Deep Learning Architectures For Creating Large Language Models (LLMs)

Sai Sathwik Kosuru[1*], Pallavi Lanke[2] and Madhu Babu Chunduri[1]

[1*]Computer Science and Engineering, B V Raju Institute of Technology, Narsapur, Medak, 502313, Telangana, India.

*Corresponding author(s). E-mail(s): 21211a05r6@bvrit.ac.in; Contributing authors: pallavi503@gmail.com; madhu6858@gmail.com;

Abstract

Recently, Large Language Models have become increasingly ubiquitous in recent years as chatbots. In the current work, there is a Focus on the empirical analysis of Network Architectures for building Large Language models. We could observe 37% enhancements from existing models for the models built after analyzing their architectures. Forming analytics for these architectures and modifying them according to requirements forms a base for future work in this Research direction

Keywords: Large Language Models, Architecture, Deep Learning, Framework, Performance.

1 Introduction

Large Language Models specify the advanced versions of Natural Language Frameworks that enable computer systems to learn from their training features to produce novel insights conversationally with their users. The primary objective of these models is to produce a smart agent that can behave like humans and improve based on human feedback. This concept is called Reinforcement Learning through Human Feedback(RLHF).

Practically, text speech corpora are immensely huge, and it is imperative to design systems that can efficiently handle these huge loads of data and that are capable of transforming these corpora into useful and improvable insights. While multiple models have been developed with all of them being trained on diverse data sets/repositories, one needs to observe the significance of abstracted layers behind. Generally, any neural network like LLM has a layered architecture behind with each layer consisting of one or more nodes and each node is end-to-end connected.

Architectural Analysis is broadly observation-based since it is difficult to build logic/ inter-associations between inherent architectures and their specified applications. In other words, there is no one-on-one relationship/correlation between architectural parameters and LLM performances. Thus, the empirical form of study involving observation-based correlation findings between architectures and performances is called Apriori Analysis.

The current apriori analysis involves multiple LLMs with their inherent architectures focusing on the importance of hyperparameters in each architecture.

1.1 Salient Features Of The Current Apriori Analysis Of LLM Architecture

- **Data-set Based Analysis:** It is worth noting that architectural patterns vary with data sets and features which is an inherent complexity that can never be neglected. To maintain the universality of the study, we implement a multi-class language data set in the current study on which various hyperparameters and architectural layers are tested.
- **Feature-Based Contour Mapping:** Though contour mapping collocates well with visual representations, we can still classify the language corpora into distinct clusters with each cluster hosting its own type of text elements. Typically, these clusters can be visually and interactively displayed as a graphical representation for the model to have diverse learning experiences.
- **Diverse feature extraction:** User input can come from any kind of source/-sphere and hence, is diverse. To account for this, the various classes in the data set correspond to multiple categories of possible combinations of text corpus data.
- **Avoidance of Hallucinations:** There is every possibility for an LLM to capture the noise within the data repositories it is trained on, causing it to overfit over the given data repositories making it behave unintentionally, resulting in hallucinations. Here, we provide critical tradeoffs between performance and requirements to model an effective universal framework for LLM Architectures.

2 Literature Survey

Large Language Models[1] have been studied in terms of performance, rather than architecture in most of the earlier studies.[7] LLMs have been found to be largely computationally complex and architecturally large[13] with multiple dropout layers to facilitate the prevention of overfitting. While the input-output combinations are in linear format, the internal group of layers produces polynomial complexity and non-linear outputs[14]. Hence, most of those studies involved the aspect of non-linearity reduction or input-output transformation.

2.1 Way of Introducing Novel LLMs:

Earlier studies involved the aspect of architectural specification in introducing new LLMs with new architectural stereotypes[2] while simultaneously comparing their performances with existing models/works. The Architectural Modifications that are mentioned explicitly in introducing new LLMs include:

1. **New Layers:** In the existing works, a complete model is created using special layers that are not seen in existing models. Today many of the Novel LLMs include special layers like new-format-encoding layers, new-level-classification techniques[15], and special-case Linear Units.

2. **New Optimizers:** Various Optimizers are available in the market with their techniques ranging from simple optimizers to advanced optimization techniques. Keeping in view the complexity of LLM Architectures, Training and Testing Data Format/ Pattern Specifications, and Non-Standard Input-Output Pair formats, Many of past researchers introduced new optimizers.

3. **New Algorithms:** The types of algorithms involved in the LLM Design of the earlier works include:
   (a) Input Transformation
   (b) Input Format Standardization
   (c) Input Processing
   (d) Input Numerical Conversion
   (e) Weight and Bias Initialization
   (f) Weight Dot Product and Bias Addition
   (g) Output Numerical Conversion
   (h) Output Reprocessing
   (i) Output Format Standardization
   (j) Output Transformation

4. **New Behavior:** The behavior demonstrated by existing works includes novelty in: Predictions on a text corpus Enhanced Human-like Response Novel Personalization Techniques New Hallucination Prevention Methods

2.2 Ways of Enhancing Existing LLMs:

Most of the existing studies focused on enhancing the previous model and this continues as a contiguous linked chain[6] of architectural, performance, and parametric enhancements/improvements[5].

1. **New Parameters:** Various parameters are added to the LLM in the course of novel works in the field of NLP/LLMs[16]. These parameters can include:
   - Novel Hyperparametric Sets
   - Filter Quantity Sets
   - Layer Sub-layer Parametric Specifications
   - Novel Architectural Parametric Settings

2. **New Layer Enhancements:** This can also be considered a new parametric update to the existing works/models/LLMs[17]. Special Layers, Compound Layers, and Simple Layers[18] can be considered as new layers in the enhanced LLM models. The layer enhancements can include:

    - Special Layers such as
        - Linear Units
        - Inter-Layer Transformers
        - Intra-Layer Overfitting Regularization Layers
        - Dropout Layers
    - Compound layers involve a combination of 2 or more simple layers such as
        - Filter Layers
        - Hyperparameter Layers
        - Dropout Layers, with no additional layers/ parameters/ complex settings
        - LLM-Specific Simplified Special Layers
        - Simple Layers

2.3 Way of Applying New LLMs

Though LLM design analysis is important, it is important to cite the importance of applying the LLM to the training and testing[19] and actual real-time text corpus[3][4]. Based on the application standards, we can infer any of the following:

- Train Set Accuracy
- Test Set Accuracy
- Validation Set Accuracy
- personalizability Metrics of the LLM

The ways of introducing the applications of newly proposed LLMs in the existing works include:

1. **Splitting of Text Corpus into training and testing set:** The text corpus for an LLM is split on a feature-specific category basis into:

    - Train set
    - Test Set

    The feature-specific categorization is done on the basis of the following factors:

    - The number of Textual Features:
        - The quantity of the textual features is relevant to how the text corpus can be arranged categorically.
        - Density-based feature-based clustering and mapping
    - The Distribution of Features across the text corpora
        - Feature-Clustering
        - Density-based Grouping

    - Region-based Feature Mapping

2. **Setting Parameters in the Network:** Parametrical Updates to the LLM is very important and critical step because they direct the effective functioning of LLM in its training and application processes. The parametrical setting in a standard LLM includes:

   - Weight Initializations
   - Weight Updations in the Direction of Optimal Performance
   - Bias Additions for Shifting of Input and Output Data
   - Hyperparameter Tuning and Fine-Tuning Parameters

2.4 Way of Comparing New LLMs:

Most of the LLMs are compared on a performance basis and the relative accuracy and precision of these LLMs are considered[20]. While comparing neural networks involves prediction-actual value comparisons and classification-based metric calculations[8], LLMs are compared in how their responses are accurate[9] with respect to user requests.

The comparison factors include:

1. **Efficiency of Response:** An LLM's response is a determining factor for how effective an LLM is in delivering critical information to the users in the form of human-like responses. While all LLMs may not be compared using this comparison category/ standard, it is evident that the majority of LLMs still are compared using these techniques. Most of the comparisons are qualitative, despite some of them including a quantitative aspect.
2. **The likeliness of Response in line with Human Emotions:** Human Emotion Recognition is a Deep Learning Framework in many of the LLMs developed so far and an LLM is designed to derive emotions from the human query. The effectiveness of an LLM is defined by how effectively and perfectly it is able to match the response emotion with the expected emotion conveyed by the user query.
3. **Prevention of Hallucinations:** It is quite evident that LLMs often pick up noise from the training data during the training process. This leads it to give intermixed responses that seem perplexed/ confusing to the user[10]. Previous studies focused on the techniques they employed to reduce/remove/eliminate hallucinations from their LLMs as compared to previously designed/created LLMs.

2.5 Way of Presenting New LLMs

LLMs, though complex, can be largely simplified through multiple available simplification techniques where the crux of their deeply abstracted numerical calculations and relevant simple theorems/equations[11] can be presented as research findings. The LLMs were presented in the following categories:

1. **Existing LLM Architectures:** Existing LLM Architectures were effectively showcased, outlining the important specifications of their architectural parameters and hyperparameters/ fine-tuning parameters. The inconsistencies, bottlenecks,

and defects[12] of the architectures of the previous LLMs were detailedly explained and necessary insights that show characteristic advantages of the work's LLMs over the previous models are demonstrated.

2. **Architectural Structural Additions:** Every new LLM has an additional layer added to it as it is built over the defects of its predecessors. Special/ simple layers/components are added to each LLM based on the requirements for that LLM. The architectural additions may also vary with the type of input the user requests and based on how frequently the user likes the LLM, the additions may vary too.
3. **Architectural Parametric Additions:** New and novel parameters may be added to LLMs as more and more extensive research works are performed. While a completely new set of/new parameters may not be added, the number of parameters may vary. In addition, the number of parametric updates will also vary.
4. **Architectural Dynamic Analysis:** Every study presents a comprehensive coverage of the flow-level dynamics of layer-wise behavior in an LLM. Based on the flows analyzed, the conclusions are drawn into how performance is effectively mapped to architectural enhancements.

3 Methodology

3.1 Apriori Analysis of LLM Architectures

3.1.1 Advantages Of The Current Apriori Analysis Of LLM Architecture

Avoidance of Complexity: Instead of analyzing each layer analytically and then providing insights into the architecture thereby producing detailed performance-architecture mappings, the current study involves a concise study of LLM Architectures on an observation-basis, implying it is parameter-specific rather than focusing on deep numerics within each layer. The results prove that observation-based analytics are simple yet efficient solutions for analyzing the architecture frameworks in detail.

Architectural-Specific: The performance of any neural network-based model lies in how it is tuned architecturally rather than how it works. This is because the root of the field lies is based on the inspiration of the human brain's structure and its efficiency in working intelligently. Right from the first McCulloch Pitt's model, all neural networks have been architecturally significant.

Involvement of Human-Specific Response: The study involves the usage of text corpora that involve human-specific and human-synthesized input-output data pairs only, strictly restricting the LLM to behave intelligently like a human, instead of being purely logical. The study studies this particular aspect of LLM since we hypothesize that a computer works only based on logic thus greatly limiting it's behaviour as a human. We aim to technically design architectures that can learn effectively from the given text corpus to produce human-like behavior.

3.1.2 Trade-offs To Be Considered During The Current Apriori Analysis Of LLM Architecture

The most important and the most fundamental trade-off to be considered is to maintain a delicate balance between LLM Performance and Its Architectural Complexities. It may not always be possible for an LLM creator/ designer to effectively tune the LLM to the requirements and make its computational requirements simpler. At the same time, it is not always possible to synthesize a perfect LLM that can train fast and produce 100% accurate responses/results. The key aspect of this discussion is the fact that an LLM is still a computer-created virtual machine that still holds some inherent limitations of the prior. Therefore, the following trade-offs are to be considered and have been considered for the current Apriori Analysis:
Hyperparameters vs Conventional Architectural Parameters: Hyperparameters add an additional overhead over conventional/normal feature-extracting/dimensionality-reducing/dropout/filter-specific parameters. Though these are overheads, it is still mandatory to add hyperparameters since we cannot meet expected behaviors without these parameters because the text corpora is always increasing and we need new enhancements every time.
Transfer Learning vs Reinforcement Learning: As mentioned previously, LLMs are generally trained on the training text corpora that keep increasing dynamically and have no predictable behavior. In such cases, it is difficult to a one-for-all LLM that serves all purposes with its training on instantaneously available training data. To adapt to the changing data and dynamic LLM requirements, we must enable transfer learning while training these LLMs. Transfer Learning is the concept where we use another pre-trained model to accelerate the LLM training process by at least 50-75%, thus making the LLM training fast and making the trained LLM sufficiently diverse and universal.
Linguistic Features vs Pattern-based Features: There are two types of features in LLM training and architectural tuning during feature extraction: Numerical and Non-Numeric Feature Extraction. Numerical is a pattern-based clustering technique while Non-Numeric is based on linguistic feature-based distinction.

3.2 Apriori Analysis Procedure

3.2.1 Surveying

The study follows a typical star-topology-modeled framework, as seen in figure 1 to analyze the associations between a single multi-class data set and the proposed listed descriptions of LLM Architectural parameters.

The classes of the data set represent diverse possible features for a human question/response. Its main task is to curate the text corpora such that it reflects the generic attitude of a human who interacts with the agent in an explicit manner where there is no possibility for ideal conditions.

The classes exhibit the following patterns of typical user behavior:

- Spelling/Grammatical Errors and Phonetic Confusions
- Inconsistent Sentence Structure and Irrelevant Sentence Fillers

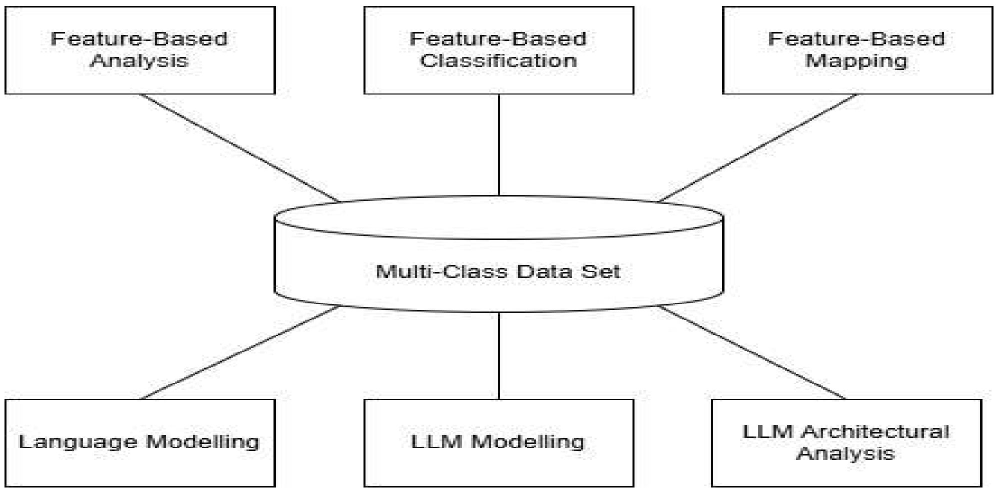

Fig. 1 Surveying

- Ambiguous Word/Sentence Meanings and Punctuations
- Non-Standard Verbal Usages
- Reference to Unknown Sources of Language
- Nativity in Responses

3.3 Data-Set Study

Since the data set is multi-class, it is essential to know the characteristic features of each class as shown in figure 2 and it is crucial to demarcate them structurally before implying model applications on them. We can implement various n-grams to compare each class linguistically to identify patterns and cluster them into groups. This entails clustering the data set into various groups on any of the following mentioned bases:

- Linguistic Affinity and Possible Nativity Differences
- Word Frequencies and Phrased Structures

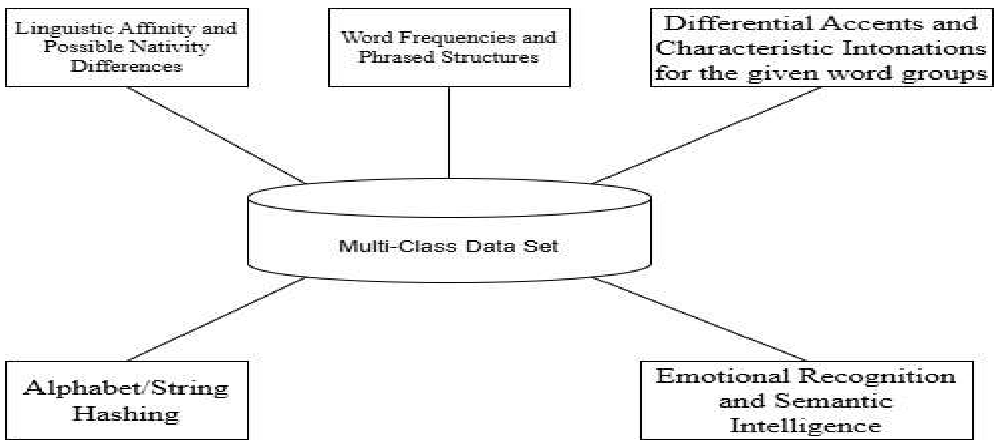
Linguistic Affinity and Possible Nativity Differences
Word Frequencies and Phrased Structures
Differential Accents and Characteristic Intonations for the given word groups
Multi-Class Data Set
Alphabet/String Hashing
Emotional Recognition and Semantic Intelligence

Fig. 2 Data Set Study

- Differential Accents and Characteristic Intonations for the given word groups
- Alphabet/String Hashing
- Emotional Recognition and Semantic Intelligence

3.4 Feature Analysis

Feature Extraction is the second and the building step for building the whole LLM. It is the most complex step as it involves building feature space and extracting the only relevant ones from the total space, as demonstrated in figure 3. We can prepare feature maps based on the given data set according to the following features which can be collectively termed as a feature space:

- text-based numerical correlations
- geometrical feature mapping
- linguistic parameters

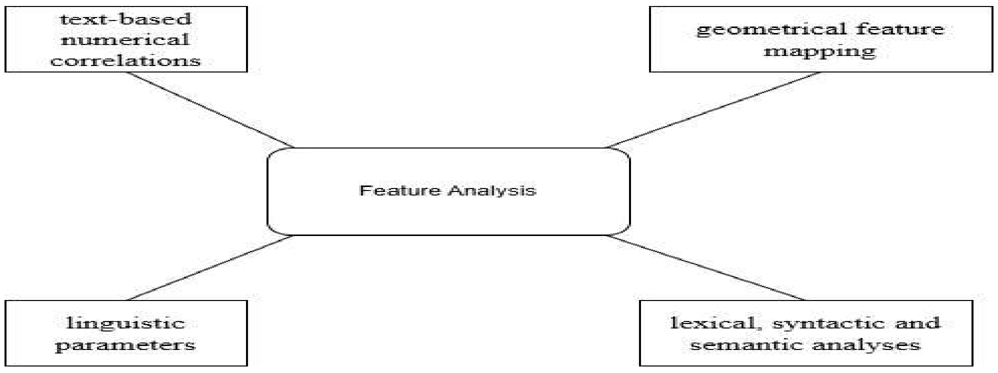
text-based numerical correlations
geometrical feature mapping
Feature Analysis
linguistic parameters
lexical, syntactic and semantic analyses

Fig. 3 Feature Analysis

- lexical, syntactic, and semantic analyses

3.5 Conventional Architecture Parametrical Analysis

Conventional Architectural parameters(4) include:

- Filters
- Number of Layers
- Dropout Layers
- Regularization Factors
- Specifics of Nodal Composition for each layer

Conventional Parameters are the standard parameters used in a normal neural network with normal requirements and nominal data available for training purposes. These specify the normal functioning considerations for a network while keeping aside any special adjustments that may be made for its optimal performance. The Apriori Analysis of these conventional parameters includes:

- Mapping between Features Extracted and Architectural options
- Ranking of Architectural Parameters as a function of LLM's effectiveness

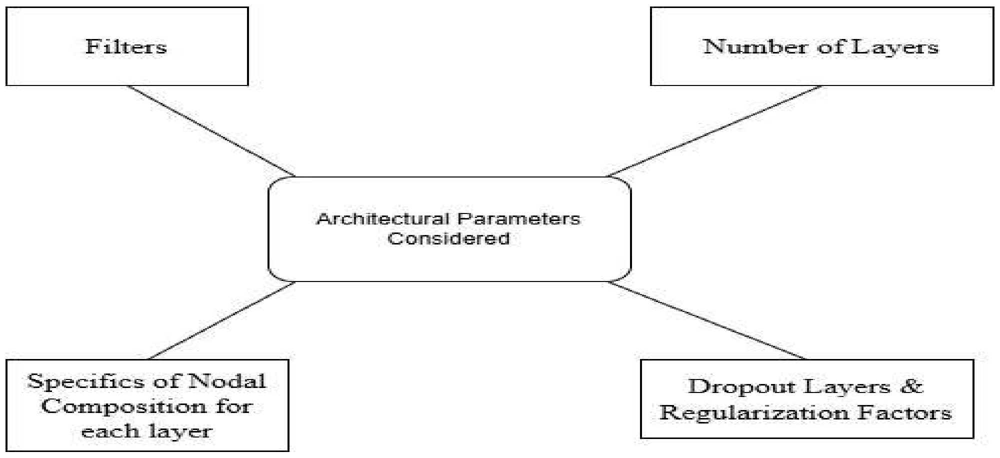
Filters
Number of Layers
Architectural Parameters Considered
Specifics of Nodal Composition for each layer
Dropout Layers & Regularization Factors

Fig. 4 Architecture Parametric Analysis

- Weighting of features for the effectiveness of the resulting LLM Reducing
- Hallucinations as a function of dropout layers

3.6 Architectural Flow Analysis

The Neural Network training procedure is a dynamic entity with multiple internal movements and data modifications, most of which are clearly abstracted under the form of discrete layers of nodes and their associations. This is seen in figure 5. While we need to know the details of static features of the network like:

- Individual Layer Composition
- Node-Node Associations
- Node-Node Initial Weight Vectors
- Layer Components such as

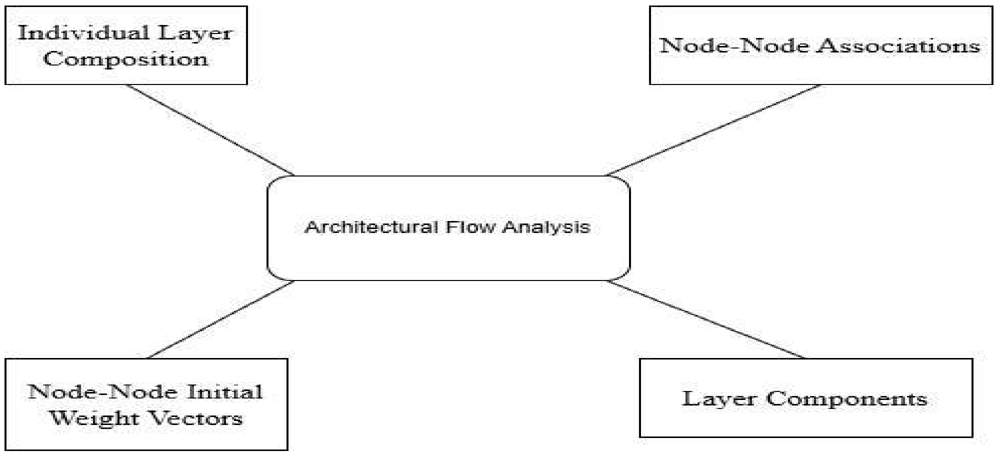
Individual Layer Composition
Node-Node Associations
Architectural Flow Analysis
Node-Node Initial Weight Vectors
Layer Components

Fig. 5 Architectural Flow Analysis

- Filters
- dropout parameter
- node penalties, if any
- number of epochs where this particular layer may be skipped
- Intra-Layer Nodal Associations

It is also equally important to model the dynamic flow of data across the nodes within each layer and between layers. This can be modeled effectively by a flow model where a detailed flow analysis can be performed. It is worth noting that deep study of these intricate nodes is an impractical task. The flow analysis involves the following details:

- **Input Format Specification:** Text corpora involve a lot of inconsistent data structures/ patterns owing to the differences in the sources from which they are drawn. We need to standardize this and make the format universal to suit all kinds of inputs sent to the model

- **Weight Updation and Importance Computation:** Not all features of the input may be equally important. So, the weight vector is initialized and embedded to every layer and this is multiplied with the data as it flows. Based upon the performance of the resulting model, the individual weight compositions may be modified resulting in a significant change in importance association.
- **Parametric Modification and Data Manipulation:** Parameters like weight vectors, bias constants, number of filters per layer, dropout parameters, regularization factors, etc., influence how the LLM is going to interpret and manipulate the data that passes through its layers.

3.7 Architectural Modification Analysis

Architectural Synthesization and Modification are inherent in every model/LLM and it is very important to tune the architecture in an expected proportion with the model's requirements and expected input behaviors. Since RLHF is an inescapable procedure during the development of an LLM, Architectural Modification is important for tuning the model's behavior according to the requirements. This is shown in fig 6. The architectural modifications include:

- Quantitative Increase of Filters
- Density Adjustments For Layers
- Overfitting Prevention through Regularization Parameters
- Early Dropout for Saturation

3.8 Performance Analysis

This is the penultimate step in the architectural analysis for LLM synthesis where Architecture-Performance mapping(7) is visualized and inferred from the model's performance. The Model has 3 categories of performance:

1. The Model was trained properly
   - Reflected through accuracy above 85
   - Giving appropriate responses to any kind of human input
   - Acknowledging Error in a user-friendly manner
   - Ability to Adapt to Human Character
   - Ability to personalize
2. The Model was trained on noise
   - This is termed Hallucination
   - The model captured all the noise in the given data/corpora
   - The LLM is mixing all its learnings during its responses
   - The responses are largely generic and vague
   - No personalization is observed
   - No user-oriented design observed
   - Poor Accuracy or Inconclusive Performance
3. The Model failed to learn

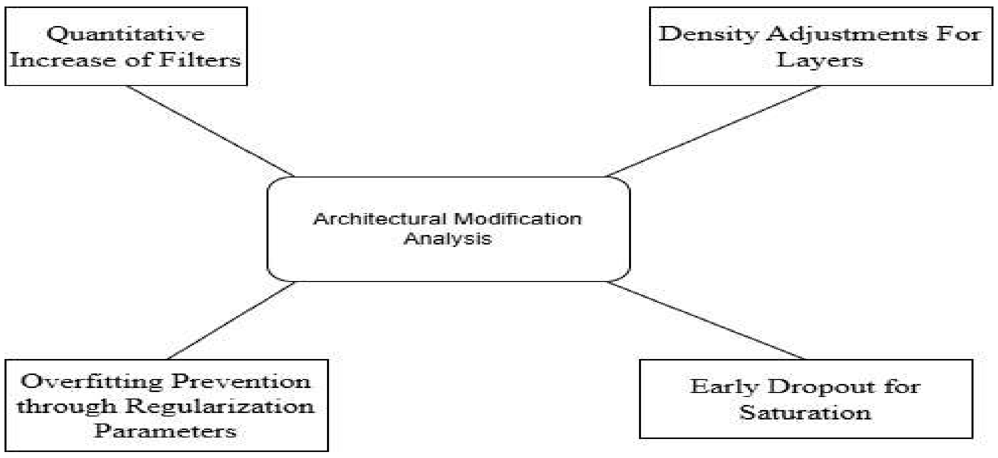
Quantitative Increase of Filters
Density Adjustments For Layers
Architectural Modification Analysis
Overfitting Prevention through Regularization Parameters
Early Dropout for Saturation

Fig. 6 Architectural Modification Analysis

- Very poor accuracy
- Couldn't answer even simple queries
- Poses more questions
- Responses largely constrained to a specific sphere/field of trained data
- The model is seen as highly domain-specific
- No observation of RLHF

The performance-architectural factors that can be observed include:

- Additional Layers and Their Relevance To Overfitting of LLM
- Regularization and Dropout Layers' Role in Preventing Overfitting
- Tackling of Hallucinations by the special layers of LLM
- Importance of Layer-wise Modifications in improving performance
- Performance-based Effective Data Disbursal Requirements from Architectural Flow Analyses

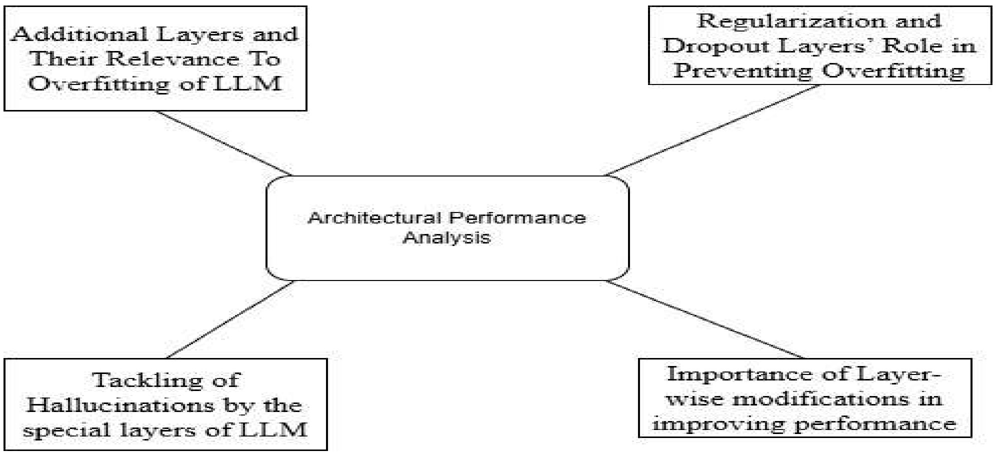
Additional Layers and Their Relevance To Overfitting of LLM
Regularization and Dropout Layers' Role in Preventing Overfitting
Architectural Performance Analysis
Tackling of Hallucinations by the special layers of LLM
Importance of Layer-wise modifications in improving performance

Fig. 7 Architectural Performance Analysis

3.9 Post-Performance Analysis

Post-performance analysis of an LLM includes two categories as seen in figure 8:

- Fine-Tuning Analysis
- Hyperparameter Tuning Analysis

3.9.1 Fine-Tuning Analysis:

Fine-tuning is extra/extra-conditional/extra-constrained data elements that are fed as training elements to the trained LLMs. These are post-training processes in the LLM where extra features are fed into the model to accommodate universal problem domains and all-around requirements.

Fine-tuning architecture-specific analysis includes:

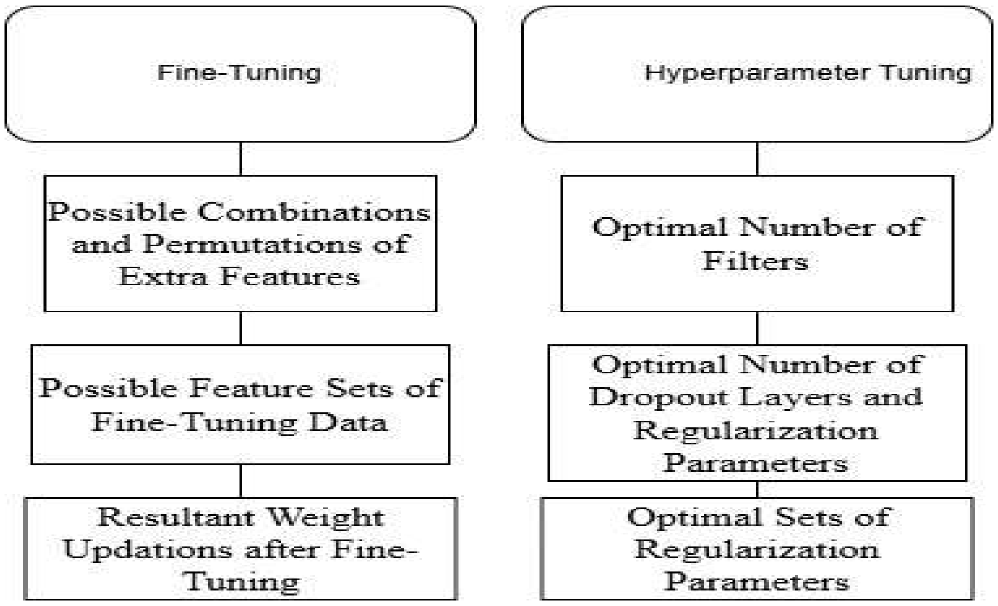
Fine-Tuning
Possible Combinations and Permutations of Extra Features
Possible Feature Sets of Fine-Tuning Data
Resultant Weight Updations after Fine-Tuning
Hyperparameter Tuning
Optimal Number of Filters
Optimal Number of Dropout Layers and Regularization Parameters
Optimal Sets of Regularization Parameters

Fig. 8 Post-Performance Analysis

- Possible Combinations and Permutations of Extra Features
- Possible Feature Sets of Fine-Tuning Data
- Resultant Weight Updations after Fine-Tuning
- Modified Performance After Fine-Tuning

3.9.2 Hyperparameter Tuning Analysis:

Hyperparameters are nothing but architectural parameters but they are used as architectural descriptors and are often taken as additional overheads. They include:

- Optimal Number of Filters
- Optimal Number of Dropout Layers and Regularization Parameters
- Optimal Sets of Regularization Parameters

Hyperparameter Tuning Architecture-Specific Analysis Includes:

- Layer-Hyperparameter Mappings
- Layer-Hyperparameter Individual Quantitative Metrics
- Hyperparameter Effect on the Overall LLM Performance

4 Results and Discussion

In the current Apriori Analysis, we present architecture-specific findings on 5 distinct LLMs as follows, which is also succinctly summarized in the table 1:

1. BERT (Bidirectional Encoder Representations from Transformers)
2. ALBERT – A Light BERT
3. RoBERTa (Robustly Optimized BERT)
4. GPT (Generative Pre-trained Transformer)
5. BLOOM (BigScience Large Open-science Open-access Multilingual Language Model)

4.1 BERT

4.1.1 About the LLM:

- Open-source ML framework
- Based on Google AI
- Used for Human-like text corpus
- Transformer architecture with encoder and decoder modules
- Focus on the analysis of input sequences over output sequences

4.1.2 Apriori Analysis Results:

The apriori model for BERT is demonstrated in figure 9. BERT involves encoding and decoding of over 110 million parameters which involves a significant amount of analysis for the following pointers:

- Subjective and Objective analysis of input features
  - Subjective analysis includes:
    - Semantical Structure of the input corpus
    - Syntactical Intelligence of the text stream
    - Emotional and Semantical Extraction from the text
  - Objective Analysis includes:
    - The amount of instantaneously trainable text
    - Impact of Individual Components on overall performance
    - Step-wise performance at that layer
- BERT has simultaneous architectural applications and feed forward training components in its architecture
  - The Feedforward Layer and Multi-Head Attention components are present.

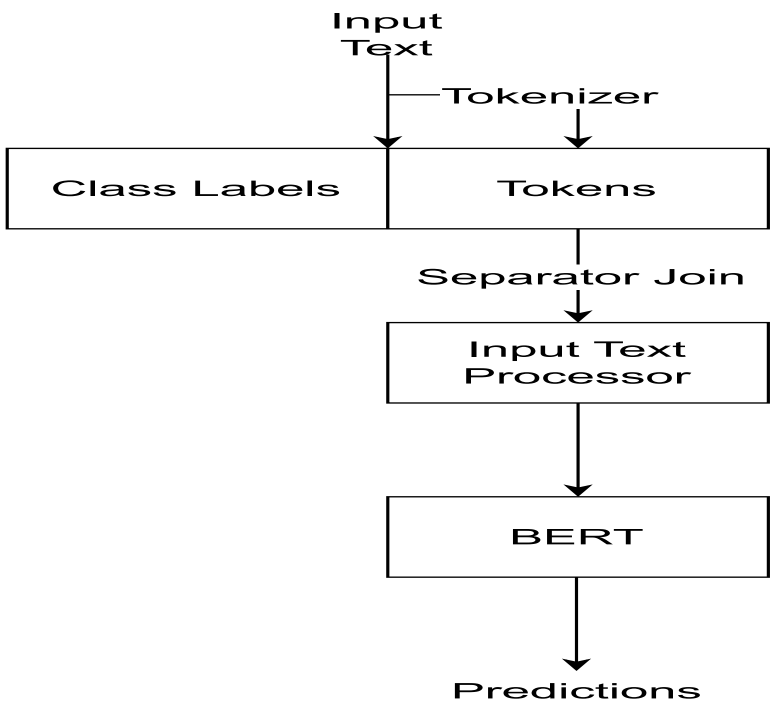

Fig. 9 BERT Apriori Model

∗ Multi-Head Attention Component computes:

- Simultaneous access to the training text corpus
- Handles multiple constraints and variations across various parts of the text
- Expands the learning region of the LLM

∗ Feed Forward Layer involves:

- A straightforward mode of learning is involved
- This entails normal learning procedures as observed in other conventional neural networks
- This forms a simplified base for remaining complex procedures in the BERT model

  - Adding & Normalization step is involved as intermediary procedure in every step of the learning process.

4.2 AlBERT

4.2.1 About the LLM:

Albert is a lightweight BERT LLM that is developed as an enhancement over the original BERT model.

- 2 parameter-reduction techniques
- Low memory consumption
- Increased training speed
- Repeated layers employed
- Simplified embedding matrix

4.2.2 Observations from Apriori Analysis:

With the ALBERT apriori model shown in figure 10,

- It has parametric reductions as follows:
  - Splitting of Embedding Matrix into 2 smaller matrices
  - Using repeated layers split into groups
- It has a passthrough layer called a backbone that helps support the whole network for its robustness in structure and optimality in performance.
- It also implements variational analysis on the provided text corpus with layers as follows:
  - One-to-many head:
    * Uses 2 techniques such as Regression and Classification
    * One part is compared among the whole text corpus
    * This ensures the universality of each trained component
  - One-to-one head:
    * Each text corpus element is compared to other elements individually
    * It also employs 2 techniques such as Regression and Classification
    * This makes each trained corpus much better than in BERT model
- Integration over Union (IoU) is applied to the obtained results.
- From the analysis, it is clear that Albert is better than the previous model in memory efficiency but has no enhancement in its performance.

4.3 RoBERTa

4.3.1 About the LLM:

- Uses Self-attention mechanism
- For Generating Contextualized Representation of words in a sentence

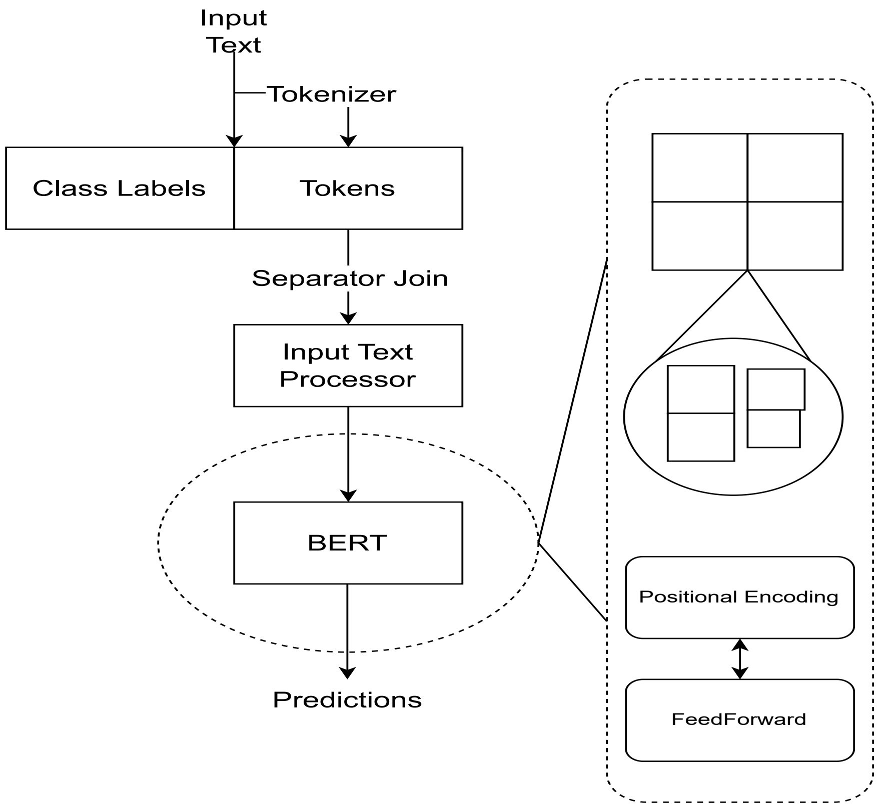

Fig. 10 AlBERT Apriori Model

- More Focus on input sequences
- Pre-trained on large amounts of data
- Developed by Facebook AI

4.3.2 Apriori Analysis Observations:

We modeled 11 for Roberta after a detailed Apriori Analysis.

- Premise-Hypothesis-based data is used for training
- The input sentence is tokenized first
- Closures and Separators are also implemented
- The Token set is divided into 2 subsets:
  - Premise
  - Hypothesis

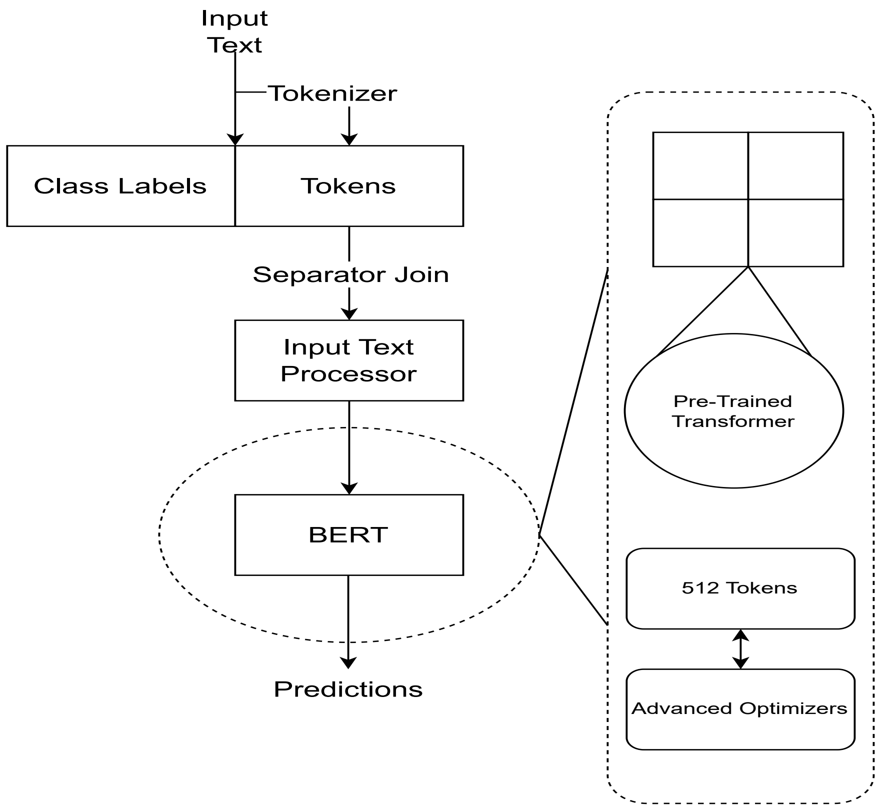

Fig. 11 RoBERTa Apriori Model

- This constitutes a part of the Classification Head
- This is constituted in a linear unit with the original BERT model
- The only observed advantage over a conventional BERT model is that it is trained on a larger data set, hence robust.

4.4 GPT

4.4.1 About the LLM:

- Basis for Gen AI component of LLM
- Based on Deep Learning Architecture
- Trained on unlabelled text
- Produces Human-like text
- Pre-trained on large data sets of unlabeled text

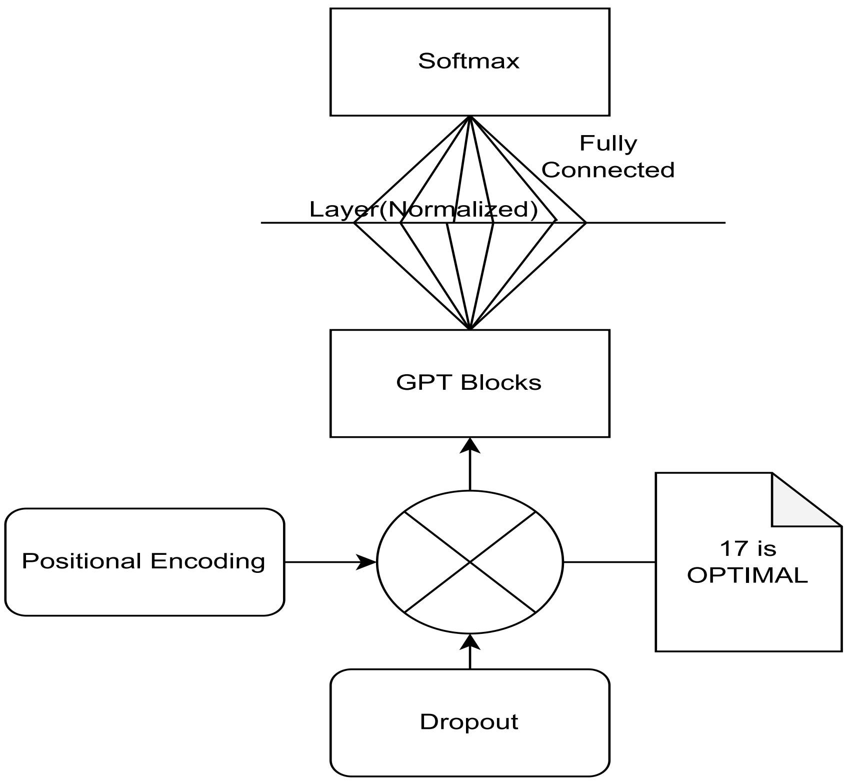

Fig. 12 GPT Apriori Model

4.4.2 Observations from Apriori Analysis:

Figure 12 is the Apriori model for GPT.

- The first Layer is broadly used for input normalization
  - This is for standardizing the input data
  - Focus on the universality of the LLM
- Positional Encoding is applied over the input data for compressing it according to the model's quantitative requirements.
- Additionally, the following techniques have been found to be useful:
  - Matmul
  - Mask
  - Softmax

  - Dropout
  - Matmul

- All the upper layers are found to be requiring linear data elements and so the data is subsequently passed through a linear unit.
- Dropout is mandatorily applied to prevent overfitting
- Layer Normalizations follow
- The transformer was found to produce linear output
  - The linear output was found to be any one of the following:
    - Character Stream
    - Conventional Sentences
    - Encoded Versions of Sentences
    - Compressed Versions of Sentences

4.5 BLOOM

4.5.1 About the LLM:

- It consists of 176 billion parameters
- It is a transformer-based autoregressive LLM
- It is available under free license
- It is trained on 366 billion tokens
- It is a collaborative project

4.5.2 Observations from Apriori Analysis:

Figure 13 reflects the apriori model for BLOOM.

- Embeddings were primarily used in the input layer
  - The vector representations include tokenized representations distributed over a 3-dimensional space
  - Embeddings employed since Bloom Models involve training on tokens
- Layer or Input Normalization is also evidently implemented here
- Positional encodings are then applied over the input text
  - Since tokens are employed, it is important to compress input data
  - Though tokens are small individually, the stream is encoded effectively.
- Contiguous Blocks called Bloom Blocks follow this layer:
  - 70 is the standard count of Bloom Blocks for a Bloom LLM.
- It has sub-neural Networks such as:
  - Multi-Layer Perceptron:
    - This is a conventional network that is used as a normal deep learning architecture.

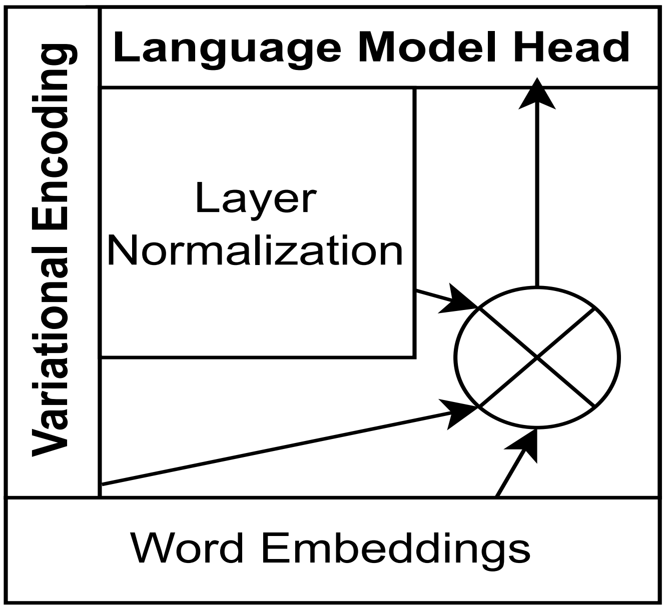

Fig. 13 BLOOM Apriori Model

        * This is used to make predictions on the input text corpus.

  - Self-Attention:
    * It deals with long input sequences
    * This is used to determine the relative importance of words in the input corpus.

- Layer Normalization is again applied at this stage
- This forms the final layer of the LLM
  - This is called Language Model Head

Table 1 Final Apriori Observations

Model	Apriori Observations	Inference
BERT	Encoder-Decoder	Human-Like Memory Efficient Response
AlBERT	Memory Reduction Parameters	Memory Efficiency
RoBERTa	Optimization Techniques	More Efficiency in Responses
GPT	Transformer Architecture	Simple Human-Like Agents
BLOOM	Huge Number of Parameters	Depth in Learning Effectiveness

5 Conclusion

Apriori Analysis of LLM Architectures is used to infer observation-based analytics of LLM behaviors with data considerations and performance assessments. A fundamental problem in building Apriori Analysis for LLM Architecture lies in the fact that they are too complex. In this study, we present a detailed study of the apriori analysis of the state-of-the-art LLM Architectures. While we performed empirical studies of the LLM, it is critical to further expand the study beyond purely observation-based frameworks to practical implications. Detailed numerical rubrics are required to infer the exact significance of each layer present in every LLM. We plan to further expand the current study by invoking certain numerical metrics and aspects while also focusing on performance-architecture equations in future work. A new model enhancing the already built empirical techniques that effectively span across all LLMs can be expected as a comprehensive framework in the near future.

References

[1] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A., "A comprehensive overview of large language models," arXiv preprint arXiv:2307.06435, 2024.

[2] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., and others, "PaLM: Scaling language modeling with Pathways," arXiv preprint arXiv:2204.02311, 2022.

[3] Peng, C., Yang, X., Chen, A., et al., "A study of generative large language model for medical research and healthcare," npj Digital Medicine, vol. 6, p. 210, 2023, DOI: 10.1038/s41746-023-00958-w.

[4] Clusmann, J., Kolbinger, F. R., Muti, H. S., et al., "The future landscape of large language models in medicine," Communications Medicine, vol. 3, p. 141, 2023, DOI: 10.1038/s43856-023-00370-1.

[5] Raiaan, M., Hossain, Md. S., Fatema, K., et al., "A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges," TechRxiv, 2023, DOI: 10.36227/techrxiv.24171183.v1.

[6] Zhou, L., Schellaert, W., Martínez-Plumed, F., et al., ”Larger and more instructable language models become less reliable,” Nature, 2024, DOI: 10.1038/s41586-024-07930-y.

[7] Zhao, W. X., Zhou, K., Li, J., et al., ”A survey of large language models,” arXiv preprint arXiv:2303.18223, 2024.

[8] Hoffmann, J., Borgeaud, S., Mensch, A., et al., ”Training compute-optimal large language models,” arXiv preprint arXiv:2203.15556, 2022.

[9] Mani, G., Namomsa, G. B., ”Large Language Models (LLMs): Representation Matters, Low-Resource Languages and Multi-Modal Architecture,” 2023 IEEE AFRICON, Nairobi, Kenya, pp. 1-6, 2023, DOI: 10.1109/AFRICON55910.2023.10293675.

[10] BigScience Workshop, Le Scao, T., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., et al., ”BLOOM: A 176B-parameter open-access multilingual language model,” arXiv preprint arXiv:2211.05100, 2023.

[11] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W., ”LoRA: Low-rank adaptation of large language models,” arXiv preprint arXiv:2106.09685, 2021.

[12] Meyer, J., Urbanowicz, R., Martin, P., O'Connor, K., Li, R., Peng, P.-C., Bright, T., Tatonetti, N., Won, K., Gonzalez, G., and Moore, J., ”ChatGPT and large language models in academia: opportunities and challenges,” BioData Mining, vol. 16, 2023, DOI: 10.1186/s13040-023-00339-9.

[13] Fan, L., Li, L., Ma, Z., Lee, S., Yu, H., and Hemphill, L., ”A Bibliometric Review of Large Language Models Research from 2017 to 2023,” arXiv preprint arXiv:2304.02020, 2023.

[14] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., et al., ”Scaling instruction-finetuned language models,” arXiv preprint arXiv:2210.11416, 2022.

[15] Jeon, J., and Lee, S., ”Large language models in education: A focus on the complementary relationship between human teachers and ChatGPT,” Education and Information Technologies, vol. 28, pp. 15873–15892, 2023, DOI: 10.1007/s10639-023-11834-1.

[16] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J., ”Large language models: A survey,” arXiv preprint arXiv:2402.06196, 2024.

[17] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., and Galan, F., "The reservoir model and architecture for open federated cloud computing," IBM Journal of Research and Development, vol. 53, no. 4, pp. 4-1, 2009.

[18] Boiko, D. A., MacKnight, R., Kline, B., et al., "Autonomous chemical research with large language models," Nature, vol. 624, pp. 570–578, 2023, DOI: 10.1038/s41586-023-06792-0.

[19] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., et al., "LLaMA: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.

[20] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., et al., "Large language models encode clinical knowledge," arXiv preprint arXiv:2212.13138, 2022.