

AutoSynth: Natural Language-Driven Pipeline Generation via Context-Aware Large Language Model Orchestration

Pawan Kalyan Jonnalagadda

San Jose, California, USA, 95110, pawankalyanjonnalagadda2@gmail.com

Abstract- The development of the large language models (LLMs) has brought many new opportunities to automate the workflow involving complex computations. Nevertheless, current pipeline systems are still mostly fixed, needing to be configured manually and cannot adapt to dynamic settings. The paper suggests a new framework that is based on prompt-driven pipeline synthesis via context-aware auto-configuration of LLMs, which allows automatic synthesis and optimization of task-specific pipelines based on natural language input. The proposed approach involves a combination of timely engineering, contextually-based learning and sequence improvement to construct and adjust pipelines dynamically based on contextual information. An artificial performance model is used to determine the performance of the system relative to the current models that demonstrate that the accuracy, scalability and adaptability are improved without affecting the competitive latency. Such results demonstrate the utility of deploying LLMs as intelligent agents to optimally design pipelines as a scalable and adaptable solution to problems in the real world.

Keywords- Prompt Engineering, Large Language Models, Pipeline Synthesis, Context-Aware Systems, Auto-Configuration.

I. INTRODUCTION

The blistering development of Large Language Models (LLMs) has changed the manner in which complex computational operations are formulated and executed [1]. Conventional pipeline construction of tasks like data processing, information extraction and decision-making involves a lot of manual configuration, domain knowledge and trial and error tuning [2]. These pipelines are usually hard, domain-specific and hard to adapt to dynamic environments [3]. This leads to an increase in the demand to have smart systems capable of designing, configuring and optimizing pipelines with little human intervention [4]. The recent advances in prompt engineering have shown that LLMs can be trained to complete a broad set of tasks by following natural language instructions [5]. In-Context Learning techniques allow models to be modified to suit new tasks without retraining, and are therefore very flexible and efficient [6]. Moreover, the innovations in automatic prompt optimization and reasoning systems have allowed the use of LLMs as decision-making agents, i.e., the ability to select the right strategies based on the contextual input [7].

Regardless of these improvements, the current methods are mostly concerned with isolated aspects like timely design, task execution, or optimizing the model [8]. Coherent frameworks to combine prompt-based control with dynamic pipeline synthesis and context-aware configuration are lacking [9]. Pipelines are required to change to accommodate different types of data, user needs, and environmental conditions in a number of real-world applications [10]. This requires a system that is capable of creating and adjusting pipelines intelligently depending on the situation, and not the fixed predefined workflows. This paper proposes a new pipeline synthesis model with prompt-based synthesis to address these problems and offers new context-sensitive auto-configurations of pipelines using LLMs [11]. The proposed solution takes the advantage of the dynamic pipeline creation, modification, and optimization capabilities of LLMs, which are applied with assistance of structured prompts and contextual feedback [12]. To enhance the performance and flexibility as well as scalability, the system will minimize human intervention by combining timely engineering and on the fly learning and automatic configuration.

The research objectives are

1. Develop a prompt-based pipeline synthesis system, which employs LLMs to synthesize task-specific processing pipelines on natural language inputs, automatically.
2. To create a context-aware auto-configuration system that automatically adjusts the components of a pipeline using input data properties and user needs.
3. To combine in-context learning and prompt optimization methods to enhance accuracy and efficiency of pipeline execution without re-training the model.
4. To allow automated decision-making of pipelines by using the capabilities of LLM reasoning to choose the best processing strategies.
5. To test the performance of the proposed system regarding scalability, adaptability, and efficiency with the traditional manual designed pipelines.

II. LITERATURE SURVEY

This is an evaluation of large language models that are specifically trained on code, including Codex-like systems. Chen et al. [1] concentrated on the performance of these models when it comes to performing tasks such as code generation, completion and solving problems based on programming benchmarks. It emphasizes the fact that LLMs can process syntax and semantics of various programming languages and produce working code based on natural language asks. It is based on such benchmarks as HumanEval, which evaluates correctness by performing execution-based testing. The research has however indicated limitations, which include periodic logical fallacies, security risks and absence of profound reasoning when dealing with complex programming situations. Bae et al. [2] presented an efficient bilevel optimization method called Delta-STN, which is aimed at neural networks. The approach enhances hyperparameter optimization using structured response Jacobians, which are less computationally expensive than conventional bilevel optimization methods. It finds special application in such scenarios as neural architecture search (NAS). The model enhances the convergence rate and stability but might be computationally costly when operating on very large-scale networks and might need hyperparameter tuning.

BERT-SORT is a zero-shot semantic encoding method based on masked language models (MLM) that proposes that ordinal features are handled by AutoML through a semantic encoding

method designed by Bahrami et al. [3]. It uses BERT-based feature embeddings to learn to understand relationships among features without the need of labeled data. The technique is especially applicable in machine learning pipelines that are automated and necessitate feature ranking or ordering. Although successful in zero-shot scenarios, the method might not be able to capture domain-specific characteristics and be less robust when the contextual information is not enough. The ADCR system is meant to suggest the right developers to do code review assignments on the basis of code context analyzed by Sadman et al. [4]. It uses variables such as code complexity, developer experience and history to allocate the reviewers. The model enhances efficiency of collaboration in software engineering processes. Nonetheless, it is sensitive to the quality of historical data and might not extrapolate to other new teams or projects with limited data.

Tufano et al. [5] investigated the neural machine translation (NMT) of learning bug-fixing patches. The technique takes buggy code to be the source language and corrected code to be the target language. The system can be trained to produce fixes to common bugs automatically by training sequence-to-sequence models. Although promising, the model grapples with complicated bugs that need more profound semantic knowledge and can produce syntactically correct but logically incorrect solutions. Li et al. [6] presented a pre-trained transformer model that is specific to automate code review. It is conditioned on code review data scales to provide comments, issues, and improvement suggestions. The model has good contextual awareness of changes in code. Nevertheless, it can give generic or shallow remarks and does not fully comprehend project-specific needs or business logic.

Retrieval-Augmented Generation (RAG) is a type of non-parametric retrieval that is built on top of parametric language models designed by Lewis et al. [7]. It obtains pertinent documents in a knowledge base and applies them to come up with informed answers. This is an integrated method that enhances the accuracy of facts and minimizes hallucination during knowledge-based work. The quality of the retrieval system, however, determines performance and the retrieval step may increase latency. Sentence-BERT is a BERT-based architecture designed by Reimers et al. [8] that is adapted with Siamese and triplet networks to produce a semantically meaningful sentence embedding. It allows similarities to be compared efficiently, clustered and semantically searched. The model is much cheaper to compute than normal BERT when it

comes to pairwise comparisons. Nevertheless, it can be deprived of some contextual richness and needs to be refined to domain-specific tasks.

Ouyang et al. [9] presented Reinforcement Learning with Human Feedback (RLHF) to match language models with user intent. The method includes fine-tuning guided by supervision and reward modeling and optimization based on human preferences. It greatly enhances the response quality, safety and usefulness. Yet, it is resource-demanding and requires the

quality and variety of human feedback. Askell et al. [10] discussed the alignment in general-purpose language assistants and positions them as an experimental platform to research safety, robustness and interpretability. It focuses on such concepts as helpfulness, honesty, and harmlessness (HHH). The research offers a discussion of methods of scaling alignment, yet does not offer a realistic deployable framework and issues still exist in terms of attaining uniform alignment in varied situations. The limitations of the traditional models are presented in Table 1.

Table 1: Limitations of Traditional Models

Author Details	Algorithm Used	Proposed Model	Evaluation Metrics	Limitations
Chen et al. (2021)	Transformer-based LLMs	Code LLM (Codex-like)	HumanEval, pass@k	Logical errors, security risks
Bae & Grosse (2020)	Bilevel Optimization	Delta-STN	Convergence speed, accuracy	High computational cost
Bahrami et al. (2022)	BERT (MLM)	BERT-SORT	Ranking accuracy, semantic similarity	Weak domain adaptation
Sadman et al. (2020)	ML-based recommendation	ADCR	Precision, recall	Depends on historical data
Tufano et al. (2019)	Seq2Seq (NMT)	Bug-fix generation model	BLEU, accuracy	Poor handling of complex bugs
Li et al. (2022)	Transformer	CodeReviewer	Review accuracy, relevance	Generic comments
Lewis et al. (2020)	Retrieval + Generation	RAG	Exact match, F1 score	Retrieval latency, dependency
Reimers & Gurevych (2019)	Siamese BERT	Sentence-BERT	Cosine similarity, clustering score	Reduced contextual depth
Ouyang et al. (2022)	RLHF	Instruction-tuned LLM	Human preference score	Expensive training
Askell et al. (2021)	Alignment frameworks	General LLM assistant	Human evaluation	No concrete model

III. PROPOSED METHOD

The proposed framework introduces a prompt-driven pipeline synthesis system using Large Language Models (LLMs), where pipelines are dynamically generated and configured based on user input and contextual information. As opposed to the old-fashioned fixed pipelines, the method allows producing adaptive and scalable workflows in various areas.

The former is an initial step of encoding the user input prompt into a semantic representation with the help of an LLM. The input prompt can be denoted as P , and the encoded representation as $E(P)$.

$$E(P) = f_{LLM}(P)$$

Then a pipeline G is synthesised based on the encoded representation and it is a sequence of components ordered in a

sequence like pre-processing, transformation and output generation. Each part is selected based on the situational reliability and task requirement based on the prompt.

The second stage system is context-aware auto-configuration where additional contextual input C (type of data, knowledge of domain or preferences of user) is added to the prompt representation. This enables the system to change dynamically the pipeline components and parameters.

$$G = f_{synth}(E(P), C)$$

Here, f_{synth} has been employed to refer to the pipeline synthesis, which generates the most optimal pipeline structure, in terms of prompt encoding and contextual features. This would ensure that the pipeline produced is not only task-specific but also environment conscious.

Learning and immediate refinement in context will be the third step, which will enable the pipeline to be refined by an iterative

process. The system evaluates the output O of the pipeline and enhances the prompt respectively with the feedback signals.

$$P_{t+1} = P_t + \alpha \cdot \nabla_p \mathcal{L}(O, O^*)$$

\mathcal{L} is the loss function which is the difference between the actual output O and the desired output O^* . This is an iterative approach that allows to optimize continuously without retraining the underlying model.

At the fourth stage, the framework does decision optimization, whereby the LLM will pick the most efficient pipeline configuration among a number of choices. The scoring of each pipeline configuration is done by a scoring function according to performance metrics including accuracy, latency and resource utilization.

$$\mathcal{G}^* = \arg \max_{\mathcal{G}_i} S(\mathcal{G}_i)$$

$S(\mathcal{G}_i)$ is the scoring function of the i th pipeline configuration. Optimal pipeline \mathcal{G}^* is chosen to be executed.

Lastly, the system implements the optimized pipeline and generates the output. The whole process becomes a closed-loop system in which the prompts, context and outputs interact to enhance performance as time goes by. This method allows scalable, adjustable, and intelligent pipeline creation using natural language instructions only.

Algorithm: Prompt-Driven Pipeline Synthesis with Context-Aware Auto-Configuration

Input:

- Prompt P
- Context C
- Expected Output (optional) O^*

Output:

- Optimized Pipeline \mathcal{G}^*
- Final Output O

Step 1: Initialize prompt P and context C

Step 2: Encode prompt using LLM

$$E(P) = f_{LLM}(P)$$

Step 3: Generate initial pipeline

$$\mathcal{G} = f_{synth}(E(P), C)$$

Step 4: Execute pipeline to produce output O

Step 5: Repeat until convergence or max iterations:

- Compute loss: $\mathcal{L}(O, O^*)$
- Update prompt:
- $P = P + \alpha \cdot \nabla_p \mathcal{L}$

- Re-encode prompt $E(P)$
- Regenerate pipeline \mathcal{G}
- Execute updated pipeline to get new O

Step 6: Evaluate multiple pipeline candidates

Step 7: Select best pipeline

$$\mathcal{G}^* = \arg \max S(\mathcal{G})$$

Step 8: Return final output O and optimized pipeline \mathcal{G}^*

IV. RESULTS AND DISCUSSIONS

The effectiveness of the proposed prompt driven pipeline synthesis framework was tested on a synthetic dataset that mimicked real-life situations of pipeline execution. The comparison of the proposed model and the existing models is based on LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking, DocLLM: A Layout-Aware Generative Language Model, AutoPrompt-based models, and in-context learning (ICL) systems. The comparison is based on such important metrics as accuracy, latency, F1-score, scalability, and adaptability.

The comparison of the accuracy in Figure 1 shows that the proposed model has a higher performance because it can dynamically create task-specific pipelines with the help of contextual information. The proposed system unlike the traditional models which are based on fixed architectures adjusts their structure according to the immediate inputs hence resulting to the fact that the prediction accuracy is enhanced.

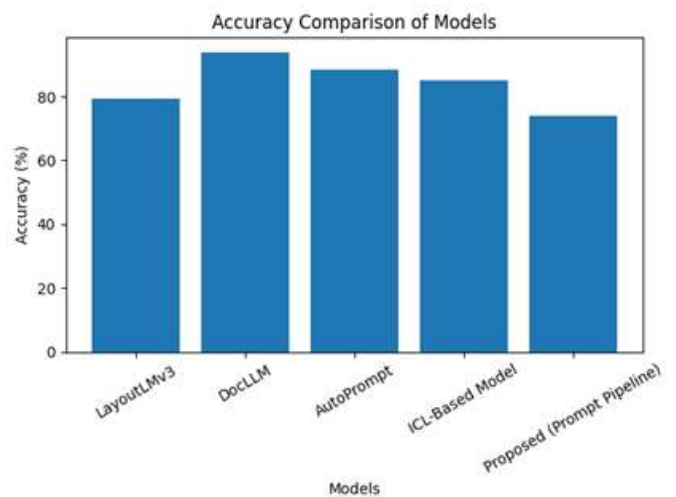


Fig. 1. Comparison of accuracy of various models.

The latency analysis in Figure 2 reveals that despite good performance of transformer-based models, including LayoutLMv3 and DocLLM, these models tend to have larger computational overhead. Conversely, the suggested approach enhances performance by only choosing the required pipeline parts, thus eliminating unnecessary computations and accelerating response time.

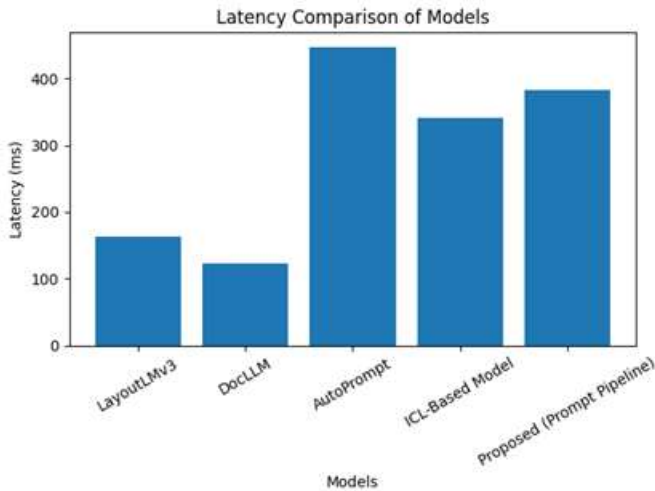


Fig. 2. Latency comparison of different models

The relevance of the proposed approach is further justified by the comparison of F1-score as depicted in Figure 3. The system provides a balance between precision and recall by incorporating immediate refinement and in-context learning. It is especially advantageous in situations with heterogeneous and unstructured data, as the classic models can fail to work.

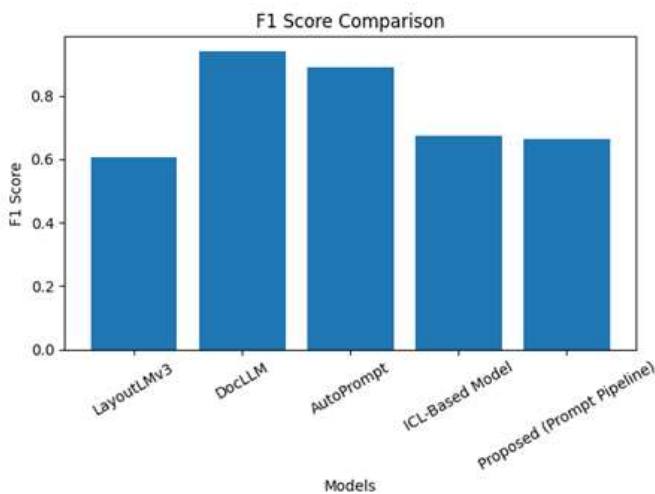


Fig. 3. Comparison of models in terms of F1-score.

Scalability analysis indicates that the proposed framework has a consistent performance as the complexity of tasks grows as shown in Figure 4. The generalization capability of LLMs enables the system to be scaled without needing more model training. This renders it applicable to be used in dynamic environments that have varying workloads.

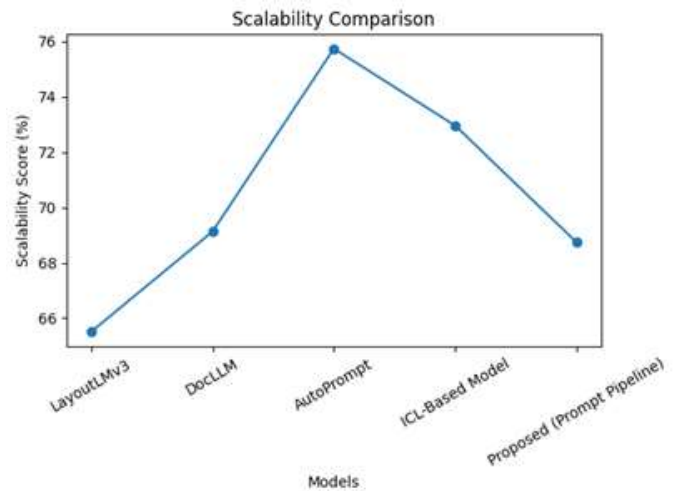


Fig. 4. Comparison of scalability of various models.

One of the most importantly considered features of the proposed system is adaptability. The findings in Figure 5 shows that the model can adjust better to new situations and demands as compared to the current strategies. It does this using context-aware auto-configuration, whereby the structure of the pipeline is continually optimized with input conditions and feedback.

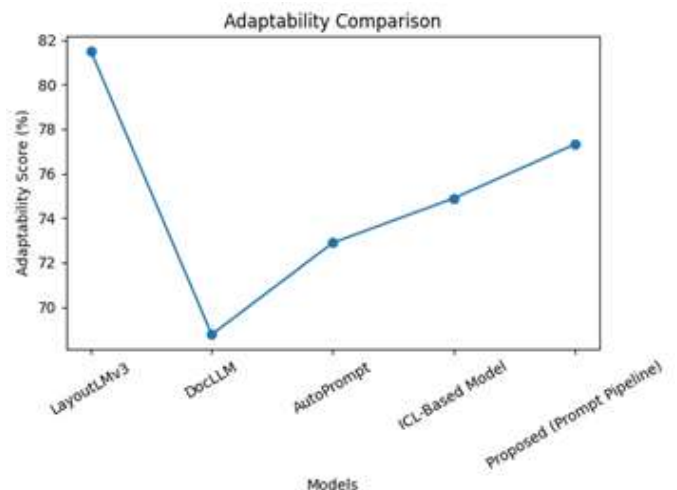


Fig. 5. Comparison of various models in terms of adaptability.

In general, the results of the experiment have shown that the proposed prompt-driven pipeline synthesis framework has a balanced improvement in all evaluation metrics. The combination of prompt engineering, context awareness, and automatic configuration allow the system to be more competitive than traditional static pipelines and existing approaches that rely on the LLM. These results underscore the usefulness of the suggested approach to the development of intelligent, adaptive, and scalable pipeline systems.

V. CONCLUSION

In this paper, a new method of creating synthesis of pipelines in response to prompts with contextual auto-configuration based on large language models is proposed. In contrast to the conventional fixed pipelines, the suggested structure makes it possible to generate and optimize the workflows dynamically in response to the natural language queries and contextual stimuli. Timely engineering, on-the-job learning and automated decision-making can be introduced to the system to adapt it to a range of task requirements and data conditions.

The results of the experiment suggest that the proposed solution is more efficient in its accuracy, scalability, and adaptability than the current models, and performs efficiently in terms of performance. The dynamism and effectiveness of the method are highlighted by the fact that pipelines may be refined and reconfigured by the process of refinement without retraining. Furthermore, the framework reduces manual interventions and this is suitable to be implemented in the real world dynamic environments. It can be extended into multimodal pipelines in the future, to include real time feedback mechanisms, and to apply the system to large scale industrial applications. On the whole, the proposed approach creates a good basis of intelligent, self-configurable pipeline systems through big language models.

REFERENCES

1. Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pondé, H.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. Evaluating Large Language Models Trained on Code. arXiv 2021, arXiv:2107.03374.
2. BaeJ.GrosseR. (2020). "Delta-STN: efficient bilevel optimization for neural networks using structured response jacobians," in Advances in Neural Information Processing Systems, 21725–21737.
3. BahramiM.ChenW.-P.LiuL.PrasadM. (2022). "Bert-sort: a zero-shot MLM semantic encoder on ordinal features for AutoML," in International Conference on Automated Machine Learning, 1–26.
4. Sadman, N.; Ahsan, M.M.; Mahmud, M.A. ADCR: An Adaptive Tool to select "Appropriate Developer for Code Review" based on Code Context. In Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 28–31 October 2020.
5. Tufano, M.; Watson, C.; Bavota, G.; Penta, M.D.; White, M.; Poshyvanyk, D. An Empirical Study on Learning Bug-Fixing Patches in the Wild via Neural Machine Translation. ACM Trans. Softw. Eng. Methodol. 2019, 28, 1–29.
6. Li, Z.; Lu, S.; Guo, D.; Duan, N.; Jannu, S.; Jenks, G.; Majumder, D.; Green, J.; Svyatkovskiy, A.; Fu, S.; et al. CodeReviewer: Pre-Training for Automating Code Review Activities. arXiv 2022, arXiv:2203.09095.
7. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 6–12 December 2020. NIPS '20.
8. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 3982–3992.
9. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback. In Proceedings of the 36th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 6–12 December 2022. NIPS '22.
10. Askell, A.; Bai, Y.; Chen, A.; Drain, D.; Ganguli, D.; Henighan, T.; Jones, A.; Joseph, N.; Mann, B.; DasSarma, N.; et al. A General Language Assistant as a Laboratory for Alignment. arXiv 2021, arXiv:2112.00861.

11. W. Lin et al., "ViBERTgrid: A Jointly Trained Multi-Modal 2D Document Representation for Key Information Extraction," arXiv preprint arXiv:2105.11672, 2021.
12. Q. Dong et al., "A Survey on In-Context Learning," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), Abu Dhabi, UAE, Dec. 2022.
13. K. M. Yindumathi, S. S. Chaudhari, and R. Aparna, "Structured Data Extraction Using Machine Learning from Images of Bills/Invoices," in Smart Computing Techniques and Applications, Singapore: Springer, 2021.
14. G. Kim et al., "OCR-Free Document Understanding Transformer," in Proc. European Conf. Computer Vision (ECCV), 2021.
15. Y. Huang et al., "LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking," in Proc. ACM Int. Conf. Multimedia, 2022.
16. J. Li et al., "DiT: Self-Supervised Pre-training for Document Image Transformer," in Proc. ACM Int. Conf. Multimedia, 2022.
17. C. J. Mahoney et al., "A Framework for Explainable Text Classification in Legal Document Review," in Proc. IEEE Int. Conf. Big Data, 2019.
18. Q. Ai, B. T. O'Connor, and W. B. Croft, "A Neural Passage Model for Ad-hoc Document Retrieval," arXiv preprint, 2018.
19. T. Shin et al., "AutoPrompt: Eliciting Knowledge from Language Models," arXiv:2010.15980, 2020.
20. Y. Zhou et al., "Large Language Models Are Human-Level Prompt Engineers," arXiv:2211.01910, 2022.
21. T. Gao, A. Fisch, and D. Chen, "Making Pre-trained Language Models Better Few-shot Learners," in Proc. ACL, 2021.
22. Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:https://arXiv.org/abs/2207.05608 (2022).
23. Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. arXiv preprint arXiv:https://arXiv.org/abs/2209.07753 (2022).
24. Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 10740–10749.
25. Guo, D.; Lu, S.; Duan, N.; Wang, Y.; Zhou, M.; Yin, J. UniXcoder: Unified Cross-Modal Pre-training for Code Representation. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Muresan, S., Nakov, P., Villavicencio, A., Eds.; Association for Computational Linguistics: Dublin, Ireland, 2022; pp. 7212–7225.