The Case for Developing a Foundation Model for Planning-like Tasks from Scratch

Biplav Srivastava, Vishal Pallagani

Artificial Intelligence Institute, University of South Carolina
{biplav.s@, vishalp@mailbox.}sc.edu

Abstract

Foundation Models (FMs) have revolutionized many areas of computing including Automated Planning and Scheduling (APS). For example, a recent study found them to be useful for eight aspects of planning problems: plan generation, language translation, model construction, multi-agent planning, interactive planning, heuristics optimization, tool integration, and brain- inspired planning. Besides APS, there are a number of seemingly related tasks involving generation of a series of actions with varying guarantee of their executability to achieve intended goals, that we collectively call *planning-like* (PL) tasks like business processes, programs, workflows, and guidelines, where researchers have considered using FMs. However, previous works have primarily focused on pre-trained, off-the-shelf FMs and optionally fine-tuned them. In this paper, we discuss the need for a comprehensive FM for PL tasks from scratch and explore the design considerations for it. We argue that such a FM will open new and efficient avenues for PL problem solving just like LLMs are creating for APS.

Introduction

Foundation Models (FMs (Uncbiag 2024)), or Large Language Models (LLMs (Hannibal046 2024)) when referring to them in textual format, have revolutionized many areas of computing. We will use the terms FM and LLM interchangeably to refer to models like GPT (OpenAI 2023), PaLM (Chowdhery et al. 2022), LLaMA (Touvron et al. 2023) and ImageBind (Girdhar et al. 2023). These are deep-learning based models trained using large datasets on a set of tasks (see Figure 1), and optionally fine-tuned using datasets of a particular domain. An FM model so trained can be used for a variety of tasks including conversation (chat), i.e., dialog management. ChatGPT and Bard are examples of such chatbots. When testing such a model, neither the training procedure, consisting of data or tasks, may be known nor the reason why the chatbot generated an utterance - thus, they are truly *blackbox* systems.

Automated Planning and Scheduling (APS) is a branch of Artificial Intelligence (AI) that focuses on the creation of action sequences, also called plans or policies, for execution by intelligent agents. Besides APS, a number of other tasks involve generating a series of actions with varying guarantee for execution semantics, that we collectively call *planning-like* (PL) tasks including business processes (OMG 2011), dialogs (Vlad Serban et al. 2015), guidelines (Field, Lohr et al. 1990), instructions, design diagrams (Wu, Xiao, and Zheng 2021), programs (Kernighan and Ritchie 1988; Ziegler et al. 2024), workflows (van der Aalst and van Hee 2004). For example, a recent instruction generation task for travel planning (Xie et al. 2024) was introduce in the Natural Language Processing (NLP) community that can be processed by humans. But the application scenario of booking travel packages in a travel agency was standardized as back as in 2002 [1] (Figure 2) and spans from a simple, closed-world travel example into a dynamic, integrated solution.

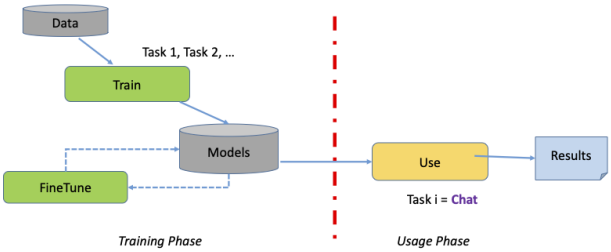

Figure 1: Overview of developing an FM-based system - e.g., an LLM-based chatbot.

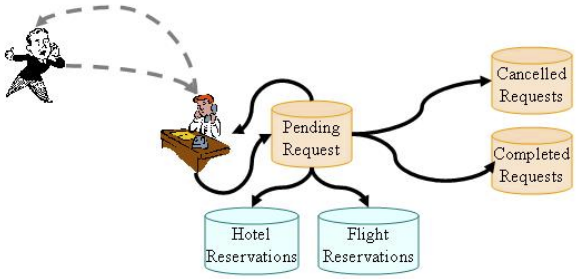

Figure 2: The travel agency example from (Srivastava and Koehler 2003)

In the *closed-world* case, a customer talks to the travel agent who notes the customer's requests and generates a *trip request* document that may contain several needed *flight* and *hotel reservations*. The travel agent performs all bookings

[1] See http://www.w3.org/2002/04/17-ws-usecase.

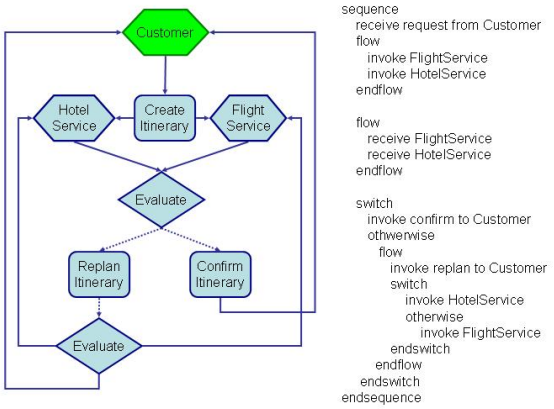

Figure 3: The travel plan in business process notations, from (Srivastava and Koehler 2003)

and when he is done, he puts the *trip request* either into the *cancelled requests* or the *completed requests* data base. A completed document is sent to the customer as an answer to his request. If the booking fails, the customer is contacted again and the whole process re-iterates. Now if we allow the travel agency to cooperate with external specialized service providers that offer hotel and flight reservations, the process requires to reorganize the entire processing of customer requests. In the *open-world* case now, new services have to be integrated and all services must correctly interact with each other. Different settings of travel planning were popularly tackled in APS (Agarwal et al. 2008), business processes literature (see Figure 3), and web services (Srivastava and Koehler 2003).

Researchers have started considering FMs for PL tasks extensively. In APS, a recent study found LLMs to be relevant for various aspects of planning problems: plan generation, language translation, model construction, multi-agent planning, interactive planning, heuristics optimization, tool integration, and brain-inspired planning (Pallagani et al. 2024b). Their use has also lead to many ongoing debates. Consider plan generation. Although there is growing consensus that LLMs cannot reason but act as approximate information retrievers (Valmeekam et al. 2023b,a, 2022; Valmeekam, Marquez, and Kambhampati 2023), they can still be useful to find plans under various LLM/ fine tuning settings (Pallagani et al. 2022a, 2023b,a). Furthermore, they can be productively used in conjunction with conventional planners for effective plan generation (Fabiano et al. 2023).

However, most of the previous works have used pretrained, off-the-shelf FMs and optionally fine-tuned them. In this paper, we discuss the need for a comprehensive FM for PL tasks from scratch and explore the design considerations for it. We argue that such a FM will open new and efficient avenues for PL problem solving just like LLMs are creating for APS. We begin by providing background on FMs, APS and planning-like tasks. Then, we outline the desiderata for a Planning FM and its benefits over current models. We then discuss design considerations, data choices and training options, and conclude.

Background

In this section, we give relevant background for classical planning problem in APS, PL tasks, FMs and LLMs, and LLM training so that the motivation and design options for a ground-up FM for PL is well contextualized.

Automated Planning and Scheduling

The simplest APS task is a classical planning problem (CPP). A CPP is a tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$ - where F is a set of fluents that define a state $s \subseteq F$, and A is a set of actions - and initial and goal states $\mathcal{I}, \mathcal{G} \subseteq F$ (Russell and Norvig 2003). Action $a \in A$ is a tuple $(c_a, \mathit{pre}(a), \mathit{eff}^{\pm}(a))$ where c_a is the cost, and $\mathit{pre}(a), \mathit{eff}^{\pm}(a) \subseteq F$ are the preconditions and add/delete effects, i.e., $\delta_{\mathcal{M}}(s, a) \models \perp s$ *if* $s \not\models \mathit{pre}(a)$; *else* $\delta_{\mathcal{M}}(s, a) \models s \cup \text{eff}^{+}(a) \setminus \text{eff}^{-}(a)$ where $\delta_{\mathcal{M}}(\cdot)$ is the transition function. The cumulative transition function is $\delta_{\mathcal{M}}(s, (a_1, a_2, \ldots, a_n)) = \delta_{\mathcal{M}}(\delta_{\mathcal{M}}(s, a_1), (a_2, \ldots, a_n))$. A plan for a CPP is a sequence of actions $\langle a_1, a_2, \ldots, a_n \rangle$ that transforms the initial state $\mathcal{I}$ into the goal state $\mathcal{G}$ using the transition function $\delta_{\mathcal{M}}$. Traditionally, a CPP is encoded using a symbolic representation, where states, actions, and transitions are explicitly enumerated. This symbolic approach, often implemented using Planning Domain Definition Language or PDDL (McDermott et al. 1998), ensures precise and unambiguous descriptions of planning problems. This formalism allows for applying search algorithms and heuristic methods to find a sequence of actions that lead to the goal state, which is the essence of the plan.

Planning Like Tasks

We now clarify terminology for prominent tasks that bear resemblance to planning and we seek to include them in the proposed FM. They are summarized in Table 1.

The most general representation is a **workflow** which is a formalized representation of activities involved in accomplishing a well-defined objective using automated and manual actions consisting of control and/or data flow (van der Aalst and van Hee 2004). So, workflows can represent manufacturing operations (**design**), machine learning pipelines, complex operations in a financial institution, **dialogs** generated by chatbots, or scripts written for automation. In the context of *automated* action execution, they are referred to as **plans** or **policies** in AI community or **programs** in the software engineering community, and in the context of business improvement, they are also called **business processes**. When tasks are described for managing or communicating with people, they are also represented as **instructions** or **guidelines**.

Thus, in summary:

- Business Processes: a set of coordinated activities described within a business context across organizations - e.g., customers and vendors (OMG 2011; Srivastava and Mazzoleni 2010a,b).

Table 1: Characterizing Select Planning Like Tasks. Auto Gen: Automatic Generation, Auto Exec: Automatic Execution

Name	State	Control Flow	Data Flow	Auto Gen	Auto Exec	Comments	Reference
Business Process		x	x			Data, failure, compensation conditions and cross-organization responsibilities are richly represented; execution can be manual	(OMG 2011)
Design drawing		x				Multi-modal data, geometry constraints	(Wu, Xiao, and Zheng 2021)
Dialogs		x		*		Multi-agent conversation with at least one human, *: trust issues are prominent	(Vlad Serban et al. 2015)
Guidelines		x				Focused on normative (domain) preferences and constraints	(Field, Lohr et al. 1990)
Instructions		x				Step-by-step sequence of actions, emphasizing simplicity and clarity for interpretation and action.	(Weitzenhoffer 1974)
Plan	x	x	x	x	x	Full initial state, partial goal state, minimal data representation, execution guarantee (soundness)	(McDermott et al. 1998)
Program		x	x		x	States, including goals, are not formally represented.	(Kernighan and Ritchie 1988)
Web Services		x	x	*	x	Web-based endpoints, *: with semantic descriptions added	(Chinnici et al. 2007; Agarwal et al. 2008)
Workflow		x	x		x	Data, failure conditions and handling, compensations elaborately represented	(van der Aalst and van Hee 2004)

- Design drawings: Computer-aided design (CAD) diagrams represent a sequence of operations for manufacturing new physical items. They can have both text and visual representations and now FMs are being considered for them (Wu, Xiao, and Zheng 2021; Vuruma et al. 2024).
- Dialogs: represent interaction between a human and automated system with the collective goal of solving a problem (Vlad Serban et al. 2015; McTear, Callejas, and Griol 2016), and are a popular task with both FMs (Jalil et al. 2023) and application of planning (Botea et al. 2019; Muise et al. 2019; Pallagani and Srivastava 2021). Trust issues are a key concern with them (Henderson et al. 2018; Srivastava 2021).
- Guidelines: a sequence of expert-defined information consisting of questions and constraints (e.g. clinical practice guidelines) (Field, Lohr et al. 1990).
- Instructions: a sequence of activities a human can follow to achieve a desirable goal (Weitzenhoffer 1974).
- Plans: a set of coordinated activities where states and action models are formally represented, guaranteeing soundness of execution for any plan found.
- Programs: a set of functions implementing activities and executable on a computer (computable platform) (Kernighan and Ritchie 1988); Copilot is a popular FM-based tool that has been shown to improve coding productivity (Ziegler et al. 2024).
- Web Services: a set of functions run on computers connected over a network (e.g., internet) (Srivastava and Koehler 2003; Chinnici et al. 2007).
- Workflows: a structured sequence of activities and tasks involving automated systems and human participants designed to achieve a specific goal or process efficiently.

We observe from Table 1 that control flow is the common information across PL tasks. However, since other information may be missing, the workflows - referring to the PL outputs collectively - may be acceptable for humans but not for automation. Ensuring automated systems precisely interpret and execute complex, multi-step tasks requires understanding task-specific actions and contextualizing these within the physical world (**grounding**). It also demands alignment with objectives and adaptability to various contexts (**alignment**),

ensuring coherent and efficient progression towards goals. Additionally, the ability to dynamically adapt to new instructions in changing environments (**instructability**) is crucial, necessitating advanced capabilities to bridge the gap between formal task representations and real-world execution. While intrinsic to APS, developing these capabilities in FMs is an active research area.

Foundation and Large Language Models

Large language models are neural network models with upwards of $\sim$ 3 billion parameters trained on extremely large corpora of natural language data (trillions of tokens/words). These models are proficient in interpreting, generating, and contextualizing human language, leading to applications ranging from text generation to reasoning tasks. The Transformer (Vaswani et al. 2017) architecture, which is fundamental to advancements in language modeling, has undergone modifications to develop LLMs that are adept at a diverse range of tasks. Three notable architectural variants in language modeling include causal, masked, and sequence-to-sequence models.

Causal Language Modeling (CLMs): CLMs, such as GPT-4 (OpenAI 2023), are designed for tasks where text generation is sequential and dependent on the preceding context. They predict each subsequent word based on the preceding words, modeling the probability of a word sequence in a forward direction. This process is mathematically formulated as shown in Equation 1.

$$P(T) = \prod_{i=1}^{n} P(t_i|t_{<i}) \tag{1}$$

In this formulation, $P(t_i|t_{<i})$ represents the probability of the i-th token given all preceding tokens, $t_{<i}$. This characteristic makes CLMs particularly suitable for applications like content generation, where the flow and coherence of the text in the forward direction are crucial.

Masked Language Modeling (MLMs): Unlike CLMs, MLMs like BERT(Vaswani et al. 2017) are trained to understand the bidirectional context by predicting words randomly masked in a sentence. This approach allows the model to learn both forward and backward dependencies in language structure. The MLM prediction process can be represented as Equation 2.

$$P(T_{\text{masked}}|T_{\text{context}}) = \prod_{i \in M} P(t_i|T_{\text{context}}) \tag{2}$$

Here, T_{masked} is the set of masked tokens in the sentence, T_{context} represents the unmasked part of the sentence, and M is the set of masked positions. MLMs have proven effective in NLP tasks such as sentiment analysis or question answering.

Sequence-to-Sequence (Seq2Seq) Modeling: Seq2Seq models, like T5 (Raffel et al. 2020a), are designed to transform an input sequence into a related output sequence. They are often employed in tasks that require a mapping between different types of sequences, such as language translation or summarization. The Seq2Seq process is formulated as Equation 3.

$$P(T_{\text{output}}|T_{\text{input}}) = \prod_{i=1}^{m} P(t_{\text{output}_i}|T_{\text{input}}, t_{\text{output}_{<i}}) \tag{3}$$

In Equation 3, T_{input} is the input sequence, T_{output} is the output sequence, and $P(t_{\text{output}_i}|T_{\text{input}}, t_{\text{output}_{<i}})$ calculates the probability of generating each token in the output sequence, considering both the input sequence and the preceding tokens in the output sequence.

In addition to their architectural variants, the utility of LLMs is further enhanced by model adaptation strategies. One key strategy is fine-tuning, which involves further training pre-trained LLMs on a smaller, task-specific dataset, thereby adjusting the neural network weights for particular applications. This process is mathematically represented in Equation 4.

$$\theta_{\text{fine-tuned}} = \theta_{\text{pre-trained}} - \eta \cdot \nabla_\theta L(\theta, D_{\text{task}}) \tag{4}$$

Here, $\theta_{\text{fine-tuned}}$ are the model parameters after fine-tuning, $\theta_{\text{pre-trained}}$ are the parameters obtained from pre-training, η is the learning rate, and $\nabla_\theta L(\theta, D_{\text{task}})$ denotes the gradient of the loss function L with respect to the parameters θ on the task-specific dataset D_{task}.

In contrast to fine-tuning, in-context learning offers a distinct adaptation strategy for LLMs, notably seen in CLMs. This method allows models to tailor responses to immediate prompts without additional training. Given a context C, the model generates text T that is contextually relevant, as shown in Equation 5. Here, $P(T|C)$ is the probability of generating text T given the context C, and $P(t_i|t_{<i}, C)$ is the probability of generating the i-th token t_i given the preceding tokens $t_{<i}$ and the context C.

$$P(T|C) = \prod_{i=1}^{n} P(t_i|t_{<i}, C) \tag{5}$$

Which model to start with for plan generation? (Pallagani et al. 2023a) identify that pre-trained LLMs cannot generate plans for classical planning problems off the shelf. The selection of pre-training data and the methodology employed for model adaptation play pivotal roles in improving the quality of plan generation. It is observed that LLMs pre-trained on programming languages, as opposed to solely natural language, when fine-tuned with incrementally hard planning problems (Pallagani et al. 2022b), exhibit enhanced capabilities in generating optimal plans for domains in which they have been fine-tuned. Despite these advancements, the fine-tuned models demonstrate a limitation in their ability to generalize to domains not included in the fine-tuning process.

Training LLMs

There are tutorials on training CLMs, Masked, and Seq2Seq LLMs for NLP from scratch, including (Karpathy 2023) and (HuggingFace 2024). Training LLMs requires a vast and high-quality dataset, as the data's quality directly influences model performance. With adequate data, LLMs can achieve near-human proficiency across various domains (Gunasekar

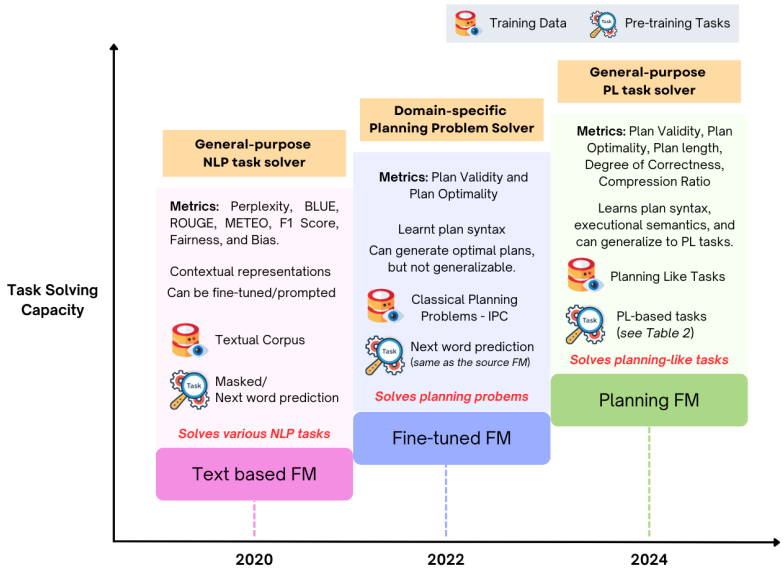

Figure 4: A timeline capturing the progression of FMs for planning and the need for designing a Planning FM from ground up for solving PL tasks.

et al. 2023). Training data typically involves diverse sources such as Books, CommonCrawl, Reddit feeds, Wikipedia, and Code. While next-word prediction remains the primary pre-training task for LLMs, there is a shift towards incorporating domain-specific pre-training tasks. Such tasks, designed for specialized fields like code synthesis and Chess strategy analysis, offer more targeted learning pathways for the models.

For planning, there is a recent tutorial on fine-tuning LLMs (Pallagani et al. 2024a) with planning problems for plan generation. Additionally, there has been progress in adapting the Transformer architecture to A* search dynamics for puzzles like Sokoban (Lehnert et al. 2024). However, a significant gap persists in the availability of a comprehensive training corpus for PL tasks and novel pre-training tasks apt for an FM to learn the execution semantics and the syntactic knowledge intrinsic to a PL task.

The Need for a Planning Foundation Model

Current approaches to FMs in NLP and AI research have predominantly centered around generic pre-training tasks such as masked or next-word/sentence prediction, as shown in Figure 4. These models are trained on diverse text corpora, enabling them to perform various downstream tasks, from sentiment detection to question answering. However, when these models are applied to PL tasks, their effectiveness is inherently limited by their generic training paradigms. This limitation is due to the complex requirements of PL tasks, which not only necessitate the understanding of textual information but also require an intricate capture of state, control flow, and data flow alongside the constrained generation and execution semantics unique to each task (see Table 1). Current pre-training tasks lack the specificity to model these detailed, dynamic relationships inherent in PL tasks, leading to a gap in effectiveness when transitioning from general text understanding to specialized plan generation and execution.

The finetuning of FMs, exemplified by models like Plansformer, represents a step towards domain-specific adaptation, where generic FMs are trained on PL tasks. While this method offers improvements in performing PL tasks by leveraging domain-specific data, it still falls short in fully capturing the nuances of plan generation, validation, and generalizing across all PL tasks due to the limitation in training data and the inherent general-purpose pre-training. Prompting is another approach that has gained popularity recently. Despite its wide adoption, it has demonstrated inferior plan generation performance compared to fine-tuning and often relies heavily on the human-in-the-loop's skill in crafting effective prompts.

The need for a Planning FM arises from these limitations and new opportunities. Note from Table 1 that PL tasks have a complementary focus, which may be leveraged with an FM. For example, business processes and dialogs focus on multi-party interaction, design focuses on geometry; plans focus on soundness, and workflows and business processes focus on failure handling. An FM trained on diverse considerations may bring these insights learned from data to bear in its output. It can also focus on PL-specific tasks during both pre-training and downstream and metrics to enable better execution of semantics associated with PL tasks. Such an FM may be critical for effectively generating, summarizing, and generalizing PL tasks. We note that in the past, in code generation, FMs trained for that objective (Wang et al. 2021) performed better than fine-tuning text-based FMs (Raffel et al. 2020b) with code - a trend seen in many other areas.

Table 2: Novel Pre-training Tasks for PL Tasks

Pre-training Task	Description	Relevance
Next Action Prediction	Training the model to make context-based decisions, simulating step-by-step plan execution.	Mimics real-world planning and workflow execution.
Conditional Branching Prediction	Predicting outcomes from multiple possible branches in a given scenario.	Reflects decision-making in complex processes.
Action and Effect Modeling	Understanding causal relationships between actions and their consequences.	Fundamental for realistic plan generation.
Constraint Satisfaction	Identifying and applying constraints to optimize outcomes.	Crucial for generating efficient and viable plans.
Hierarchical Task Planning	Generating plans involving tasks at multiple levels of abstraction.	Addresses complex tasks with sub-tasks in detailed planning.
Cross-Domain Understanding	Recognizing entities and relationships across different planning domains.	Enhances model's versatility, applicability, and generalizability.
Execution Simulation	Simulating the execution of plans or code to predict outcomes.	Improves the model's execution semantics understanding.
Error Detection	Identifying and correcting errors in plans or instructions.	Enhances reliability and correctness of outputs.
Multi-modal Contrastive Learning	Incorporating visual, textual, and auditory data for plan understanding and generation.	Relevant for tasks relying on multi-modal inputs.

Discussion

We now discuss how the proposed FM may be developed. The envisioned Planning FM distinguishes itself through a specialized focus on PL tasks and its design principles of compactness, generalizability, and an intrinsic awareness of temporal and execution semantics.

Training Procedure

We provide a detailed description of the comprehensive training procedure involved in constructing the Planning FM, encompassing critical aspects from tokenization and model architecture to the application of the Planning FM in various downstream tasks.

Tokenization and Embeddings: Designing a tokenizer for the Planning FM, which is specialized for handling PL tasks requires a nuanced approach to understand and encode diverse data types and structures. Such a tokenizer can be designed as follows:

- **Pre-processing and Normalization**
  - Input Normalization: The initial stage involves standardizing heterogeneous inputs, including textual data from various PL task documents and non-textual data such as design drawings. Textual data undergo normalization processes, including case normalization, special character removal, and spelling correction. Non-textual data is processed through Optical Character Recognition (OCR) techniques to extract textual information, ensuring a unified text-based input for further processing.
  - Semantic Augmentation: Employing domain-specific ontologies such as the Planning Ontology, the text is augmented with semantic annotations. This involves the identification and annotation of domain-relevant entities and their interrelations, enhancing the model's comprehension of planning-specific lexicon and structures.
- **Tokenization Strategy**
  - Adaptive Subword Tokenization: A subword tokenization algorithm, tailored to the planning domain through the training on the pre-processed and normalized corpus, is employed. Techniques such as Byte-Pair Encoding (BPE), Unigram Language Model, or SentencePiece can be considered, with the aim to capture the lexical characteristics of PL tasks.
  - Multi-Modal Tokenization: For incorporating non-textual information, a multi-model tokenization approach is adopted. This involves the extraction of visual features using Convolutional Neural Networks (CNNs), subsequently mapped into a discrete token space compatible with textual tokens, facilitating integrated multi-modal analysis.
- **Special Tokens and Embeddings**
  - Integration of Domain-Specific Tokens: The tokenizer incorporates special tokens designed to signify critical PL constructs (e.g., START_OF_PLAN, END_OF_TASK) and entities (e.g., OBJECTS, COST). These tokens are crucial for the model to recognize and prioritize PL task elements within the

data.

– Enhanced Embedding Layer: The embedding layer for the Planning FM incorporates rotary position embeddings (RoPE)(Su et al. 2024) to effectively encode sequential information in PL tasks. RoPE distinguishes itself by directly encoding positional information into the embeddings, allowing the model to maintain the relative order of tokens without losing the context of their placement.

This approach to tokenizer for Planning FM provides for a blend of linguistic processing, domain-specific tailoring, and multi-modal integration, enabling the model to be adept at PL tasks.

Model Architecture A Seq2Seq architecture will be the preferred choice for the Planning FM, as prior research has demonstrated its capabilities in generating structured sequences compared to CLMs or MLMs. The Seq2Seq framework, originally devised for machine translation, excels at comprehending and producing complex sequences, making it an optimal choice for PL tasks.

The Seq2Seq architecture comprises two principal components: an encoder and a decoder. The encoder processes the input sequence, capturing its semantic and structural essence into a comprehensive context vector. For PL tasks, this involves encoding the initial state, objectives, constraints, and available resources. The inclusion of a self-attention mechanism enables the encoder to assess the relative importance of different elements within the input sequence, a critical function for understanding complex planning instructions and dependencies.

The decoder is responsible for generating the output sequences, such as plans or summaries, from the encoded representation. It predicts the subsequent action or step in the plan sequentially, considering both the current generation and the overarching goal articulated by the encoder. The decoder employs a cross-attention mechanism, particularly allowing it to concentrate on pertinent portions of the input sequence while formulating each step of the plan. A significant advancement in the Seq2Seq architecture for the Planning FM will be the incorporation of RoPE. RoPE enhances the Planning FM's ability to maintain and exploit the temporal and sequential dependencies more effectively than the traditional positional encoding methods utilized in modern LLMs.

Pre-training Objectives The novel pre-training objectives for the Planning FM are outlined in Table 2. We depart from conventional next-word prediction objective predominantly used to train LLMs to objectives that are inherently aligned with the intricacies of PL tasks. Our proposed pre-training objectives encompass tasks such as Next Action Prediction, Conditional Branching Prediction, and Action-Effect Modeling, equipping the Planning FM to learn context-based decision-making, understanding complex process dynamics, and modeling causal relationships between actions and their consequences. Furthermore, tasks like Constraint Satisfaction and Hierarchical Task Planning target the generation of optimal and executable plans by navigating constraints and orchestrating tasks across multiple levels of abstraction. The inclusion of Cross-Domain Understanding and Execution Simulation extends the model's versatility and applicability across various PL scenarios, enhancing its predictive accuracy and execution semantics. Additionally, Error Detection and Multi-modal Contrastive Learning further refine the model's reliability and adaptability to multi-modal inputs, ensuring the generation of robust and error-free plans.

Evaluation The training regimen of the Planning FM, featuring novel pre-training tasks specifically devised for PL tasks, necessitates the introduction of sophisticated evaluation metrics to accurately gauge its efficacy. These metrics aim to capture essential dimensions of the model's output, such as *Plan Validity*, which scrutinizes the feasibility and compliance of the generated plans with predefined constraints and objectives. *Plan Optimality* assesses the efficacy and resource efficiency of the plans, ensuring goals are met with minimal expenditure of time and resources. *Plan Length*, defined as a numerical value representing the number of actions or steps in a generated plan. *Degree of Correctness* measures the ratio of successfully achieved goals to the number of specified goals within a plan. Notably, *Compression Ratio* is introduced as a metric to assess the model's efficiency in condensing detailed planning instructions into succinct, actionable summaries without losing critical information. This suite of metrics offer a comprehensive framework for assessing the Planning FM's performance, ensuring it delivers practical, optimal, and executable plans suited to a wide array of real-world planning scenarios.

Downstream Tasks The Planning FM is designed for a variety of downstream tasks, reflecting its utility in planning-related applications. Key downstream tasks include:

- Plan Generation: Creating detailed plans based on specific goals and constraints.
- Completing a Partial Plan: Filling in missing elements of an incomplete plan to ensure its comprehensiveness and actionability.
- Replanning: Adjusting plans in response to changes in objectives, constraints, or conditions.
- Predicting Plan Validity: Determining the feasibility and coherence of plans, identifying potential execution issues.
- Plan Summarization: Condensing comprehensive plans into brief summaries that retain essential information (Srivastava 2010).
- Resource Optimization: Ensuring efficient use of resources within plans to achieve objectives with minimal waste.
- Error Detection and Correction: Identifying and correcting inaccuracies or inconsistencies within plans.

The model can be further adapted to any downstream PL task by fine-tuning or prompting the Planning FM with with an appropriate quantity and quality of data.

Model Properties

We envisage the proposed FM to be compact, general and aware of PL needs. We now describe specific steps towards the same.

Compactness: Conventional FMs are assumed to be usable in a resource-rich setting like residing remotely on a cloud and invoked on-demand via API calls. However, many real world applications are resource-constrained and we there is a rising demand for compact models that can be deployed effectively in both cloud as well as edge settings (Vuruma et al. 2024) without compromising performance.

Prominent techniques for these are:

- **Model Pruning:** The process of removing non-critical and redundant components of a model without a significant loss in performance. With respect to LLMs, this can mean removing weights with smaller gradients or magnitudes and parameter reduction among others. Novel pruning methods like Wanda (Sun et al. 2023) and LLM-Pruner (Ma, Fang, and Wang 2023) present optimal solutions for making LLMs smaller. Furthermore, parameter sharing across different parts of the model and pruning redundant or non-contributory neurons post-training contribute to a compact yet powerful model architecture.
- **Quantization:** A key strategy is mixed precision training, which utilizes a combination of 16-bit (half-precision) and 32-bit (single precision) floating-point operations during the training process or representing model parameters such as weights in a lower precision, i.e. using fewer bits to store the value (Gholami et al. 2021). This results in a smaller model size, faster inference, and a reduced memory footprint. LLM Quantization can be achieved either in the post-training phase (Dettmers et al. 2022) or during the pre-training or fine-tuning phase (Liu et al. 2023).
- **Knowledge Distillation:** Transferring the knowledge of a large teacher model to a smaller learner model to replicate the original model's output distribution difference. Knowledge Distillation has been widely used to reduce LLMs like BERT into smaller distilled versions DistilBERT (Sanh et al. 2020). More recently, approaches like MiniLLM (Gu et al. 2023) and (Hsieh et al. 2023) further optimize the distillation process to improve the student model's performance and inference speed.

Generalizability: refers to the ability to generate output beyond the specific training data. We seek to attain it across the wide range of PL tasks by training the Planning FM on a diverse corpus that encompasses not just traditional planning datasets like the International Planning Competition (IPC (ICAPS 2023)) but also extends to other tasks like business processes (Camunda 2024), dialogues (Vlad Serban et al. 2015), design (Wu, Xiao, and Zheng 2021), and workflows (Allard et al. 2019) form Table 1. Note that the business process and design data are inherently multi-modal consisting of image and text. It can be further enriched by knowledge from a planning ontology that captures inter-relationships between different metadata (Muppasani et al. 2023). This broad training base, combined with task-agnostic pre-training objectives such as next-action prediction, conditional branching prediction, and execution simulation, equips the Planning FM with a robust foundation of knowledge. The ontological knowledge facilitates the model's ability to adapt to various planning contexts without requiring extensive task-specific retraining. Moreover, the incorporation of transfer learning techniques allows the model to leverage knowledge acquired from one task to improve performance on related tasks, enhancing its generalizability. (Torrey and Shavlik 2010).

Awareness of temporal and execution considerations is cultivated by deliberately designing pre-training tasks that mimic real-planning scenarios as outlined in Table 2. With the help of novel pre-training tasks, the Planning FM develops a nuanced comprehension of how plans unfold over time and how individual actions interrelate. Training the model to recognize and predict the impact of various actions within different execution contexts further ensures that the generated plans are theoretically sound and practically executable.

LLM Properties

The development of Planning FM introduces notable concerns regarding the executional guarantees traditionally offered by APS systems. FMs inherently do not provide sound and complete solutions due to their probabilistic frameworks. Therefore, leveraging FM architectures for PL tasks requires a thorough investigation into the Planning FM's properties of alignment, instructability, and grounding - areas where FMs often encounter limitations.

Grounding: refers to the model's ability to base its planning and reasoning processes on real-world knowledge and data. This necessitates the incorporation of extensive domain-specific knowledge into the model, potentially with the help of a knowledge graph (Sheth et al. 2022). The APS planners may also help in verifying physical constraints just as information retrievers do in RAG (Lewis et al. 2020).

Alignment: is critical for ensuring that the progression of Planning FM towards goals is both coherent and efficient. It involves the model's capability to accurately interpret and execute PL tasks in a manner that not only meets predefined objectives but also dynamically adjusts to changes in the environment.

Instructability: involves the model's responsiveness to nuanced directives and its capacity to adapt its planning strategies accordingly. Enhancing the Planning FM's instructability requires fine-tuning with preferences (Ziegler et al. 2019) that encompass a broad range of PL tasks and user intents.

Conclusion

In this paper, we considered various PL tasks together and argued for the need for a FM to be developed from scratch. We explored various design considerations including training tasks, data and its behavioral properties. We argue that such a FM will open new and efficient avenues for PL problem solving just like LLMs are creating for APS.

Acknowledgements

We wish to thank Amitava Das, Kaushik Roy, Manas Gaur, Jianhai Su and Sai Vuruma for discussions related to foundation models and NLP, and Faez Ahmed on the role of FM for design. This work was partially funded with a gift from JPMorgan Chase Faculty Research Award.

References

Agarwal, V.; Chafle, G.; Mittal, S.; and Srivastava, B. 2008. Understanding approaches for web service composition and execution. In *Proceedings of the 1st Bangalore annual Compute conference*, 1–8.

Allard, T.; Alvino, P.; Shing, L.; Wollaber, A.; and Yuen, J. 2019. A dataset to facilitate automated workflow analysis. In *PLoS ONE 14(2): e0211486. https://doi.org/10.1371/journal.pone.0211486*.

Botea, A.; Muise, C.; Agarwal, S.; Alkan, O.; Bajgar, O.; Daly, E.; Kishimoto, A.; Lastras, L.; Marinescu, R.; Ondrej, J.; Pedemonte, P.; and Vodolan, M. 2019. Generating Dialogue Agents via Automated Planning. In *https://arxiv.org/abs/1902.00771*.

Camunda. 2024. BPMN for research. In *https://github.com/camunda/bpmn-for-research*.

Chinnici, R.; Moreau, J.-J.; Ryman, A.; and Weerawarana, S. 2007. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. World Wide Web Consortium, Recommendation REC-wsdl20-20070626.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; Schuh, P.; Shi, K.; Tsvyashchenko, S.; Maynez, J.; Rao, A.; Barnes, P.; Tay, Y.; Shazeer, N.; Prabhakaran, V.; Reif, E.; Du, N.; Hutchinson, B.; Pope, R.; Bradbury, J.; Austin, J.; Isard, M.; Gur-Ari, G.; Yin, P.; Duke, T.; Levskaya, A.; Ghemawat, S.; Dev, S.; Michalewski, H.; Garcia, X.; Misra, V.; Robinson, K.; Fedus, L.; Zhou, D.; Ippolito, D.; Luan, D.; Lim, H.; Zoph, B.; Spiridonov, A.; Sepassi, R.; Dohan, D.; Agrawal, S.; Omernick, M.; Dai, A. M.; Pillai, T. S.; Pellat, M.; Lewkowycz, A.; Moreira, E.; Child, R.; Polozov, O.; Lee, K.; Zhou, Z.; Wang, X.; Saeta, B.; Diaz, M.; Firat, O.; Catasta, M.; Wei, J.; Meier-Hellstern, K.; Eck, D.; Dean, J.; Petrov, S.; and Fiedel, N. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311.

Dettmers, T.; Lewis, M.; Belkada, Y.; and Zettlemoyer, L. 2022. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. arXiv:2208.07339.

Fabiano, F.; Pallagani, V.; Ganapini, M. B.; Horesh, L.; Loreggia, A.; Murugesan, K.; Rossi, F.; and Srivastava, B. 2023. Fast and Slow Planning. *arXiv preprint arXiv:2303.04283*.

Field, M. J.; Lohr, K. N.; et al. 1990. Clinical practice guidelines. *Directions for a new program*, 90(8).

Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M. W.; and Keutzer, K. 2021. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630.

Girdhar, R.; El-Nouby, A.; Liu, Z.; Singh, M.; Alwala, K. V.; Joulin, A.; and Misra, I. 2023. ImageBind: One Embedding Space To Bind Them All. In *CVPR*.

Gu, Y.; Dong, L.; Wei, F.; and Huang, M. 2023. Knowledge Distillation of Large Language Models. arXiv:2306.08543.

Gunasekar, S.; Zhang, Y.; Aneja, J.; Mendes, C. C. T.; Del Giorno, A.; Gopi, S.; Javaheripi, M.; Kauffmann, P.; de Rosa, G.; Saarikivi, O.; et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Hannibal046. 2024. Awesome-LLM: a curated list of Large Language Model. In *https://github.com/Hannibal046/Awesome-LLM*.

Henderson, P.; Sinha, K.; Angelard-Gontier, N.; Ke, N. R.; Fried, G.; Lowe, R.; and Pineau, J. 2018. Ethical Challenges in Data-Driven Dialogue Systems. In *Proc. of AAAI/ACM Conference on AI Ethics and Society (AIES-18), New Orleans, Lousiana, USA*.

Hsieh, C.-Y.; Li, C.-L.; Yeh, C.-K.; Nakhost, H.; Fujii, Y.; Ratner, A.; Krishna, R.; Lee, C.-Y.; and Pfister, T. 2023. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. arXiv:2305.02301.

HuggingFace. 2024. Training a causal language model from scratch. *https://huggingface.co/learn/nlp-course/en/chapter7/6*.

ICAPS. 2023. International Planning Competitions. In *https://www.icaps-conference.org/competitions/*.

Jalil, S.; Rafi, S.; LaToza, T. D.; Moran, K.; and Lam, W. 2023. Chatgpt and software testing education: Promises & perils. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 4130–4137. IEEE.

Karpathy, A. 2023. Let's build GPT: from scratch, in code, spelled out. *https://www.youtube.com/watch?v=kCc8FmEb1nY*.

Kernighan, B. W.; and Ritchie, D. 1988. Reference Manual. In *The C Programming Language*, chapter Appendix A, 191–240. Prentice Hall, second edition.

Lehnert, L.; Sukhbaatar, S.; Mcvay, P.; Rabbat, M.; and Tian, Y. 2024. Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping. *arXiv preprint arXiv:2402.14083*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546.

Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; and Chandra, V. 2023. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. arXiv:2305.17888.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. arXiv:2305.11627.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL-the planning domain definition language.

McTear, M.; Callejas, Z.; and Griol, D. 2016. Conversational Interfaces: Past and Present. In *The Conversational Interface. Springer, DOI: https://doi.org/10.1007/978-3-319-32967-3_4*.

Muise, C.; Chakraborti, T.; Agarwal, S.; Bajgar, O.; Chaudhary, A.; Lastras-Montano, L. A.; Ondrej, J.; Vodolan, M.;

and Wiecha, C. 2019. Planning for Goal-Oriented Dialogue Systems. In *https://arxiv.org/abs/1910.08137*.

Muppasani, B.; Pallagani, V.; Srivastava, B.; and Mutharaju, R. 2023. Building and using a planning ontology from past data for performance efficiency. In *Proc. ICAPS'23 Workshop PLanning And onTology wOrkshop (PLATO)*.

OMG. 2011. Business Process Model and Notation (BPMN), Version 2.0.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Horesh, L.; Srivastava, B.; Fabiano, F.; and Loreggia, A. 2022a. Plansformer: Generating symbolic plans using transformers. *arXiv preprint arXiv:2212.08681*.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Horesh, L.; Srivastava, B.; Fabiano, F.; and Loreggia, A. 2022b. Plansformer: Generating Symbolic Plans using Transformers. In *On Arxiv at: https://arxiv.org/abs/2212.08681*.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Srivastava, B.; Horesh, L.; Fabiano, F.; and Loreggia, A. 2023a. Understanding the Capabilities of Large Language Models for Automated Planning. *arXiv preprint arXiv:2305.16151*.

Pallagani, V.; Muppasani, B.; Srivastava, B.; Rossi, F.; Horesh, L.; Murugesan, K.; Loreggia, A.; Fabiano, F.; Joseph, R.; Kethepalli, Y.; et al. 2023b. Plansformer Tool: Demonstrating Generation of Symbolic Plans Using Transformers. In *IJCAI*, volume 2023, 7158–7162. International Joint Conferences on Artificial Intelligence.

Pallagani, V.; Murugesan, K.; Srivastava, B.; Rossi, F.; and Horesh, L. 2024a. Harnessing Large Language Models for Planning: A Lab on Strategies for Success and Mitigation of Pitfalls. In *AAAI Conference on Artificial Intelligence, https://github.com/VishalPallagani/LLMsforPlanningLab-AAAI24*.

Pallagani, V.; Roy, K.; Muppasani, B.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; and Sheth, A. 2024b. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). In *ICAPS, Calgary, Canada*.

Pallagani, V.; and Srivastava, B. 2021. A Generic Dialog Agent for Information Retrieval Based on Automated Planning Within a Reinforcement Learning Platform.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020a. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Russell, S.; and Norvig, P. 2003. *Artificial Intelligence, A Modern Approach. Second Edition*.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

Sheth, A.; Gaur, M.; Roy, K.; Venkataraman, R.; and Khandelwal, V. 2022. Process knowledge-infused ai: Toward user-level explainability, interpretability, and safety. *IEEE Internet Computing*, 26(5): 76–84.

Srivastava, B. 2010. Processes Summarization. In *Conference on Management of Data (COMAD), India. Code at: https://github.com/biplav-s/processes-summarizer*.

Srivastava, B. 2021. Did chatbots miss their "Apollo Moment"? Potential, gaps, and lessons from using collaboration assistants during COVID-19. In *Patterns, Volume 2, Issue 8, 100308*.

Srivastava, B.; and Koehler, J. 2003. Web service composition-current solutions and open problems. In *ICAPS 2003 workshop on Planning for Web Services*, volume 35, 28–35.

Srivastava, B.; and Mazzoleni, P. 2010a. An APQC-PCF based framework to compare service offerings in business transformation projects. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, 73–78.

Srivastava, B.; and Mazzoleni, P. 2010b. Business Driven Consolidation of SOA Implementations. In *2010 IEEE International Conference on Services Computing*, 49–56. IEEE.

Su, J.; Ahmed, M.; Lu, Y.; Pan, S.; Bo, W.; and Liu, Y. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568: 127063.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A Simple and Effective Pruning Approach for Large Language Models. arXiv:2306.11695.

Torrey, L.; and Shavlik, J. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, 242–264. IGI global.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Uncbiag. 2024. Awesome Foundation Models. In *https://github.com/uncbiag/Awesome-Foundation-Models*.

Valmeekam, K.; Marquez, M.; and Kambhampati, S. 2023. Can Large Language Models Really Improve by Self-critiquing Their Own Plans? *arXiv preprint arXiv:2310.08118*.

Valmeekam, K.; Marquez, M.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2023a. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). *arXiv preprint arXiv:2206.10498*.

Valmeekam, K.; Sreedharan, S.; Marquez, M.; Olmo, A.; and Kambhampati, S. 2023b. On the planning abilities of

large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*.

van der Aalst, W. M.; and van Hee, K. 2004. Workflow Management: Models, Methods, and Systems. In *ISBN:978-0262720465, MIT Press*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vlad Serban, I.; Lowe, R.; Henderson, P.; Charlin, L.; and Pineau, J. 2015. A Survey of Available Corpora for Building Data-Driven Dialogue Systems. *ArXiv e-prints*.

Vuruma, S. K. R.; Margetts, A.; Su, J.; Ahmed, F.; and Srivastava, B. 2024. From Cloud to Edge: Rethinking Generative AI for Low-Resource Design Challenges. arXiv:2402.12702.

Wang, Y.; Wang, W.; Joty, S.; and Hoi, S. C. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.

Weitzenhoffer, A. M. 1974. When is an "instruction" an "instruction"? *International Journal of Clinical and Experimental Hypnosis*, 22(3): 258–269.

Wu, R.; Xiao, C.; and Zheng, C. 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 6772–6782.

Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; and Su, Y. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Ziegler, A.; Kalliamvakou, E.; Li, X. A.; Rice, A.; Rifkin, D.; Simister, S.; Sittampalam, G.; and Aftandilian, E. 2024. Measuring GitHub Copilot's Impact on Productivity. *Commun. ACM*, 67(3): 54–63.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.