SysCaps: Language Interfaces for Simulation Surrogates of Complex Systems

Patrick Emami, Zhaonan Li[*], Saumya Sinha[*], Truc Nguyen[*]
National Renewable Energy Lab
{Patrick.Emami, Zhaonan.Li, Saumya.Sinha,Truc.Nguyen}@nrel.gov

Abstract

Surrogate models are used to predict the behavior of complex energy systems that are too expensive to simulate with traditional numerical methods. Our work introduces the use of language descriptions, which we call "system captions" or SysCaps, to interface with such surrogates. We argue that interacting with surrogates through text, particularly natural language, makes these models more accessible for both experts and non-experts. We introduce a lightweight multimodal text and timeseries regression model and a training pipeline that uses large language models (LLMs) to synthesize high-quality captions from simulation metadata. Our experiments on two real-world simulators of buildings and wind farms show that our SysCaps-augmented surrogates have better accuracy on held-out systems than traditional methods while enjoying new generalization abilities, such as handling semantically related descriptions of the same test system. Additional experiments also highlight the potential of SysCaps to unlock language-driven design space exploration and to regularize training through prompt augmentation.

1 Introduction

Data-driven surrogates enable computational scientists to efficiently predict the results of expensive numerical simulations that run on supercomputers [30, 7]. Surrogates are particularly valuable for emulating simulations of *complex energy systems* (CES), which model dynamic interactions between humans, earth systems, and infrastructure. Examples of CES include buildings [54, 9, 5], electric vehicle fleets [55], and microgrids [13]. Advancing the science of CES contributes to reducing emissions and accelerating the adoption of clean energy, which is needed to address the impacts of climate change.

These models perform a fairly standard regression task, predicting simulation output quantities of interest from *a*) an input system configuration and *b*) a deployment scenario. For example, we might want to predict the amount of energy a building will consume given *a*) a list of building characteristics and *b*) a weather timeseries spanning an entire year. In this case, this involves performing long sequence timeseries regression, which traditional regression techniques such as gradient-boosted decision trees have difficulty with [5, 62].

Surrogate models are not only used by experts. Surrogates are also used to inform highly consequential policy and investment decisions about complex systems made by non-experts in industry and governments [42], such as when planning to build and deploy a new renewable energy system [21]. In this work, we design and analyze *language interfaces* for such surrogates. Intuitively, language interfaces make surrogate models more accessible, particularly for non-experts, by simplifying how we inspect and alter a complex system's configuration. Language interfaces are powerful—they

[*]Equal contribution.

ground interactions between humans and machines in the human's preferred way [53]. The idea of using language to create interfaces for complex data or models is not new [23, 40], but interest has renewed due to the success of large language models (LLMs) and their demonstrated ability to generate high-quality synthetic natural captions [47, 12, 36, 22]. Our work defines a "system caption", or **SysCap**, as text-based descriptions of *knowledge about the system* being simulated. The only available knowledge our work assumes is the system configuration found in simulator metadata files as lists of attributes.

In general, it is unknown whether textual inputs, and particularly *natural language* inputs, are suitable for real-world *tabular* regression tasks. Tabular data, such as the system attributes in question, are sets of both discrete (categorical, binary, or string) and continuous (numeric) variables. Previous work demonstrated inconclusive evidence when using language models to do tabular regression from text-encoded inputs, with and without modifications to the architecture [11, 27, 4, 59], motivating further study. Regression with text-encoded tabular inputs is promising because *a*) language is a more intuitive and flexible user interface than traditional encoding strategies (e.g., one-hot encodings), and *b*) using language embeddings to encode system attributes unlocks the ability to exploit the semantic information contained in SysCaps to generalize across related systems.

Our paper introduces a framework for training *multimodal* surrogates for CES with text (for system attributes) and timeseries inputs (for the deployment scenario) and makes contributions towards addressing the following technical challenges:

- We introduce a simple and lightweight multimodal surrogate model architecture for timeseries regression that *a*) fuses text embeddings obtained from pretrained language models (LMs) with *b*) timeseries encoded by a bidirectional sequence encoder. We expect this to be insightful for future multimodal text and timeseries studies.
- To address the lack of human-labeled natural language descriptions of complex systems, we use LLMs to generate high-quality natural language SysCaps from simulation metadata. Although LLMs have previously been used to generate text captions from metadata [12, 37], we believe our application to multi-modal surrogate modeling is novel.
- We develop an automatic evaluation strategy to assess caption quality–specifically, we estimate the rate at which ground truth attributes appear in the synthetic description with a multiclass attribute classifier.

Our experiments are based on two real-world CES simulators of buildings and wind farm wake. We rigorously evaluate accuracy on held-out systems and show that SysCaps-augmented surrogates have better accuracy than one-hot baselines. We also show generalization beyond the capabilities of traditional regression approaches enabled by the use of text embeddings, e.g., robustness to replacing attributes names with synonyms in test captions. We qualitatively show that text interfaces unlock system design space exploration via language. As there are no standard benchmarks for comparing surrogate modeling performance for CES, we open-source data and code to facilitate future work: `https://github.com/NREL/SysCaps`.

2 Related Work

Language interfaces for scientific machine learning: An increasing amount of work is exploring language interfaces for scientific machine learning (SciML) models, including protein representation learning [58], protein design [32], and activity prediction for drug discovery [48]. LLM-powered natural language interfaces are also being designed for complicated scientific workflows including synchrotron management [39], automated chemistry labs [6], and fluid dynamics workflows [29]. We add to this work by studying language interfaces for lightweight surrogate models.

Large language models for regression: Another line of work asks whether LLMs can perform regression with both text inputs and outputs (numbers encoded as tokens), such as for tabular problems [11] or black-box optimization [49, 33]. We do not use an LLM to do regression directly, but rather train a lightweight multimodal architecture that predicts continuous outputs instead of tokens. Moreover, one study [11] found mixed results and highlighted difficulties with interpolation, raising questions about the effectiveness of LLM-based regression.

Multimodal text and timeseries modeling: Surrogate modeling can be cast as a timeseries forecasting problem when the goal is to train a model to emulate a dynamical system and predict its future behavior (e.g., predicting future energy demand from past energy usage). Previous work has explored multimodal timeseries forecasting where auxiliary text data is introduced as covariates to improve the forecasting accuracy [45, 14, 28]. Notably, Time-LLM [28] "reprograms" an LLM to process both text prompts and timeseries. However, our timeseries regression setting (Section 3) differs in that our surrogates are trained to map simulator inputs (e.g., building characteristics and a weather timeseries) to simulator outputs (e.g., energy usage). In our problem, models critically depend on the system information encoded as text, whereas in Time-LLM the text only contains auxiliary information that slightly improves forecasting accuracy. This dependence partially motivates our development of a custom architecture for fusing text and timeseries embeddings. Recent evidence [38, 51] also brings into question whether LLMs are useful at all for reasoning about temporal data; thus, research on lightweight multimodal models is warranted. Contrastive pretraining objectives have also been explored for modeling text and timeseries data [1, 26, 31, 63]. For example, Agostinelli et al. [1] aligns embeddings of captions that describe the audio. In our setting, captions describe system attributes, and the timeseries inputs are exogenous simulator inputs. Here, the text and timeseries inputs share no information with which to learn a shared embedding space.

Knowledge-enhanced PDE surrogates: Numerical simulation of partial differential equations (PDEs) is extremely computationally intensive, and thus a large body of SciML work is focused on developing neural PDE surrogates which are efficient to evaluate. Recent work has tried to encode knowledge about the physical system under study (the PDE) into a surrogate to facilitate generalization within and across families of PDEs. Specifically, methods that embed equation parameters (i.e., the system attributes) within the architecture to generalize to unseen parameters include CAPE [50] and those explored in Gupta & Brandstetter [19]. Others embed structural knowledge about the PDE into the surrogate model architecture [41, 61] or the loss [43]. Concurrent work has explored "PDE captions" [34, 60], which are a type of SysCaps for neural PDE surrogates where the system knowledge is PDE equations encoded as text.

3 Problem Statement

Our goal is to learn a surrogate $f : \mathcal{X} \times \mathcal{Z} \rightarrow \mathcal{Y}$ that regresses the outputs of a simulator F directly from its inputs. We are given a dataset D of pairs of simulator inputs and outputs. The inputs are a deployment scenario $x_{1:T} \in \mathcal{X}$ (a timeseries) and the tabular system attributes $z \in \mathcal{Z}$. The outputs are a timeseries $y_{1:T} \in \mathcal{Y}$. For simplicity, we consider only univariate timeseries outputs in this work ($y_t \in \mathbb{R}$). However, the number of timesteps T may be large (thousands of steps), the timeseries inputs $\mathcal{X}$ are multivariate, and the mapping f which approximates the simulator is highly nonlinear. To summarize, we have a timeseries regression problem modeled as $y_{1:T} = f(x_{1:T}, z)$. By conditioning the surrogate on system knowledge z, it can potentially generalize to new system configurations. However, learning transferable representations of variable-length, heterogeneous input features such as z is notoriously difficult for deep neural networks, and is a key focus of tabular deep learning (see survey [3]). In our work, we develop a framework for multimodal surrogates where tabular z is encoded as text. Some simulators may have inputs that are not clearly distinguishable into what is $\mathcal{X}$ and $\mathcal{Z}$, for example, if a dynamical system simulation is configured to be in steady-state or assumes fixed exogenous conditions. In these cases, we allow $\mathcal{X}$ to be a vector of real-valued scalars (a timeseries with $T = 1$), or, simply an empty set (leaving only $\mathcal{Z}$).

Example: In many CES, the timeseries $\mathcal{X}$ are exogenous inputs to the system such as weather timeseries consisting of temperature or wind speed. Attributes $\mathcal{Z}$ of a wind farm might include the number of turbines in the wind farm and turbine blade length.

4 Synthesizing System Captions (SysCaps) with LLMs

Our work is motivated by the idea that language interfaces for surrogates represent a path towards improving the accessibility of these models for expert and non-expert users, e.g., when using them for downstream system design tasks [53]. Our proposed framework for augmenting surrogates with language interfaces is visualized in Figure 1. During training, we create SysCaps out of system attributes specified in simulation metadata. To create large amounts of synthetic natural language SysCaps, we use LLMs. The ultimate goal is to enable scientists to "chat" with the multi-modal surrogate model at test time via text prompting. In this section, we describe two approaches for converting system attributes into text: key-value templates and natural language.

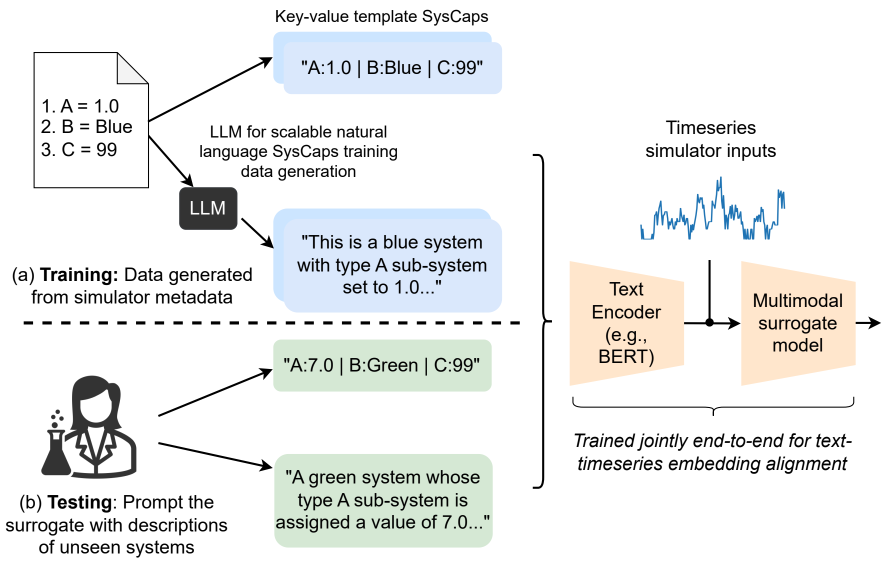

Key-value template SysCaps
"A:1.0 | B:Blue | C:99"
1. A = 1.0
2. B = Blue
3. C = 99
LLM for scalable natural language SysCaps training data generation
LLM
"This is a blue system with type A sub-system set to 1.0..."
(a) Training: Data generated from simulator metadata
"A:7.0 | B:Green | C:99"
"A green system whose type A sub-system is assigned a value of 7.0..."
(b) Testing: Prompt the surrogate with descriptions of unseen systems
Timeseries simulator inputs
Text Encoder (e.g., BERT)
Multimodal surrogate model
Trained jointly end-to-end for text-timeseries embedding alignment

Figure 1: Our pipeline for augmenting multimodal simulation surrogates with language interfaces using "system captions", or SysCaps. SysCaps are text descriptions of knowledge about the system being simulated. In our work, the SysCaps describe the system's characteristics, as found in simulation metadata files. During training (a), we create paired datasets of temporal simulator inputs with key-value template SysCaps or LLM-generated natural language SysCaps. At test time (b), we prompt the surrogate model with one or more key-value template captions or natural language captions. LLMs are only used to generate synthetic training data; we use a lightweight BERT-style text encoder and an efficient long-sequence encoder to keep the computational cost of our surrogate low.

For the key-value approach, attributes are described as key-value pairs `key:value` and joined by a separator "|" (**SysCaps-kv**). For example, if a simulation has attributes A=1.0 and B=blue, we create the string `A:1.0|B:blue`. Generating these strings is easy to do and incurs a negligible amount of extra computational overhead. In the natural language approach (**SysCaps-nl**, Figure 1), attributes are described in a conversational manner, which we believe is more flexible and expressive than key-value captions and thereby more accessible for non-experts. However, we do not have access to large quantities of natural language descriptions for each system and simulation. We avoid the time-consuming task of enlisting domain experts to create this data by instead prompting a powerful LLM to generate synthetic natural language descriptions given attributes. In our work, we use the open-source LLM `llama-2-7b-chat` [52]. The details of the prompt are provided in Appendix A.

5 Text and Timeseries Surrogate Model

We now describe a lightweight multimodal surrogate model for timeseries regression. The surrogate f (Figure 2) is a composition of a multimodal encoder function $g_\psi : (\mathcal{Z}, \mathcal{X}) \to \{\mathbb{R}^d\}_{1:T}$ and a top model $h_\theta : \mathbb{R}^d \to \mathbb{R}$, where for simplicity, the model parameters θ are shared across timesteps to predict each timeseries output y_t. The training objective is to minimize the expected mean square error averaged over simulation timesteps,

$$\min_{\theta,\psi} \mathbb{E}_{(z,x_{1:T},y_{1:T})\sim D}\left[\frac{1}{T}\sum_{t=1}^{T}\Big([h_\theta(g_\psi(z,x_{1:T}))]_t - y_t\Big)^2\right]. \tag{1}$$

Although more sophisticated loss functions than Eq. 1 could be used that account for predictive uncertainty, we left this extension for future work to simplify our exposition and experiments.

Multimodal encoder g_ψ: A text encoder g_ψ^{text} extracts an embedding $\hat{z}$ from a SysCap z then broadcasts and concatenates this embedding with the timeseries inputs to create a sequence of multimodal feature vectors. These features get processed by a bidirectional sequence encoder g_ψ^{seq} to

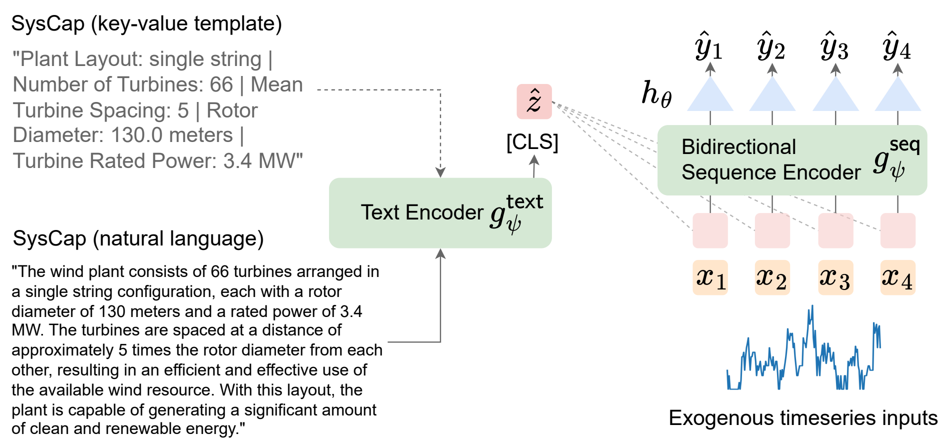

SysCap (key-value template)
"Plant Layout: single string | Number of Turbines: 66 | Mean Turbine Spacing: 5 | Rotor Diameter: 130.0 meters | Turbine Rated Power: 3.4 MW"
SysCap (natural language)
"The wind plant consists of 66 turbines arranged in a single string configuration, each with a rotor diameter of 130 meters and a rated power of 3.4 MW. The turbines are spaced at a distance of approximately 5 times the rotor diameter from each other, resulting in an efficient and effective use of the available wind resource. With this layout, the plant is capable of generating a significant amount of clean and renewable energy."
Text Encoder g_ψ^{text}
[CLS]
$\hat{z}$
h_θ
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
Bidirectional Sequence Encoder g_ψ^{seq}
x_1 x_2 x_3 x_4
Exogenous timeseries inputs

Figure 2: Building blocks of our surrogate model, $f = h_\theta \circ g_\psi$, that includes a multimodal encoder, g_ψ, and a top model, h_θ. The multimodal encoder, $g_\psi = g_\psi^{\text{seq}} \circ g_\psi^{\text{text}}$, is a composition of a text encoder, g_ψ^{text}, and a bidirectional sequence encoder, g_ψ^{seq}, for timeseries inputs. The text embedding vector $\hat{z}$ is broadcasted (dashed lines) to create a sequence that is concatenated with the timeseries input. This multimodal sequence is the input to the sequence encoder.

produce a sequence of time-dependent fused multimodal features $e_{1:T}$, $e_{1:T} = g_\psi^{\text{seq}}(g_\psi^{\text{text}}(z), x_{1:T})$, which are finally used to regress outputs.

Text encoder g_ψ^{text}: To encode textual inputs we use pretrained BERT [10] and DistilBERT [46] models that are relatively more efficient than LLMs. We use the model's default pretrained tokenizer. Tokenized sequences are bracketed by `[CLS]` and `[EOS]` tokens, and we use the final activation at the `[CLS]` token position to produce a text embedding $\hat{z} \in \mathbb{R}^d$. Following standard fine-tuning practices, all layers for BERT are fine-tuned while only the last layer of DistilBERT is fine-tuned.

Bidirectional sequence encoder g_ψ^{seq}: We broadcast the text embedding $\hat{z}$ to create a sequence of length T, $\hat{z} \rightarrow \{\hat{z}_t\}_{t=1}^T$, and concatenate each $\hat{z}_t$ with the timeseries input $x_{1:T}$, $\{\hat{z}_t; x_t\}_1^T$. This simplifies the task of learning timestep-specific correlations between system attributes $\hat{z}$ and timeseries $x_{1:T}$ in the multimodal encoder g_ψ. To efficiently embed long timeseries with thousands of timesteps, we explore both bidirectional LSTMs [25] and bidirectional SSMs [16] for g_ψ^{seq}. Our bidirectional SSM uses stacks of S4 blocks [18] without downpooling layers. We use the last layer's hidden states as temporal features $e_{1:T}$ for the top model. If $T = 1$ or for non-sequential surrogate models, we instead use an MLP with residual layers (ResNet MLP) to embed each $\{\hat{z}_t; x_t\}$ per-timestep to get e_t.

Top model h_θ: The multimodal encoder g_ψ produces T feature vectors $e_{1:T}$. For simplicity, the output $\hat{y}_t$ at each timestep is predicted from e_t by a shared MLP with a single hidden layer.

6 Experiments

This section presents our experimental results, which include: quantifying the quality of LLM-generated SysCaps (Section 6.1), accuracy on held-out systems (Section 6.2), generalization under distribution shifts (Section 6.3), a design space exploration application (Section 6.4), and a study on SysCaps prompt augmentation (Section 6.5). These experiments are based on two real-world CES simulators for buildings and wind farms. All SysCaps are synthetically generated in this work. We provide additional qualitative examples of SysCaps in Appendix C.

Setup: Our main experiments focus on training building stock surrogate models for the building energy simulator EnergyPlus [8]. Given an annual hourly weather timeseries (T = 8,760) with 7 variables and a list of tabular building attributes, surrogates predict the building's energy consumption at each hour of the year. We use the commercial building split of the Buildings-900K dataset [15], which are building stock simulation runs representative of the United States commercial building stock. Commercial building stock surrogates can provide significant speedups compared to EnergyPlus,

Table 1: **Accuracy.** We show the mean NRMSE across 3 random seeds. Lower NRMSE is better. Building-hourly is the NRMSE normalized per building and per hour. Stock-annual first sums the predictions and targets over all buildings and hours, before computing the NRMSE (equivalent to the normalized mean bias error—see Appendix B.1). When using a BERT encoder instead of DistilBERT, SysCaps surrogates achieve better accuracy than one-hot baselines. We trained one SSM without any attribute information (Attribute Encoding "X") as an ablation study; the poor accuracy shows that our multimodal architecture successfully learns to fuse the text-based attribute and timeseries inputs.

Model	Text Encoder	Attribute Encoding	Buildings-Hourly (NRMSE)	Stock-Annual (NRMSE)
SSM	-	X	1.712$\pm$ 0.003	0.658$\pm$ 0.008
SSM	-	onehot	0.450$\pm$ 0.019	0.041$\pm$ 0.021
LSTM	-	onehot	0.449$\pm$ 0.025	0.045$\pm$ 0.024
ResNet	-	onehot	0.634$\pm$ 0.009	0.072$\pm$ 0.008
LightGBM	-	onehot	0.679$\pm$ 0.014	0.094$\pm$ 0.003
SSM	DistilBERT	SysCaps-nl	0.532$\pm$ 0.010	0.069$\pm$ 0.010
SSM	BERT	SysCaps-nl	0.543$\pm$ 0.011	0.035$\pm$ 0.005
SSM	DistilBERT	SysCaps-kv	0.454$\pm$ 0.012	0.046$\pm$ 0.003
SSM	BERT	SysCaps-kv	0.450$\pm$ 0.007	**0.020**$\pm$ 0.012
LSTM	DistilBERT	SysCaps-kv	0.489$\pm$ 0.021	0.063$\pm$ 0.005
LSTM	BERT	SysCaps-kv	**0.439**$\pm$ 0.037	0.022$\pm$ 0.011
ResNet	DistilBERT	SysCaps-kv	0.633$\pm$ 0.020	0.081$\pm$ 0.011
ResNet	BERT	SysCaps-kv	0.670$\pm$ 0.043	0.049$\pm$ 0.015

e.g., 96% [62]. Since this dataset only provides energy timeseries, we extract the building metadata and weather timeseries from the End-Use Load Profiles database [57] for each building. We use the top 13 attributes per commercial building as ranked by recursive feature elimination [20]. Our training set is comprised of 330K buildings, and we use 100 buildings for validation, and 6K held-out buildings for testing. We also reserved a held-out set of 10K buildings for RFE. We carefully tune the hyperparameters of all models (details in Appendix B.2). We created three SysCaps datasets: a "medium" caption length dataset of 4-6 sentences, a "short" dataset using 2-3 sentences, and a "long" dataset using 7-9 sentences. The SSMs in our experiments are trained with medium captions. Generating these datasets with `llama-2-7b-chat` used $\sim$1.5K GPU hours on a cluster with 16 NVIDIA A100-40GB GPUs.

6.1 Evaluating caption quality

The LLM that generates natural language SysCaps may erroneously ignore or hallucinate attributes, which may negatively impact downstream performance. To test this, we propose evaluating generated captions by estimating the fraction of attributes which the LLM successfully includes per caption. To compute this metric, we train a multi-class classifier to predict each categorical attribute in a caption from its text embedding. We used a held-out validation set of captions to check that the classifier was not overfitting. The test rate of missing or incorrect attributes is around 9-12% across the "short", "medium", and "long" caption types, with "short" captions having the highest error (Table 2). This increases our confidence that our LLM-based approach for generating natural language SysCaps preserves sufficient information for surrogate modeling.

6.2 Accuracy On Held-Out Systems

Following Emami et al. [15], we use the normalized root mean square error (NRMSE) metric to compare model accuracy, averaged across 3 random seeds. In addition to comparing the ResNet MLP, LSTM, and SSM, we also trained a tuned LightGBM baseline. **Does the sequential architecture matter?** *Yes*—Table 1 shows that the LSTM and SSM encoders outperform both the ResNet and our carefully tuned LightGBM baseline, and the SSM outperforms the LSTM. **How do different system attribute encoding approaches compare?** Surprisingly, the best SSM and LSTM with key-value SysCaps achieve comparable building-hourly test accuracy to one-hot baselines and the *best* accuracy at the stock-annual aggregation level (equivalent to the normalized mean bias error—see

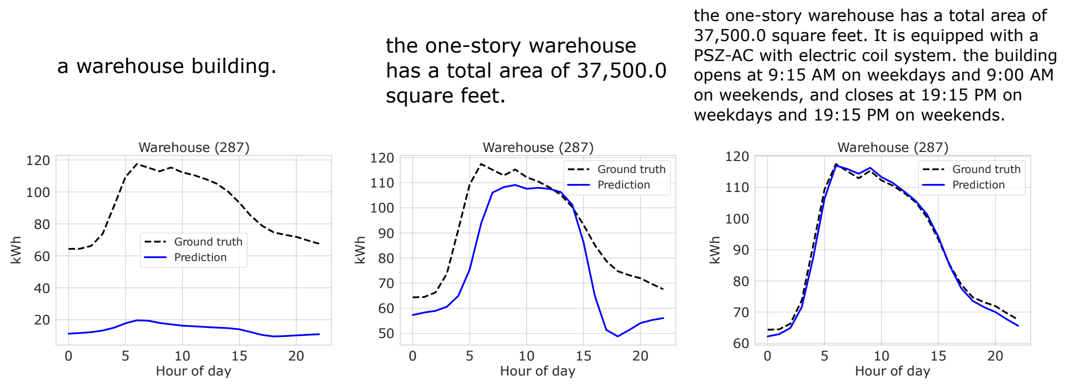


Figure 3: **System captions unlock text-prompt-style surrogate modeling for complex systems.** We show building stock daily load profiles aggregated for Warehouse building type, created with caption templates. From left to right, we use captions with one, three, and six attributes.

Table 2: **Caption quality**. We estimate the presence of each attribute in a SysCap, measured by the average test accuracy of a multi-class classifier trained to predict each categorical attribute. Our metric suggests ∼9-12% of attributes are missing or incorrect per SysCap, due to errors made by `llama-2-7b-chat`.

Caption length (13 attributes)	Accuracy (%)
Short (2-3 sent.)	88.90
Medium (4-6 sent.)	90.90
Long (7-9 sent.)	90.38

Table 3: **SysCaps zero-shot length generalization**. NRMSE is per-building-hourly. Results are for the SSM model trained with medium-length SysCaps and evaluated zero-shot on short and long captions.

SysCaps length	NRMSE
Short	0.57$\pm$ 0.02
Medium	**0.53**$\pm$ 0.01
Long	0.64$\pm$ 0.02

Appendix B.1). The SSM and LSTM with natural language SysCaps has slightly worse accuracy than with the key-value SysCaps, yet they comfortably outperform the non-sequential models (including LightGBM) and the one-hot baselines at the stock-annual aggregation level. We initially expected to see a non-negligible drop in regression accuracy for SysCaps models, even key-value SysCaps, because the text encoder has to compress the caption into a single embedding vector. However, the BERT encoder is sufficiently expressive to mitigate this. We observe that the stock-level NRMSE reduces by about half when switching from DistilBERT to BERT. We believe the performance gap between key-value templates and natural language SysCaps is mostly explained by the caption quality (Table 2). We ablate the importance of the system attributes by training an SSM baseline with these inputs removed (SSM with Attribute Encoding "X"); this model is unable to learn this task. Figure 3 qualitatively shows how a SysCaps model performs with natural language captions (created with a sentence template) provided to the model. Note that we did not train our models on any captions with missing attributes. Prediction accuracy improves as more information is given; notably, there is a large jump in accuracy once the building square footage is known.

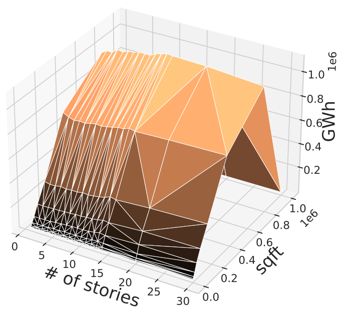
6.3 Caption Generalization

Length generalization: We assess how accuracy varies when surrogates are provided with natural language SysCaps having different lengths than seen during training. We evaluate zero-shot generalization to the short and long captions. The results (Table 3) show a small increase in error for shorter captions with a larger increase in error for longer captions, as might be expected. The error on long captions remains lower than the error achieved by our tuned LightGBM baseline.

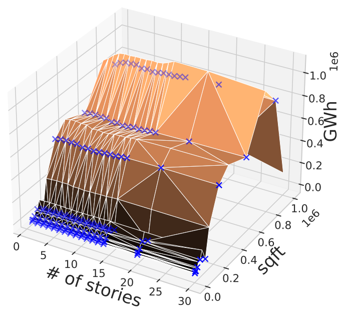
Attribute synonyms: To quantitatively evaluate the extent to which natural language SysCaps surrogates gain a level of robustness to distribution shifts such as word order changes, synonyms, or writing style [24], we created captions for the held-out systems where the "building type" attribute is

Table 4: **Generalization to attribute synonyms.** We quantify how text embeddings make our models robust to the use of attribute synonyms. The metric is the difference in NRMSE between the original caption and the modified caption. We bold the best method, i.e, closest to 0. Column 3 replaces the building type with a synonym, column 4 removes the building type and sub-type attributes from the caption, and column 5 randomly swaps the correct building type attributes with incorrect ones.

Building Type	Synonym	With Synonym	Without Building Type	Random Building Type
FullServiceRestaurant	FineDiningRestaurant	**0.52**± 0.05	0.93± 0.01	1.17± 0.07
RetailStripmall	ShoppingCenter	**0.01**± 0.00	0.68± 0.02	0.28± 0.04
Warehouse	StorageFacility	**0.35**± 0.30	0.55± 0.31	4.02± 0.32
RetailStandalone	ConvenienceStore	**0.00**± 0.01	0.30± 0.04	0.40± 0.03
SmallOffice	Co-WorkingSpace	0.03± 0.01	**0.02**± 0.02	1.95± 0.30
PrimarySchool	ElementarySchool	**0.00**± 0.01	0.38± 0.02	0.52± 0.17
MediumOffice	Workplace	0.08± 0.02	**0.03**± 0.04	0.91± 0.11
SecondarySchool	HighSchool	**-0.01**± 0.04	0.52± 0.06	0.67± 0.33
Outpatient	MedicalClinic	**0.02**± 0.01	0.55± 0.09	0.32± 0.06
QuickServiceRestaurant	FastFoodRestaurant	**0.10**± 0.07	0.83± 0.01	0.85± 0.01
LargeOffice	OfficeTower	**0.12**± 0.13	0.23± 0.03	0.29± 0.11
LargeHotel	Five-Star Hotel	**0.03**± 0.01	0.46± 0.06	0.29± 0.09
SmallHotel	Motel	**0.26**± 0.07	0.88± 0.07	0.70± 0.14
Hospital	HealthcareFacility	**0.03**± 0.04	0.62± 0.12	0.20± 0.07



(a)



(b)

Figure 4: **Design space exploration using language**. a) We show that the model has learned a physically plausible relationship between building square footage (sqft) and number (#) of stories. b) Failure case: When tested on unseen values of sqft (blue crosses), the model's predictions appear to be physically implausible—the model underestimates the energy consumption at these sqft values.

replaced by a synonym. We avoid biasing the choice of building type synonym by 5-shot prompting `llama-2` to suggest the synonyms. There are two baselines we compare the synonym caption accuracy against: 1) accuracy when testing the model on captions with the building type attribute removed, and 2) accuracy when testing the model on captions with a random building type. Examples and results are shown in Table 4, where for 11/14 building type synonyms the increase in NRMSE is less than 13%, while the average increase for the two baselines is 54% and 90%, respectively.

6.4 Design Space Exploration Application Using Language

Figure 4a visualizes the results of a language-based sensitivity analysis on two system attributes, as might be performed for an early-stage design space exploration task. We use a template to create a caption for each test building that enumerates all combinations of the number of stories and square footage attributes, totaling 160 configurations; the entire analysis requires simulating 960K buildings, and took 1 hour on a single NVIDIA A100 GPU. We observe that the model has learned physically plausible relationships between these two attributes. However, the model fails to predict the energy usage for buildings over 100K square feet—such buildings are in the "long tail" of the training data distribution. Figure 4b also shows a failure case where the model underestimates energy usage at unseen numeric building square footage attribute values.

6.5 Prompt Augmentation: Wind Farm Wake

This experiment uses the Wind Farm Wake Modeling Dataset [44], made with the FLORIS simulator, to train a surrogate to predict a wind farm's power generation in steady-state atmospheric conditions. The difficulty of this task is in modeling losses due to wake effects, given only a coarse description of the wind farm layout. There are three numeric simulator inputs x specifying atmospheric conditions, and five system attributes which include categorical variables indicating wind farm shape (four different layout types), number of turbines, and average turbine spacing. We do not use RFE. In this dataset, there are only 500 unique system configurations (split 3:1:1 for train, val, test), although each configuration is simulated under 500 distinct atmospheric conditions.

Table 5: **Wind farm accuracy.** The base architecture is ResNet. Average across 3 random seeds.

Model	NRMSE
LightGBM	0.196$\pm$0.000
one-hot	0.212$\pm$ 0.009
SysCaps-kv	0.054$\pm$ 0.024
SysCaps-nl	**0.036**$\pm$ 0.001

We explore generating multiple captions for each system configuration through prompt augmentation to increase diversity. Specifically, we modify our prompt (Appendix A) to encourage different description styles by adding phrases such as: *with an objective tone*, *with an objective tone (creative paraphrasing is acceptable)*, *to a colleague*, and *to a classroom*. This simulation assumes steady-state conditions so we tune hyperparameters for and train the non-sequential ResNet models. The ResNet baseline with one-hot encoded attributes suffers from severe overfitting (Table 5), likely due to the small number (300) of training systems, whereas the SysCaps models generalize better to unseen systems. This suggests SysCaps can have a regularizing effect in small data settings. Notably, the prompt augmentation helps the natural language SysCaps model to achieve the lowest NRMSE.

7 Conclusion

In this work, we introduced a lightweight, multimodal text and timeseries surrogate models for complex energy systems such as buildings and wind farms, and described a process for using LLMs to synthesize natural language descriptions of such systems, which we call SysCaps. Our experiments show SysCaps can achieve better accuracy than standard feature engineering while also enjoying the advantages of text embeddings such as robustness to caption paraphrasing. For a problem with only a *small* number of training systems available, SysCaps prompt augmentation can have a regularizing effect that helps mitigate overfitting. Overall, these results underscore that language is a viable interface for interacting with real-world surrogate models of complex systems.

Limitations: Current BERT-style tokenizers struggle with numerical values [56] and can interpolate poorly to unseen numbers (Figure 4b). Orthogonal research on improving number encodings for language model inputs [17, 59] can benefit our framework. Another potential concern is with creating SysCaps for simulators with a large number of attributes (e.g., over 100). A more powerful LLM than `llama-2-7b-chat` with a longer context window may be needed in this case. In general, we expect that more powerful LLMs will further improve the quality of the training captions.

Broader impacts and future work: Improving surrogate models for CES has the potential to accelerate the transition to cleaner energy sources. To avoid unfair outcomes from decisions made with CES surrogates, care should be taken when deciding what simulation runs to use as training data and when selecting an LLM to use for caption generation. A future extension of this work might explore how to use language to also manipulate the non-text (e.g., timeseries) simulator inputs. For example, to study how the complex system behaves when the average temperature increases by five degrees. Another promising extension is to conduct studies with domain scientists to evaluate the quality of SysCaps in the context of a downstream task such as system design optimization. An important question is how we might create *surrogate foundation models* that generalize not only across system configurations for a single simulator, but also generalize across different simulators.

Acknowledgments

This work was authored by the National Renewable Energy Laboratory (NREL), operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported by the Laboratory Directed Research and Development (LDRD) Program at NREL. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. The research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

References

[1] Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[2] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[3] Badaro, G., Saeed, M., and Papotti, P. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*, 11: 227–249, 2023.

[4] Bellamy, D. R., Kumar, B., Wang, C., and Beam, A. Labrador: Exploring the limits of masked language modeling for laboratory data. *arXiv preprint arXiv:2312.11502*, 2023.

[5] Bhavsar, S., Pitchumani, R., Reynolds, M., Merket, N., and Reyna, J. Machine learning surrogate of physics-based building-stock simulator for end-use load forecasting. pp. 113395, 2023. ISSN 03787788. doi: 10.1016/j.enbuild.2023.113395. URL `https://linkinghub.elsevier.com/retrieve/pii/S0378778823006254`.

[6] Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A., and Schwaller, P. Augmenting large language models with chemistry tools. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[7] Carter, J., Feddema, J., Kothe, D., Neely, R., Pruet, J., Stevens, R., Balaprakash, P., Beckman, P., Foster, I., Iskra, K., et al. Advanced research directions on ai for science, energy, and security: Report on summer 2022 workshops. 2023.

[8] Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., et al. Energyplus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, 2001.

[9] Dai, T.-Y., Niyogi, D., and Nagy, Z. Citytft: Temporal fusion transformer for urban building energy modeling. *ArXiv preprint*, abs/2312.02375, 2023. URL `https://arxiv.org/abs/2312.02375`.

[10] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Dinh, T., Zeng, Y., Zhang, R., Lin, Z., Gira, M., Rajput, S., Sohn, J.-y., Papailiopoulos, D., and Lee, K. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35:11763–11784, 2022.

[12] Doh, S., Choi, K., Lee, J., and Nam, J. LP-MusicCaps: LLM-based pseudo music captioning. *ArXiv preprint*, abs/2307.16372, 2023. URL `https://arxiv.org/abs/2307.16372`.

[13] Du, Y. and Li, F. Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning. *IEEE Transactions on Smart Grid*, 11(2):1066–1076, 2019.

[14] Emami, H., Dang, X.-H., Shah, Y., and Zerfos, P. Modality-aware transformer for time series forecasting. *arXiv preprint arXiv:2310.01232*, 2023.

[15] Emami, P., Sahu, A., and Graf, P. Buildingsbench: A large-scale dataset of 900k buildings and benchmark for short-term load forecasting. *Advances in Neural Information Processing Systems*, 2023.

[16] Goel, K., Gu, A., Donahue, C., and Ré, C. It's raw! audio generation with state-space models. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7616–7633. PMLR, 2022. URL `https://proceedings.mlr.press/v162/goel22a.html`.

[17] Golkar, S., Pettee, M., Eickenberg, M., Bietti, A., Cranmer, M., Krawezik, G., Lanusse, F., McCabe, M., Ohana, R., Parker, L., et al. xval: A continuous number encoding for large language models. *arXiv preprint arXiv:2310.02989*, 2023.

[18] Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[19] Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[20] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine learning*, 46:389–422, 2002.

[21] Harrison-Atlas, D., Glaws, A., King, R. N., and Lantz, E. Artificial intelligence-aided wind plant optimization for nationwide evaluation of land use and economic benefits of wake steering. *Nature Energy*, pp. 1–15, 2024.

[22] Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pp. 5549–5581. PMLR, 2023.

[23] Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., and Slocum, J. Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS)*, 3(2):105–147, 1978.

[24] Hendrycks, D., Liu, X., Wallace, E., Dziedzic, A., Krishnan, R., and Song, D. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

[25] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[26] Huang, Q., Jansen, A., Lee, J., Ganti, R., Li, J. Y., and Ellis, D. P. W. MuLan: A joint embedding of music audio and natural language, 2022. URL `https://arxiv.org/abs/2208.12415`.

[27] Jablonka, K. M., Schwaller, P., Ortega-Guerrero, A., and Smit, B. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, pp. 1–9, 2024.

[28] Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., and Wen, Q. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[29] Kumar, V., Gleyzer, L., Kahana, A., Shukla, K., and Karniadakis, G. E. Mycrunchgpt: A llm assisted framework for scientific machine learning. *Journal of Machine Learning for Modeling and Computing*, 4(4), 2023.

[30] Lavin, A., Krakauer, D., Zenil, H., Gottschlich, J., Mattson, T., Brehmer, J., Anandkumar, A., Choudry, S., Rocki, K., Baydin, A. G., et al. Simulation intelligence: Towards a new generation of scientific methods. *ArXiv preprint*, abs/2112.03235, 2021. URL `https://arxiv.org/abs/2112.03235`.

[31] Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D. Audioldm: Text-to-audio generation with latent diffusion models. *ArXiv preprint*, abs/2301.12503, 2023. URL `https://arxiv.org/abs/2301.12503`.

[32] Liu, S., Zhu, Y., Lu, J., Xu, Z., Nie, W., Gitter, A., Xiao, C., Tang, J., Guo, H., and Anandkumar, A. A text-guided protein design framework. *arXiv preprint arXiv:2302.04611*, 2023.

[33] Liu, T., Astorga, N., Seedat, N., and van der Schaar, M. Large language models to enhance bayesian optimization. *International Conference on Learning Representations*, 2024.

[34] Lorsung, C., Li, Z., and Barati Farimani, A. Physics informed token transformer for solving partial differential equations. *Machine Learning: Science and Technology*, 2024.

[35] Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2017.

[36] Mei, X., Meng, C., Liu, H., Kong, Q., Ko, T., Zhao, C., Plumbley, M. D., Zou, Y., and Wang, W. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *arXiv preprint arXiv:2303.17395*, 2023.

[37] Mei, X., Meng, C., Liu, H., Kong, Q., Ko, T., Zhao, C., Plumbley, M. D., Zou, Y., and Wang, W. WavCaps: A ChatGPT-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *ArXiv preprint*, abs/2303.17395, 2023. URL `https://arxiv.org/abs/2303.17395`.

[38] Merrill, M. A., Tan, M., Gupta, V., Hartvigsen, T., and Althoff, T. Language models still struggle to zero-shot reason about time series. *arXiv preprint arXiv:2404.11757*, 2024.

[39] Potemkin, D., Soto, C., Li, R., Yager, K., and Tsai, E. Virtual scientific companion for synchrotron beamlines: A prototype. *ArXiv preprint*, abs/2312.17180, 2023. URL `https://arxiv.org/abs/2312.17180`.

[40] Quamar, A., Efthymiou, V., Lei, C., Özcan, F., et al. Natural language interfaces to data. 11(4): 319–414, 2022.

[41] Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[42] Rackauckas, C. V. and Abdelrehim, A. Scientific machine learning (sciml) surrogates for industry, part 1: The guiding questions. https://doi.org/10.31219/osf.io/p95zn, 2024. Accessed: 2024-03-22.

[43] Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[44] Ramos, D., Glaws, A., King, R., , and Harrison-Atlas, D. Flow redirection and induction in steady state (floris) wind plant power production data sets, 2023. URL `https://data.openei.org/submissions/5884`.

[45] Rodrigues, F., Markou, I., and Pereira, F. C. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion*, 49:120–129, 2019.

[46] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108, 2019. URL `https://arxiv.org/abs/1910.01108`.

[47] Schick, T. and Schütze, H. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6943–6951, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.555. URL `https://aclanthology.org/2021.emnlp-main.555`.

[48] Seidl, P., Vall, A., Hochreiter, S., and Klambauer, G. Enhancing activity prediction models in drug discovery with the ability to understand human language. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30458–30490. PMLR, 2023. URL `https://proceedings.mlr.press/v202/seidl23a.html`.

[49] Song, X., Li, O., Lee, C., Peng, D., Perel, S., Chen, Y., et al. Omnipred: Language models as universal regressors. *ArXiv preprint*, abs/2402.14547, 2024. URL `https://arxiv.org/abs/2402.14547`.

[50] Takamoto, M., Alesiani, F., and Niepert, M. Learning neural pde solvers with parameter-guided channel attention. In *International Conference on Machine Learning*, pp. 33448–33467. PMLR, 2023.

[51] Tan, M., Merrill, M. A., Gupta, V., Althoff, T., and Hartvigsen, T. Are language models actually useful for time series forecasting? *arXiv preprint arXiv:2406.16964*, 2024.

[52] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023. URL `https://arxiv.org/abs/2307.09288`.

[53] Vaithilingam, P., Arawjo, I., and Glassman, E. L. Imagining a future of designing with ai: Dynamic grounding, constructive negotiation, and sustainable motivation. *arXiv preprint arXiv:2402.07342*, 2024.

[54] Vazquez-Canteli, J., Demir, A. D., Brown, J., and Nagy, Z. Deep neural networks as surrogate models for urban energy simulations. In *Journal of Physics: Conference Series*, volume 1343, pp. 012002. IOP Publishing, 2019.

[55] Vepsäläinen, J., Otto, K., Lajunen, A., and Tammi, K. Computationally efficient model for energy demand prediction of electric city bus in varying operating conditions. *Energy*, 169: 433–443, 2019.

[56] Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. Do NLP models know numbers? probing numeracy in embeddings. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5307–5315, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1534. URL `https://aclanthology.org/D19-1534`.

[57] Wilson, E. J., Parker, A., Fontanini, A., Present, E., Reyna, J. L., Adhikari, R., Bianchi, C., CaraDonna, C., Dahlhausen, M., Kim, J., et al. End-use load profiles for the us building stock: Methodology and results of model calibration, validation, and uncertainty quantification. Technical report, National Renewable Energy Lab (NREL). URL `https://www.nrel.gov/docs/fy22osti/80889.pdf`.

[58] Xu, M., Yuan, X., Miret, S., and Tang, J. Protst: multi-modality learning of protein sequences and biomedical texts. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[59] Yan, J., Zheng, B., Xu, H., Zhu, Y., Chen, D., Sun, J., Wu, J., and Chen, J. Making pre-trained language models great on tabular prediction. *arXiv preprint arXiv:2403.01841*, 2024.

[60] Yang, L., Liu, S., and Osher, S. J. Fine-tune language models as multi-modal differential equation solvers. *arXiv preprint arXiv:2308.05061v4*, 2024.

[61] Ye, Z., Huang, X., Chen, L., Liu, H., Wang, Z., and Dong, B. Pdeformer: Towards a foundation model for one-dimensional partial differential equations. *arXiv preprint arXiv:2402.12652*, 2024.

[62] Zhang, L., Plathottam, S., Reyna, J., Merket, N., Sayers, K., Yang, X., Reynolds, M., Parker, A., Wilson, E., Fontanini, A., Roberts, D., and Muehleisen, R. High-resolution hourly surrogate modeling framework for physics-based large-scale building stock modeling. 75:103292, 2021. ISSN 22106707. doi: 10.1016/j.scs.2021.103292. URL `https://linkinghub.elsevier.com/retrieve/pii/S2210670721005680`.

[63] Zhou, Y., Yang, J., Zou, H., and Xie, L. Tent: Connect language models with iot sensors for zero-shot activity recognition. *arXiv preprint arXiv:2311.08245*, 2023.

A SysCaps Prompt Details

Prompt design: We append a carefully written instruction template to a list of system attributes to help guide `llama-2-7b-chat` in generating a caption via prompting (see Figure 6 for an example). The system prompt is: *You are a <CES> expert who provides <CES> descriptions <STYLE>.* The user prompt is: *Write a <CES> description based on the following attributes. Your answer should be <NUM> sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.* The last part is added to discourage the LLM from changing or omitting attributes. The tags <CES>, <STYLE>, <NUM> are filled in with the CES type (e.g., *buildings*), the style of the description (e.g, *with an objective tone*), and the number of sentences to use in the description (e.g., "4-6"), respectively.

Attribute subset selection: Simulations of real-world systems may have attributes that only weakly correlate with the output quantity of interest, or have a large number of attributes, which can be challenging for deep learning approaches. Since the length of a SysCap is proportional to the number of attributes, the computational burden incurred by text-based encodings of attributes can grow significantly in these cases. In these cases, reducing the number of attributes can be handled with classic feature selection methods such as recursive feature elimination (RFE) [20] or by recommendations from domain experts, as a pre-processing step.

B Additional Experiment Details

B.1 Metrics

We use the normalized root mean square error (NRMSE) to capture the accuracy of the surrogate model. NRMSE is also known as (CV)RMSE.

The **building-hour NRMSE** is where the NRMSE is normalized by building and by hour, where B is the number of buildings in the building stock and T is the number of hours in a year:

$$:= \frac{1}{\frac{1}{BT}\sum y_t^b}\sqrt{\frac{1}{BT}\sum_{b=1,t=1}^{B,T}(y_t^b - \hat{y}_t^b)^2}. \tag{2}$$

The **stock-annual NRMSE**, is where the NRMSE is normalized by the annual stock energy consumption:

$$:= \frac{1}{\sum y_t^b}\sqrt{\left((\sum_{b=1,t=1}^{B,T} y_t^b) - (\sum_{b=1,t=1}^{B,T} \hat{y}^b) \right)^2}. \tag{3}$$

Notice that the square and square root cancel, making the stock-annual NRMSE equivalent to the normalized mean bias error.

We use the AdamW [35] optimizer with β_1 = 0.9, β_2 = 0.98, ϵ = 1e-9, and weight decay of 0.01 for all experiments. The early stopping patience is 50 for all experiments. All models are trained with a single NVIDIA A100-40GB GPU. The longest training runs take 1-2 days and the shortest 2-3 hours.

B.2 Buildings model hyperparameters

There are 13 attributes after RFE, which are one-hot encoded into a 336-dimensional feature vector that gets embedded into 128 dimensions, whereas the text embeddings are 768-dimensional. We concatenate cyclically encoded calendar features to 7 weather variables, creating a 103-dimensional vector [15]. For the one-hot models (Figure 5), this creates a 128 + 103 = 231 dimensional input for the bidirectional sequence encoder, and for the text models, it is a 768 + 103 = 871 dimensional input. See Table 6 for grid search details for all models.

LightGBM: As LightGBM does not support batch training out of the box, the entire training data needs to be loaded into the memory to train a LightGBM model. With the train dataset containing 340k buildings, each with 8759 hours and 347 features, we randomly extract 438 hours per building (which is about 5% of total hours) to limit memory usage. This results in $340,000 \times 438$ hours in total for the train dataset, which consumes about 380 GB of memory when being loaded into a

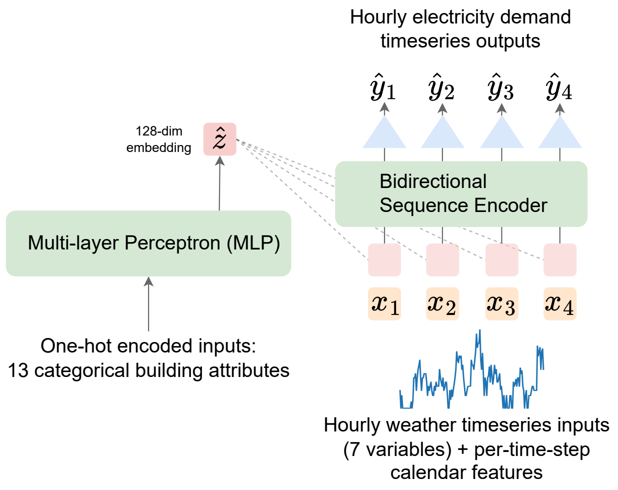

Hourly electricity demand timeseries outputs
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
128-dim embedding
$\hat{z}$
Bidirectional Sequence Encoder
Multi-layer Perceptron (MLP)
x_1 x_2 x_3 x_4
One-hot encoded inputs: 13 categorical building attributes
Hourly weather timeseries inputs (7 variables) + per-time-step calendar features

Figure 5: A visualization of the sequential surrogate model baseline with one-hot encoded system attributes for the buildings experiment.

NumPy object. For the validation and test splits, we retain the full number of hours per building. The LightGBM model is tuned with Optuna [2] across 30 trials and achieves the best validation NRMSE of 0.667.

B.3 Wind farm model hyperparameters

LightGBM: The training, validation, and test split for the wind dataset gives us datasets of size 148,650, 49,600, and 49,250 respectively with 190 features after one-hot encoding. The training dataset is loaded into memory to train the LightGBM model. We use Optuna [2] to tune the hyperparameters and the best validation NMRSE achieved is 0.189.

See Table 7 for grid search details for all models.

C Additional SysCaps examples

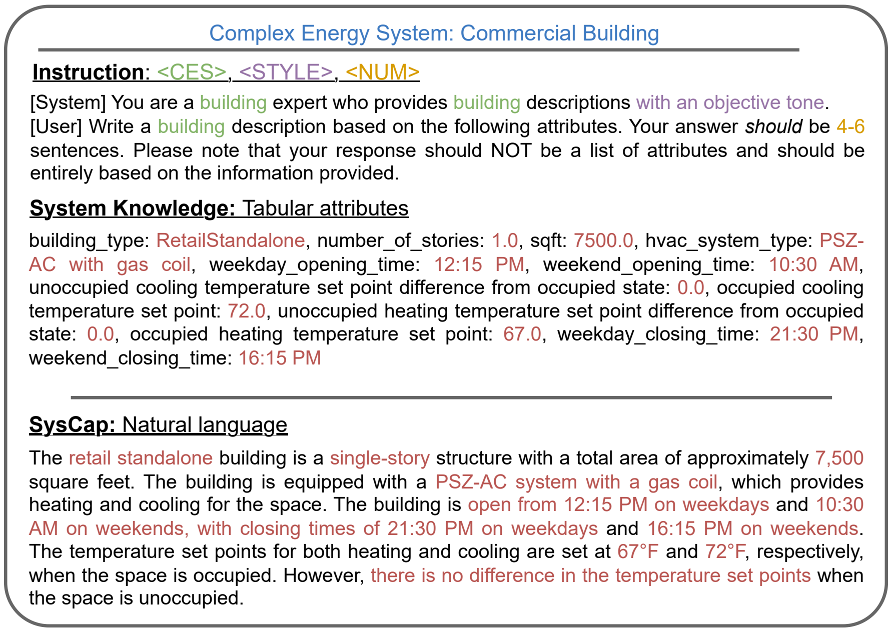
In this section we visualize examples of prompts and the corresponding SysCaps outputs for the buildings simulator (Figures 6, 7, and 8) and the wind farm simulator (Figures 9, 10, and 11).

Table 6: Buildings: hyperparameters.

Model	**Hyperparameter**	**Grid search space**	**Best values**
LightGBM	Learning rate	From 0.01 to 0.1	0.066
	Number of leaves	From 40 to 150	149
	Subsample	From 0.05 to 1.0	0.178
	Feature fraction	From 0.05 to 1.0	0.860
	Min number of data in one leaf	From 1 to 100	12
ResNet (one-hot)	Hidden layers size	256, 1024, 2048	1024
	Number of layers	2, 8	2
	Batch size	128, 256, 512	512
	Learning rate	0.0001, 0.0003, 0.001	0.0003
ResNet (SysCaps)	Hidden layers size	256, 1024, 2048	256
	Number of layers	2, 8	8
	Batch size	128, 256, 512	256
	Learning rate	0.0001, 0.0003, 0.001	0.0003
Bidirectional LSTM (one-hot)	[Hidden layer size, Batch size]	[128, 64], [512, 32], [1024, 32]	[128, 64]
	Number of layers	1, 3, 4, 6, 8	4
	MLP dimension	256	256
	Learning rate	0.00001, 0.0003, 0.001	0.001
Bidirectional LSTM (SysCaps)	[Hidden layer size, Batch size]	[128, 64], [512, 32], [1024, 32]	[512, 32]
	Number of layers	1, 3, 4, 6, 8	1
	MLP dimension	256	256
	Learning rate	0.00001, 0.0003, 0.001	0.0003
Bidirectional S4 (one-hot)	[Hidden layer size, Num. layers]	[64,8] , [128,4]	[128,4]
	MLP dimension	256	256
	Batch size	32, 64	64
	Learning rate	1e-5, 3e-4, 1e-3	3e-4
Bidirectional S4 (SysCaps)	[Hidden layer size, Num. layers]	[64,8] , [128,4]	[128, 4]
	MLP dimension	256	256
	Batch size	32, 64	32
	Learning rate	1e-5, 3e-4, 1e-3	3e-4

Table 7: Wind: hyperparameters.

Model	**Hyperparameter**	**Grid search space**	**Best values**
LightGBM	Learning rate	From 0.01 to 0.1	0.039
	Number of leaves	From 40 to 120	108
	Subsample	From 0.6 to 1.0	0.963
	Feature fraction	From 0.6 to 1.0	0.997
	Min number of data in one leaf	From 20 to 100	96
ResNet (one-hot)	Hidden layers size	[256,1024]	256
	Number of layers	[2,8]	2
	Batch size	[128,256]	128
	Learning rate	[1e-5, 3e-4, 1e-3]	1e-5
ResNet (SysCaps-kv)	Hidden layers size	[256,1024]	1024
	Number of layers	[2,8]	8
	Batch size	[128,256]	256
	Learning rate	[1e-5,3e-4,1e-3]	3e-4
ResNet (SysCaps-nl)	Hidden layers size	[256,1024]	1024
	Number of layers	[2,8]	8
	Batch size	[128,256]	256
	Learning rate	[1e-5,3e-4,1e-3]	1e-5



Complex Energy System: Commercial Building

Instruction: <CES>, <STYLE>, <NUM>

[System] You are a building expert who provides building descriptions with an objective tone.
[User] Write a building description based on the following attributes. Your answer *should* be 4-6 sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.

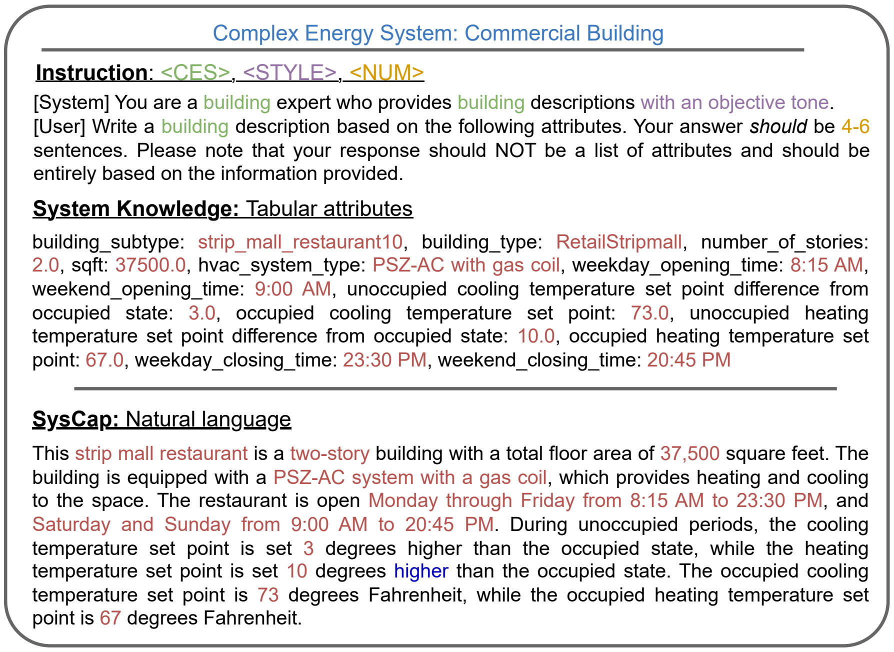
System Knowledge: Tabular attributes

building_type: RetailStandalone, number_of_stories: 1.0, sqft: 7500.0, hvac_system_type: PSZ-AC with gas coil, weekday_opening_time: 12:15 PM, weekend_opening_time: 10:30 AM, unoccupied cooling temperature set point difference from occupied state: 0.0, occupied cooling temperature set point: 72.0, unoccupied heating temperature set point difference from occupied state: 0.0, occupied heating temperature set point: 67.0, weekday_closing_time: 21:30 PM, weekend_closing_time: 16:15 PM

SysCap: Natural language

The retail standalone building is a single-story structure with a total area of approximately 7,500 square feet. The building is equipped with a PSZ-AC system with a gas coil, which provides heating and cooling for the space. The building is open from 12:15 PM on weekdays and 10:30 AM on weekends, with closing times of 21:30 PM on weekdays and 16:15 PM on weekends. The temperature set points for both heating and cooling are set at 67°F and 72°F, respectively, when the space is occupied. However, there is no difference in the temperature set points when the space is unoccupied.

Figure 6: Natural language building SysCap. The instruction and list of key-value attributes (first two paragraphs) are provided to the LLM as the prompt. We observe that the LLM automatically converts numbers to more human-interpretable descriptions (e.g., the number of stories is changed from 1.0 in the prompt to "single-story"). The LLM also succinctly (and correctly) states that "there is no difference in the temperature set points when the space is unoccupied" in the output by summarizing the "unoccupied heating/cooling temperature set point difference from occupied state: 0.0" attributes.


Complex Energy System: Commercial Building

Instruction: <CES>, <STYLE>, <NUM>

[System] You are a building expert who provides building descriptions with an objective tone.
[User] Write a building description based on the following attributes. Your answer *should* be 4-6 sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.

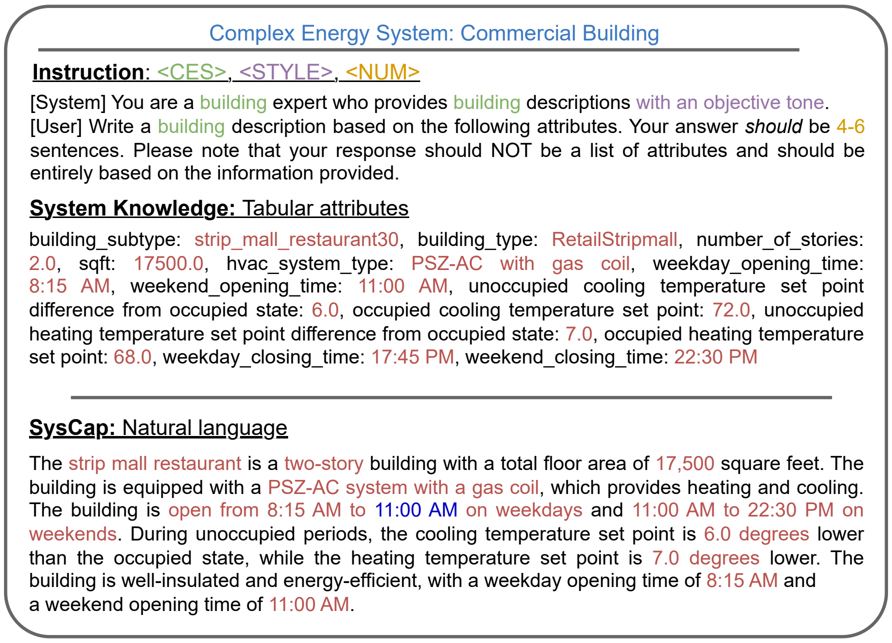
System Knowledge: Tabular attributes

building_subtype: strip_mall_restaurant10, building_type: RetailStripmall, number_of_stories: 2.0, sqft: 37500.0, hvac_system_type: PSZ-AC with gas coil, weekday_opening_time: 8:15 AM, weekend_opening_time: 9:00 AM, unoccupied cooling temperature set point difference from occupied state: 3.0, occupied cooling temperature set point: 73.0, unoccupied heating temperature set point difference from occupied state: 10.0, occupied heating temperature set point: 67.0, weekday_closing_time: 23:30 PM, weekend_closing_time: 20:45 PM

SysCap: Natural language

This strip mall restaurant is a two-story building with a total floor area of 37,500 square feet. The building is equipped with a PSZ-AC system with a gas coil, which provides heating and cooling to the space. The restaurant is open Monday through Friday from 8:15 AM to 23:30 PM, and Saturday and Sunday from 9:00 AM to 20:45 PM. During unoccupied periods, the cooling temperature set point is set 3 degrees higher than the occupied state, while the heating temperature set point is set 10 degrees higher than the occupied state. The occupied cooling temperature set point is 73 degrees Fahrenheit, while the occupied heating temperature set point is 67 degrees Fahrenheit.

Figure 7: Natural language building SysCap. The LLM makes a subtle logical error here—the unoccupied heating temperature set point difference from occupied state suggests the set point should be *lower* by 10 degrees, not higher (highlighted in blue). Logical errors such as this may have contributed to the slightly worse accuracy of natural language SysCaps compared to key-value SysCaps. Collaborating with domain experts able to validate the LLM's outputs is important for catching these errors. We expect that using more powerful LLMs will help reduce logical errors.


Complex Energy System: Commercial Building

Instruction: <CES>, <STYLE>, <NUM>

[System] You are a building expert who provides building descriptions with an objective tone.
[User] Write a building description based on the following attributes. Your answer *should* be 4-6 sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.

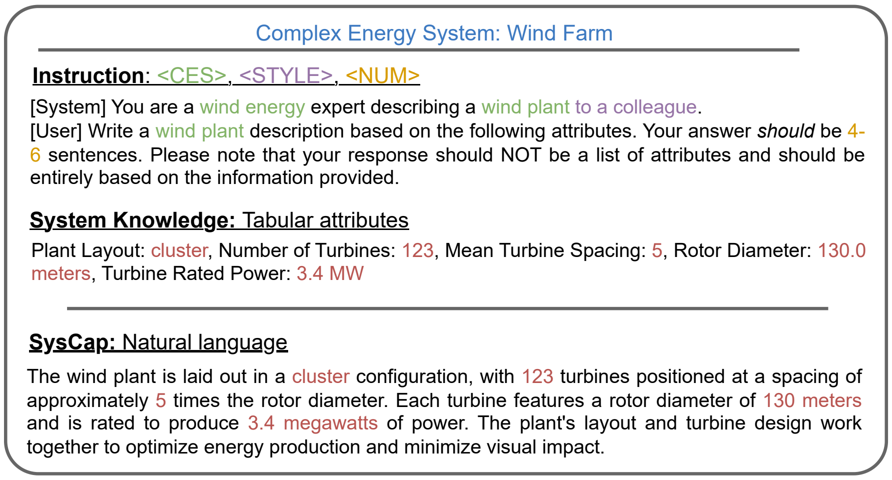
System Knowledge: Tabular attributes

building_subtype: strip_mall_restaurant30, building_type: RetailStripmall, number_of_stories: 2.0, sqft: 17500.0, hvac_system_type: PSZ-AC with gas coil, weekday_opening_time: 8:15 AM, weekend_opening_time: 11:00 AM, unoccupied cooling temperature set point difference from occupied state: 6.0, occupied cooling temperature set point: 72.0, unoccupied heating temperature set point difference from occupied state: 7.0, occupied heating temperature set point: 68.0, weekday_closing_time: 17:45 PM, weekend_closing_time: 22:30 PM

SysCap: Natural language

The strip mall restaurant is a two-story building with a total floor area of 17,500 square feet. The building is equipped with a PSZ-AC system with a gas coil, which provides heating and cooling. The building is open from 8:15 AM to 11:00 AM on weekdays and 11:00 AM to 22:30 PM on weekends. During unoccupied periods, the cooling temperature set point is 6.0 degrees lower than the occupied state, while the heating temperature set point is 7.0 degrees lower. The building is well-insulated and energy-efficient, with a weekday opening time of 8:15 AM and a weekend opening time of 11:00 AM.

Figure 8: Natural language building SysCap. The LLM confuses the weekday closing time with the weekend open time (highlighted in blue).


Complex Energy System: Wind Farm

Instruction: <CES>, <STYLE>, <NUM>

[System] You are a wind energy expert describing a wind plant to a colleague.
[User] Write a wind plant description based on the following attributes. Your answer *should* be 4-6 sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.

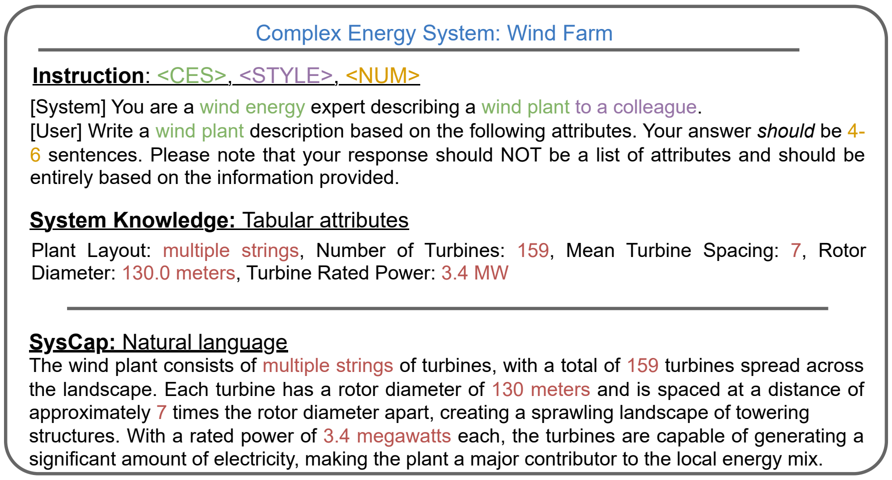
System Knowledge: Tabular attributes

Plant Layout: cluster, Number of Turbines: 123, Mean Turbine Spacing: 5, Rotor Diameter: 130.0 meters, Turbine Rated Power: 3.4 MW

SysCap: Natural language

The wind plant is laid out in a cluster configuration, with 123 turbines positioned at a spacing of approximately 5 times the rotor diameter. Each turbine features a rotor diameter of 130 meters and is rated to produce 3.4 megawatts of power. The plant's layout and turbine design work together to optimize energy production and minimize visual impact.

Figure 9: Natural language wind farm SysCap.


Complex Energy System: Wind Farm

Instruction: <CES>, <STYLE>, <NUM>

[System] You are a wind energy expert describing a wind plant to a colleague.
[User] Write a wind plant description based on the following attributes. Your answer *should* be 4-6 sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.

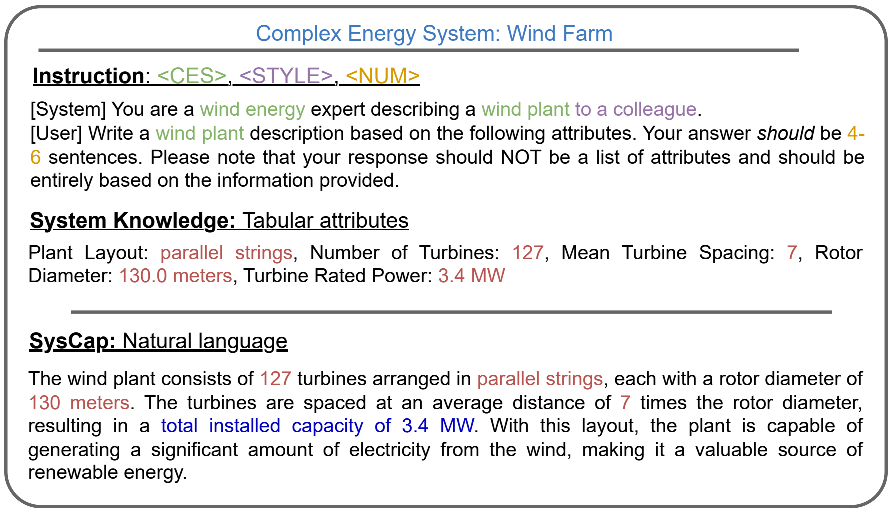
System Knowledge: Tabular attributes

Plant Layout: multiple strings, Number of Turbines: 159, Mean Turbine Spacing: 7, Rotor Diameter: 130.0 meters, Turbine Rated Power: 3.4 MW

SysCap: Natural language

The wind plant consists of multiple strings of turbines, with a total of 159 turbines spread across the landscape. Each turbine has a rotor diameter of 130 meters and is spaced at a distance of approximately 7 times the rotor diameter apart, creating a sprawling landscape of towering structures. With a rated power of 3.4 megawatts each, the turbines are capable of generating a significant amount of electricity, making the plant a major contributor to the local energy mix.

Figure 10: Natural language wind farm SysCap.



Complex Energy System: Wind Farm

Instruction: <CES>, <STYLE>, <NUM>

[System] You are a wind energy expert describing a wind plant to a colleague.
[User] Write a wind plant description based on the following attributes. Your answer *should* be 4-6 sentences. Please note that your response should NOT be a list of attributes and should be entirely based on the information provided.

System Knowledge: Tabular attributes

Plant Layout: parallel strings, Number of Turbines: 127, Mean Turbine Spacing: 7, Rotor Diameter: 130.0 meters, Turbine Rated Power: 3.4 MW

SysCap: Natural language

The wind plant consists of 127 turbines arranged in parallel strings, each with a rotor diameter of 130 meters. The turbines are spaced at an average distance of 7 times the rotor diameter, resulting in a total installed capacity of 3.4 MW. With this layout, the plant is capable of generating a significant amount of electricity from the wind, making it a valuable source of renewable energy.

Figure 11: Natural language wind farm SysCap with a logical error where it says "total installed capacity", but it just states the capacity of a single turbine (highlighted in blue). The correct total installed capacity is number of turbines times 3.4, or 431.8 MW.