The Evolving Ecosystem of Language: A Comparative Overview of LLM Architectures and Their Expanding Applications

Mr. Harsh Virani (Mtr. Nr. - 10030225)
MSc Artificial Intelligence & Robotics
Hof University of Applied Sciences
Hof, Germany
harsh.virani@hof-univeristy.de

***Abstract*—The field of Natural Language Processing (NLP) has been significantly revolutionized by the advent and rapid progress of Large Language Models (LLMs). These advanced deep learning models have demonstrated state-of-the-art (SOTA) performance across a wide spectrum of language-related tasks, profoundly influencing both natural language understanding (NLU) and natural language generation (NLG). This paper presents a comparative overview of different LLMs, focusing on their fundamental architectures, their diverse use cases across various domains, and a discussion of their inherent strengths and limitations. Through a literature review and comparative analysis of existing LLMs based on their architectural types, functionalities, and performance in various domains, this study highlights key distinctions. Notably, decoder-only architectures are predominantly employed for generative tasks, encoder-only models excel in understanding and extracting information, while encoder-decoder architectures are well-suited for sequence-to-sequence tasks like translation and summarization. Furthermore, the paper emphasizes the specialization of Code LLMs in generating source code from natural language descriptions, showcasing their growing importance in software development. Finally, the paper briefly addresses the challenges associated with LLMs, including biases, the generation of factually incorrect information (hallucinations), and the high computational cost, while also touching upon future research directions such as enhancing interpretability, addressing ethical concerns, and expanding multimodal and multilingual capabilities. This comparative analysis aims to provide a concise yet thorough understanding of the current landscape of LLMs and their implications for various applications.**

***Index Terms*—Large Language Models, Survey, Overview, Natural Language Processing (NLP), LLM Applications, LLM Architectures**

I. Introduction

The past few years have seen an unprecedented leap in Natural Language Processing, driven by Large Language Models (LLMs) that surpass traditional models in both understanding (NLU) and generation (NLG). By learning complex language representations in a self-supervised manner on vast text corpora, LLMs now generate and interpret human language with remarkable fluency and coherence, signaling a shift from narrow, task-specific systems to versatile, general-purpose models.

Central to this evolution is the dramatic scale-up of model size—from millions to trillions of parameters—which has unlocked emergent capabilities such as in-context learning, instruction following, and multi-step reasoning. Underpinned by the Transformer architecture and its attention mechanism, contemporary LLMs fall mainly into three categories—decoder-only, encoder-only, and encoder-decoder—each suited to different language tasks.

The broad applicability of LLMs spans creative text generation, automated code synthesis (notably through specialized "Code LLMs" for NL2Code), question answering, information retrieval, translation, and complex data analysis. Their specialized variants demonstrate tangible benefits in domains ranging from software development to research and customer support.

This paper offers a comparative overview of LLMs, examining their architectures, domain-specific use cases, and respective strengths and limitations. It is organized as follows: Section 2 defines essential terminology; Section 3 surveys major architectural types; Section 4 explores task-oriented applications; Section 5 delivers a cross-domain performance analysis; Section 6 discusses challenges and future research directions; and Section 7 concludes.

II. Key Terminologies

This section establishes the fundamental vocabulary necessary for a clear understanding of Large Language Models (LLMs) discussed throughout this paper. We will define and elaborate on several core concepts that are crucial to grasping the nature, capabilities, and development of these powerful language models.

A. Language Model (LM)

A **Language Model (LM)** is a statistical model that learns to predict the probability of a sequence of words occurring in a given language. Traditionally, these models assigned probabilities to sequences of words based on their co-occurrence patterns observed in training data. Earlier statistical methods like N-grams computed word- or phrase-level statistics to be used as features in supervised models [1]. However, these approaches often lacked contextual information and suffered

from the curse of dimensionality. Recurrent Neural Network based language models (RNN-LMs) represented an advancement by processing sequences of words and maintaining a hidden state to capture contextual information.

B. Large Language Model (LLM)

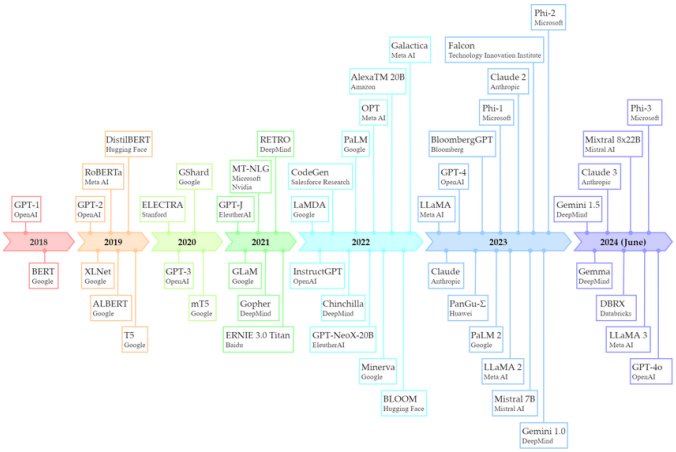

Fig. 1. The timeline of the development of LLMs from 2018 to 2024 (June), showcasing key advancements and notable models. [2]

The emergence of deep learning has resulted in the creation of more advanced language models, culminating in Large Language Models (LLMs). LLMs represent a significant evolution from traditional LMs and are characterized by several key attributes:

Scale of Parameters: LLMs contain a remarkably large number of parameters, often ranging from billions to trillions, enabling them to learn complex patterns and representations of language. Examples of LLMs with varying parameter counts include GLM (515 Million), GPT-J (6 Billion), YaLM (100 Billion), BLOOM (176 Billion), Gopher, and models exceeding a trillion parameters.

Massive Training Datasets: These models are trained on enormous quantities of text data, often comprising diverse sources such as webpages, books, news articles, and code. The sheer size and diversity of the training data enable LLMs to acquire a broad understanding of language and the world.

Transformer Architecture: The vast majority of modern LLMs are built upon the **Transformer architecture**, which has become the dominant architecture in NLP. The Transformer, presented in "Attention Is All You Need", depends extensively on attention mechanisms to determine the significance of various words within a sequence during processing [3]. This architecture allows for parallel processing of input tokens, significantly reducing training time compared to sequential models like RNNs.

Emergent Capabilities: Due to their scale and the Transformer architecture, LLMs exhibit **emergent abilities**, such as in-context learning, that are not readily apparent in smaller language models. **In-context learning** refers to the ability of LLMs to learn from a few examples provided in the prompt (without explicit fine-tuning) and apply that learning to generate relevant output.

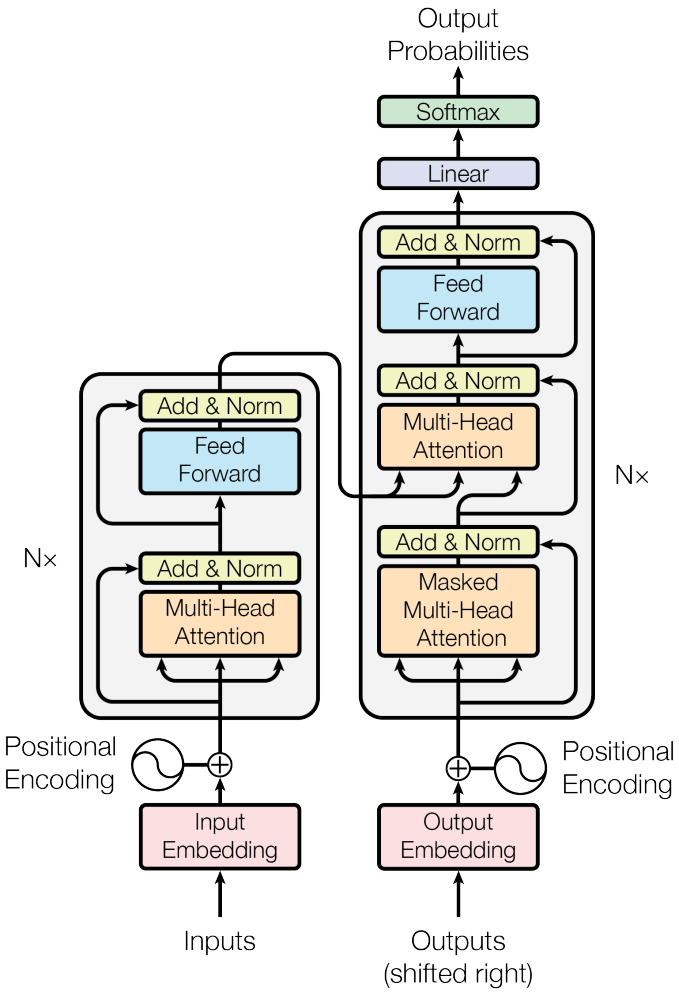

Fig. 2. The Transformer - model architecture.

Understanding and Generation: LLMs possess significant capabilities in both **natural language understanding (NLU)** and **natural language generation (NLG)**. They can comprehend the nuances of human language and generate coherent, contextually relevant text for a wide range of tasks.

The development of LLMs represents a significant step towards creating more versatile and powerful language processing systems, moving beyond earlier limitations of task-specific models.

C. The Transformer Architecture

The **Transformer architecture** has become the fundamental building block for most modern Large Language Models (LLMs). This framework, presented in 2017 by Vaswani and colleagues in their groundbreaking paper "Attention Is All You Need", represented a major departure from earlier sequence-to-sequence architectures such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [3].

A fundamental breakthrough of the Transformer is its dependence on attention mechanisms, especially self-attention,

which enables the model to assess the relevance of various portions of the input sequence while processing individual words [1]. This enables the Transformer to effectively capture **long-range dependencies** in text, a challenge for earlier recurrent architectures.

Unlike RNNs that process tokens sequentially, the Transformer architecture is designed for **parallel processing** of the entire input sequence, leading to significant improvements in training speed and efficiency, especially with the use of GPUs [4], [5]. The initial Transformer architecture employed an encoder-decoder framework. The encoder converts the input sequence into a continuous representation, which the decoder then uses to produce the output sequence [4]. Although later LLMs may employ just the encoder or decoder portions of the Transformer, the fundamental principles of the complete architecture—such as the **self-attention mechanism** and **positional encoding** for managing sequence ordering—continue to be essential to their operation. The Transformer's remarkable capacity for parallel computation and its robust representational abilities have established it as the standard architecture for numerous Natural Language Processing (NLP) applications [6].

D. Pre-training Explained

Pre-training is a **critical phase** in the development lifecycle of Large Language Models (LLMs). It is the **very first step** in the training pipeline. During this phase, LLMs acquire **fundamental language understanding capabilities**.

This procedure is carried out using **self-supervised learning techniques** on enormous collections of unlabeled textual data and corpora. The goal is to establish the network's foundational parameters, essentially building a comprehensive language representation. This self-supervised methodology utilizes unlabeled datasets to equip the model with both language comprehension and text generation abilities [1].

The purpose of pre-training is to enable the model to acquire and understand language patterns and knowledge. This acquired knowledge may be broad-based or specialized to particular domains, and is subsequently applied to various downstream Natural Language Understanding (NLU) and Natural Language Generation (NLG) applications.

Common pre-training objectives include:

- **Autoregressive Language Modeling** (alternatively called next-token prediction), in which the model develops the ability to forecast the subsequent token in a sequence using the tokens that come before it [4].
- **Masked Language Modeling** (also known as denoising autoencoding), in which the model learns to predict concealed words within a sequence by analyzing the contextual information surrounding them [4].

After pre-training is complete, this base model is usually customized or fine-tuned for particular downstream applications. The pre-training methodology proves more efficient and results in quicker and superior generalization performance compared to building a model entirely from the ground up [1]. Pre-training employs unlabeled data through self-supervised learning methods, whereas the following transfer learning (fine-tuning) stage generally uses labeled data tailored to the specific downstream application.

E. Defining Fine-tuning

Following initial self-supervised pre-training on vast unlabeled corpora, fine-tuning adapts a Large Language Model (LLM) to specific downstream tasks by updating its parameters using task-relevant labeled data. This process leverages the broad linguistic patterns and world knowledge acquired during pre-training, tailoring the model's behavior for applications such as text classification, question answering, code generation, or machine translation.

Whereas pre-training uses generic, unlabeled text and optimizes for next-token prediction, fine-tuning employs a supervised learning setup on a narrower, task-specific data distribution. By exposing the model to examples directly aligned with the target task, fine-tuning accelerates convergence and yields better generalization than training from scratch. It also permits incorporation of proprietary or domain-specific knowledge absent from the original pre-training corpus.

Full-parameter fine-tuning (FFT) updates all model weights but can be computationally demanding and sometimes hinders transferability. Parameter-efficient fine-tuning (PEFT) techniques—such as LoRA's low-rank adapters—address these limitations by freezing most parameters and training only a small subset or added modules. Instruction tuning further refines the model's ability to follow natural-language commands by fine-tuning on examples framed as task descriptions. Multitask fine-tuning extends this concept, adapting one model to perform well across multiple objectives simultaneously.

In essence, fine-tuning is an indispensable post-pre-training step: it sharpens a generalist LLM's capabilities to excel on precisely defined tasks through supervised learning on labeled, task-aligned data.

F. Briefing In-context Learning

Going beyond the conventional pre-training and fine-tuning framework, **In-context Learning (ICL)** has become a significant emergent capability of Large Language Models (LLMs). Introduced in models such as GPT-3, ICL enables LLMs to **acquire new tasks directly from a limited number of examples included in the prompt during inference**.

The fundamental principle of ICL is that the model is conditioned on **in-context information** – typically a task instruction followed by a few input-label examples – to generate the desired output for a new, unseen input. Unlike fine-tuning, ICL **does not involve any parameter updates or architectural modifications**; the model leverages the vast knowledge acquired during its pre-training phase [1].

The primary advantage of ICL is the **minimal need for task-specific labeled data**, making it particularly useful for tasks with scarce datasets. Based on the number of examples provided in the prompt, ICL is often categorised into:

- **Zero-Shot Learning:** The model is given only the task instruction with no examples. This is useful when no task-specific data is available.

- **One-Shot Learning:** One example is provided as context alongside the task description.
- **Few-Shot Learning:** A small number of examples (ranging from a few to typically 10-100, or as many as fit within the model's context window) are included in the prompt.

While initial results from ICL, especially with larger models like the 175B parameter GPT-3, showed promise and improved out-of-domain generalization [1], the performance of ICL can still be **inferior to that achieved through fine-tuning**. The effectiveness of few-shot ICL can also be influenced by factors such as the specific examples chosen, their order, and the prompt format. Despite these points, ICL represents a significant step towards more adaptable and data-efficient LLM usage.

III. Primary Types of LLM Architectures

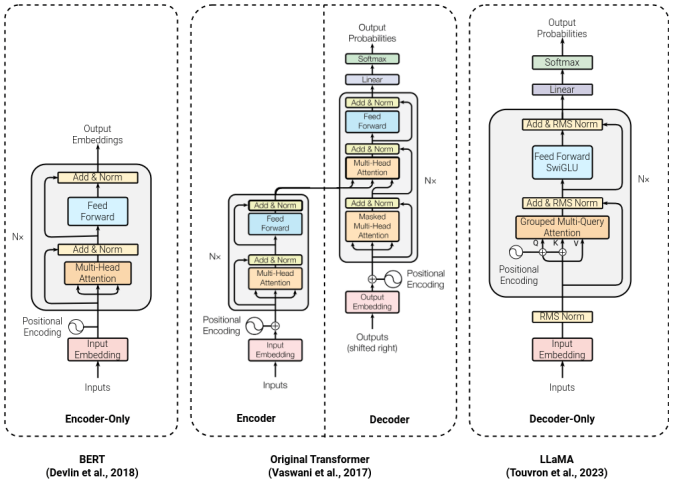

Fig. 3. Comparison between architectures of influential models from different modelling methods.

Large Language Models (LLMs) predominantly leverage the foundational **Transformer architecture**. Based on the core components of the Transformer that are primarily utilized, LLMs can be broadly categorized into three main architectural designs: **Decoder-only Models**, **Encoder-only Models**, and **Encoder-Decoder Models**. This classification is based on their structural construct and the tasks they are best suited for.

A. Decoder-only Models

Decoder-only models are characterized by their architecture, which consists solely of the **decoder part** of the Transformer. These models are inherently **autoregressive**. Their main training objective is usually **Causal Language Modeling (CLM)**, also called unidirectional language modeling or next-token prediction, in which the model develops the ability to predict the following token in a sequence using exclusively the tokens that precede it [6].

As a result of this training objective, decoder-only models are **well-suited for Natural Language Generation (NLG) tasks** [1]. They excel at generating new text, continuing sequences, and tasks that require producing output in a sequential manner. The **input and target tokens are concatenated** before processing, and their representations are built concurrently layer by layer.

Prominent examples of decoder-only LLM families include the **GPT series (e.g., GPT-1, GPT-2, GPT-3, GPT-4)**, **LLaMA (e.g., Llama 2)**, and **PaLM**. Other notable decoder-only models include **Jurassic-1**, **Pangu-Alpha**, **GPT-J**, **BLOOM**, **Falcon**, **Mixtral-8x7B**, **YaLM**, and the **Phi series (e.g., Phi-1, Phi-1.5, Phi-3)**. For code generation specifically, decoder-only models like **StarCoder** and **Code Llama** are widely used.

A notable advantage of larger decoder-only models is their capacity for **in-context learning**, enabling them to execute new tasks using examples given directly within the prompt without needing explicit fine-tuning. Nevertheless, their reliance on unidirectional attention mechanisms may create constraints for tasks that involve extremely long sequences such as summarization [1].

B. Encoder-only Models

Encoder-only models are built using solely the **encoder component** of the Transformer architecture [5]. These models employ **bidirectional attention**, meaning that when processing a token, the attention mechanism can access context from both the preceding and subsequent tokens in the input sequence. This bidirectional nature allows them to effectively **capture and integrate contextual information** into the data representation.

Encoder-only models are usually pre-trained through methods such as Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Through MLM, the model develops the ability to predict hidden tokens by analyzing the contextual information around them, whereas NSP enables the model to learn how two sentences relate to each other [4].

Encoder-only models are particularly effective for **Natural Language Understanding (NLU)** applications because they can process input text in both directions simultaneously, giving them a comprehensive view of the entire context. This bidirectional processing capability makes them ideal for tasks that demand thorough contextual comprehension, including categorizing text, identifying specific entities within text, answering questions by extracting relevant information from passages, and conducting detailed text analysis.

The most prominent example of an encoder-only model is **BERT (Bidirectional Encoder Representations from Transformers)**. Several variants have been developed based on BERT, including **RoBERTa**, **ALBERT**, **DeBERTa**, **XLM**, **XLNet**, **UNILM**, and **ERNIE**. **ELECTRA** is another model in this category.

A limitation noted for encoder-only models is that their inherent context dependency and composition without a decoder can be a hindrance for text generation. Downstream task adaptation for these models typically necessitates fine-tuning.

C. Encoder-Decoder Models

Encoder-Decoder models, occasionally called sequence-to-sequence models [4], employ the full Transformer architecture, incorporating both encoder and decoder components. The encoder converts the input sequence into a continuous representation, while the decoder uses this representation to generate the output sequence. To enable communication and information transfer between the encoder and decoder, a **cross-attention mechanism** is incorporated between their corresponding layers [1].

These models are designed to handle tasks that involve understanding an input sequence and generating a different output sequence based on that understanding. They are most effective for **conditional generation applications**, including **machine translation**, **text summarization**, and **generative question answering**.

Encoder-decoder models **amalgamate the strengths of both encoder-only and decoder-only architectures**, combining natural language understanding capabilities from the encoder with generation competencies from the decoder [5].

Examples of representative encoder-decoder models include the original **Transformer model**, **T5 (Text-to-Text Transfer Transformer)**, its multilingual variant **mT5**, **BART**, and **MASS**. Models like **CodeT5**, **CodeT5+**, and **CodeRL** are examples within the code generation domain. The **UL2** model is also mentioned as having an encoder-decoder variant.

While encoder-decoder models might end up having more parameters than single-stack models, they often maintain similar computational costs. In contrast to decoder-only models which learn to produce either the original input or its continuation, encoder-decoder models are designed to generate specific target outputs based on the provided input conditions. Different masking strategies are used in this architecture, with fully visible masking often used in the encoder and causal masking in the decoder [1].

IV. Use Cases of LLMs Based on Specialities

The inherent capabilities of Large Language Models (LLMs) to understand, generate, and manipulate human language, largely enabled by their foundational Transformer architectures, have led to their application across a wide spectrum of domains. Different architectural designs and training methodologies empower LLMs to excel in specific types of tasks. This section elaborates on the primary use cases of LLMs, categorized by their specialities.

A. General-Purpose Language Understanding and Generation

Due to their pre-training on massive and diverse text datasets, LLMs exhibit remarkable general-purpose abilities in both **Natural Language Understanding (NLU)** and **Natural Language Generation (NLG)**. These capabilities form the basis for a multitude of applications that are not tied to highly specific technical domains.

Broad applications include:

- **Text Summarization:** Condensing lengthy texts into shorter, coherent summaries.
- **Question Answering:** Understanding natural language queries and providing relevant answers based on their training data or provided context.
- **Content Creation:** Generating creative writing, articles, dialogue, emails, reports, and other forms of textual content. Models like **GPT-3** are examples of those excelling in versatile text generation. Tools based on models like **GPT-4** and **Bard** are for creative writing assistance.
- **Language Translation:** Translating text from one natural language to another, often a task handled effectively by encoder-decoder models.
- **Text Analysis:** Performing various analytical tasks such as sentiment analysis and text classification.
- **Conversational AI:** Powering chatbots and virtual assistants capable of engaging in human-like conversation.

These general capabilities stem from the models' ability to learn complex language representations in a self-supervised manner during pre-training.

B. Code Generation

A particularly impactful application area for LLMs is the automation of code-related tasks, most notably **Code Generation**. This involves generating source code from natural language descriptions, a task often referred to as **Natural Language to Code (NL2Code)**. This area has attracted considerable attention from both academic researchers and industry professionals because of its promise to improve efficiency and productivity in software development processes [6].

While general-purpose LLMs can sometimes generate code, **specialized LLMs, known as Code LLMs**, have emerged. These models are specifically pre-trained or continually pre-trained and fine-tuned on large-scale code corpora, often alongside text and mathematical data, to better assimilate coding principles and logic [6].

Examples of prominent Code LLMs include:

- **CodeGPT**
- **Codex** (trained using a combination of GPT-generated data and publicly available code repositories from GitHub, with performance assessed through the HumanEval benchmark)
- **AlphaCode** (targets complex competitive programming problems)
- **CodeBERT** (an encoder-only model suitable for code comprehension)

The "Survey on Large Language Models for Code Generation" provides a comprehensive review of this specific area. Evaluation of these models often involves benchmarks like HumanEval, MBPP, and BigCodeBench, among others.

C. Multimodal Applications

A sophisticated category of LLMs, called **Multimodal Large Language Models (MLLMs)**, expands the functionality of conventional LLMs by enabling them to handle and

understand information from various data types beyond text alone, including images, audio, and video [5]. This enables MLLMs to handle tasks that involve understanding and relating information across different types of data.

MLLMs can process both natural language and other forms of data as input and often generate text or other modalities as output. Their capabilities include:

- **Visual Question Answering (VQA):** Answering questions about the content of images.
- **Image Captioning:** Generating textual descriptions for images.
- **Text-based Image Description:** Understanding text queries to describe image content.
- **Document Reading (OCR):** Processing text within images.
- **Embodied Tasks:** Potential applications in robotics to accomplish physical tasks guided by language and visual input.

Examples of models and frameworks facilitating multimodal capabilities mentioned include:

- **PaLM-E** (an evolution of PaLM using Google's Pathways architecture, capable of deciphering natural language and imagery, with applications in robotics)
- **Visual-LLaMA**
- **Gemini 1.5** (recognized for enabling multimodal comprehension across extensive contexts spanning millions of tokens)
- **DALL-E**, **DALL-E 2**, **DALL-E 3** (primarily for image generation based on text)
- **Whisper** (for audio-to-text conversion)

The development of MLLMs represents a significant step towards more comprehensive AI systems capable of interacting with the world across different data types.

D. Domain-Specific LLMs

Beyond general applications, LLMs can be tailored or specialized for performance within particular domains. This often involves training on domain-specific datasets or fine-tuning techniques to adapt the models' knowledge and capabilities to the nuances of a field.

Examples of domain-specific LLMs and their applications include:

Medicine: Models like **Med-PaLM** and **BioMegatron** are developed to handle medical texts, understand clinical queries, and potentially assist in diagnostics or information retrieval within the healthcare domain.

Science and Mathematics: LLMs are being applied to tasks requiring scientific and mathematical reasoning. Models like **Minerva**, fine-tuned from PaLM on a large dataset, are noted for their mathematical reasoning capabilities. The use of LLMs to create "code-based solutions" for complex mathematical problems can mitigate limitations in performing intricate mathematical computations [6]. **PaLM-U** and **EnsureMath** are additional examples in this space.

Finance: The LLMs are finding utility across diverse domains, including finance. Models tailored for the financial sector include examples such as **FinMA-7B**, **FinMA-30B**, different versions of **Fin-GPT (V1/V2/V3)**, **Instruct-FinGPT**, and **Fin-LLaMA**. The concept involves tailoring LLMs for tasks like financial text analysis, forecasting, or report generation, implicitly supported by the general mention of finance as a domain of utility for LLMs.

Developing domain-specific LLMs allows for improved accuracy and relevance within specialized fields where general models might lack the necessary depth of knowledge or understanding of domain-specific terminology and context.

V. Comparative Analysis of LLMs Across Domains

This section provides a direct comparison of different Large Language Models (LLMs) or families of LLMs, evaluating their performance across various domains. Benchmarks and evaluation metrics play a crucial role in assessing the capabilities and identifying the strengths and weaknesses of different models.

A. Coding

TABLE I
SELECTED LLM PERFORMANCE ON HUMANEVAL BENCHMARK (PASS@1) [6]

Model	**HumanEval** (Pass@1) %
GPT-4	87.8
WizardCoder 33B	79.9
GPT-3.5	76.2
Gemini Ultra	74.4
Gemini Pro	67.7
Code Llama 70B	53
phi-1 1.3B	50.6
OctoCoder	46.2
PaLM Coder	36
InstructCodeT5+ 16B	35

TABLE II
SELECTED LLM PERFORMANCE ON CLASSEVAL BENCHMARK (PASS@1) (GREEDY SAMPLING) [7]

Model	**Class-level Pass@1**	**Method-level Pass@1**
GPT-4	37.6%	62.8%
GPT-3.5	29.6%	50.4%
WizardCoder	12.2%	35.2%
Instruct-StarCoder	10.2%	23.1%
SantaCoder	8.6%	27.7%
Instruct-CodeGen	8.2%	24.9%
CodeGeeX	7.2%	21.2%
InCoder	6.2%	21.1%
Vicuna	3.0%	11.0%
ChatGLM	1.4%	8.2%
PolyCoder	1.4%	13.2%

LLMs have made significant strides in code-related tasks, especially code generation. Evaluations on benchmarks like HumanEval, MBPP, and BigCodeBench compare general-purpose LLMs to specialized Code LLMs, with the latter often achieving superior results.

More recently, ClassEval was introduced to assess more complex, class-level code generation. Models must complete provided Class Skeletons—including class information and method designs—and manage interdependencies between methods and fields.

Common evaluation metrics include execution-based Passk, which measures the proportion of correctly solved problems within k attempts, and automatic text-based metrics such as Exact Match, BLEU, ROUGE, and METEOR. CodeBLEU was specifically developed to evaluate code synthesis quality.

Table I presents a comparative performance of various LLMs on HumanEval, illustrating progressive improvements in code generation. Table II displays selected ClassEval results, showing that only GPT-4 and GPT-3.5 perform best using a holistic (entire class) strategy; other models achieve higher success with incremental, method-by-method generation. Across models, generating code dependent on fields yields higher success rates than handling method-to-method dependencies.

B. Analysis (Understanding) and Reasoning

Evaluating LLMs on tasks requiring deep contextual understanding and reasoning is crucial. Encoder-based models were initially designed for such tasks. Benchmarks assess capabilities such as commonsense reasoning, symbolic reasoning, and world knowledge.

Commonsense reasoning denotes the ability to use prior knowledge in combination with reasoning skills. Benchmarks like HellaSwag and OBQA are used for evaluation. **World knowledge** is assessed using benchmarks like TriviaQA, NaturalQ, WebQ, and ARC. **Symbolic reasoning** can be evaluated using benchmarks like Cobjects and Penguins. **Arithmetic reasoning** is evaluated using benchmarks like GSM8K and MATH.

C. Text Generation

Text generation is a primary application for many LLMs, particularly those with a decoder-only architecture, which excel in predicting the next token in a sequence and generating fluent and coherent text. Decoder-only models like the GPT series are noted for their strengths in text generation.

The quality of generated text can be influenced by factors such as model size and the scale and diversity of the training data. Evaluation of generated content can involve metrics-based approaches (like token-matching metrics such as BLEU, ROUGE, METEOR) or potentially generative evaluation methods using another LLM.

D. Question Answering

LLMs are also applied to Question Answering (QA) tasks, which can involve extracting information from text or answering general knowledge questions. Encoder-based models are well-suited for understanding queries and retrieving relevant information.

Benchmarking for QA can involve metrics like Exact Match (EM), **ROUGE-2**, and **ROUGE-L**, which evaluate the overlap and similarity between the generated answer and a human reference. These metrics quantify the models' ability to reproduce meaningful word sequences. The process can involve providing context and a list of queries to an LLM to extract structured information.

Table III includes performance on some world knowledge benchmarks (TriviaQA, NaturalQ, ARC, WebQ) that involve question answering.

TABLE III
SELECTED LLM PERFORMANCE ON WORLD KNOWLEDGE BENCHMARKS (INCLUDING QA)

Model	TriviaQA	NaturalQ	WebQ	ARC
GPT-4	92.1	48.6	58.6	98.6
PaLM 2-L	86.1	37.5	28.2	89.7
PaLM-540B	81.4	39.6	43.5	87.1
LLaMA 2 70B	85	33	-	-
GPT-3 175B	71.2	29.9	41.5	85.2
Mistral 7B	69.9	28.8	-	55.5
LLaMA 65B	72.6	39.9	-	-
BLOOM 176B	-	-	-	50.85

Developing domain-specific LLMs, as discussed in section IV, allows for improved accuracy and relevance within specialized fields compared to general models which might lack the necessary depth of knowledge. This comparative analysis across domains underscores that the "best" LLM is often task-dependent, with specialized models or models excelling on specific benchmarks being preferred for particular applications.

VI. CHALLENGES AND FUTURE DIRECTIONS

Despite the remarkable advancements and revolutionary impact of Large Language Models (LLMs) across various domains, particularly in areas like natural language processing and code generation, numerous significant **challenges** are still remaining to be addressed. These challenges not only limit their current capabilities but also raise crucial ethical and practical considerations for their widespread deployment. Simultaneously, these challenges pave the way for exciting **future research directions** aimed at enhancing LLMs' robustness, reliability, fairness, and overall utility.

A. Challenges Associated with LLMs

LLMs face several inherent limitations and issues that require ongoing research and development. Some of the critical challenges include:

Biases and Toxic Content: LLMs are susceptible to reproducing and amplifying existing biases present in their massive training datasets. This can lead to the generation of toxic, offensive, or stereotypical content related to factors like race, gender, religion, or socioeconomic status. Underrepresentation or misrepresentation of certain groups in the data can result in biased language generation and inaccurate outputs for those groups [1]. Addressing these issues is complex, and evaluating these harms often requires specific benchmarks, such as CrowS-Pairs, StereoSet, and the Parity benchmark.

Ensuring fairness and unbiased behaviour is crucial as LLMs are increasingly used daily.

Hallucinations: A significant issue where LLMs generate information that is false or incorrect, often presented confidently. This can manifest as intrinsic hallucination (contradicting source text) or extrinsic hallucination (unverifiable against the source). Reasons include misunderstanding source information, conflicts between contextual information and pretraining knowledge (parametric knowledge bias), or the fundamental probabilistic nature of these models. While training methods like instruct tuning and Reinforcement Learning from Human Feedback (RLHF) aim to improve factuality, the issue persists. Evaluating hallucination is an active area, and benchmarks like the **Hughes Hallucination Evaluation Model (HHM) Leaderboard** are used to assess models.

Computational Cost and Efficiency: Training and deploying LLMs require substantial computational resources, including numerous costly GPU machines. This leads to high financial costs and significant energy consumption, contributing to environmental impact (carbon footprint). The sheer size of the models presents challenges for distributed computation and can affect service level agreements, particularly in terms of latency for the largest models. Research is ongoing to develop more efficient models and training practices.

Explainability and Interpretability: Due to their complex, black-box nature, understanding how LLMs arrive at specific decisions or predictions is challenging. This lack of transparency hinders trust, accountability, and widespread acceptance. While some inherent interpretability exists (like attention weights indicating focus areas), the current research aims to develop methods that make the model's decision-making process and internal workings understandable to humans.

Security and Ethical Concerns: Beyond biases, LLMs face security vulnerabilities like adversarial attacks and can be misused, for instance, to spread misinformation or through the creation of covert channels. Addressing ethical concerns, ensuring fairness and responsibility, handling sensitive information responsibly, and protecting against potential threats are critical research areas. This includes mitigating risks associated with the training data itself, such as potential privacy violations and the ethical implications of their use in specific domains like code generation, where trustworthiness, safety, and responsibility are paramount.

B. Future Research Directions

The challenges outlined above, along with the rapid evolution of the field, highlight several promising avenues for future research and development. Key future directions include:

Development of More Efficient Architectures and Training Methods: Research is focused on creating smaller and more efficient LLMs, exploring new post-attention architectural paradigms, and developing techniques to reduce computational cost and increase energy efficiency, potentially through methods like quantization.

Improving Factual Accuracy and Reducing Hallucinations: Mitigating the issue of hallucinations is a major focus. Techniques like Retrieval-Augmented Generation (RAG), which allows LLMs to access external knowledge sources, are being investigated to reduce the generation of incorrect or unverifiable information. Continued research into prompting and augmentation techniques is expected.

Addressing Bias and Enhancing Fairness: Ongoing efforts are required to ensure LLMs are fair, unbiased, and handle sensitive information responsibly. This involves developing methods to identify and mitigate biases in training data and model outputs [1].

Enhancing Interpretability and Explainability: Future work aims to make LLM decision-making processes more transparent and understandable to users and developers, fostering greater trust and facilitating responsible deployment.

Expanding Multimodal and Multilingual Capabilities: A major emerging direction involves creating Multimodal Large Language Models (MLLMs) capable of handling and combining diverse forms of data such as text, visual content, sound, and video footage. Additionally, there is important ongoing research focused on expanding LLM functionality to support a broader spectrum of global languages, especially those with limited digital resources, in order to promote greater linguistic diversity and accessibility.

Creating LLMs that Can Interact with Tools and Environments (Language Agents): An exciting direction involves developing LLM-based agents and multi-agent systems that can access external tools and make decisions, moving towards more autonomous and capable AI systems.

These challenges and future directions highlight that while LLMs have made remarkable strides, there is still considerable room for improvement and responsible innovation in this rapidly evolving field.

VII. Conclusion

Large Language Models (LLMs) represent a transformative advancement across numerous fields, with a particularly remarkable impact on areas like natural language processing and automated code-related tasks. This paper has provided a systematic overview and comparative analysis of LLMs, exploring their foundational architectures, diverse use cases, key challenges, and promising future directions.

We highlighted the prevalence of Transformer-based architectures in modern LLMs, noting variations such as encoder-decoder models capable of handling sequence-to-sequence tasks like translation, decoder-only models often favoured for generative tasks, and models incorporating specific pre-training objectives like span corruption or fill-in-the-middle capabilities relevant to code generation.

The survey delved into the burgeoning field of Code LLMs, which have demonstrated significant capabilities in generating source code from natural language descriptions (NL2Code). Their utility extends beyond generation to other software engineering tasks such as code completion, translation, and repair.

Despite these considerable achievements, the path forward for LLMs is marked by significant challenges. Issues such as the presence and amplification of biases from training data, the problem of generating false or nonsensical information (hallucinations), the substantial computational resources and environmental impact required for training and deployment, the lack of transparency and interpretability in their decision-making, and various security and ethical concerns, including the generation of potentially harmful or biased code, all necessitate ongoing research.

These challenges, however, simultaneously define critical avenues for future research and development. Future directions include the pursuit of more computationally efficient models and training techniques, strategies to enhance factual accuracy and mitigate hallucinations (e.g., through Retrieval-Augmented Generation), continued efforts to address bias and improve fairness, and the development of methods to increase model interpretability. Expanding capabilities into multimodal and multilingual domains, as well as the development of LLM-based agents that can interact with tools and environments, represent exciting frontiers.

In conclusion, while LLMs have fundamentally altered the landscape of AI and automation, particularly in code generation, they are still evolving. Addressing the identified challenges responsibly and effectively is crucial for unlocking their full potential and ensuring their beneficial integration into practical applications. We remain optimistic that LLMs will continue to advance, eventually transforming all facets of coding and potentially enabling the automatic creation of safe, accurate, trustworthy, and controllable code, akin to or even surpassing human expertise for certain problems.

REFERENCES

[1] R. Patil and V. Gudivada, "A review of current trends, techniques, and challenges in large language models (llms)," *Applied Sciences*, vol. 14, no. 5, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/5/2074

[2] C.-N. Hang, P.-D. Yu, R. Morabito, and C.-W. Tan, "Large language models meet next-generation networking technologies: A review," *Future Internet*, vol. 16, p. 365, 10 2024.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[4] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," 2025. [Online]. Available: https://arxiv.org/abs/2402.06196

[5] M. Shao, A. Basit, R. Karri, and M. Shafique, "Survey of different large language model architectures: Trends, benchmarks, and challenges," *IEEE Access*, vol. 12, pp. 188 664–188 706, 2024.

[6] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A survey on large language models for code generation," 2024. [Online]. Available: https://arxiv.org/abs/2406.00515

[7] X. Du, M. Liu, K. Wang, H. Wang, J. Liu, Y. Chen, J. Feng, C. Sha, X. Peng, and Y. Lou, "Evaluating large language models in class-level code generation," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3639219