optimized for different applications. The Phi-3 Mini-4K-Instruct model was trained using a combination of synthetic data and filtered publicly available datasets, ensuring high-quality, reasoning-dense inputs. This model excels in tasks requiring common sense, language understanding, math, and logical reasoning. It supports context lengths of up to 4,000 tokens and showcases robust performance across multiple benchmarks when compared to other models with fewer than 13 billion parameters.

LLaMA 3 8B The LLaMA 3 8B model, developed by Meta [17], is an 8 billion parameter language model known for its strong performance in a wide range of NLP tasks. This model is built on the foundation of the LLaMA architecture, which emphasizes efficiency and scalability. The version I choose to evaluate is the instruct version of the model: LLaMA 3 8B - Instruct model [17]. This version, like the phi3 model described before, is a fine-tuned version specifically optimized for instruction-following tasks. Instruct models [5] incorporates additional supervised fine-tuning to improve its performance in scenarios where understanding and executing instructions is crucial. The fine-tuning process involves using high-quality, diverse instructional datasets to ensure that the model can handle complex directives with high accuracy and reliability.

Table 2: Model Names and Corresponding Hugging Face IDs

Model Name in Paper	**Hugging Face ID**
Phi 3	`microsoft/Phi-3-mini-4k-instruct`
LLaMA Inst	`meta-llama/Meta-Llama-3-8B-Instruct`

Chat vs non-chat I chose the instruct model to determine whether it was more effective to use the model for simple next-token prediction or to prompt it specifically for time series forecasting.

Therefore, a system to handle various prompts was necessary to make the codebase flexible and facilitate experimentation with different prompts. To achieve this, I developed a custom prompt loader that seamlessly loads Jinja2 prompts and passes them to the preprocessor function. This allows the transformation of simple time series in numerical form into more complex, text-like prompts.

For this analysis, I chose a straightforward prompt designed to persuade the model to function as a time series forecaster. This approach ensures the model returns a series of numbers rather than a message-like response.

.

> Given the historical time series data, provide the forecasted values for the next {{ target_size }} period. Return just predicted numerical values separated by spaces.
> {% for row in data.target %}{{ row }} {% endfor %}

.

where {{ target_size }} will be substituted with the target size for the specific experiment and {% for row in data.target %}{{ row }} {% endfor %} will iterate over the target data to produce the forecasted values.

4 Experiment setup and results

4.1 Hyperparameter Selection for Model Training

To evaluate the model specific hyperparameters were chosen for each dataset to effectively test different kind of settings and to facilitate comparison with prompt-based forecasting.

These hyperparameters were selected to align with the unique characteristics and challenges of each dataset:

- **One-Step-Ahead Prediction Testing:**
  - **CT and SG Datasets:**
    - `target_size: 1`, `window_size: 15`, `batch_size: 64`
    - These parameters were chosen to focus on short-term, one-step-ahead predictions. The small window size of 15 allows the model to capture recent trends.
- **Multi-Step Long-Term Forecasting and Long Context Windows:**
  - **ETTh1, ETTh2, ETTm1, ETTm2 Datasets:**
    - `target_size: 48`, `window_size: 168`, `batch_size: 8`, `stride: 8`
    - These datasets are well-known for multi-step long-term forecasting tasks, therefore I am going to use a large window size of 168 timesteps that captures extensive historical context and allows the model to learn the weekly seasonality. The target size of 48 reflects the long-term forecasting goal. The chosen batch size and stride ensure that the model can handle the extensive data efficiently. Those two values were chosen based on an extensive survey of many paper. These usually use 512 window size to predict 96, 192, 336, 720 [15] [8] [14] Quote ts mixer and other paper using that. These usual values were changed to the ones choosen for a matter of computational complexity of the models being evaluated.
- **Granularity and Weak Seasonality in Various Domains:**
  - **M4-quarter, M4-month, M4-year Datasets:**
    - **M4-quarter:** `target_size: 8`, `window_size: 20`,
    - **M4-month:** `target_size: 18`, `window_size: 60`
    - **M4-week:** `target_size: 12`, `window_size: 54`,
    - All the dataset above were evaluated with a `batch_size of 64` and by limiting the number of series to the first 100
    - These datasets from the M4 competition were chosen to evaluate how LLMs perform in settings with different granularities and weak seasonality across various domains. The specific target sizes and window sizes were selected to match the forecasting horizons used in the M4 competition [2], allowing for a meaningful comparison of model performance.
- **Business Usability**
  - **M5 Dataset:**
    - `target_size: 28`, `window_size: 28`, `batch_size: 16`
    - The M5 dataset, characterized by many zero values, is crucial for understanding the business applicability of models in predicting sales. The chosen parameters allow the model to learn from the sparse data effectively and provide forecasts that are highly relevant for business planning.

– **GWT Dataset:**

* `target_size: 1`, `window_size: [28, 56]`, `batch_size: 16`,
* This dataset helps in evaluating model performance in scenarios with weekly seasonality and strong trends. The window size were selected to allow the model to capture these two characteristics of the series.

These carefully chosen hyperparameters ensure that each model is well-suited to the specific characteristics and forecasting challenges of its respective dataset, providing a robust foundation for both one-step-ahead prediction testing and comprehensive comparative analysis.

4.2 Computational resources

As expected, working with large language models (LLMs) has presented significant challenges. Running inference with the selected models required expensive computational resources. To face this issue, the evaluation pipeline was run on a machine with the following specifications: a g2-standard-8 machine type, equipped with an NVIDIA L4 GPU with 25 GB of ram and 100 GB standard disk(to store the large models weights).

4.3 Results

As described in Section 4.1, I divided the benchmark into four distinct macro-areas. These areas are based on varying levels of difficulty a forecasting model might encounter and the specific applications they pertain to. In this section, I report and analyze the results for the four different benchmarks.

Table 3: Results of One-Step-Ahead Prediction

Dataset	Metric	Mean	sNaive	Exp. Smt	Phi3	C-Phi3	LLama
CT	RMSE	4.916	3.715	3.843	4.604	4.054	5.76
	MAE	3.789	2.877	2.877	3.017	3.023	3.402
	mMAPE	5.185	3.977	3.977	4.157	4.181	4.697
SG	RMSE	9.528	12.26	10.97	11.33	11.852	13.06
	MAE	6.746	8.658	7.470	7.880	8.246	8.911
	mMAPE	25.41	31.83	27.79	28.49	30.05	31.625

One-step-ahead prediction The first benchmark I conducted was the evaluation of the models on the CT and SG datasets. This evaluation was performed to compare our results with the PromptCast paper [12], which was one of the first papers to introduce the concept of leveraging text models to handle tabular data using prompt engineering and the zero-shot capabilities of those models.

As shown in Table 3, our models consistently underperformed across all metrics compared to at least two baseline models. This suggests that using LLMs for zero-shot forecasting is not feasible.

Nonetheless, our models consistently outperformed the benchmarks presented in Table X(Figure 2 in Appendix) of the PromptCast paper, indicating that LLMs are still more capable than the models used in that study.

Furthermore, we found that Llama performed significantly worse than Phi3. Indeed, I observed that Llama often generated missing values by attempting to answer with text instead of numbers, behavior that was not observed in running Phi3 without chat. Due to this issue and the excessive computational resources it consumed, we decided to exclude Llama from further evaluations.

Table 4: Results of Multi-step long-contex prediction

Dataset	Metric	Mean	sNaive	Exp. Smt	Phi3	Phi3-Chat
ETTh1	RMSE	4.209	5.235	3.879	3.956	5.506
	MAE	3.403	4.406	2.874	2.962	4.368
	mMAPE	9.690	12.756	8.611	8.629	12.613
ETTh2	RMSE	5.902	6.513	5.532	9.262	7.642
	MAE	4.651	5.422	4.361	4.775	6.306
	mMAPE	10.583	12.542	10.142	10.910	14.573
ETTm1	RMSE	3.043	3.536	3.031	9.499	4.072
	MAE	2.023	2.567	1.989	3.576	3.173
	mMAPE	5.992	7.686	5.982	9.696	9.493
ETTm2	RMSE	4.032	4.874	4.084	4.258	5.033
	MAE	3.286	3.901	3.026	3.177	3.942
	mMAPE	11.477	13.519	11.039	11.157	13.80

Multi-step forecasting with long context Contrary to my expectations, the Phi3 model's performance was not as poor as anticipated. In many cases, it outperformed the mean and sNaive benchmarks and was sometimes very close to the Exponential Smoothing model. However, it is evident that on the ETTh2 and ETTm1 datasets, Phi3 performed slightly worse in terms of mMAPE and RMSE, suggesting the presence of larger errors, as these metrics are more sensitive to such deviations. In contrast, the MAE values for Phi3 were generally in line with the baseline models, indicating consistent performance across different datasets. Additionally, as observed in the previous benchmark, Phi3-Chat exhibited variability in its performance, consistently showing worse metrics across all datasets compared to the non-chat version.

Evaluation on M4 competition Again, evaluating Phi3 on the M4 dataset for this benchmark showed performances similar to those observed in the ETT dataset. Despite the shorter context and target lengths, and the differing granularities (minutes and hours in ETT versus week, quarter, and month in M4), Phi3's performance remained comparable to the Exponential Smoothing model, while largely outperforming the mean and naive approaches.

Similar to the previous benchmark, Phi3-Chat demonstrated variability in its performance during the M4 competition evaluation. The Phi3-Chat model consistently showed worse metrics across all datasets compared to the non-chat version, reinforcing the pattern observed earlier. Specifically, Phi3-Chat's RMSE, MAE, and mMAPE values were significantly higher, indicating its tendency to produce less accurate forecasts.

Table 5: Results of M4 Competition

Dataset	Metric	Mean	sNaive	Exp. Smt	Phi3	Phi3-Chat
M4-quarter	RMSE	519.849	592.505	309.461	321.653	498.95
	MAE	317.709	303.047	184.895	195.695	333.974
	mMAPE	10.301	9.536	6.245	6.598	11.526
M4-month	RMSE	1461.884	1397.994	1071.647	1111.910	1596.62
	MAE	1071.882	922.503	715.554	717.289	1162.39
	mMAPE	18.535	16.475	13.465	13.389	21.635
M4-week	RMSE	151.682	225.648	64.760	64.763	125.97
	MAE	123.751	192.204	41.683	39.128	92.64
	mMAPE	4.225	6.548	1.453	1.378	3.19

Table 6: Results of Real Life Applications

Dataset		Metric	Mean	sNaive	Exp. Smt	Phi3	Phi3-Chat
M5		RMSE	0.537	0.632	0.548	0.617	1.03
		MAE	0.299	0.308	0.301	0.230	0.426
		mMAPE	62.731	70.676	63.210	97.588	74.69
GWT	28	RMSE	31.140	42.015	32.692	34.099	33.31
		MAE	13.234	16.940	13.284	13.255	13.787
		mMAPE	76.052	91.767	76.394	55.054	78.019
	56	RMSE	29.759	42.059	30.439	33.357	51.08
		MAE	13.048	16.887	12.805	12.795	23.309
		mMAPE	75.850	89.051	71.941	51.835	133.25

Real life application of LLMs Finally, the ultimate test was evaluating the model on practical business applications, such as web traffic forecasting and sales forecasting. The results were mixed: our model outperformed all others on the MAE metric but performed worse on every other metric. Additionally, on the GWT dataset, increasing the context size improved the model's performance. Overall, the model appears to be a relatively good alternative to simpler models, suggesting that for managers and business users unable to perform statistical analysis, using an LLM for quick predictions could be a viable solution.

5 Findings

Even though the performance of the models compared to the very simple benchmark models was poor in many scenarios, this study presents several valuable insights.

Prediction quality Most important aspect of the study is definitely the prediction quality. I underscored how in Table 6, the GWT dataset showed improved performance with an increased context size. This insight confirms our observations when comparing the performance on the PromptCast dataset against the M4 and ETT benchmarks. Indeed, the Phi3 model performed better on multi-step predictions with larger step sizes rather than one-step predictions with very small window sizes. This is a significant finding, as it provides strong evidence that the model does not randomly guess the next

prediction based on token frequency alone but instead identifies some form of relationship between input and output. Furthermore, the zero-shot capability of LLMs still presents advantages for certain high-level analytics. As discussed in Section 4.3, Phi3 showed mixed results in practical business applications like web traffic forecasting and sales forecasting. While it outperformed all others on the MAE metric, it performed worse on every other metric. Notably, increasing the context size on the GWT dataset improved the model's performance. This suggests that LLMs can be a relatively good alternative to simpler models. For managers and business users unable to perform statistical analysis, using an LLM for quick predictions could be a viable solution. In addition to that, we found that the LLama model, despite coming from the same company, using the same tokenizer, and being larger than Phi3, performed clearly worse. This indicates that larger models are not necessarily better at learning these kinds of patterns. While this difference may be attributed to variations in training data, it underscores that larger models do not automatically result in better predictions. Finally, there was a significant discrepancy in prediction quality between using the model as a simple sequence-to-sequence model and utilizing its chat capabilities.

Ease of use One reason to choose a zero-shot model over training a specific model for time series forecasting is ease of use. However, performing predictions with these models proved to be anything but easy. Consistently obtaining numerical outputs was particularly challenging (with sequence-to-sequence models performing exponentially better than chat models, see Table 7). This difficulty often negates the advantages of zero-shot capabilities, as it necessitates the creation of a reliable and robust output handling algorithm. Furthermore, as highlighted on many occasions, and as is well known, LLMs are extremely heavy and computationally expensive. They require significant investments in setup and dedicated machines to run effectively.

Table 7: Comparison of Missing Value Fraction between Phi3 and Phi3-Chat (**in percentages**)

Dataset	**Phi3 NaN Fraction (%)**	**Phi3-Chat NaN Fraction (%)**
CT (PromptCast)	0.0084	0.2609
SG (PromptCast)	0.0	0.5349
ETTh1 (ETDataset)	0.1207	0.7987
ETTh2 (ETDataset)	0.0	0.3275
ETTm1 (ETDataset)	0.0	0.3177
ETTm2 (ETDataset)	0.02198	0.3057
M4-Month	0.0	1.2308
M4-Week	0.0	0.3716
M4-Quarter	0.0	1.10
M5 Competition	0.0	0.011
Google Wiki Traffic (GWT)	0.0742	0.5

6 Analysis limitations and further work

This study faces several limitations, primarily due to the high computational demands of LLMs and the restricted computational resources available.

One of the most evident limitations is the necessity to use only the smaller versions of the models in this study. I was able to compare only two relatively small models, and although I observed that the smaller model performed better, this may have been due to differences in training data rather than model size itself. This aspect warrants further research to determine the true impact of model size on performance.

The study revealed mixed results when applying the Phi3 model to practical business applications like web traffic forecasting and sales forecasting. While the model outperformed others on the MAE metric, it underperformed on all other metrics. This discrepancy suggests that the model's prediction quality may vary significantly depending on the metric used, potentially biasing the evaluation of its overall performance. This limitation highlights the need for further research to optimize LLMs for specific practical applications, ensuring that performance improvements in one metric do not come at the expense of others.

Another limitation observed was the significant discrepancy in prediction quality between using the model as a simple sequence-to-sequence model and utilizing its chat capabilities. This difference indicates that the model's performance can be heavily influenced by the method of utilization and by the prompt used, which may have biased the research findings. Further investigation is required to understand how different model utilization methods and prompts impact prediction accuracy and to develop best practices for employing LLMs in various forecasting tasks.

6.1 Futher work

One of the key advancements that can be derived directly from the code base I developed is the ability to modify and optimize the prompts. Prompt engineering is well-known for enhancing the performance of large language models, particularly for instruct models [18]. In my research, the focus was on evaluating the zero-shot performance of these models on univariate forecasting tasks. However, by employing prompt engineering techniques, it is highly probable that the models' performance could be significantly improved.

Prompt engineering involves crafting specific prompts that guide the model to generate more accurate and relevant outputs. For instance, carefully designed prompts can provide context or examples that help the model better understand the task at hand. This method has been shown to be particularly effective in few-shot learning scenarios, where providing a few examples within the prompt can enhance the model's understanding and performance. Additionally, advanced prompt engineering techniques such as Chain-of-Thought (CoT) prompting, introduced by Wei et al. (2022) [11], can further boost model performance. CoT prompting guides the model to break down complex tasks into intermediate reasoning steps, leading to more structured and logical outputs. This technique has proven beneficial for tasks requiring detailed and multi-step reasoning, which can be highly relevant for improving forecasting accuracy.

Moreover, this study has demonstrated that even with the use of smaller and computationally limited models, there is a potential for these models to somewhat predict time series data. Although the predictions are currently weak, this finding is promising and indicates that with further enhancements, these models could become more robust and accurate. One such enhancement could be the fine-tuning of these models.

Fine-tuning involves taking a pre-trained model and training it further on a specific dataset or task to improve its performance. Given that the initial results have shown some degree of predictive capability, fine-tuning could significantly enhance the accuracy

and robustness of the models. By training the models on time series-specific data, we could adapt them more closely to the nuances and patterns of such data, potentially overcoming some of the limitations observed in this study. Fine-tuning would allow the models to learn from the particular characteristics of the datasets used, leading to better generalization and performance in real-world applications.

Furthermore, fine-tuning can be combined with advanced prompt engineering techniques to further enhance model performance. By tailoring the training process and prompts specifically for time series forecasting, it is possible to maximize the models' predictive accuracy. This approach could address the current weaknesses and leverage the strengths demonstrated in the initial zero-shot evaluations, paving the way for more effective and reliable time series forecasting models in future research.

7 Conclusion

Although the prediction quality results are not particularly encouraging, the findings highlighted in Section 5 give us hope that further fine-tuning could enhance these models to better understand the underlying data distributions. The results provide strong evidence that the model is indeed capable of understanding the relationship between input and output, even though we are handling tabular data as text. This capability was not guaranteed and thus represents a significant positive outcome. This indicates that the challenges of text generation and time-series forecasting may indeed be related, and advancements in one area could potentially benefit the other. By fine-tuning these models, we could bridge the gap between their current performance and the benchmarks, ultimately leading to more accurate and reliable forecasting models.

While we found that these models are very easy to use and set up, they are also quite expensive. Scaling this model to an enterprise-grade level could be challenging due to the difficulty in obtaining reliable and straightforward outputs. Additionally, the high computational costs and resource requirements could hinder widespread adoption in large-scale applications.

To address these challenges, future research should focus on optimizing model efficiency and output consistency. Developing more sophisticated parsing algorithms and refining prompt engineering techniques could help mitigate the issues related to output reliability. Furthermore, exploring alternative architectures or hybrid models that balance performance and computational efficiency could provide a more viable path for enterprise deployment.

Investing in these improvements could make these advanced models more accessible and practical for real-world applications, ultimately enhancing their utility and impact across various industries. The potential of LLMs in time series forecasting, especially in scenarios where quick adaptability and minimal data preprocessing are crucial. The zero-shot capabilities of LLMs provide a significant advantage in terms of reducing the need for extensive model-specific training data and computational resources. This could democratize access to advanced forecasting tools, particularly benefiting smaller organizations and industries with limited data.

References

[1] Robert G. Brown, Richard F. Meyer, and D. A. D'Esopo. "The Fundamental Theorem of Exponential Smoothing". In: *Operations Research* 9.5 (1961), pp. 673–687. ISSN: 0030-364X. URL: https://www.jstor.org/stable/166814 (visited on 06/02/2024).

[2] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M4 Competition: Results, findings, conclusion and way forward". en. In: *International Journal of Forecasting* 34.4 (Oct. 2018), pp. 802–808. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2018.06.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169207018300785 (visited on 05/07/2024).

[3] Haoyi Zhou et al. *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*. arXiv:2012.07436 [cs]. Mar. 2021. URL: http://arxiv.org/abs/2012.07436 (visited on 05/07/2024).

[4] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "M5 accuracy competition: Results, findings, and conclusions". en. In: *International Journal of Forecasting* 38.4 (Oct. 2022), pp. 1346–1364. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2021.11.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169207021001874 (visited on 05/07/2024).

[5] Long Ouyang et al. *Training language models to follow instructions with human feedback*. arXiv:2203.02155 [cs]. Mar. 2022. URL: http://arxiv.org/abs/2203.02155 (visited on 06/02/2024).

[6] Edward Beeching et al. *Open LLM Leaderboard*. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard. 2023.

[7] Konstantinos Benidis et al. "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey". In: *ACM Computing Surveys* 55.6 (July 2023). arXiv:2004.10240 [cs, stat], pp. 1–36. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3533382. URL: http://arxiv.org/abs/2004.10240 (visited on 05/07/2024).

[8] Vijay Ekambaram et al. "TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. arXiv:2306.09364 [cs]. Aug. 2023, pp. 459–469. DOI: 10.1145/3580305.3599533. URL: http://arxiv.org/abs/2306.09364 (visited on 06/12/2024).

[9] Nate Gruver et al. *Large Language Models Are Zero-Shot Time Series Forecasters*. arXiv:2310.07820 [cs]. Oct. 2023. URL: http://arxiv.org/abs/2310.07820 (visited on 05/01/2024).

[10] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. URL: http://arxiv.org/abs/1706.03762 (visited on 05/01/2024).

[11] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903 [cs]. Jan. 2023. URL: http://arxiv.org/abs/2201.11903 (visited on 06/03/2024).

[12] Hao Xue and Flora D. Salim. *PromptCast: A New Prompt-based Learning Paradigm for Time Series Forecasting*. arXiv:2210.08964 [cs, math, stat]. Dec. 2023. URL: http://arxiv.org/abs/2210.08964 (visited on 05/06/2024).

[13] Marah Abdin et al. *Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone*. arXiv:2404.14219 [cs]. May 2024. URL: http://arxiv.org/abs/2404.14219 (visited on 06/02/2024).

[14] Abhimanyu Das et al. *A decoder-only foundation model for time-series forecasting*. arXiv:2310.10688 [cs]. Apr. 2024. URL: http://arxiv.org/abs/2310.10688 (visited on 06/12/2024).

[15] Ming Jin et al. *Time-LLM: Time Series Forecasting by Reprogramming Large Language Models*. arXiv:2310.01728 [cs]. Jan. 2024. URL: http://arxiv.org/abs/2310.01728 (visited on 05/01/2024).

[16] John A. Miller et al. *A Survey of Deep Learning and Foundation Models for Time Series Forecasting*. arXiv:2401.13912 [cs]. Jan. 2024. URL: http://arxiv.org/abs/2401.13912 (visited on 05/01/2024).

[17] AI@Meta. "Llama 3 Model Card". In: *2024* (). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[18] "Language Models are Unsupervised Multitask Learners". In.

8 Appendix

8.1 Dataset seasonality

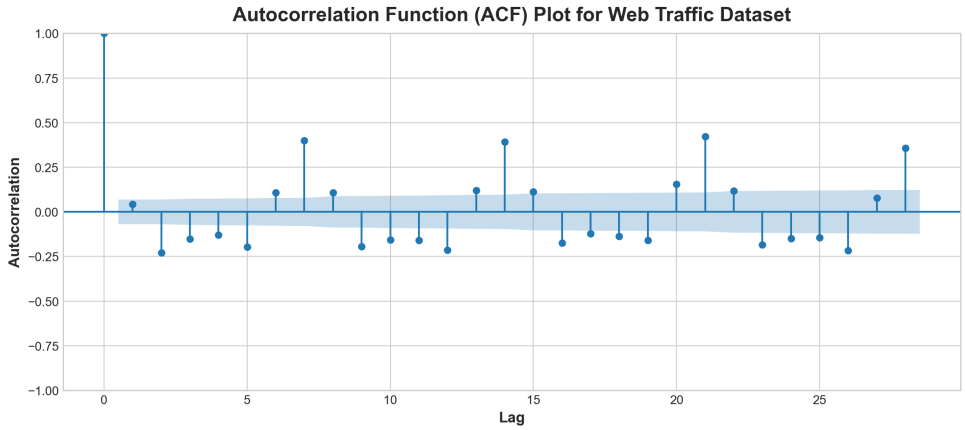

Figure 1: Autocorrelation Function (ACF) Plot for Web Traffic Dataset. This plot illustrates the autocorrelation of the web traffic dataset over different lags. The significant spikes at lag 7 and multiples thereof (e.g., lag 14, 21) indicate a weekly seasonality pattern, reflecting the recurring weekly trends in the dataset.

8.2 PromptCast minimal benchmark

Prompt	Model	CT				ECL				SG			
		RMSE		MAE		RMSE		MAE		RMSE		MAE	
		mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Basic	PromptCast (Bart)	6.512	0.016	4.790	0.012	603.571	2.461	380.583	1.695	8.677	0.055	5.912	0.013
	PromptCast (Pegasus)	6.496	0.012	4.767	0.007	**595.942**	2.246	**366.677**	1.373	**8.530**	0.032	**5.879**	0.010
	PromptCast (Bigbird)	**6.478**	0.033	**4.752**	0.019	609.315	3.071	378.100	2.112	8.576	0.035	5.922	0.009

Figure 2: Table X from the PromptCast paper illustrates the performance of different models (Bart, Pegasus, Bigbird) using basic prompts across three datasets: CT, ECL, and SG. The table reports RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) with both mean and standard deviation values. This table was chosen to provide a comparative baseline for evaluating the effectiveness of leveraging text models for tabular data through prompt engineering and zero-shot capabilities.