A Survey of Open Research Problems in AI, Hybrid Cloud, Quantum and Security

Arkadeep Acharya, Krish Agrawal, Dibyanayan Bandyopadhyay, Kanishka Bansode, Jitbitan Baroi, Utkarsh Bhatt, Debarpan Bhattacharya, Debasmita Bhoumik, Kushagra Bhushan, Atharva Deshmukh, Adrija Dhar, Meet Doshi, Sannzay Gaddam, Baban Gain, Priyanshul Govil, Parthivi Gupta, Sarthak Harne, Anubhav Jana, Aaqilah A J, Ishant Kohar, Simran Kumari, Vishal Kumar, Kishan Maharaj, Lakshmi Mandal, Vivek Mathur, Alokendu Mazumder, Yash Mehan, Aaryav Mishra, Ananya Mishra, Sidhaarth Sredharan Murali, Anoushka Nag, Pingal Pratyush Nath, Tejomay Kishor Padole, Shobhit Pandey, Penumalla Aditya Pavani, Sameer Pimparkhede, Deepika Ponnana, Kiran Pradeep, Shubh Prakash, Adnan Qidwai, A Eashaan Rao, Dhruva Singh Sachan, Himanshu Sahu, Arunima Sarkar, Lakshita Saxena, Vandit Shah, Suchi Sharma, Shivangi Shreya, Satyam Shukla, Akanksha Singal, Abhishek Kumar Singh, Sidharth Sinha, Ameya Srivastav, Arjun Temura, Soumya Teotia, Aryamaan Thakur, Pradhuman Tiwari, Aditya Tomar, Tirth Vamja, Subhash Veggalam, Samidha Verma

IBM Research

{arkadeep.acharya, agarwalkrish29, dibyanayan, kanishka.bansode, jitbitan, utkarsh.bhatt, debarpanb, debasmita.bhoumik, kushagra, atharva.deshmukh, adrija.dhar, meetdoshi, baban.gain, priyanshul, sarthak.harne, gupta.parthivi, anubhav.jana, aaqilahaj, ishantkohar, simran.k, vishal.kumar73, kishan.maharaj, lakshmi.mandal, vivek.mathur, mazumder, yash.mehan, aaryav.mishra, ananya.mishra2, ssm13, anoushka.nag, pingalnath, tejomaypadole, shobhit.pandey, pavani.penumalla, sameer.pimparkhede1, deepika.ponnana, shubhprakash, adnanqidwai, rao.eashaan, dhruva.singh, arunima.sarkar, lakshita.saxena, shah.vandit.pankaj, suchi.sharma, shivangi.shreya1, akanksha.singal, abhisheksingh, sidharthsinha, ameya, arjun.temura, soumya.teotia, aryamaan.thakur, pradhuman.tiwari, tirth.vamja, subhash.veggalam, vsamidha}@ibm.com, {sannzay.gaddam, kiran.pradeep, himanshu.sahu, satyam.shukla, aditya.tomar}@partner.ibm.com

1 Artificial Intelligence

1.1 ESG Sentiment in Forecasting Stocks

- Vivek Mathur

The integration of Environmental, Social, and Governance (ESG) factors into stock price prediction models is a critical challenge in the financial sector (Aldowaish et al. 2022) . This project aims to quantify ESG sentiment from news articles and company disclosures (Chowdhury et al. 2023), which are essential in assessing a company's long-term sustainability and ethical impact(de Souza Barbosa et al. 2023). By fine-tuning models like ClimateBERT (Webersinke et al. 2021) on climate data, the project generates ESG scores that are integrated into time series AI models. Additionally, the project incorporates Financial Shock indices and Governance Risk by integrating the Global Policy Uncertainty (GPR) (Caldara and Iacoviello 2022), Economic Policy Uncertainty (GEPU) (Bossman, Gubareva, and Teplova 2023), and Volatility Index (VIX) (Liu et al. 2022) indices. It also leverages the classification powers of models such as ESGify (Kazakov et al. 2023) and IBM's Tiny Time Mixer Model (TTM) (Ekambaram et al. 2024a). This innovative approach enhances the predictive accuracy of stock prices, offering valuable insights to investors and companies about the potential financial impact of their ESG performance.

1.2 IndicRetrieval

-Arkadeep Acharya

With over half a billion Hindi-speaking individuals globally, there is a pressing need for effective Hindi information retrieval systems. Despite ongoing research in this domain, we realised that there remains a lack of a comprehensive benchmark for evaluating retrieval models on Hindi language. To bridge this gap, we aim to introduce and release the Indic version of the BEIR (Thakur et al. 2021) benchmark, starting with Hindi and then extending to other Indian Languages. We would evaluate baseline models on this benchmark, identifying domain-specific challenges and performance discrepancies. This benchmark would help us to gain insights into the limitations and capabilities of Hindi retrieval models, fostering advancements in this critical area. Further we aim to come up with a method to handle low resource Indian languages for retrieval tasks which can be used to develop a retrieval model which is tailored to handle these languages and performs well on our Indic Benchmark.

1.3 Joint Retriever Reader

-Sidhaarth Sredharan Murali

Creating a single model that excels across a variety of tasks has long been an AI goal, with the argument that all text-based problems can be managed by a single LLM through generation. However, tasks like search engines and chatbots, which organize information using embeddings,

have not been fully explored within LLMs. Instead of multiple specialized tools, we propose a unified "joint retriever reader" model based on decoder-only architectures to handle both retrieval and generation, enhancing Retrieval-Augmented Generation (RAG) (Lewis et al. 2020) performance. This approach addresses the sufficiency of current small language models (SLMs), typically with 1 to 3 billion parameters, for joint retriever-reader tasks. Inspired by past unified models (Muennighoff et al. 2024) that displayed better performance across different tasks by training with a single target, our approach is to introduce a weighted objective function and a novel pooling strategy. In modern times, there is increasing need for adaptable AI systems that are high performing, which such a unified model could greatly simplify the application and efficiency of AI in many areas, from search engines to interactive chatbots; thus being very valuable.

1.4 GraphQL Schema Generation using RL

-Lakshmi Mandal

GraphQL (Wittern, Cha, and Laredo 2018) is a query language and execution engine for APIs which is having several advantages over REST APIs (Ganesan et al. 2024). In GraphQL users can specify exactly what data they need, it is easy to aggregate data from multiple sources. Further, GraphQL uses a type system to describe data rather than endpoints. Two important components of GraphQL are GraphQL schema and GraphQL query. A GraphQL Schema serves as a blueprint for the API that defines what data can be queried from the server, and what are the different types of that data. They define a contract of API between a server and a client.

However, there is no efficient and effective way to generate the GraphQL schema automatically given a natural language user utterance. Thus, we propose a Reinforcement Learning (RL) based approach to solve the above mentioned issue.

1.5 *"Any-length" Light weight Time Series Forecaster*

- Alokendu Mazumder

In time series modelling, a significant challenge is that traditional models can train and infer on the same length of context and forecast lengths. They don't have the flexibility of training and inferencing over different contexts and forecast lengths. In recent years, several models have been developed to address these issues. MOMENT (Das et al. 2023) and TimesFM (Goswami et al. 2024) handle this problem by fixing a longer context length for training and padding with zeros for data points that have shorter total lengths than the desired context length. During inference, models like TimesFM use auto-regressive decoding, similar to large language models, to predict any forecast length.

Our research aims to develop an "any-length" lightweight time series forecaster. Our proposed model will be able to train on any arbitrary context length and perform inference on any arbitrary context length. The training context length and inference context length need not be the same, and this flexibility also applies to the forecast length. For our backbone, we use the lightweight *Tiny Time Mixer*(TTM) (Ekambaram et al. 2024a). To address the aforementioned issues, we propose using a looping mechanism over any decided context length. The entire time series is first divided into smaller segments, and each small segment is processed through a deep network of the backbone. In the later stages, the outputs of each segment from the backbone are fused using lightweight self-attention and convex combinations across the backbone's encoder output. To handle the variable forecast length, a similar looping mechanism is employed over the TTM decoder. This allows our model to handle "any-length" context and forecast lengths effectively during training and inference!

1.6 Diffusion Models for Code Generation

-Tejomay Kishor Padole

Diffusion models are the current state-of-the-art approach for modeling real-world continuous data distributions (like a distribution over images). Inspired by non-linear statistical physics (Sohl-Dickstein et al. 2015), the approach is based on defining a Markov chain of transitions that iteratively adds noise to the data until it reaches pure random noise distribution. A neural network is then employed to reverse this stochastic process by iterative denoising from the noise distribution back to the original data distribution. While other generative models like VAEs and GANs learn a mapping from noise to data, diffusion models break down that mapping into iterative denoising steps, intuitively simplifying the neural network's job. This iterative generation process has also allowed diffusion models to generate more diverse data samples and handle multimodality in complex real-world distributions.

While diffusion models have seen immense success in the continuous domain [(Rombach et al. 2022), (Dhariwal and Nichol 2021)], they have also started garnering attention for generating discrete data (like text and code). But why use diffusion models when LLMs exist?

Current LLMs use autoregressive generation strategy to model text and code. While this has worked pretty well, it still has some issues. Due to the autoregressive nature of LLMs, they generate data sequentially, which can become quite slow for long sequence generation. Another drawback of sequential generation is that LLMs suffer from "sampling drifts," i.e., an error made in mid-generation will propagate further as the generation continues. In contrast, diffusion models generate data non-autoregressively by iterative refinement of the entire sequence. Recent work on diffusion models for text has shown that they do a much better job following constraints (Li et al. 2022) while generating and exhibiting high output diversity in unconstrained situations (Gong et al. 2023). While the work is promising, not much has gone into employing diffusion models for text. This is even more true for generating structured discrete data like code.

Considering what diffusion models have to offer over current LLM-based modeling, we have started exploring diffusion models for code generation. We also want to ex-

plore whether current pre-trained code-LLMs can be used along with diffusion models in a transfer learning setting. Instead of requiring pre-training for code diffusion models, we hope to use pre-trained code-LLMs and further enhance them with diffusion-based training.

1.7 Improving Learned Sparse Retriever

-Meet Doshi

Learned sparse retrieval (LSR) or sparse neural search is an approach to text search that uses a sparse vector representation of queries and documents. It borrows techniques both from lexical bag-of-words and vector embedding algorithms and is claimed to perform better than either alone. Current state-of-the-art systems like Splade-v3 and Splade-doc (Lassance et al. 2024) maintain semantic information as a bag of words. This type of similarity matching loses a lot of sequential and semantic information. This can be extended to decoder language models, which generally have better representations based on exposure to extensive training data. We explore whether during inference this search can be improved using synonyms, root words, and clustering words with similar embeddings. We also focus on how we can reweigh samples during training so that $P_\theta(X_{train}, Y_{train}) \approx P_\theta(X_{test}, Y_{test})$. This is done via Distributed data sampling (DDS) (Wang et al. 2019), a way to reweigh a learned sampling distribution in case of multiple training sets where some subset may or may not represent the test set distribution. This is done using a bilevel optimization which is very hard to converge. Practically this is very hard to improve due to excessive compute and sparse weight problems but we are look at approximate solutions which yield good results.

1.8 Evaluating Code Summaries for Hallucination

-Kishan Maharaj

In recent years, large language models have shown significant advancements in understanding the code. With this advancement, hallucination has become a prevalent problem hampering the progress towards code-related tasks like code summarization, code generation, etc (Liu et al. 2024). In general, a summary evaluation methodology can have multiple aspects to it; this includes coherence (highlights the capability of the model to retain the context of the input), completeness (highlights the capability of the model to cover the complete context of the input), and correctness (highlights the capability of the model to generate the output grounded to the original input). We would like to progress towards the evaluation of summary for its correctness.

As the language models progress further, the detection of hallucination becomes more challenging because the language model tends to hallucinate more convincingly (Huang et al. 2023). Hallucination detection in code summarisation is specifically challenging due to the complexities of natural language. Also, the challenge arises due to the presence of two different text domains: code and natural language. The interaction between these domains can become very complicated to handle. In our preliminary experiments, we observe that the current language models have the tendency to guess the summary based on just the lexical interpretation of its entities like variable name, function name etc. We also observe that this behaviour is more profound when the input code is longer and the logic conveyed by the code is more complicated.

In our work, we aim to progress in this non-trivial problem by mapping the summary to code for hallucination detection at the entity level. We aim to leverage all the recent state-of-the-art language models; and also some available efficient pre-trained models trained on code data. This will potentially lead to an interpretable hallucination detection approach for grounding the code summary to the code. This work is directly aligned with various IBM products, such as IBM watsonx Code Assistant for Z and the Granite code models.

1.9 Constrained generation evaluation for LLMs

-Sameer Pimparkhede

Large language models (LLMs) have shown promising results (Brown et al. 2020) in comprehending and subsequently generating coherent text and code in zero and few-shot settings, especially for resource-rich languages. However, their practical utility depends on their ability to follow constraints at various granularity, encompassing user and system requirements. Recent work (Sun et al. 2023) shows LLMs have difficulty in understanding fine-grained hard constraints represented as a natural language in zero and few shot settings for tasks like paraphrase generation and numerical planning. While in the code domain, encoding use-case-specific constraints in code format is prevalent and crucial for enterprises to maintain the integrity of the code typically written in DSLs such as JSON, YAML, and XML. Such DSLs are widely adopted for data exchange and configuration of systems such as Kubernetes, and Ansible. Depending on the use case and system, the constraints that are often fine-grained are articulated as schemas in various code languages like JSON schema, YAML, or Python Pydantic to validate DSL code. A typical schema holds fine-grained hard and soft constraints such as data types, required and optional fields, default values, numerical constraints, etc. LLMs must be cognizant of such constraints for practical use in producing reliable code. Therefore, we aim to study and understand the controllability of LLMs in generating code when fine-grained constraints are represented in code format.

1.10 Utilizing Question Decomposition for Efficient Text-to-SQL Generation

-Kiran Pradeep

Text-to-SQL generation involves converting a natural language query into an executable SQL query. Humans generally break complex questions into simpler manageable sub-questions and process them sequentially to solve such tasks. This breaking down of complex questions into simpler manageable sub-questions is termed as question decomposition. We aim to induce similar thinking in machines, enhancing their capability to generate accurate SQL queries for a given

natural language query. Apart from the challenges in the Text-to-SQL parsing, question decomposition introduces its own challenges such as the granularity of decomposition, optimal decomposition selection, ensuring complete coverage of the aspects of the query asked, and the execution sequence order. And there are no specific metrics to evaluate this which poses another major challenge.

Wolfson, Deutch, and Berant., (2022) is one of the few works that leverage question decomposition for solving text-to-SQL parsing. Their work focuses on solving the problem of the existence of multiple valid SQL queries for a given natural language query. To this extent, they synthesize the SQL query using the weak signal received from comparing the answer generated by the synthesized SQL query and the gold one. But this method is not generalizable which is evident from their experimental results. Zhang et al.., (2023) leverages question decomposition for Explainable Question Answering (XQA). They create a Hierarchical question decomposition tree that facilitates reasoning and solves the granularity problem while decomposing. However, their model does not fare well with other LLM-based models.

We plan to evaluate our method on the BIRD-SQL (Li et al. 2024a) and Spider (Yu et al. 2018) datasets. Currently, IBM tops the BIRD-SQL leaderboard. But there is still a marginal gap between the human performance. This research aims to reduce this gap by employing question decomposition for efficient Text-to-SQL generation.

1.11 Multi-lingual Code Summarization

-Arunima Sarkar

Large Language Models(LLMs) are state-of the art for many natural language and also code related tasks. However these models treat input as sequence of tokens which might not well capture the code structure or syntax involved. As a result Large Code Models (LCMs) have been introduced what also takes the structure and syntax into the context for tasks like Code Summarization, Code generation and many more. A real world scenario includes many programmers contributing or developing codes with non-english variable or function names. Hence there is a growing requirement of models and datasets that can handle multi-lingual codebases (Wang et al. 2023) and convert them to English codebase for accessibility to a broader community. Most of these LCM are predominantly trained on codes written in English owing to the larger availability of English corpus. Hence, their performance on downstream tasks like code summarization (Haldar and Hockenmaier 2024) is more accurate on codes in English compared to codes written in other languages. Our initial experiments conclude a similar observation , claiming better quality code summary from English codes rather than non-english code. However there is a lot of overlap between the summaries generated. Our hypothesis : LLMs might decompose the code summarization process into two parts: (1) In LLM first the input code in non-english is getting translated into a English code, (2) Then LLMs summarize that English code. Currently, we are extensively experimenting with multiple languages to understand the overlap among summaries and also overlap of summaries and code.

1.12 Time Series Event Extraction For Forecasting Stock Price Trends

-Vishal Kumar

Stock price data alone, despite encapsulating typical time series characteristics, are insufficient for accurately forecasting stock price trends. Supplementing price and volume data with company announcements and market sentiment derived from news events becomes crucial. In our multi-intern project, our focus is on extracting pertinent events from company announcements. Integrating these event signals as a distinct modality enhances the predictive capability of our time series models for forecasting stock price trends.

1.13 Dynamic Question Generation for Code Related Tasks: Bug Identification, Localisation and Rectification

-Ameya Srivastav

Introduction In software development, code frequently fails to pass all test cases or exhibit the desired behavior. Traditional debugging methods are often time-consuming, require significant expertise, and are impractical for large or complex systems. To address these challenges, we propose a novel approach that utilizes large language models (LLMs) to dynamically generate tailored questions for specific code snippets. This method aims to streamline the debugging process by guiding the model to **identify** and **rectify bugs** more efficiently.

Motivation We need an efficient, guided debugging approach beyond generic questions due to the diverse nature of real-world codebases. This method can narrow down the debugging task, making it more accessible and effective.

Problem Statement Generic debugging questions are not customized to the specific context of each unique code snippet, leading to irrelevance and inefficiency in identifying bugs.
Our objective is to enhance debugging by **dynamically generating questions** tailored to a specific code snippet (typically in Python or C++) that contains **logical, reference, or syntactical bugs**. By framing these questions specifically for the given code, we provide a structured approach that guides the model through the **debugging** process. As the model answers these targeted questions and examines the buggy code, it can more effectively identify the bug's location, rectify the issue, and ultimately generate the correct code . We leverage **LLMs**(typically Llama-3-70b) to generate customized questions tailored to the specific context of each code snippet. The entire process typically involves:
1. Using a buggy dataset (Liu and Wan 2021).
2. Generating context-specific questions (Tian et al. 2024).
3. Assessing the relevance of these questions to improve bug identification and correction (Fernando et al. 2023).
4. Localise and rectify the bug based on the questions.

We use parameters such as question relevance and final code evaluation using test cases to assess the model's efficiency and performance. By refining prompts based on feedback (Fernando et al. 2023), we iteratively improve the model until the desired outcome is achieved.

1.14 Conversation Disentanglement

-Shivangi Shreya

Consider a chat involving 'n' people discussing 'k' different topics. In this group, some people will talk about topic 1, others about topic 2, and so on. Since the chat occurs on a shared platform, these conversations will be intermingled, making it difficult to trace discussions within a specific topic. The process of separating these interwoven conversations is known as conversation disentanglement. This is crucial for multi-party conversational NLP applications, such as question answering and summarization.

To address this problem, various approaches have been proposed, including different architectures and new datasets. Researchers use diverse datasets, from Ubuntu and Linux (Kummerfeld et al. 2019) logs to TV show screenplays (Chang, Chen, and Bamman 2023), to create more robust models.

Specifically, conversation disentanglement involves assigning a message (target message) to the correct conversation. This can mean identifying the message (parent message) to which target message replies or recognizing the entire conversation under a specific topic.

Traditionally, conversation disentanglement is treated as a classification problem. A target message is paired with all other messages in the chat log, and each pair is classified as either a target-parent message pair or not. Each pair is considered independently, meaning the decision for one pair does not affect others.

We are investigating whether leveraging decisions made on previous pairs can enhance the performance when deciding on new pairs. Instead of framing this as a classification problem, we approach conversation disentanglement as a generation task utilizing large language models.

1.15 Anomaly Detection for Time-series using Time-series Foundation Models (TSFM)

-Debarpan Bhattacharya

Problem statement: Time series anomaly detection plays a crucial role in various fields by identifying unusual patterns or events that deviate from expected behavior within temporal data. This research problem holds immense importance across industries such as finance, healthcare, and cybersecurity, where early detection of anomalies can prevent fraud, diagnose critical health conditions, or protect against cyber threats. By leveraging advanced algorithms and statistical methods, researchers aim to develop robust models capable of accurately distinguishing anomalies from normal variations in time series data. Ultimately, improving the effectiveness of anomaly detection systems not only enhances operational efficiency but also ensures timely intervention, leading to better decision-making and risk management in complex and dynamic environments.

Motivation: The three main challenges that make time-series anomaly detection challenging are:

- The time-series data are often consisting of multiple channels, are of different periods (hourly, monthly, yearly etc.) which makes detection of time-series anomaly a sophisticated problem.
- The anomalies should be distinguished from outliers. The outliers are in-distribution but rarely occuring samples. However, anomalies are out-of-distribution and generally occurs due to some fault in the process which is generating the time-series.
- The training data is simply a set of normal samples, without any anomaly. So, the anomaly comes during the inference only. Hence, during training, we have only data coming from one class and during inference, there are two classes (normal vs anomaly) which makes the problem tricky to solve.

Baselines: A few recent and prominent time-series foundation models capable of time-series anomaly detection are GPT4TS (Zhou et al. 2023) and MOMENT (Goswami et al. 2024).

1.16 Knowledge Incremental Continual Learning in LLMs

-Baban Gain

Continual learning focuses on the capability of models to adapt to new data as it becomes available without forgetting the capabilities of the previous tasks. It has been observed that fine-tuning the large language models (LLMs) on unseen data increases the chances of hallucination (Gekhman et al. 2024). Generally, it is assumed that data for a specific task or domain is fully available at the beginning, and new tasks or domains are added over time. However, in real-life settings, data for the tasks are not readily available at the beginning but are added incrementally over time as the project/annotation progresses. This creates a dynamic learning environment where the model must be capable of integrating new data as it arrives while maintaining the performance of previously learned tasks. Further, during a training cycle, the amount of training data for different tasks may not be consistent, where the dataset created for one task can be of multiple magnitudes of other tasks, making it dominate over others. The objective of the research project is to continually learn from the added data without forgetting already acquired knowledge. This could help in lowering significant training efforts compared to full-data fine-tuning at every training cycle.

1.17 AutoEDA using LLMs

-Parthivi Gupta

Exploratory Data Analysis (EDA) is a critical yet time-consuming phase in data science, involving the summarization and visualization of data to uncover patterns and anomalies. This project presents AutoEDA, an innovative

approach leveraging Large Language Models (LLMs) to automate the EDA process, thereby accelerating data-driven decision-making and making it accessible to non-experts. Earlier work includes multiple methods like the ones used in QuickInsights (Ding, R., et al. 2019) and ATENA (Bar El, O., et al. 2020)

AutoEDA employs a pipeline utilizing Large Language Models, that automates query generation and insight extraction. The methodology involves a multi-agent system for query formation and iterative prompting, followed by algorithms for segregating out the best queries and scoring them for being used in an insight search. By utilizing the extensive knowledge base and question-forming capabilities of Large Language Models, AutoEDA ensures the creation of diverse and coherent queries, streamlining the EDA process. This automated approach not only enhances efficiency but also democratizes data analysis, enabling more rapid and informed decisions in various applications.

1.18 Detecting and mitigating factual inconsistencies in grounded generation

-Pradhuman Tiwari

This project focuses on enhancing the factual accuracy of IBM's Data Story Generation system by addressing errors in two key modules: SQL2NLG and Summarization. In the SQL2NLG module, which converts SQL query results into natural language inferences, the goal is to detect and mitigate discrepancies between the generated text and the SQL output, incorrect inferences, fabricated entities, and omitted information.

In the Summarization module, which synthesizes a coherent summary from a list of insights, the project aims to identify and correct variations in values, erroneous inferences, fabricated entities, and missing information in the summary. To achieve these goals, prominent techniques in the landscape of hallucination detection, including black-box methods (e.g., Chain-of-Knowledge (Li et al.), Semantic-aware consistency checks (Zhang et al.)) and gray-box methods (e.g., tracking attention on query tokens (Yuksekgonul et al.)), will be tested, and the best performing ones will be employed. This work will significantly improve the reliability and accuracy of the Data Story Generation system, providing users with trustworthy narratives grounded in their data.

1.19 Bias and Stereotype Detection in Multi-Task Learning Framework

-Aditya Tomar

Stereotypes and biases deeply embedded within Language Models (LMs) pose significant challenges to the fairness and reliability of AI technologies. This research investigates the effectiveness of multitask learning in identifying stereotypes and biases in natural language processing (NLP) tasks. Utilizing a meticulously annotated dataset containing labels for both stereotypes and biases, we conduct a comparative analysis of two training approaches: single-task learning, where classifiers are trained separately on stereotype and bias annotations, and multitask learning, where classifiers are trained simultaneously on both tasks. Acknowledging the nuanced disparities between bias and stereotype, often blurred in prior studies, we examine their distinct impacts and interactions within the multitask learning framework. Our hypothesis suggests that while treating bias and stereotype as distinct entities, stereotypes can aid in bias detection, given their intertwined nature. We employ two datasets, Stereoset ((Nadeem, Bethke, and Reddy 2021)) and Crows-pairs ((Nangia et al. 2020)), labeling them for both stereotype and bias annotations. Our findings illuminate the potential of multitask learning to improve stereotype and bias detection, emphasizing its importance in addressing harmful linguistic patterns in NLP applications.

1.20 Multimodal Time-Series Forecasting

– Priyanshul Govil

There has been considerable progress in forecasting time-series data with Transformer-based models (Wu et al. 2021; Zhou et al. 2022, 2021). However, Zeng et al. (2023) show that a simple linear model is able to outperform most of the Transformer-based models, questioning their effectiveness. Moreover, the Transformer architecture is both memory and compute intensive. To address these shortcomings, TSMixer (Ekambaram et al. 2023) builds a multi-layer perceptron (MLP) based model, and highlight the importance of patching time-series data. IBM's TTM (Ekambaram et al. 2024b) builds on top of this to develop a lightweight model architecture for time-series forecasting. TTM has significant performance on zero/few-shot forecasting. However, TTM still takes as input a single data modality, i.e., time-series. Real-world scenarios present themselves in various forms. For instance, traffic prediction is affected by both the historic traffic data, as well as road construction schedules. We aim to make TTM compatible with multi-modal inputs, focusing on prediciting stocks as a downstream application. We aim to achieve this by exploring various modalities in the stock market such as sentiment, news, and financial reports.

1.21 Domain Adaptation for Conversational Agents

– Kushagra Bhushan

Conversation Models can be helpful in a variety of general tasks, but, when it comes to domain specific scenarios we see that their performance rarely holds up. The current go-to method of augmenting LLMs with domain knowledge is Retrieval-Augmented Generation. In this certain parts of your domain data (usually unstructured text documents) are given in context along with the user query. This leverages the LLMs in-context learning capability to answer questions. Though effective it has its drawbacks. Some of them being improper/incorrect retrieval of documents and inability to answer complex questions that require multiple different sections of your domain.

To this end, we are working on a novel way to ingest the domain's knowledge directly into the parameters of the base model. We hypothesise that if done well enough the LLM would be able to reason over the entire domain and would

not require externally retrieved passages. Any additional information provided would be a cherry-on-top just improving the model's performance.

We do this by creating a synthetic question-answering dataset from the domain documents making sure that there is significant coverage (about 90%) to the original documents. The generated dataset consists not only of question-answer pairs but also an explanation of the answer and evidence from the original document as a form of citation to make sure of the groundedness. We have seen positive results on training with such datasets in the model's ability to recall fine-grained facts of the domain without being provided any context in form of retrieved documents. We believe that further tweaks in the training and evaluation regimes would be able to push the performance even further.

1.22 API Sequencing and Glue Code Composition via Code generation

– Deepika Ponnana

In the realm of business automation, digital assistants/chatbots are emerging as the primary method for making automation software accessible to users in various business sectors. Access to automation primarily occurs through executions sequences of APIs and RPAs. Our project addresses this need by generating API sequencing and transformation glue code from user instructions using the IBM Granite model,drawing on methodologies similar to those discussed in the ToolLlama paper (Qin et al. 2023). This task involves converting APIs into Python helper functions and approaching the problem as one of code generation. Specifically, we aim to create a Python function that orchestrates these API calls, defining their sequence and managing their input-output relationships. By ensuring smooth data flow and correct ordering of API interactions, this function will streamline the integration of multiple APIs. Our approach simplifies the complexities of API integration, enhancing efficiency and reducing manual coding efforts, thereby facilitating seamless integration processes for developers.

1.23 Compositionality Between LLMs and Indic RAG Benchmarks

– Abhishek Kumar Singh

Compositionality Between LLMs Language models (LMs) often exhibit subpar performance in reasoning tasks, such as mathematics and coding, when applied to low-resource languages. This limitation arises primarily because LMs are predominantly trained on corpora from a few high-resource languages, resulting in low-resource languages being underrepresented as long-tail knowledge. Addressing these tasks in low-resource languages ideally requires either multilingual or language-specific LLMs, followed by instruction fine-tuning (IFT) with data in the specific low-resource language. However, the available IFT data is often insufficient in scale. In this work, we will explore the composition of LLMs, each specialized in different tasks, to leverage their unique strengths collaboratively. Our aim is to enhance the performance of LMs in solving tasks in low-resource languages, ensuring that language is no longer a barrier. Through this joint approach, we seek to improve the applicability and effectiveness of LMs in diverse linguistic contexts.

Benchmarks Large Language Models (LLMs) have demonstrated remarkable zero-shot and few-shot capabilities in unseen tasks, including context-grounded question answering (QA) in English. However, the evaluation of LLMs' capabilities in non-English languages for context-based QA is limited by the scarcity of benchmarks in non-English languages. To address this gap, we introduce Indic-QA, the largest publicly available context-grounded question-answering dataset for 11 major Indian languages from two language families. The dataset comprises both extractive and abstractive question-answering tasks and includes existing datasets as well as English QA datasets translated into Indian languages. Additionally, we generate a synthetic dataset using the Gemini model to create question-answer pairs given a passage, which is then manually verified for quality assurance.

1.24 Repository aware Code LLMs

– Dibyanayan Bandyopadhyay

Language models trained primarily on natural language data exhibit certain limitations that restrict their effectiveness in the code domain. Unlike natural language, code possesses an inherent structure and set of rules that require specific inductive biases during training to be effectively captured by language models. To address this, our approach extends language-based LLMs into the code domain by incorporating structural elements into their training process.

Our hypothesis posits that Code LLMs, when trained with an awareness of this structure, will demonstrate improved reasoning capabilities regarding repository-level code. We aim to enhance the model's performance on repository-centric tasks by explicitly organizing the training dataset to emphasize caller-callee relationships between several function modules inside different files of a repository. This approach is expected to elevate the LLM's abilities in tasks such as code completion and code translation, ultimately leading to more proficient and contextually aware code generation.

1.25 LLM Orchestrator in workflow automation

– Dhruva Singh Sachan

Vision: AI-aided **Orchestrator** for existing business **workflows**. Orchestrator has **complete knowledge** about the workflow, the processes and the entities, Act as **NL-interface** to the client and human-agents aiding their tasks.

We are mainly targeting **Data Workflows**. For client the relevant use-case could be knowing status of tasks, results etc. For the human-agent who is assigned a task by the Orchestrator, it is the first point of contact, it aids him in his task and also extracts useful results for the task from the conversation. Orchestrator acts as a NL-interface to **database**, performs various **algorithms** to solve a task, **generates visualisations** etc. Orchestrator do so by orchestrating multiple **agents** - for eg: Text2SQL, Code agent etc. The response

and conversation mode of automated agents is in NL. The goal of the project is to develop an orchestrator which can **generate sufficient context to efficiently orchestrate the automated-agents** to solve tasks. For example, the human-agent needs to know about entropy of an attribute of the dataset, and it was not calculated in the workflow, Orchestrator solves this task using a expert-planner (LLM), Text2SQL and Code agent. The scope of the project revolves around Orchestrator (Efficient communication between orch-agents to solve a task) not agents.

1.26 Structural Data Generation from Text

– Sarthak Harne

Structured data, in the form of tables, JSON files, YAML files are ubiquitous in data processing. There are many sources of text which contain structured information in the form of natural language in the form of summaries, reports and commentaries. Off-the-shelf language models do not have the understanding of structured data. This task requires semantic understanding, structured generation via reasoning. Text-to-Table (Wu, Zhang, and Li 2022) introduces special models focusing on relation embeddings to extract tabular information from text. However, creating an LLM or fine-tuning an LLM can be very expensive for such a task. Due to this, recent work has tried prompting methods specific to this problem on LLMs to extract structured data (Sundar, Richardson, and Heck 2024), (Deng et al. 2024), (Jain, Marzoca, and Piccinno 2024). Along with the recent developments with prompting (Madaan et al. 2023), we approach this problem by breaking this complex problem down into simpler ones so that it becomes solvable for an LLM along with techniques in self-refinement. Preliminary results have brought us close to fine-tuned LLMs (Wu, Zhang, and Li 2022).

Another problem we face with structured data generation is evaluation. As there a lot of different possible data types and data formats, exact match might not be a good metric to judge a model. For data types like numbers, metrics like BERTScore are not suitable. Metrics like P Score and H Score were introduced (Tang et al. 2024) to judge the structure of generated tables. We also aim to use models like MATE (Eisenschlos et al. 2021) to introduce metrics like MATEScore to evaluate generated tables.

1.27 AutoLLMReviewer: Code Explanation Evaluator

– Soumya Teotia

Understanding complex codebases often hinges on the clarity and completeness of accompanying comments and explanations. However, manually written annotations can be inconsistent or outdated, hindering developer productivity. This project introduces AutoLLMReviewer, a novel approach using Language Language Models (LLMs) as an evaluator to review code explanations and assess their reliability. Current methods rely on metrics like BLEU, ROUGE, and METEOR (Evtikhiev et al. 2023) to assess machine-generated code explanations. These metrics, while useful, often overlook semantic equivalence and may use outdated reference data from software repositories.

AutoLLMReviewer is structured into two primary phases: one evaluates the code explanation themselves, while the other ensures the robustness and reliability of these evaluations. Using advanced Language Models (LLMs) proficient in both code comprehension and natural language processing (Fu et al. 2023), we instruct the LLM to analyze input code alongside its corresponding explanations based on predefined evaluation criteria. To ensure the credibility of our evaluations, we implement an iterative self-probing approach. This methodology continuously refines evaluations through LLMs using the COT (Chain of Thought) approach, ensuring consistent and thorough assessments. Throughout this iterative process, specialized probing modules scrutinize critical aspects of the code and its explanation, including key variables, algorithmic implementations, and input/output dynamics.

Beyond improving code comprehension, AutoLLMReviewer offers significant business benefits. It reduces onboarding time by automating documentation review, boosts productivity by maintaining consistent documentation quality across large projects, and scales effortlessly with project complexity.

1.28 In-Context Learning for Code Translation

– Samidha Verma

Code written in legacy code needs to be translated to modern programming languages in order to allow scalability and maintenance. Most programmers at present are unaware of the languages in which the legacy code is written. Moreover, modern languages provide community support, enabling faster and more efficient development with the help of peer-reviewed packages. Translating huge code bases with the help of subject experts will be labor- and time-intensive and infeasible. Thus, we need neural machine translation for which we can use large language models (LLMs).

An emergent ability of LLMs (Wei et al. 2022) is in-context learning, i.e., the capability to perform a new task by "learning" from the input context without gradient update. An LLM's output can be improved by carefully curating the instruction text, using appropriate in-context examples, or both. This work focuses on improving translation quality by means of in-context examples. Thus, there is a need for a retrieval system to select appropriate examples from a corpus containing support examples such that the translation quality of the LLM for a new query improves.

A retrieval system for selecting in-context examples can be designed using LLM-agnostic or LLM-aware ranking. The LLM-agnostic ranker looks at the surface-level similarity between the query code and the examples in the corpus. This approach does not guarantee that the translation quality will improve significantly, as similar codes may lead to poor translation quality, and dissimilar codes may lead to better translation quality. An LLM-aware ranker considers the actual translation quality when LLM is given certain in-context examples in conjunction to the query. An LLM-aware retriever can be designed by training a bi-encoder

model that treats the in-context examples as independent of each other, or train a reinforcement learning based retriever that models this dependence. Both approaches have some pros and cons

The plan ahead has two steps. Step 1 is to identify whether there is a need for an LLM-aware retriever for coder translation, and Step 2 is to actually design this retriever either by RL or using Biencoder models. We have identified certain benchmark datasets such as AVATAR, CodeTransOcean etc that also have translation data among other language pairs. Apart from that we also have COBOL to Java translation corpus at IBM. We plan to test our approach on Granite-code models, CodeLlama, CodeGemma and StarCoder.

1.29 Half-truth detection

-Satyam Shukla

Half-true information refers to statements that are partially accurate but omit crucial details, leading to a skewed or misleading interpretation. These statements do not contain outright false information; rather, they present facts in a way that distorts the overall truth by leaving out key context. This problem is akin to issues in pragmatics, where the understanding of context is essential for accurate communication. Half-truths can be particularly deceptive because they exploit the fine line between truth and falsehood, making it challenging to discern their misleading nature without a deep understanding of the broader context in which they are presented.

Earlier work in this domain includes research by (Estornell, Das, and Vorobeychik 2020), who explored how an adversary can influence a principal's decision through half-truths by strategically masking or hiding bits of information. Their findings indicate that while the optimal attack strategy is complex and NP-hard to approximate, there are specific cases where efficient solutions can be found. Another significant contribution is the paper (Sandeep and Bhattacharyya 2023) This study introduced a comprehensive pipeline to detect and debunk half-truths using advanced models like T5 for controlled claim editing. Their methodology achieved impressive results, including an F1 score of 82% for half-truth detection, outperforming other models and setting a new benchmark in the field. These efforts underscore the complexity and importance of addressing half-truths in the era of information overload, demonstrating that while challenging, effective detection and mitigation strategies are achievable with advanced computational techniques.

1.30 Improving Code Generation Models using Feedback Learning

-Sidharth Sinha

The automation of code generation and bug-fixing processes through large language models (LLMs) has shown significant potential since their emergence as powerful reasoning agents in 2020. Despite this promise, a critical challenge remains in addressing the covariate shift issue, where models pre-trained on diverse external datasets struggle to adapt to specific in-house codebases. Initial experiments with fine-tuning on the HumanEval dataset yielded only a marginal 5% increase in accuracy, revealing the limitations of current fine-tuning methods.

This research aims to propose a novel approach to overcome covariate shift by incorporating feedback-driven training loops. The methodology involves allowing models to generate code, receive feedback on errors, and iteratively refine their outputs. This process is designed to produce training data that closely matches the model's original pre-training experiences and effectively targets specific bug patterns.

Direct Preference Optimization(Rafailov et al. 2023) offers a cheap alternative to quickly align LLMs. We plan to extend previous work to customize Direct Preference Optimization, to incorporate multi-candidate ranking instead of just binary.

The goal of this research is to establish a robust pipeline integrating these methodologies, particularly focusing on the agentic workflow. By enhancing the efficiency and accuracy of automated code generation and bug-fixing, this approach aims to advance the field of AI-driven software engineering.

2 Hybrid Cloud

2.1 Training Time Estimation

- Aryamaan Thakur

Accurately estimating the training time of machine learning models will be critical for efficient resource allocation and project planning. This project will aim to develop a predictive framework for estimating the training duration of models based on key parameters such as model architecture, number of GPUs, batch size, training strategy (Zhao et al. 2023) and various hyperparameters. We will propose a method to predict the training time before actual model training commences. As of now, we are focused on building this system for open-source models like Granite (Mishra et al. 2024).

2.2 Distributed Training for LLMs: A Simulation based approach

-Arjun Temura

There has been an increasing use of Large Language Model (LLMs) like ChatGPT for education, content creation, and doing repetitive tasks. LLMs have shown exponential growth in their model sizes over time. For instance, ChatGPT-3.5 currently uses 175B parameters to train their models, 1000 times of what it used to be 6 years back. Models have become more compute and memory hungry and eventually require more time to train. Training LLMs on single GPUs is not feasible. For example, Model size of ChatGPT-3.5 is much larger than the capacity of state-of-the-art (SOTA) GPUs. Therefore, LLMs are trained over multiple GPUs. Many SOTA parallelisation techniques exist to ensure distributed training of LLMs across GPUs like DeepSpeed and FSDP. Prior research shows that the choice of parallelisation technique can significantly effect training performance (Zheng et al. 2022). However, choosing the best technique for your model is a challenge. We plan

to develop a simulator that runs on some minimal hardware and will return a profiled report of our model (compute/network/memory overhead) under different parallelisation strategies. Our work will cater to two broad categories of users. Firstly, Developers will be able to decide which parallelization strategy is best for their model. Secondly, researchers can test their novel parallelization algorithms against the SOTA techniques.

2.3 Supporting QoS Class with LLM Inferencing

-Anubhav Jana

People use LLM inferencing which runs primarily on GPUs. Different kinds of users have different kinds of SLO requirements in LLM inference for different LLM Models. For e.g. for text summarization, longer initial overhead is acceptable as long as long output texts can be generated quickly whereas for e.g. in chatbot – output should start getting generated quickly, while the speed of subsequent tokens does not need to exceed the human reading speed. The high cost of GPU's prohibits dedicated GPUs for a user to meet their requirements. In the context of large language model (LLM) services, multiple user classes need to be supported, each characterized by distinct quality of service (QoS) requirements such as latency. Users generate a pre-specified load in terms of requests per minute (RPM) and the average number of tokens per request. The challenge is to efficiently allocate resources, specifically GPUs, to meet SLO requirements while minimizing costs and energy consumption.

2.4 Universal Abstract Syntax Tree for Code Data Profiling

-Adnan Qidwai

As software systems today have become increasingly complex and diverse, understanding the structure and behavior of such codebases is crucial for abstracting relevant information to build code models. However, traditional approaches to code analysis rely on language-specific grammars and syntax, limiting their applicability across different programming languages and ecosystems. We propose a framework for constructing a universal abstract syntax tree (UAST) that enables language-independent code data profiling. Our UAST captures high-level structures and relationships within software systems, abstracting away from language-specific details. Using our approach, we demonstrate how to profile code data in a language-agnostic manner (Liu et al. 2021), providing insights into code concepts, issues, API calls and package interactions. This enables developers and analysts to analyze complex software systems without requiring prior knowledge of specific programming languages or ecosystems.

2.5 Semantic Analysis of Code for Data Profiling and LLM customisation

-Adrija Dhar

Existing work in the fields of code analysis, data profiling, and Large Language Model customisation has primarily focused on syntax-based approaches, which often lack the depth needed to fully understand the meaning and context of code. Tools and techniques in static and dynamic analysis have been developed to parse and analyse code structures, but they often fall short in capturing the semantic nuances that are crucial for accurate data profiling and effective LLM customisation.The motivation behind this research project is to address the limitations of current approaches by developing a robust framework for semantic analysis of code that is language-agnostic, utilising the UAST representation. By going beyond syntax-based methods, we aim to capture the deeper meaning and context of code, which is essential for accurate data profiling and the customisation of LLMs.Our problem statement breaks down into three key components: developing a robust, language-agnostic semantic analysis framework for code using the UAST representation, applying this framework to data profiling to extract meaningful insights The integration with LLMs will involve customising these models to leverage the profiled data, improving their performance and context-awareness in various applications.

2.6 Automating Functional Testing

-Lakshita Saxena

Functional testing is a vital component of an application's lifecycle, as it ensures that the software meets its functional requirements and performs as intended. Traditionally, these tests are carried out manually or converted into automation scripts, both of which can be time-consuming and prone to errors. Automating the generation of test scripts from natural language instructions using large language models (LLMs) can significantly improve efficiency and accuracy. The challenge lies in three key stages: generating test instructions from feature descriptions, converting these instructions into automated scripts, and executing these scripts to validate functionality. Initially, input documents are preprocessed to extract natural language descriptions of application features, which are then transformed into test instructions. Next, LLMs use these detailed instructions, along with prompt guidelines, sample Gherkin tests, and application metadata, to produce accurate Gherkin tests that can be converted into executable scripts. Finally, tools like Selenium, Cucumber, and Galasa are employed to run these scripts, ensuring the application's functionality is thoroughly validated. This automated approach streamlines the testing process, reducing manual effort and enhancing reliability.

2.7 Next Configuration Recommender

- Atharva Deshmukh

Next Configuration Recommender is a framework to recommend configuration for ongoing and successive training runs. Imagine a scenario where a client, who lacks a background in machine learning, approaches us with their own dataset, model and training logs. They are eager to optimize their model's performance, but are unsure how to proceed. Our solution provides a framework that can take the client's training logs and loss curves as input, analyze the training process, and provide actionable insights for hyperparameter tuning. Our framework will help the client understand

the current state of their model's performance, but also provide recommendations for hyperparameter adjustments that can potentially improve the model's accuracy and efficiency. This project involves multiple steps starting from, analysis of all hyperparameters, then analyzing various Hyperparameter Optimization Techniques, propose a heuristic captures the relationship and finally creating a framework which recommends next best configuration for the set of hyperparameters. Currently the framework is being optimized for transformer models. Our next configuration predictor is different than traditional HPO (Yang and Shami 2020) methods in few ways:

- Traditional HPO techniques require a significant amount of computational resources. Our approach, however, aims to streamline this process by focusing on analyzing the loss curve rather than exhaustively searching through all possible hyperparameter configurations, thus saving on the computational resource as well.
- Traditional HPO methods consider value loss at a single point in time, whereas our method takes into account the entire loss curve. By examining the overall trend and shape of the loss curve, we can gain deeper insights into the performance of the model and make more informed recommendations for the next hyperparameter.

2.8 Refactor Testing

- A Eashaan Rao

Context: In software development, there are two common activities, i.e., code refactoring and software testing. *Code refactoring* involves modifying source code to improve its organization, readability, and maintainability without altering its external behavior or functionality. The goal is to enhance the code structure without impacting the final output. Whereas, *testing* involves evaluating and verifying software modules by writing test scripts to identify errors. If the software's functionality does not match the expected output, it results in test case failures.

Problem Statement: *Refactor Testing*
It refers to test case failures that occur due to changes made during code refactoring. While refactoring is intended to leave the final output unchanged, a few cases where it can cause issues due to:

- *Method Signature Changes*: Altered method signatures can cause failures.
- *Return Type Changes*: Different return data types can lead to exceptions.
- *New Functions*: Breaking long methods into new functions might cause compilation errors.

There are two main scenarios when dealing with refactor testing: a) refactored code is correct and compiles successfully, but test cases need repair to accommodate changes, b) test cases are correct and the refactored code contains issues that need fixing. Currently we are focusing on the earlier scenario and to be precise it become as test repair problem. Test Case Repair involves modifying test scripts to align with changes in the source code. This can include modifying existing test scripts, adding new test cases or deleting obsolete test cases.

To tackle this problem, our first task is to create a dataset where we are gathering the instances where test cases fail after code refactoring in open source Java projects. We are searching for refactoring commits and map them to affected test cases. However, one of the challenge is to setting up Java projects and resolving dependency issues.

Our proposed approach aims to leverage LLMs and program analysis. While LLMs can try to address various test case scenarios that current approaches handle separately, LLMs can produce incorrect code. However, it had demonstrated improvement in generating diverse test cases with oracles by balancing the intent preservation with making necessary changes. We are trying to use program analysis for accurately identifying, locating, and repairing simple test case issues and use LLMs for more intricate test repairs.

2.9 From ML Experiments to Better ML Experiments

- Anoushka Nag

Problem: WatsonX.Ai is among the best products for customers to train, fine-tune, and perform inference on their own or open-source models. Enterprises and customers use the WatsonX.Ai platform to fine-tune their models. Fine-tuning tasks are complex and often require multiple experiments to achieve desired results, such as reducing loss below a certain value. With various hyperparameters, the search space can quickly expand to thousands of experiments, making it essential to maintain detailed information about them. Sophisticated trackers such as Aim, Wandb, and MLflow facilitate this process. As we conduct thousands of experiments, we collect vast and complex data across the stack. This project aims to find a robust way to use this data and leverage it to gain valuable insights. These insights would be invaluable to customers, accelerating their product development lifecycle and giving WatsonX.Ai an edge over competing products.

Introduction: Our project will focus on transforming raw observability data, which includes AI metrics (e.g., loss over time), system metrics (e.g., GPU usage, memory usage), and metadata (e.g., model card, data card, environment) collected by trackers like Aim (Arakelyan, Soghomonyan, and The Aim team 2020), WandB (Biewald 2020), and others, into actionable insights. These insights can be used to predict model behavior during training, monitor resource usage throughout experiments, and identify ways to reduce resource overhead.

By converting raw data from tracker databases into a common format, we can apply transformations that generate derived metrics or simplify complex data into easily understandable values. This processed data is then stored in a datastore, such as a lakehouse, which can be accessed by clients or IBM directly.

Our project aims to generate actionable insights based on empirical data from real-world runs. These insights can guide users in tuning hyperparameters for faster conver-

gence or selecting model configurations that yield optimal results. By presenting these insights in an easy-to-understand and actionable manner, users can visualize their data, track performance over time, and identify areas for improvement.

The core objective of this initiative is to enhance the user experience. By providing clear, actionable insights, we empower users to make informed decisions quickly and confidently. This leads to more efficient model tuning, faster development cycles, and ultimately, better-performing models.

2.10 GPU Power Characterization

- Tirth Vamja

Introduction : Power consumed by GPU depends on various parameters like Thermal Conditions and utilization of various hardware components like CUDA(FP32 and FP64) and TENSOR cores and memory operations/accesses(L1 and L2 cache hit/miss). Based on the utilization of these components power consumption varies.

Purpose : Characterizing GPU power can help in modelling the power consumption pattern of GPU based on utilization of hardware components and memory operations.

Use Case : It can be useful in formulating the power associated with individual MIG (Multi Instance GPU) partition. (since there is no direct method available to get the power utilized by individual partition.)

2.11 Optimal Training Scripts Configurations Study

-Aaqilah A J

Training LLMs is a resource-intensive task that requires effective strategies. We are exploring various PyTorch training scripts, each with its unique approach to data processing, memory management, and system performance optimization. For instance, data processing techniques like padding, tokenizing, and packing are crucial for preparing the data efficiently. Memory and GPU optimizations involve using tools like Fully Sharded Data Parallel (FSDP) and DeepSpeed, which help distribute the workload across GPUs and boost performance with advanced algorithms. System optimizations, such as Offloading, dynamically balance workloads between CPU and GPU, ensuring optimal resource utilization. Our project rigorously tests these components to identify the best combination of settings that deliver superior performance.This involves matching configurations on a semantic level, leveraging computational graphs to ensure consistent settings across scripts.

Moreover, we address the compatibility of different tools with various infrastructures. For example, if your team prefers DeepSpeed but your company's setup is based on FSDP, our project provides a roadmap for seamless transition and equivalent performance. To illustrate, IBM's instructlab uses DeepSpeed-based training scripts, while their Full Model Stacks are built on FSDP. We ensure these scripts can be ported smoothly between the two systems without compromising performance.

In essence, our project aims to make the training of Large Language Models faster, smarter, and more efficient. By optimizing data processing, memory and GPU usage, and system configurations, we enable seamless adaptability and maximized performance in AI development.

2.12 Adaptive Filter Processor for OpenTelemetry

-Subhash Veggalam

Background: Especially in multi-cloud deployments, the number of applications running is high and the amount of telemetry data generated from them could be so huge. All the telemetry data can't be transmitted due to limited external network bandwidth. Critical telemetry data could be dropped off which isn't ideal.
Problem Statement: The goal is to transmit telemetry data efficiently without losing critical observability data in resource-constrained environment by adaptively filtering the metric and log collection based on external inputs. So, proposing a new transform for open-telemetry.
Adaptive Filter Processor dynamically filters the telemetry data based on the available budget and prioritising metrics or logs that are most relevant or indicative of critical events.

2.13 Alert Recommendation System

-Akanksha Singal

Alerts monitor the state and behavior of a system. It captures specific events that deviates from the normal or expected state. In the context of observability, alerts are automated notifications triggered by specific conditions or thresholds within the system being monitored. We need alerts to inform that something requires immediate SRE attention. Alerts are required to detect and trouble shoot errors or anomalies in a system before they become critical. They ensure the system runs smoothly and efficiently and proactively responding to issues by routing issues to appropriate teams. They are also is essential in capturing why a service fails and isolating the root cause of failure. However, Alerts are usually defined manually and require domain knowledge expertise. It is challenging to determine which alerts to set among millions of observability data types. This results in poor coverage, false positives, excessive noise and no dynamicity as the system is evolving. The current state of the art include IBM's Instana smart alert and New Relic which focuses more on the rules. MSR Intelligent Monitoring Framework for Cloud (Srinivas et al. 2024), recommends resource classes and SLO classes on which monitors should be defined. Metrics are one type of observability data type that is a measure of application and system health over a given period of time. We propose a solution by monitoring metrics and recommending which metrics should have alerts and further designing rules on these metrics to set alerts. We are using an Entropy Driven approach to select important telemetry using entropy calculation. SREs would benefit from system recommended alerts that would help meet the target SLOs.

2.14 A Time-Profiler for LLM Fine-tuning Jobs

-Yash Mehan

Given the increacing sizes of LLMs, fine-tuning LLMs can take a lot of resources - GPUs and time. Adding to the complexity of the problem is that there are many techniques for Fine-tuning (such as Full FT vs PEFT methods) and for running multi-gpu jobs (FSDP, DeepSpeed with the various configurations). Thus, it is desirable to have a system to predict the time a job configuration would take, before even starting the job. Towards this goal, the aim of this project is to build a Time Profiler to measure time taken by the various sub-process of training - be it data pre-processing, actual training (forward and backward passes), optimizer updates and communication costs in case of multi-gpu jobs. This will allow us to judge the impact of various configuration knobs on the overall time taken and enable better Fine-tuning job time predictions.

2.15 *Developer Experience of ML pipelines*

Several frameworks and libraries are available today for training and tuning of various ML models. For enterprise level model training and tuning activities, given the scale and processing complexity, programs are typically organized into a chained series. Each step in the chain acts as a discrete data processing phase. The output of one step in the program becomes the input to the next. These programs are called data pipelines (when only data processing steps are involved) or ML pipelines (when training and fine-tuning steps are also involved). The steps in an ML pipeline may deal with large amounts of data and thus execute over several hours or days. The compute and data complexity may also not be evenly distributed across the steps of a pipeline. Further, individual steps may fail or run out of resources.(Xu et al. 2021) To handle all these complexities a pipeline orchestrator module is deployed for successful execution of long running ML pipelines. Also, each individual step is often packaged into a container so that the pipeline can be deployed in a distributed fashion. As a result, the developer experience offered today to ML Engineers for creating and deploying enterprise scale ML pipelines becomes complicated. First, they need to be aware of and provide configuration for infrastructural details such as number of nodes and type of compute/memory/storage resources available. Second, they need to program meeting the specification prescribed by the chosen orchestrator framework. Third, the pipeline specification is complex due to the previous two requirements. Fourth, the journey from a simple process while working on local machine to migration for a distributed deployment is non-trivial.

(i) Local Execution

- Penumalla Aditya Pavani

We aim to simplify the developer experience by addressing requirements two and three above. In other words, the goal is to allow ML engineers to define ML pipelines effortlessly with a simplified pipeline specification language either prescribed by the orchestrator or providing a compilation step to target specification.

(ii) Distributed Execution

- Kanishka Bansode

This project aims to simplify the developer experience for an ML engineer by addressing problems one and four above. In other words, the goal is to abstract out the resource configuration in simplified manner and enable smooth transition from local execution to a distributed execution on clusters. It seeks to achieve this without requiring significant changes to the pipeline operators, or the pipeline definition.

3 Quantum

3.1 *de novo* genome sequencing on quantum computer

- Himanshu Sahu

Background New-generation sequencing (NGS) technologies have led to a massive increase in fast sequencing projects and the development of new methods to assemble short DNA sequences. However, there are still many technical and computational challenges in putting together these sequences from scratch. Although many new ideas and solutions have been proposed to address these problems in both lab and computer-based settings (Liao et al. 2019; Boev et al. 2021).
Purpose The assembly of a genome from billions of short DNA sequences, or reads, can be simplified to finding the Hamiltonian path on the overlap-layout-consensus (OLC) graph. The Hamiltonian path problem belongs to the complexity class of NP-complete problems, making it challenging to solve using classical computers. Previous literature, such as (Fang et al. 2024), has explored the possibility of achieving a quantum advantage for solving these problems. However, these approaches still lack efficiency in terms of space and time. In this work, we aim to overcome these limitations and experimentally demonstrate genome assembly on an IBM quantum processor.
Method Classical preprocessing is used to extract reads from raw data and construct the OLC graph. The problem is then framed as a combinatorial optimization problem, and a Hamiltonian is constructed. We use the variational quantum eigensolver (VQE) framework to find the optimal assembly sequence.
Contributions We proposed a qubit-efficient method based on HOBO encoding for genome assembly and demonstrated it on an IBM quantum processor. We also characterized various VQE ansatz for the problem.
Outlook In the near future, improving variational quantum algorithms and exploring their advantages will be key goals in the field of quantum computing applications, especially in bioinformatics.

3.2 *Geometric Quantum Machine Learning*

- Pingal Pratyush Nath

Brief Introduction : Integrating Representation theory into Quantum Machine Learning has led to the field of Geometric Quantum Machine Learning (GQML) (Larocca et al. 2022; Meyer et al. 2023; Ragone et al. 2022). GQML models can classify large datasets of symmetrically related elements with advantages such as avoiding barren plateaus, overparameterizing with polynomial deep circuits, and generalizing well with fewer training points(Schatzki et al. 2024).

Problem Statement : Our goal is to explore more graphical datasets and develop symmetry-invariant models for continuous symmetries, such as SO (2) rotations. One promising method for building such an ansatz is Pauli Twirling, which involves an operation over all group elements. We also aim to leverage other techniques from representation theory to build better symmetry-invariant models.

3.3 *AI4QuantumOps*

-Debasmita Bhoumik

The AI4QuantumOPs project addresses the challenge of efficiently analyzing and summarizing logs generated by hybrid quantum algorithms running on IBM's quantum stack, which includes quantum hardware, software, and middleware layers. Currently, resolving errors from these layers is time-consuming, requiring manual sifting through extensive log files, often taking up to 30% of the total resolution time. With millions of log lines generated within minutes during platform issues, this manual process hinders the support and SRE teams' ability to quickly diagnose and resolve problems. The project's goal is to leverage AI to assist various IBM Quantum teams in rapidly obtaining human-readable summaries of issues, thereby reducing the time to resolve (TTR) and improving overall efficiency. By training a Large Language Model (LLM) to analyze logs and generate concise summaries, the project seeks to reduce log analysis time from hours to minutes, enabling the SRE team to quickly understand issues and provide potential solutions to users. This will lead to increased productivity, cost savings, and enhanced scalability of quantum cloud services. The project involves collaboration with AI OPS, SRE, and Quantum Support teams to fine-tune AI models for quantum-specific logs, ultimately creating a user-friendly portal for automated log summarization and solution suggestions.

3.4 AI4QuantumOps

-Aaryav Mishra

Running tests on quantum machines involves multiple infrastructure layers, on the private user end, quantum hardware, and the Kubernetes pods on the global cloud. To run these machines, data is hosted on several services within this network. Unlike the errors affecting classical computational tasks, quantum errors when encountered affect a broad service system, debugging these errors is crucial for the widespread application of these machines. The debugging process is characterized by the involvement of multiple teams, namely the Quantum Support team, at the user end, who help with errors within the user's code, the SREs, or Software Reliability Engineers, who deal with errors on the Qiskit runtime, and the DevOps and Infra teams, who deal with errors on the cloud, which includes the Kubernetes pods where the entire infrastructure is hosted. The interpretation of the extensive log lines generated after the failure of computation takes up to 5-6 hours, thereby increasing the mean TTR(Time-to-Resolve) of the encountered errors. The proposed framework seeks to address this very problem of high mean TTR due to error interpretation, using automation. We seek to generate a classification of error types through golden signal classification within a refined set of loglines, further generate a service map, which is a knowledge graph representation of all services that are affected by one another, and finally generate a short readable summarization using an LLM all in a consolidated model. Hence, generating the relevant interpretations for each of the involved teams and thereby increasing overall productivity and decreasing costs.

3.5 Circuit Equivalence using ML

-Shobhit Pandey

Context : When we want to implement any Quantum Algorithm, we make a Quantum Circuit, which is a series of unitary operations that we do on qubits. On real hardware the Quantum Circuit the user creates, must first be compiled to a low-level representation that is compatible with the hardware. This process is known as transpilation. One issue that we can face is to tell if the new circuit operates similar to the circuit designed by the user. That is where circuit equivalence comes in. This problem has been solved for big clifford circuits (circuits made up of clifford gates, note, the clifford set is not universal) and small general circuits, however, the case of 2 large circuits from a universal set is yet to be solved.

Problem Statement : Given a pair of arbitrary Quantum Circuits, our job is to tell whether they are the same or not. This is, given any possible statevector, the action of the quantum circuits should be identical on it. Currently, we are leveraging the Directed Acyclic Graph representation of Quantum Circuits and Machine Learning to tackle this problem.

3.6 Quantum Safe Explorer Differentiators

-Jitbitan Baroi, Krish Agrawal, Suchi Sharma, Utkarsh Bhatt

Introduction The *Quantum Safe Explorer* constitutes a sophisticated tool designed to scrutinize application source codes through the identification of cryptographic usages, the detection of vulnerabilities, and the facilitation of remedial actions. Its primary objective is to conduct an examination of cryptographic implementations within applications, a process essential for ensuring their safe and secure deployment. This tool distinguishes itself by employing a methodological focus on static code analysis, thereby offering a novel approach to the assessment of cryptographic security within software applications.

Purpose At present, no cryptographic discovery tools equivalent to Explorer exist, which employ comprehensive programmatic static analysis methodologies at the forefront of technological advancement. Unlike API hooking and network scanning, source code scanning represents the solitary approach facilitating immediate remediation at the code level.

Implementation The basic operation of the QS Explorer is as follows:

The Explorer starts by scanning source codes in the applications, breaking them down into an Abstract Syntax Tree

(AST). The variable types and signatures of all methods are all resolved subsequently. Imported methods are handled explicitly. Each variable in their respective scopes is analyzed from the AST (a graph model) with the flow of the code. A Call Graph is managed separately and represents function calls. The explorer examines each function call by its name and signature in its knowledge base. If a function call node maps to a knowledge base, the QS Explorer crawls backward to find previous statements. This backward tracing makes slice transitions, and the explorer can extract key properties like the algorithm used and key length through these slices.

3.7 Smart Hardware Selection for Quantum Computing

-Ishant Kohar

Quantum computers hold immense potential for solving problems beyond the reach of classical computers. However, the current quantum devices are noisy, with various parameters characterizing different sources of noise. Selecting the optimal hardware to execute quantum circuits is challenging for users unfamiliar with these parameters. Additionally, due to high demand, there is often a significant queue for accessing hardware, especially those with lower noise profiles. Users face a critical decision: whether to wait longer for potentially better devices or to obtain quicker results on suboptimal ones, and how much the results may deviate if they choose the latter. This project aims to assist users by providing information on queuing times for available hardware and estimating the degradation in outcome quality when selecting sub-optimal hardware. This helps users balance the trade-off between waiting time and result quality.

3.8 Quantum Error correction

-Sannzay Gaddam

Motivation: As it is widely known and understood that Quantum information which is contained in the building blocks that are qubits, is very fragile and is prone to all sorts of noise and there is only so much we can do intrinsically to mitigate the errors. And with all the errors floating, quantum computers turn in garbage as results. So this predicament catapulted people working in the domain to protect the quantum information using error correcting codes, taking inspiration from classical error correction techniques. Although classical error correction is long established, the Quantum novelty associated with quantum information makes it non-trivial to generate highly efficient codes. The research is ongoing and people around the world are working on multiple classes of codes to achieve fault tolerance. This would let us go from the NISQ era to Utility scale era.

Work: We're currently working on a particular QLDPC code construction, namely, gross code $[[144, 12, 12]]$, which belongs to the family of Bivariate Bicycle codes. This work is an extension on the recently published paper by IBM on error correcting codes. We are trying to implement Syndrome decoding using machine learning and neural network techniques on the said code.

4 Security

4.1 Timed Cryptography

- Simran Kumari

A timed cryptographic primitive allows us to send messages to the future. This is useful in scenarios where we would want to delay disclosing a sensitive information until a time bound T for instance in electronic voting, closed-bid auctions or generating randomness in a distributed setting. Time Lock Puzzles (TLP) and Verifiable Delay Functions (VDF) are two fundamental primitives in this domain. In a recent work (Lai and Malavolta 2024) the authors propose a sequential function based on lattices and consequently an accepted submission in CRYPTO24 by Malavolta et. al. have built TLPs from lattices. The focus of this internship is to build the notion of VDF from quantum safe assumptions and in particular from a lattice based sequential assumption. All the currently known works on VDFs are not quantum safe.

4.2 Crypto Agility

- Ananya Mishra

We currently use symmetric and asymmetric encryption for security. However, the advent of quantum computing has galvanised the global security community into action, as it poses a significant threat to current encryption methods. Olaf Grote et al. talk about this in detail (Grote, Ahrens, and Benavente-Peces 2019). The world will soon need to move to quantum-safe encryption techniques. The first step in that transition is discovering. Think of it like repairing a damaged water reservoir: before you can fix the leaks, you first need to locate where they are. Similarly, before you can implement quantum-safe encryption, you need to identify all the points where cryptographic methods are used in your systems. This is where we come in. Now that we know what our goal is, let us understand the challenges we need to overcome. All real-world systems have telemetry and operational logs. However currently, these details are scattered across various libraries and key stores. Also Operational data is time-series. Data from different sources might be recorded at different intervals, making it difficult to correlate events accurately. The configuration of data sources might not be consistent across the system as they were developed independently. Our goal is address these challenges to create a common data model , the Integrated Data Model (IDM), a unified view that informs users about the cryptographic standards used in different parts of their application or product. Additionally, we are developing a correlation engine to aggregate and correlate data from various sources to populate the IDM. Our efforts are aimed at designing a model that is future-proof, easy to extend, and capable of evolving to serve multiple purposes.

4.3 Zero Knowledge from Lattices

- Shubh Prakash

Zero knowledge proofs are useful in a wide variety of applications, for instance, blockchain technology. ZK allows lots of expensive computations to be verified very efficiently

using a short proof. For this internship, the focus is specifically on building commitment schemes, as these are the main building blocks of all the ZK proofs. From commitment schemes one can achieve more advanced primitives like accumulators and lookup arguments. There are a lot of efficient commitment schemes known in the literature. However, they are not secure against Quantum algorithms. Our aim is to get efficient commitment schemes with nice properties which are based on quantum safe assumptions. Specifically, we want a commitment scheme which resembles KZG (Kate, Zaverucha, and Goldberg 2010) in its properties and is based on lattice assumptions. Lattice assumptions are the most promising among the various (believed) quantum safe assumptions. This is because the hardness of the average case lattice problems useful for cryptography can be obtained from the hardness of certain lattice problems in the worst case (Ajtai 1999). There has been a lot of recent work on lattice based zero knowledge such as (Attema, Cramer, and Kohl 2021), (Lyubashevsky, Nguyen, and Plancon 2022) and (Beullens and Seiler 2022). However, none of these works directly leads to a commitment scheme which is efficient and has nice properties like KZG.

4.4 Reinforcement Learning for Design of Optimal VQC's

-Vandit Shah

Currently we are in noisy intermediate scale quantum (NISQ) era and Variational Quantum Eigen Solvers VQE's are proving to be an effective approach as they may lead to real world applications of near-term quantum devices.The optimising VQE Ansatz relies on the depth and expressivity of the ansatz, we require maximum expressivity of ansatz for minimum depth possible. There are many methods for VQE structure but the area of utilising Machine Learning for the problem is not explored much. Taking forward the work of (Ostaszewski et al. 2021) Our aim is to achieve the application of reinforcement learning that autonomously search the vast ansatz space for any given problem and also to deliver a strong understanding of which reinforcement learning technique works. We have defined a very simple yet effective reward function. We are using DDQN RL algorithm as of now and most effective strategy until now is previous best circuit initialisation of environment. Also after determining the deterministic nature of VQC (i.e given a VQC the performance measure (in our case f1 score) remains nearly constant over many runs.) we are probing various state representations for ansatz that aid RL to search the VQC space efficiently and effectively, uptill now we have used represenation from (Ostaszewski et al. 2021), (Kipf and Welling 2016), Tensor Based Encoding (TBE's) from (Kundu 2024) and Learnable Vector Embedding.

Acknowledgements

HS would like to thank Kalyan Dasgupta, Jaya Vasavi P, Ashwini K, and Siddhi for various useful discussions and comments about this and related works.

Arjun Temura would like to thank Seep Goel, Priyanka Naik, Kavya G. and Chander G. for their invaluable discussions and guidance.

Lakshita Saxena would like to extend heartfelt thanks to Atul Kumar, Saravanan Krishnan and Diptikalyan Saha for their invaluable insights and support.

Atharva Deshmukh want to express my deep gratitude to Padmanabha Venkatagiri Seshadri, Akash Nayak, Harikrishnan Balagopal and Ashok Pon Kumar Sree Prakash for their invaluable guidance and support.

Pradhuman Tiwari would like to thank Ankush Gupta for various useful discussions and comments about my project and related works.

Kushagra Bhushan would like to thank Yatin Nandwani, Dinesh Khandelwal, Dinesh Raghu, for providing a space where my ideas are heard and giving me insights and guidance that have been of great help during this project.

Simran Kumari would like to thank Sikhar Patranabis for his guidance and insightful discussions on this project.

Vishal Kumar would like to thank Himanshu Gupta for fruitfull discussions and esteemed guidance on this project.

Sameer Pimparkhede would like to thank Mehant Kammakomati, Srikanth Tamilselvam, and Prince kumar, for providing a space where my ideas are heard and giving me insights and guidance that have been of great help during this project.

Aaryav Mishra would like to thank Anupama Ray for providing me with excellent insights and guidance at every step of this project.

Aditya Tomar would like to thank Rudra Murthy for various useful discussions and comments about my project and related works.

Sidhaarth Murali wishes to extend profound thanks to Vishwajeet Kumar, Rudra Murthy, and Jaydeep Sen for their exceptional guidance and steadfast support in making this exciting project possible.

Ananya Mishra would like to thank Akshar Kaul and Abhishek Singh for their constant guidance, patience and expertise in the field. Their support has been invaluable not only for this project but also for my personal and professional growth. Working alongside them has been a profoundly enriching experience, fostering significant learning and development.

Anoushka Nag would like to extend my sincere gratitude to Dushyant Behl and Ashok Pon Kumar Sree Prakash for their invaluable guidance and unwavering support.

Vivek Mathur would like to express my sincere gratitude to my mentors, Manikandan Padmanaban and Ayush Jain. Additionally, I extend my heartfelt thanks to Jagabandhu Hazra and the Climate and Sustainability team at IBM Research Bengaluru. Their guidance, invaluable insights, assistance in refining code contributions, and support in validating and reviewing my progress have been instrumental throughout this project. I deeply appreciate their unwavering support and encouragement.

Adrija Dhar would like to extend my sincere gratitude to Pankaj Thorat and Aishwarya Chakrabarty for their invaluable guidance, mentorship, and unwavering support throughout my project.

Ameya Srivastav would like to extend my heartfelt thanks to Suranjana Samanta, Oishik Chatterjee and Amar Prakash for their invaluable insights and support.

Abhishek kumar singh would like to extend my sincere gratitude to Viswajeet kumar, Rudra Murthy and Jaydeep sen for their invaluable guidance and unwavering support for this exciting project.

Penumalla Aditya Pavani would like to extend my sincere gratitude to Arun Kumar, Vikas Agarwal and Dasari Surya Sai Venkatesh for their invaluable guidance and support throughout the project.

Kanishka Bansode would like to extend my sincere gratitude to Arun Kumar, Vikas Agarwal and Dasari Surya Sai Venkatesh for their invaluable guidance and support throughout the project.

Adnan Qidwai would like to extend my heartfelt gratitude to Pankaj Thorat and Aishwariya Chakraborty for their constant insight, support and expertise.

Deepika Ponnana would like to extend my sincere gratitude to Kushal Mukharjee sir for the invaluable guidance, mentorship, and unwavering support throughout my project.

Parthivi Gupta would extend my heartfelt gratitude to my mentors, Akella Ashlesha and Brij Chavda, Krishnasuri Narayanam and Abhijit Manatkar for their unparalleled support and guidance and helping me with key insights for the project.

Baban Gain would like to extend my sincere gratitude to Varad Bhatnagar, Saswati Dana, and Dinesh Garg for their consistent guidance and invaluable insights.

Shubh Prakash would like to extend my heartfelt gratitude to Nitin Singh and Sikhar Patranabis for their constant guidance and support and insightful discussions.

A. Eashaan Rao would like to express gratitude towards my mentors, Shivali Aggarwal, Sandeep Hans, Vini Kanwar and Devika Sondhi for their guidance, support and technical discussions to help me progress in my work.

Subhash Veggalam would like to express gratitude to Priyanka Naik, Seep Goel and Mudit Verma for their invaluable guidance and support in my work.

Shobhit Pandey would like to thank Dhiraj Madan, Anupama Ray and Venkata Subramaniam for their unwavering support, guidance and invaluable insight.

Shivangi Shreya would like to thank Karan Bhukar, Dinesh Raghu and Harshit Kumar for their invaluable guidance and support in my work.

Satyam Shukla would like to thank Rudra Murty for his constant support and guidance in solving the problem. With his input it is possible to address the issue effectively.

Samidha Verma would like to thank Vijay Arya for his invaluable guidance and insightful discussions on the project and how to approach a research problem in general.

Kiran Pradeep would like to express gratitude to Nishtha Madaan and Kirushikesh D B for offering exceptional insights and guidance throughout every stage of this project.

Ishant Kohar would like to express deepest gratitude to Anupama Ray and Ritajit, for their invaluable guidance, and encouragement throughout this project. Their expertise and insights have been instrumental in shaping the direction of the project.

Vandit Shah would like to express gratitude to Dhiraj Madan, Dhinakaran Vinayagamurthy and Anupama Ray for offering invaluable insights, guidance and support throughout every stage of this project.

Akanksha Singal would like to thank Mudit Verma for his invaluable support and guidance. Additionally, I extend sincere gratitude to Pratibha Moogi, Kaustabha Ray, and Felix George for enriching discussions, insights and learning opportunities.

Sidharth Sinha would like to thank Pooja Aggarwal and Oishik Chatterjee for their unwavering support, guidance and invaluable insight.

Priyanshul Govil would like to thank Arindam Jati, Pankaj Dayama, and Vijay Ekambaram for their support and encouragement.

Debarpan is grateful to Sumanta Mukherjee, Kamanchi Chandramouli, Vijay Ekambaram, Arindam Jati and Pankaj Dayama for their valuable guidance and mentorship.

References

Ajtai, M. 1999. Generating Hard Instances of the Short Basis Problem. In *International Colloquium on Automata, Languages and Programming*.

Aldowaish, A.; Kokuryo, J.; Almazyad, O.; and Goi, H. C. 2022. Environmental, Social, and Governance Integration into the Business Model: Literature Review and Research Agenda. *Sustainability*, 14(5).

Arakelyan, G.; Soghomonyan, G.; and The Aim team. 2020. Aim.

Attema, T.; Cramer, R.; and Kohl, L. 2021. A Compressed Σ-Protocol Theory for Lattices. Cryptology ePrint Archive, Paper 2021/307. https://eprint.iacr.org/2021/307.

Beullens, W.; and Seiler, G. 2022. LaBRADOR: Compact Proofs for R1CS from Module-SIS. Cryptology ePrint Archive, Paper 2022/1341. https://eprint.iacr.org/2022/1341.

Biewald, L. 2020. Experiment Tracking with Weights and Biases. Software available from wandb.com.

Boev, A. S.; Rakitko, A. S.; Usmanov, S. R.; Kobzeva, A. N.; et al. 2021. Genome assembly using quantum and quantum-inspired annealing. *Scientific Reports*, 11(1): 13183.

Bossman, A.; Gubareva, M.; and Teplova, T. 2023. Economic policy uncertainty, geopolitical risk, market sentiment, and regional stocks: asymmetric analyses of the EU sectors. *Eurasian Economic Review*, 13: 321–372.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Caldara, D.; and Iacoviello, M. 2022. Measuring Geopolitical Risk. *American Economic Review*, 112(4): 1194–1225.

Chang, K. K.; Chen, D.; and Bamman, D. 2023. Dramatic Conversation Disentanglement. *arXiv preprint arXiv:2305.16648*.

Chowdhury, M. A. F.; Abdullah, M.; Azad, M. A. K.; Sulong, Z.; and Islam, M. N. 2023. Environmental, social and governance (ESG) rating prediction using machine learning approaches. *Annals of Operations Research*.

Das, A.; Kong, W.; Sen, R.; and Zhou, Y. 2023. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*.

de Souza Barbosa, A.; da Silva, M. C. B. C.; da Silva, L. B.; Morioka, S. N.; and de Souza, V. F. 2023. Integration of Environmental, Social, and Governance (ESG) criteria: their impacts on corporate sustainability performance. *Humanities and Social Sciences Communications*, 10(1): 410.

Deng, Z.; Chan, C.; Wang, W.; Sun, Y.; Fan, W.; Zheng, T.; Yim, Y.; and Song, Y. 2024. Text-Tuple-Table: Towards Information Integration in Text-to-Table Generation via Global Tuple Extraction. arXiv:2404.14215.

Dhariwal, P.; and Nichol, A. Q. 2021. Diffusion Models Beat GANs on Image Synthesis. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 8780–8794.

Eisenschlos, J.; Gor, M.; Müller, T.; and Cohen, W. 2021. MATE: Multi-view Attention for Table Transformer Efficiency. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 7606–7619. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Ekambaram, V.; Jati, A.; Dayama, P.; Mukherjee, S.; Nguyen, N. H.; Gifford, W. M.; Reddy, C.; and Kalagnanam, J. 2024a. Tiny Time Mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *arXiv [cs.LG]*.

Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 459–469.

Ekambaram, V.; Jati, A.; Nguyen, N. H.; Dayama, P.; Reddy, C.; Gifford, W. M.; and Kalagnanam, J. 2024b. TTMs: Fast Multi-level Tiny Time Mixers for Improved Zero-shot and Few-shot Forecasting of Multivariate Time Series. *arXiv preprint arXiv:2401.03955*.

Estornell, A.; Das, S.; and Vorobeychik, Y. 2020. Deception through half-truths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 10110–10117.

Evtikhiev, M.; Bogomolov, E.; Sokolov, Y.; and Bryksin, T. 2023. Out of the BLEU: How should we assess quality of the Code Generation models? *Journal of Systems and Software*, 203: 111741.

Fang, J.-K.; Lin, Y.-F.; Huang, J.-H.; Chen, Y.; et al. 2024. Divide-and-Conquer Quantum Algorithm for Hybrid *denovo* Genome Assembly of Short and Long Reads. *PRX Life*, 2: 023006.

Fernando, C.; Banarse, D.; Michalewski, H.; Osindero, S.; and Rocktaschel, T. 2023. PROMPTBREEDER: SELF-REFERENTIAL SELF-IMPROVEMENT VIA PROMPT EVOLUTION. *arXiv:2309.16797*.

Fu, J.; Ng, S.-K.; Jiang, Z.; and Liu, P. 2023. GPTScore: Evaluate as You Desire. arXiv:2302.04166.

Ganesan, B.; Ghosh, S.; Gupta, N.; Kesarwani, M.; Mehta, S.; and Sindhgatta, R. 2024. LLM-powered GraphQL Generator for Data Retrieval. In *IJCAI*.

Gekhman, Z.; Yona, G.; Aharoni, R.; Eyal, M.; Feder, A.; Reichart, R.; and Herzig, J. 2024. Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations? arXiv:2405.05904.

Gong, S.; Li, M.; Feng, J.; Wu, Z.; and Kong, L. 2023. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. In *The Eleventh International Conference on*

Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Goswami, M.; Szafer, K.; Choudhry, A.; Cai, Y.; Li, S.; and Dubrawski, A. 2024. MOMENT: A Family of Open Time-series Foundation Models. *arXiv preprint arXiv:2402.03885*.

Grote, O.; Ahrens, A.; and Benavente-Peces, C. 2019. A Review of Post-quantum Cryptography and Crypto-agility Strategies. In *2019 International Interdisciplinary PhD Workshop (IIPhDW)*, 115–120.

Haldar, R.; and Hockenmaier, J. 2024. Analyzing the Performance of Large Language Models on Code Summarization. *arXiv preprint arXiv:2404.08018*.

Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Jain, P.; Marzoca, A.; and Piccinno, F. 2024. Structsum Generation for Faster Text Comprehension. arXiv:2401.06837.

Kate, A.; Zaverucha, G. M.; and Goldberg, I. 2010. Constant-Size Commitments to Polynomials and Their Applications. In Abe, M., ed., *Advances in Cryptology - ASIACRYPT 2010*, 177–194. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kazakov, A.; Denisova, S.; Barsola, I.; Kalugina, E.; Molchanova, I.; Egorov, I.; Kosterina, A.; Tereshchenko, E.; Shutikhina, L.; Doroshchenko, I.; Sotiriadi, N.; and Budennyy, S. 2023. ESGify: Automated classification of environmental, social, and corporate governance risks. *Doklady Mathematics*, 108(S2): S529–S540.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kummerfeld, J. K.; Gouravajhala, S. R.; Peper, J. J.; Athreya, V.; Gunasekara, C.; Ganhotra, J.; Patel, S. S.; Polymenakos, L. C.; and Lasecki, W. 2019. A Large-Scale Corpus for Conversation Disentanglement. In Korhonen, A.; Traum, D.; and Màrquez, L., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3846–3856. Florence, Italy: Association for Computational Linguistics.

Kundu, A. 2024. Reinforcement learning-assisted quantum architecture search for variational quantum algorithms. *arXiv preprint arXiv:2402.13754*.

Lai, R. W. F.; and Malavolta, G. 2024. Lattice-Based Timed Cryptography. *IACR Cryptol. ePrint Arch.*, 540.

Larocca, M.; Sauvage, F.; Sbahi, F. M.; Verdon, G.; Coles, P. J.; and Cerezo, M. 2022. Group-invariant quantum machine learning. *PRX Quantum*, 3(3): 030341.

Lassance, C.; Déjean, H.; Formal, T.; and Clinchant, S. 2024. SPLADE-v3: New baselines for SPLADE. *CoRR*, abs/2403.06789.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. 2024a. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Li, X.; Thickstun, J.; Gulrajani, I.; Liang, P.; and Hashimoto, T. B. 2022. Diffusion-LM Improves Controllable Text Generation. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Li, X.; Zhao, R.; Chia, Y. K.; Ding, B.; Joty, S.; Poria, S.; and Bing, L. 2024b. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. arXiv:2305.13269.

Liao, X.; Li, M.; Zou, Y.; Wu, F.-X.; Yi-Pan; and Wang, J. 2019. Current challenges and solutions of de novo assembly. *Quantitative Biology*, 7(2): 90–109.

Liu, C.; and Wan, X. 2021. CodeQA: A Question Answering Dataset for Source Code Comprehension. *arXiv:2109.08365*.

Liu, F.; Liu, Y.; Shi, L.; Huang, H.; Wang, R.; Yang, Z.; and Zhang, L. 2024. Exploring and Evaluating Hallucinations in LLM-Powered Code Generation. *arXiv preprint arXiv:2404.00971*.

Liu, L.; Nguyen, H.; Karypis, G.; and Srinivasan Sengamedu, S. 2021. Universal representation for code. In *PAKDD 2021*.

Liu, Z.; Liu, J.; Zeng, Q.; and Wu, L. 2022. VIX and stock market volatility predictability: A new approach. *Finance Research Letters*, 48: 102887.

Lyubashevsky, V.; Nguyen, N. K.; and Plancon, M. 2022. Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General. Cryptology ePrint Archive, Paper 2022/284. https://eprint.iacr.org/2022/284.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Welleck, S.; Majumder, B. P.; Gupta, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. arXiv:2303.17651.

Meyer, J. J.; Mularski, M.; Gil-Fuster, E.; Mele, A. A.; Arzani, F.; Wilms, A.; and Eisert, J. 2023. Exploiting symmetry in variational quantum machine learning. *PRX Quantum*, 4(1): 010328.

Mishra, M.; Stallone, M.; Zhang, G.; Shen, Y.; Prasad, A.; Soria, A. M.; Merler, M.; Selvam, P.; Surendran, S.; Singh, S.; Sethi, M.; Dang, X.-H.; Li, P.; Wu, K.-L.; Zawad, S.; Coleman, A.; White, M.; Lewis, M.; Pavuluri, R.; Koyfman, Y.; Lublinsky, B.; de Bayser, M.; Abdelaziz, I.; Basu, K.; Agarwal, M.; Zhou, Y.; Johnson, C.; Goyal, A.; Patel, H.; Shah, Y.; Zerfos, P.; Ludwig, H.; Munawar, A.; Crouse, M.; Kapanipathi, P.; Salaria, S.; Calio, B.; Wen, S.; Seelam, S.; Belgodere, B.; Fonseca, C.; Singhee, A.; Desai, N.; Cox,

D. D.; Puri, R.; and Panda, R. 2024. Granite Code Models: A Family of Open Foundation Models for Code Intelligence. arXiv:2405.04324.

Muennighoff, N.; Su, H.; Wang, L.; Yang, N.; Wei, F.; Yu, T.; Singh, A.; and Kiela, D. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Nadeem, M.; Bethke, A.; and Reddy, S. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 5356–5371. Online: Association for Computational Linguistics.

Nangia, N.; Vania, C.; Bhalerao, R.; and Bowman, S. R. 2020. CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1953–1967. Online: Association for Computational Linguistics.

Ostaszewski, M.; Trenkwalder, L. M.; Masarczyk, W.; Scerri, E.; and Dunjko, V. 2021. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34: 18182–18194.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; Zhao, S.; Hong, L.; Tian, R.; Xie, R.; Zhou, J.; Gerstein, M.; Li, D.; Liu, Z.; and Sun, M. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. arXiv:2307.16789.

Ragone, M.; Braccia, P.; Nguyen, Q. T.; Schatzki, L.; Coles, P. J.; Sauvage, F.; Larocca, M.; and Cerezo, M. 2022. Representation theory for geometric quantum machine learning. *arXiv preprint arXiv:2210.07980*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 10674–10685. IEEE.

Sandeep, S.; and Bhattacharyya, P. 2023. Detecting and Debunking Fake News and Half-truth: A Survey. *arXiv:2308.07973v1*.

Schatzki, L.; Larocca, M.; Nguyen, Q. T.; Sauvage, F.; and Cerezo, M. 2024. Theoretical guarantees for permutation-equivariant quantum neural networks. *npj Quantum Information*, 10(1): 12.

Sohl-Dickstein, J.; Weiss, E. A.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Bach, F. R.; and Blei, D. M., eds., *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, 2256–2265. JMLR.org.

Srinivas, P.; Husain, F.; Parayil, A.; Choure, A.; Bansal, C.; and Rajmohan, S. 2024. Intelligent Monitoring Framework for Cloud Services: A Data-Driven Approach. arXiv:2403.07927v1.

Sun, J.; Tian, Y.; Zhou, W.; Xu, N.; Hu, Q.; Gupta, R.; Wieting, J.; Peng, N.; and Ma, X. 2023. Evaluating Large Language Models on Controlled Generation Tasks. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 3155–3168. Singapore: Association for Computational Linguistics.

Sundar, A.; Richardson, C.; and Heck, L. 2024. gTBLS: Generating Tables from Text by Conditional Question Answering. arXiv:2403.14457.

Tang, X.; Zong, Y.; Phang, J.; Zhao, Y.; Zhou, W.; Cohan, A.; and Gerstein, M. 2024. Struc-Bench: Are Large Language Models Good at Generating Complex Structured Tabular Data? In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, 12–34. Mexico City, Mexico: Association for Computational Linguistics.

Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; and Gurevych, I. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663.

Tian, R.; Ye, Y.; Qin, Y.; Cong, X.; Lin, Y.; Pan, Y.; Wu, Y.; Hui, H.; Liu, W.; Liu, Z.; and Sun, M. 2024. DebugBench: Evaluating Debugging Capability of Large Language Models. *arXiv:2401.04621*.

Wang, X.; Pham, H.; Michel, P.; Anastasopoulos, A.; Neubig, G.; and Carbonell, J. G. 2019. Optimizing Data Usage via Differentiable Rewards. *CoRR*, abs/1911.10088.

Wang, Z.; Cuenca, G.; Zhou, S.; Xu, F. F.; and Neubig, G. 2023. MCoNaLa: A Benchmark for Code Generation from Multiple Natural Languages. In Vlachos, A.; and Augenstein, I., eds., *Findings of the Association for Computational Linguistics: EACL 2023*, 265–273. Dubrovnik, Croatia: Association for Computational Linguistics.

Webersinke, N.; Kraus, M.; Bingler, J. A.; and Leippold, M. 2021. ClimateBERT: A pretrained language model for Climate-Related Text. *arXiv*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; et al. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*.

Wittern, E.; Cha, A.; and Laredo, J. A. 2018. Generating graphql-wrappers for rest (-like) apis. In *International Conference on Web Engineering*, 65–83.

Wolfson, T.; Deutch, D.; and Berant, J. 2022. Weakly Supervised Text-to-SQL Parsing through Question Decomposition. In Carpuat, M.; de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *Findings of the Association for Computational Linguistics: NAACL 2022*, 2528–2542. Seattle, United States: Association for Computational Linguistics.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-

term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430.

Wu, X.; Zhang, J.; and Li, H. 2022. Text-to-Table: A New Way of Information Extraction. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2518–2533. Dublin, Ireland: Association for Computational Linguistics.

Xu, C.; Lv, G.; Du, J.; Chen, L.; Huang, Y.; and Zhou, W. 2021. Kubeflow-based Automatic Data Processing Service for Data Center of State Grid Scenario. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 924–930.

Yang, L.; and Shami, A. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415: 295–316.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; Zhang, Z.; and Radev, D. R. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. *CoRR*, abs/1809.08887.

Yuksekgonul, M.; Chandrasekaran, V.; Jones, E.; Gunasekar, S.; Naik, R.; Palangi, H.; Kamar, E.; and Nushi, B. 2024. Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models. arXiv:2309.15098.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

Zhang, J.; Cao, S.; Zhang, T.; Lv, X.; Li, J.; Hou, L.; Shi, J.; and Tian, Q. 2023. Reasoning over Hierarchical Question Decomposition Tree for Explainable Question Answering. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 14556–14570. Toronto, Canada: Association for Computational Linguistics.

Zhang, J.; Li, Z.; Das, K.; Malin, B. A.; and Kumar, S. 2024. SAC3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency. arXiv:2311.01740.

Zhao, Y.; Gu, A.; Varma, R.; Luo, L.; Huang, C.-C.; Xu, M.; Wright, L.; Shojanazeri, H.; Ott, M.; Shleifer, S.; et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. arXiv:2304.11277.

Zheng, L.; Li, Z.; Zhang, H.; Zhuang, Y.; Chen, Z.; Huang, Y.; Wang, Y.; Xu, Y.; Zhuo, D.; Xing, E. P.; Gonzalez, J. E.; and Stoica, I. 2022. Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning. arXiv:2201.12023.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, 27268–27286. PMLR.

Zhou, T.; Niu, P.; Sun, L.; Jin, R.; et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36: 43322–43355.