A

PROJECT SCHOOL REPORT

ON

LLMfAMAT: LLM FOR ADVANCED MATERIALS

Submitted By

Lakkadi Abhigna Reddy	**Roll no: 245322733091**
Alekhya Boju	**Roll no: 245322748009**
Rishit Senapati	**Roll no: 245322733176**
Mounika Navathu	**Roll no: 245322733165**
Kanchu Navyateja	**Roll no: 245322733154**
Nalla Rishitha Reddy	**Roll no: 245322733168**

Under the guidance

Of

Ramakrishna Kuppa

Professor of Practice, Science and Humanities

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

AUGUST, 2024

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the project work entitled "LLMfAMAT: LLM FOR ADVANCED MATERIALS**"** *is a bonafide work carried out by "ABHIGNA, MOUNIKA ,RISHIT, RISHITHA, NAVYATEJA"* of II-year IV semester **Bachelor of Engineering** *in CSE and "ALEKHYA"* of II-year IV semester **Bachelor of Engineering** *in CSE (AIML) during the academic year* **2023-2024 and** *is a record of bonafide work carried out by them*.

Project Mentor

Ramakrishna Kuppa

Professor of Practice

CONTENTS

ABSTRACT

Machine learning (ML) and artificial intelligence (AI) are changing the way we research materials. Instead of just using traditional methods to predict material properties from large datasets, scientists now use AI to find new and efficient ways to discover and design materials. This shift is speeding up the process and leading to significant advancements in various fields. This project aims to use AI to develop advanced, customized materials for specific applications, such as eco-friendly water bottles and sustainable fuels. We will train a Large Language Model (LLM) with a detailed dataset of polymer materials. This will help the AI suggest new and improved materials. Additionally, we will create an easy-to-use web interface using MERN-stack technology to test and validate these AI-generated suggestions.

Our project will focus specifically on polymer materials because they are versatile and have many uses. The key steps in our project include choosing the right AI model, collecting and organizing a large dataset of polymer materials, and ensuring its quality for training the model. We will then fine-tune the LLM with this dataset to improve its accuracy. After that, we will design thorough tests to evaluate the AI's suggestions and make sure they meet the required standards. Finally, we will build a user-friendly web interface for researchers and developers to interact with the AI model, submit queries, and receive material suggestions.

By combining AI with materials science, our project aims to solve real-world problems by creating new materials that offer practical solutions. Our goal is to make significant progress in developing materials that meet modern society's needs, contributing to advancements in technology, sustainability, and overall quality of life. We believe this project will not only speed up materials discovery but also inspire new methods and applications across different sectors. By integrating AI and human expertise, we aim to solve pressing challenges and promote a sustainable future.

In summary, this project represents a major step toward merging AI and materials science to unlock new potentials in material design and discovery. Through the collaborative efforts of AI and human expertise, we aim to address real-world challenges and create important new materials that can solve modern problems and improve the quality of life for people everywhere.

CHAPTER-1

INTRODUCTION

Imagine a world where the materials we use every day are tailored just for us—like a custom-made suit, but for everything from our water bottles to the fuel in our cars! Just like how humanity evolved from using stone tools to advanced silicon chips in our gadgets, we're now looking to leap forward by creating new, high-tech materials that meet our modern needs.

Think about how essential materials are in our lives: the chips that power our smartphones, the lightweight metals in sports cars, and even the packaging for our food. But here's the twist—what if we could go beyond just tweaking these materials? What if we could invent entirely new ones from scratch? That's where this project comes in!

We're using Artificial Intelligence (AI) to help us design these futuristic materials. Specifically, we'll be fine-tuning a smart AI model called a Large Language Model (LLM) to discover and design new materials. Imagine teaching a super-smart robot to come up with new ideas for materials that are not only cool but also useful, like a plastic that breaks down after you use it or a fuel that's better for the planet.

To make this happen, we'll follow a few key steps:

Choosing the AI Model:

To start this project, we picked the StabilityAI model as our main AI tool. We used a sub word tokenizer from Hugging Face, which breaks down words into smaller parts to help the AI understand materials better. Once we had the tokenizer ready, we focused on fine-tuning the StabilityAI model. We trained it with a detailed dataset about polymers to help it come up with discovery of new materials. Fine-tuning was super important to make sure the model could give us the best suggestions for designing materials.

Gathering Data:

To ensure our AI model functions effectively, we begin by collecting extensive data on polymers, specifically their glass transition temperatures (Tg), from diverse sources such as scientific research, material databases, and experimental studies. Once the data is gathered, we carefully clean and organize it by correcting any errors, standardizing measurements to a common unit like Celsius, and ensuring consistent naming conventions for polymers. This thorough preparation of the dataset is crucial, as it helps the AI model learn from accurate and well-structured information. With this organized data, the AI can make reliable predictions and assist us in designing and discovering new materials tailored to specific needs.

Training the AI:

To train the AI, we'll use the data we've gathered to help it learn how to come up with new material ideas. We'll input the cleaned data into the AI model, and it will analyze the information to understand patterns and relationships. As it learns, the AI will improve at making predictions and suggesting new materials based on what it has learned. This training is key to making sure the AI can offer creative and useful ideas for new materials.

Building a User Interface:

we will build a simple and user-friendly interface. We'll use the MERN stack, which includes MongoDB for storing data, Express.js and Node.js for handling the server, and React for creating the front-end interface. This app will allow users to easily enter information, view AI predictions, and explore new material designs. The goal is to create a straightforward platform that makes it simple for everyone to use and benefit from the AI's capabilities.

CHAPTER-2

LITERATURE SURVEY:

A literature survey is a comprehensive review and analysis of existing research, theories, and findings relevant to a specific field of study. It aims to summarize and synthesize the current state of knowledge, identify trends, gaps, and inconsistencies, and provide a context for new research. By evaluating and comparing previous works, a literature survey helps to establish a foundation for understanding the research problem and highlights the significance of the new study in relation to existing knowledge. This process not only informs the development of research questions and methodologies but also ensures that the new study builds upon and contributes to the ongoing scholarly conversation in the field.

H.G. Weyland, P.J. Hoftyzer, and D.W. Van Krevelen[1], in their paper "Prediction of the Glass Transition Temperature of Polymers," create models to estimate the glass transition temperature (Tg) of different polymers, which is important for understanding their use and behavior. They use these models to predict Tg based on the polymer's structure and composition, and their results are shown to be accurate when compared to experimental data. This paper is important for your project because it provides methods to predict Tg, which could be useful for your LLM. The models they present are helpful, though adding advanced machine learning techniques and exploring more types of polymers could improve predictions further.

Lei Tao, Guang Chen, and Ying Li's study[2], "Machine Learning Discovery of High-Temperature Polymers," utilizes a deep neural network (DNN) model trained on nearly 13,000 polymers from the PoLyInfo database to predict the glass transition temperature (Tg) of polymers. Their model, trained with 6,923 experimental Tg values, shows strong accuracy and generalization in predicting unknown Tg values. By screening one million hypothetical polymers, they identify over 65,000 with Tg greater than 200°C, significantly expanding the pool of known high-temperature polymers and offering new opportunities for polymer development.

M. Hergenrother’s 2003 review, "The Use, Design, Synthesis, and Properties of High Performance/High Temperature Polymers: An Overview,"[3] explores the development and applications of high-performance polymers. The paper focuses on key families like polyimides and polyarylene ethers, demonstrating how chemical structure influences properties. It also notes that the global market for these polymers was $4.36 billion in 2000, with polyimides representing a significant portion. The market is expected to grow with economic improvements.

In their 2020 paper[4], R. Batra, L. Song, and R. Ramprasad explore the emerging role of machine learning (ML) in materials science. They highlight how ML algorithms and expanding data repositories are enhancing the prediction of material properties and performance. The authors also discuss advancements in automated data capture and the potential of AI to transform laboratory processes, leading to more efficient material design and synthesis.

Choi, S. Yu, S. Yang, and M. Cho (2011)[5] explore the impact of silicon carbide (SiC) nanoparticles on the glass transition and thermoelastic properties of epoxy-based nanocomposites using molecular dynamics simulations. Their study reveals that embedding SiC nanoparticles improves the glass transition temperature and enhances the coefficient of thermal expansion and elastic stiffness of the nanocomposites. The effect of nanoparticle size on these properties is observed both below and above the glass transition temperature.

Palomba, Vazquez, and Díaz (2012)[6] present new descriptors for predicting the glass-transition temperature (Tg) of high-molecular-weight polymers. Their model, based on molecular modeling of polymer trimer units, uses three key descriptors: main chain surface area, side chain mass, and number of rotatable bonds. Their multi-layer perceptron neural network achieves high accuracy in predicting Tg, demonstrating the effectiveness of these descriptors in capturing polymer structure for Tg prediction.

Kim, Chandrasekaran, Jha, and Ramprasad (2019)[7] investigate active-learning frameworks for discovering polymers with high glass transition temperatures (Tg). They use Gaussian process regression and iterative selection to enhance their dataset of polymer measurements, comparing decision-making strategies (exploitation, exploration, balanced) against random approaches. Their method effectively identifies high-Tg polymers by guiding the learning process dynamically.

McKenna investigates the behavior of glass-forming materials[8], focusing on phenomena below the glass transition temperature. The study addresses the challenges posed by extremely long relaxation times and the Kauzmann paradox. McKenna explores methods to create ultra-stable glasses and examines the dynamic and thermodynamic properties of materials to question the existence of an ideal glass transition.

Chong et al. [9]review the transformative impact of machine learning (ML) on materials science in the era of big data. They highlight how ML has shifted the field from traditional trial-and-error methods to rapid screening and generation of materials based on large datasets. The review covers commonly used ML methods and applications, and discusses the potential future directions for integrating ML with materials science research.

Nguyen et al. [10] introduce a critical-material commodity database (CMCD) with a web-based platform for easy access and updates. This tool aims to help material scientists understand market impacts and improve commercialization potential. Despite its success, challenges with query design and data quality remain, and plans are underway to enhance the platform's interface and data organization.

Yu, Li, and Sun [11] perform molecular dynamics simulations on five polymers to study how chain flexibility and side groups affect the glass transition temperature. They analyze specific volume changes with temperature and examine energy components and polymer chain conformations to understand the glass transition process.

Liu et al. [12] review the impact of machine learning (ML) on advancing energy materials, highlighting how ML addresses traditional challenges such as low success rates, high time consumption, and significant computational costs. The paper covers ML fundamentals, including algorithms and open-source databases, and discusses recent progress in materials like batteries, photovoltaic materials, and carbon capture technologies.

Gibbs and DiMarzio [13] use a quasi-lattice theory to predict a second-order transition in linear molecular chains, resembling the glass transition. They find this transition temperature is influenced by chain stiffness, length, and free volume, with predictions matching experimental results for glass temperature and other properties.

Vaswani et al. [14] propose the Transformer model, which uses only attention mechanisms, avoiding recurrence and convolutions. It outperforms previous models in machine translation tasks, achieving high BLEU scores and significantly reducing training time and costs. The model also shows versatility in other tasks like English constituency parsing.

Otsuka et al. [15] present PoLyInfo, a polymer database that compiles data from academic research. It includes polymer names, structures, processing methods, and properties, aiming to aid in material design. The paper details its architecture, unique features, and usage.

Cheung et al. [16] present POLYIE, a dataset for extracting information from polymer materials literature. Annotated with entities and their relations, POLYIE addresses challenges such as diverse formats and ambiguities. It aims to advance research in scientific information extraction for polymers.

Mannodi-Kanakkithodi et al. [17] develop a machine learning approach to accelerate the design of dielectric polymers. By leveraging data from first principles computations, they create predictive

models and use genetic algorithms to optimize polymer properties. This method can be applied to various materials.

TECHNOLOGY STACK:

i. Python
ii. Pandas, Numpy
iii. Node.js, Express.js
iv. MongodB
v. React
vi. Large Language Model : stabilityai/stablelm-2-1_6b
vii. Cloud Services

ARCHITECTURE DIAGRAM:

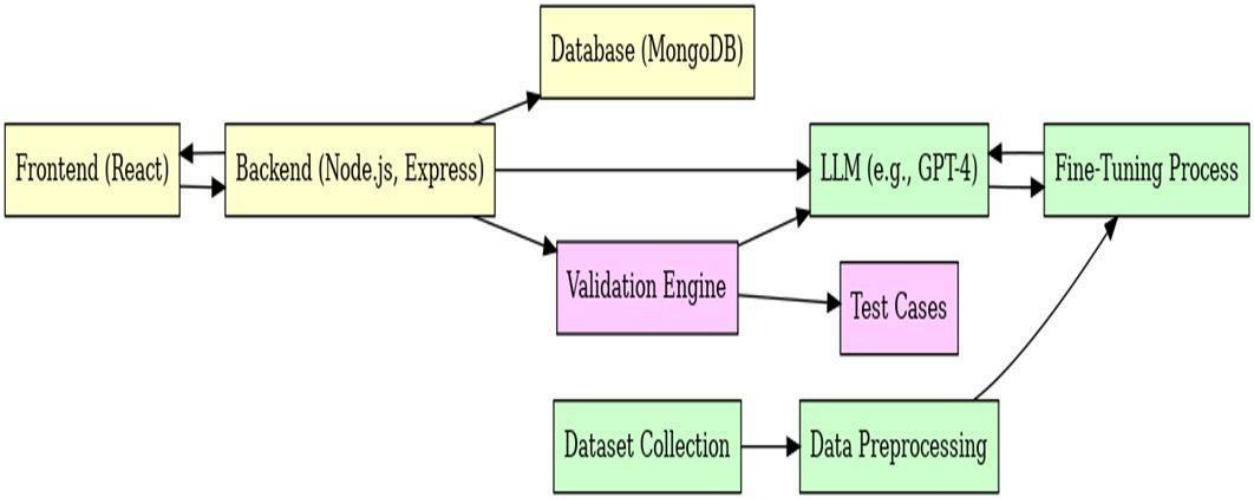

Figure 1

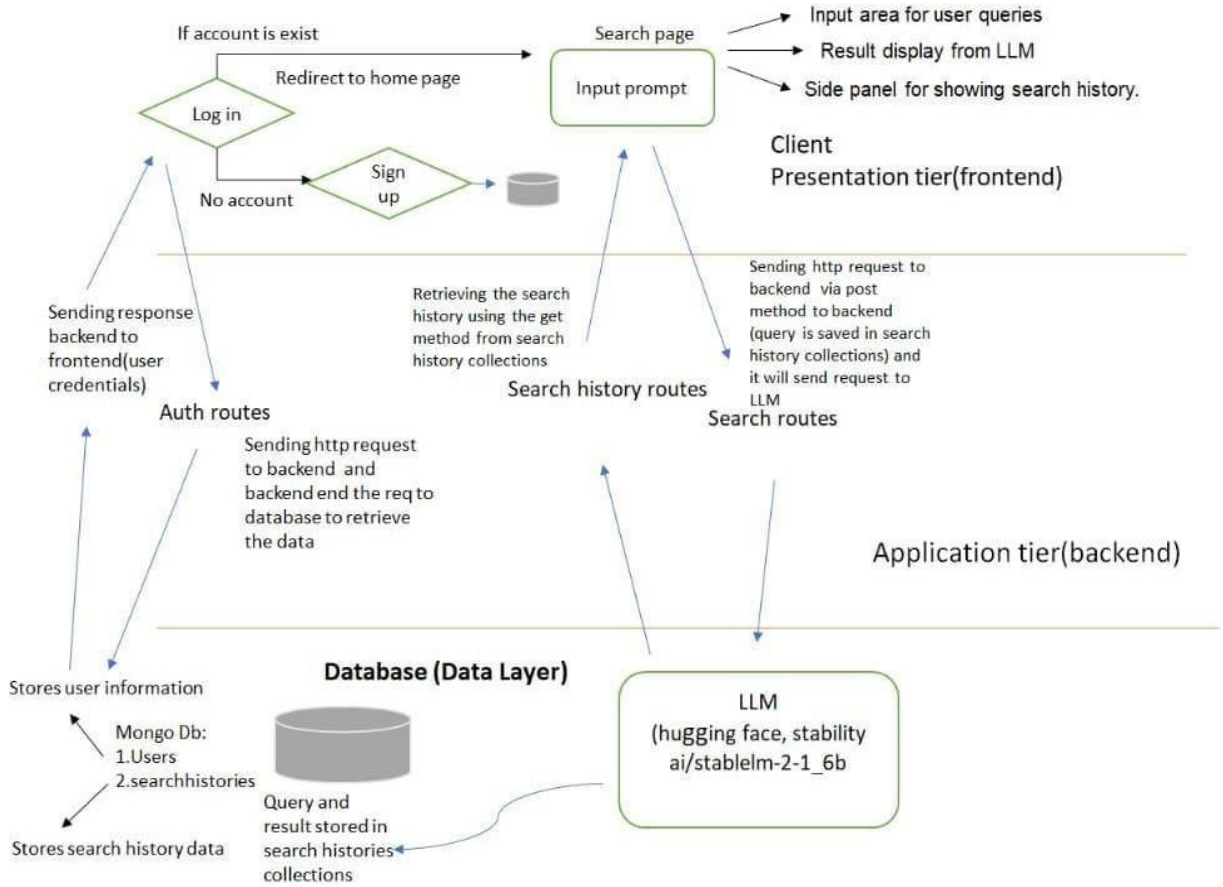

Figure 2 Architecture diagram

Frontend:

HTML: The standard markup language for creating web pages, providing the structure of the application.

CSS: A style sheet language used for describing the presentation of a document written in HTML, enabling responsive and visually appealing designs.

JavaScript: A programming language used to create dynamic and interactive user interfaces.

React: A JavaScript library for building user interfaces, providing a dynamic and responsive experience.

Backend:

Node.js: A JavaScript runtime for executing server-side code, allowing for the use of JavaScript throughout the entire stack.

Express.js: A web application framework for Node.js, designed for building APIs and web applications quickly and easily

Database:

MongoDB: A NoSQL database for storing and managing data in a flexible, JSON-like format.

Mongoose: An ODM (Object Data Modeling) library for MongoDB and Node.js, providing a straightforward way to model application data.

Machine Learning Model:

StableLM (Stable Diffusion from Stability AI): A pre-trained large language model fine-tuned for discovering and designing advanced, custom new materials.

Hugging Face Transformers: A library for natural language processing tasks, providing tools and pre-trained models for fine-tuning

CHAPTER-3

PROPOSED WORK AND IMPLEMENTATION:

Model Selection:

We picked the StabilityAI model, specifically the stabilityai/stablelm-2-1_6b which has been trained on 1.6 billion parameters, it's great at understanding and generating complex information. The stabilityai/stablelm-2-1_6b model stands out for our project due to its advanced language understanding and adaptability. Unlike many other models which are trained on less than 1.6 billion parameters which cannot grasp complex patterns and might result in incorrect responses, it's pre-trained on a wide range of data, which helps it grasp complex materials-related information effectively. Its design allows for detailed fine-tuning, making it highly customizable to specific tasks like generating innovative material ideas. Additionally, its ability to handle large datasets efficiently ensures that it performs well even with extensive materials data. This combination of robust language skills, flexibility, and efficiency makes it a superior choice for our needs in material discovery.

A pre-trained tokenizer, such as the one used in our project, plays a crucial role in processing text data before it is fed into an AI model. Specifically, we use a sub-word tokenizer, which breaks down text into smaller units (sub-words) rather than whole words. This approach helps handle rare or unknown words by decomposing them into more frequent sub-word components.There is a hugging face code that loads a pre-trained tokenizer and model from stabilityai/stablelm-2-1_6b. It tokenizes an input sentence, generates text based on the tokens using the model, and then decodes and prints the generated text.

Model Setup:

Dependencies: Ensure Python and pip are installed. You'll also need the transformers library from Hugging Face and torch for PyTorch. Install them using pip

Import Libraries: In your googlr colab, import the necessary libraries such as transformers, torch, peft, trl, datasets, pandas, and accelerate.

Load Model and Tokenizer: Use the AutoTokenizer and AutoModelForCausalLM classes to load the pre-trained StabilityAI model and tokenizer

Device Configuration: Move the model to a GPU for faster processing if available

Environment Setup:Ensure GPU Availability: Verify that a GPU is available for processing by checking torch.cuda.is_available() If not, the model will run on the CPU.

Memory Considerations: Ensure your environment has sufficient memory to handle the model and tokenization process, as large models can be memory-intensive.

Polymers Dataset:

For our project, we gathered data related to polymers from various sources, since we couldn't find a ready-made dataset that suited our needs. We manually collected information from scientific papers, industry reports, and online databases to compile a comprehensive dataset. This data primarily includes chemical properties and material structures. However, instead of including every possible property, we chose to focus on the glass transition temperature (Tg) as a key distinguishing feature. Tg is crucial for differentiating between various polymers, making it a valuable property for our fine-tuning process. By focusing on this specific property, we were able to create a more manageable and targeted dataset that aligns with our project's goals.

Glass Transition Temperature (Tg):

Glass Transition Temperature (Tg) is a critical thermal property of polymers that indicates the temperature range over which a polymer transitions from a rigid, glassy state to a more flexible, rubbery state. Below Tg, the polymer is hard and brittle, while above Tg, it becomes more pliable and elastic.

Tg is a key characteristic for distinguishing polymers because it directly impacts their physical properties and suitability for different applications. Different polymers have unique Tg values, which influence their behavior under various temperature conditions. For example, polymers with low Tg are more flexible at lower temperatures, while those with high Tg maintain rigidity even at elevated temperatures. By focusing on Tg, we can differentiate between polymers based on how they will perform in specific environments or applications, making it a valuable feature for material design and analysis.

DATA PREPROCESSING:

Data preprocessing is crucial for transforming raw data into a format suitable for analysis and machine learning models. It ensures that the data is clean, consistent, and structured, which significantly improves the quality and performance of models.

For your project, the data preprocessing involved several key steps:

Reading the Data: We loaded the text data from a file into a DataFrame to facilitate easier manipulation and cleaning.

Basic Cleaning:

Removed Extra Spaces: Extra spaces between words were eliminated to ensure uniform text formatting.

Standardized Dashes/Hyphens: Replaced inconsistent dash and hyphen characters to unify temperature range representation.

Fixed Spacing Before °C: Removed extra spaces before the °C unit for consistent temperature notation.

Custom Cleaning for Temperature Ranges:

Handled Temperature Ranges: Correctly formatted negative temperatures and ranges, ensuring accurate and consistent temperature data.

After preprocessing, the text data often needs to be converted into numerical values for machine learning models. This involves several additional steps:

Tokenization: Breaking down the text into smaller units (tokens) that the model can process.

Numerical Conversion: Converting these tokens into numerical representations using methods such as embeddings or indices.

One-Hot Encoding: This technique transforms categorical data into a format that can be provided to machine learning algorithms. Each category (or token) is represented by a binary vector where one element is '1' (indicating the presence of the category) and all other elements are '0'. This approach ensures that categorical features are effectively used in model training.

By following these preprocessing steps, the data becomes structured and ready for model fine-tuning, ensuring that the AI can accurately learn from and generate insights based on the given information.

Data Splitting:

Splitting the dataset into training and validation sets is crucial for assessing a model's performance and generalizability. The training set, comprising 90% of the data, is used to fine-tune the model, enabling it to learn patterns and relationships within the materials data. The validation set, constituting 10% of the data, serves as a proxy for unseen data, allowing for the evaluation of the model's predictive capability and tuning of hyperparameters. This 90-10 split balances providing ample data for learning while reserving a significant portion for validation, helping prevent overfitting and ensuring the model's applicability to real-world scenarios.

Training Strategy:

Fine-tuning is a process where a pre-trained model is further trained on a specific dataset to specialize it for a particular task. For instance, a language model trained on general text data can be fine-tuned using materials data to better predict and suggest new materials. This involves adjusting the model's parameters slightly to fit the new data while retaining the general knowledge it already has. Fine-tuning enhances the model's performance for the specific application without starting from scratch.

Basically the model we have used here is stabilityai/Stablem-2-1_6b

Lets start by briefing each parameters used in the fine tuning code:

num_train_epochs:

The num_train_epochs=1 argument sets the number of complete passes through the training dataset to just one. This means the model will see and learn from each training example only once. While one epoch can be sufficient for some tasks or for quick experimentation, it may not be enough for more complex problems where the model requires multiple exposures to the data to learn effectively. Generally, for effective learning, the number of epochs should range from 2 to 5. However, due to memory constraints, we confined ourselves to one epoch

Optimizer:

Optimizers are algorithms used to adjust the weights of a neural network during training to minimize the loss function. The underlying mechanism involves updating the model parameters (weights and

biases) based on the gradients computed during backpropagation. The goal is to find the set of parameters that results in the lowest possible loss.

General Mechanism:

Forward Pass: Compute the output of the network and the loss by comparing the predicted output to the actual target values.

Backward Pass: Compute the gradients of the loss with respect to each parameter in the network using backpropagation.

Parameter Update: Use the gradients to update the model parameters. This is done by subtracting a fraction of the gradient (scaled by the learning rate) from the current parameter values.

The different types of optimizers can be used are:

Gradient Descent (GD): An optimizer that updates parameters by moving in the direction of the negative gradient of the loss function based on the entire dataset.

Stochastic Gradient Descent (SGD): A variant of GD that updates parameters using a random subset (mini-batch) of the data to speed up training and handle large datasets.

Momentum: An enhancement to SGD that accumulates past gradients to accelerate convergence and smooth out the updates.

Among these AdamW is chosen because of different features like:

AdamW adapts learning rates for each parameter individually, improving efficiency.It includes weight decay, which directly penalizes large weights, helping to avoid overfitting. Compared to Gradient Descent and SGD, AdamW converges faster and handles large datasets better. Momentum improves speed but doesn't adapt learning rates or include weight decay. Adam is effective but applies weight decay indirectly through the loss function. AdamW combines the benefits of adaptive learning rates and direct regularization, offering more stability and better performance. This makes it ideal for complex models, ensuring they train effectively and generalize well.

per_device_train_batch_size:

In our project, we decided to set the per_device_train_batch_size to 1. This means that each GPU or TPU processes one training example at a time. We made this choice because our large language model, stabilityai/stablelm-2-1_6b, has 1.6 billion parameters and needs a lot of memory. Processing one example per batch helps us avoid running out of memory and allows the training to run smoothly with the hardware we have. Additionally, updating the model's parameters more frequently, even though it might take longer overall, can help the model learn better in the early stages.

Using a batch size of 1 also makes the training process more stable by reducing random fluctuations in the parameter updates. Although training with a small batch size usually takes longer, the stable learning and better memory usage make up for it. We can also use gradient accumulation to mimic larger batch sizes, making the training process more efficient without exceeding memory limits. This approach ensures that we can effectively train our large model within the constraints of our hardware, making it a good fit for our project needs.

Save_steps:

The save_steps=20,000 setting specifies that the model checkpoints will be saved every 20,000 training steps. This means that after every 20,000 steps of training, the current state of the model will be saved to disk. This helps in tracking progress and allows resuming training from the last checkpoint if needed.

fp16:

In our project, we set the fp16 parameter to True, which means we are using 16-bit floating-point precision instead of the standard 32-bit. This reduces memory usage by half because 16-bit numbers take up less space, allowing us to fit more data into the GPU memory and making the training process faster and more efficient. This is particularly important for our large model, stabilityai/stablelm-2-1_6b, which has 1.6 billion parameters and demands a lot of memory. Using FP16 also speeds up data processing, which helps the model train faster. While there may be a small loss in precision with 16-bit numbers, it usually doesn’t significantly affect the model’s performance, especially with modern techniques like mixed precision training that mitigate accuracy issues. Overall, FP16 is a valuable choice for training large models because it helps us optimize memory usage and improve training efficiency within our hardware limits.

learning_rate:

The learning_rate=2e-4 setting refers to the learning rate used during training, which is 0.0002 in scientific notation. This value controls how much the model's weights are adjusted in response to the error gradient during each update. A learning rate of 2e-4 is considered relatively small, which can be beneficial for fine-tuning, as it allows the model to make more gradual and precise updates to its weights. Using a smaller learning rate can help prevent overshooting the optimal solution and improve convergence. However, it may also increase the training time since each update is smaller. It's important to monitor the training process and adjust the learning rate as needed to ensure effective learning.

lr_scheduler_type:

In our project, we set the lr_scheduler_type to "cosine," which means we are using a cosine learning rate scheduler. This type of scheduler adjusts the learning rate based on a cosine function. It starts with a higher learning rate and then gradually reduces it in a smooth, cosine-shaped curve as training progresses.

Using the cosine learning rate scheduler has several benefits. At the beginning of training, a higher learning rate allows the model to learn quickly and explore different areas of the solution space. As training continues, the learning rate decreases gradually, helping the model fine-tune its parameters and improve accuracy. This smooth reduction can lead to better convergence and reduce the risk of overshooting the optimal solution.

Other common types of learning rate schedulers include:

- **Step Scheduler:** Reduces the learning rate by a fixed amount at predefined steps. This approach helps in making abrupt adjustments at specific training milestones.
- **Exponential Scheduler:** Decreases the learning rate exponentially over time, which can be useful for gradually reducing the learning rate.
- **Linear Scheduler:** Decreases the learning rate linearly over the course of training, providing a steady reduction.
- **Warmup Scheduler:** Starts with a low learning rate and increases it gradually to a target value before applying other schedules like cosine or step. This helps in stabilizing early training.

Overall, the cosine scheduler helps balance rapid learning with careful fine-tuning, making the training process more effective and efficient for our model.

save_strategy:

In our project, we set the save_strategy to "epoch." This means that the model will be saved at the end of each training epoch, which is a full pass through the entire training dataset. By doing this, we ensure that we have a backup of the model's state after every complete cycle of training.

Choosing save_strategy="epoch" is beneficial for several reasons. It provides us with regular checkpoints, so if we need to pause or if something goes wrong, we can resume from the last saved point. It also helps us track the model's performance at different stages of training, making it easier to evaluate how well the model is improving over time.

Other common save strategies include:

- **"steps":** This option saves the model at regular intervals based on the number of training steps, rather than waiting for a whole epoch to finish. This can be useful if we want more frequent backups without having to wait for a complete pass through the data.
- **"no":** This disables automatic saving, meaning we would need to save the model manually if needed. This might be used if we have a different saving strategy or if saving frequently isn't necessary.

Overall, setting save_strategy="epoch" helps us maintain consistent backups of the model, which is useful for managing and tracking its progress throughout the training process.

gradient_accumulation_steps:

In our project, we set the gradient_accumulation_steps parameter to 4. This means that instead of updating the model's weights after processing each small batch of data, we accumulate the gradients from 4 mini-batches before performing any updates. Here's a more detailed breakdown of how it works:

Mini-Batches: Training data is divided into smaller parts known as mini-batches. The model processes these mini-batches one at a time.

Accumulate Gradients: For each mini-batch, the model calculates gradients, which are adjustments needed for updating the model's weights. Instead of updating the weights

immediately after each mini-batch, we keep a running total of these gradients from 4 mini-batches.

Update After 4 Mini-Batches: Once we have processed 4 mini-batches and accumulated their gradients, we then update the model's weights using these combined gradients all at once.

Gradient accumulation is required mainly to ensure:

Memory Efficiency: Large models require a lot of memory. By using smaller mini-batches and accumulating gradients, we can fit the model and data into memory more efficiently. This approach allows us to simulate the effect of a larger batch size without needing to use a lot of memory at once.

Training Stability: By gathering gradients from several mini-batches before updating the model, we keep the training process more stable and effective. This approach helps manage memory better and gives us the benefits of larger batch sizes, like smoother updates and better model performance.

In summary, with gradient_accumulation_steps=4,we enhance our memory management and training stability by updating the model's weights less frequently but with more accumulated data.

Techniques for Efficient Model Adaptation:

PEFT (Parameter-Efficient Fine-Tuning) is a method designed to adapt large pre-trained models to new tasks while making minimal changes to the model's parameters. This approach avoids the need to retrain the entire model from scratch, which can be very resource-intensive and time-consuming. Instead, PEFT focuses on adjusting only a small subset of the model's parameters, making the fine-tuning process faster and more memory-efficient. This is particularly beneficial when working with large-scale models, as it helps to reduce the computational load and accelerates the adaptation process.

LoRA (Low-Rank Adaptation) is a specific technique within the PEFT framework that enhances the efficiency of model adaptation. LoRA works by adding low-rank matrices to the model's existing weight matrices. These low-rank matrices are relatively small and help the model learn new tasks or

features without altering most of the original model's parameters. This approach allows for a more efficient adaptation process, as it requires fewer additional parameters and consumes less memory.

LoRA is especially effective for tasks like text generation, where the model predicts the next word in a sentence based on previous words. By using LoRA, we can adapt the model quickly and efficiently without the need for extensive retraining.

For fine-tuning our model using LoRA, we used the following settings:

r=8: This parameter sets the rank of the low-rank matrices to 8. The rank determines how detailed the additional matrices are. A rank of 8 is a balanced choice that helps the model learn new tasks efficiently without being too large.

lora_alpha=16: This scaling factor controls how strongly the low-rank matrices influence the model's weights. A value of 16 means these matrices have a significant effect on the model's learning, helping it adapt more effectively to new tasks.

lora_dropout=0.1: This setting introduces a dropout rate of 10% to the low-rank matrices. Dropout helps prevent overfitting by randomly ignoring some parts of the matrices during training. This keeps the model from becoming too specialized to the training data.

bias="none": This specifies that no extra bias terms are used in the LoRA adaptation. This simplifies the model and focuses on adjusting the main parameters.

task_type="CAUSAL_LM": This indicates that the model is being adapted for causal language modeling. This type of task involves predicting the next word in a sentence based on the previous words, making this setting suitable for text generation and similar application.

MERN Stack Setup:

The MERN stack is a set of tools for building modern web applications. It includes:

- **MongoDB**: This is a NoSQL database that stores data in a flexible, JSON-like format. Instead of traditional tables, MongoDB uses collections and documents, which makes it easy to manage and query large amounts of data. Think of it as a smart, organized digital locker where your application's data is kept safe and can be accessed quickly.
- **Express.js**: This is a web application framework for Node.js. It simplifies the process of building the backend of your application by providing a range of built-in features and tools. With Express, you can easily set up routes, handle requests and responses, and connect to your database. It acts like a helpful assistant that manages communication between the user and the server.
- **React**: This is a JavaScript library used for creating the front end of your application. React allows you to build interactive user interfaces by breaking them down into reusable components. This means you can create complex web pages that update dynamically without having to reload the entire page. It's like building blocks that let you design engaging and responsive web experiences.
- **Node.js**: This is a runtime environment that lets you run JavaScript code on the server side. Node.js is known for its speed and efficiency, allowing you to handle multiple tasks simultaneously. It's like the engine that powers your backend, managing server operations and running your application logic smoothly.

User Interaction and Features:

Upon launching the application, users will first encounter a login or signup page. This page allows them to create a new account or access an existing one. After successfully logging in, users are directed to the home page.

On the home page, users will find detailed information about our project, which focuses on polymer materials and includes a large language model (LLM) for providing information and insights. The home page provides an overview of the project's objectives and capabilities, setting the stage for user interactions with the application.

The main feature of the application is the search page. Here, users can enter queries related to polymer materials into a search field. After submitting their query, the LLM processes it and displays relevant

responses on the same page. This allows users to quickly get information about different polymers and their properties.

Features:

- ✓ **Login/Signup Page**: The entry point of the application where users can log in to their existing accounts or sign up for new ones. This page ensures secure access to the application and personal data.
- ✓ **Home Page**: After logging in, users are greeted with the home page. It provides an overview of the project, explaining the focus on polymer materials and the integration of the LLM. This page helps users understand the purpose of the application and what to expect.
- ✓ **Search Page**: The central feature of the application where users can input their queries about polymers. The search page includes:

  Input Field: A place for users to type their questions about polymer materials.

  Response Display: After a query is submitted, the LLM generates and displays relevant information directly on the search page, helping users find the information they need efficiently.

Communication Flow for Fine-Tuning an LLM for Material Discovery

Introduction

The project aims to fine-tune a Large Language Model (LLM), specifically StabilityAI's model, to discover and design advanced materials. This involves using a React-based user interface (UI) and a Node.js backend server to communicate with the fine-tuned LLM. Understanding the communication flow between these components is crucial for the project's success.

1. Frontend to Backend Communication

React-Based UI to Node.js Server:

User Interaction and Data Input:

- **Interface Design:** The frontend of your project is built using React, which provides a dynamic and responsive user interface. Users can interact with various components like forms, input fields, and buttons to submit data related to material discovery.
- **Event Handling:** When a user performs an action, such as filling out a form or clicking a button, React triggers events. These events are handled by React components to prepare the data for transmission.

HTTP Requests:

- **Data Submission:** React communicates with the Node.js server by sending HTTP requests. This is done using libraries such as Axios or the native Fetch API. For example, if a user submits a form to test a new material, React sends a POST request to the server.
- **Request Methods and Protocols:**
  - **POST Requests:** Used to send data from the UI to the server. This could include information about the material query or new material ideas. The data is often formatted as JSON, a lightweight data interchange format.
  - **GET Requests:** Used to retrieve information from the server. For example, after submitting a query, the React app might use a GET request to fetch the results of the model's predictions.
- **Request Handling:** The HTTP requests include details like headers, body (in case of POST), and query parameters (for GET). The server endpoint (e.g., /api/test-material) receives these requests and processes the data.

2. Backend to Model Communication

Node.js Server to StabilityAI Model:

- **Server Data Processing and Communication:**
  - **Request Handling:** The Node.js server receives HTTP requests from the React frontend. It processes these requests by extracting and parsing the data. For POST requests, the server parses the JSON payload to understand the query or material data.
  - **Data Preparation:** The server prepares the data for interaction with the StabilityAI model. This may involve formatting the data according to the model's requirements,

such as converting text into a specific format or structuring the data in a particular way.

- **Model Interaction:**
  - **API Communication:** The server sends data to the StabilityAI model's API endpoint. This interaction often involves making HTTP requests to the model's service, which could be a REST API or another type of service. Authentication (such as API keys) might be required.
  - **Sending Data:** The server transmits the formatted data to the model and waits for a response. This involves constructing an appropriate HTTP request to the model's API, including necessary headers and request body.
- **Receiving and Processing Responses:**
  - **Model Response:** The StabilityAI model processes the data and returns a response, which typically includes predictions or material suggestions. This response is received by the Node.js server.
  - **Response Handling:** The server parses the model's response and formats it for the frontend. The response data is often in JSON format, which the server converts into a structure suitable for display in the React UI.
- **Sending Results to Frontend:** The server sends the processed results back to the React-based UI. This is done using HTTP responses, which include the data formatted for easy rendering in the frontend. For instance, if the model provides material suggestions, these are sent back to the UI for the user to view.

CHAPTER-4

RESULTS:

Key observations from this project are as follows:

- We have generated a set of new polymers with better mechanical properties such as thermal stability, chemical resistance etc. comparatively to current polymers used in industrial applications.
- These polymers have been identified with the help of the LLM being trained by various polymer property datasets and fine – tuned to suit the specific needs of the project, such as polymer property prediction and recommendation of new, better polymers.
- The new polymers are generated with a key aspect in mind: sustainability. The focus of the project is to generate advanced materials based on their sustainability, and this project has largely implemented this.
- The model has been successful in predicting the glass transition temperatures (Tg) of many diverse polymers and has used these existing parameters to predict the temperatures of the new polymers as well.
- By using the required Tg values, new materials with improved thermal stability and improved characteristics have been discovered with the help of the LLM model.
- We have also analyzed and researched the relationship between polymer composition and its glass transition temperature, which is very useful for understanding how these temperatures affect polymer properties.
- We use Tg to tailor some specific properties such as solubility, dissolution, elasticity etc. for certain applications, such as applications involving the chemical and plastics industry.
- The performance of the new polymers in certain temperature ranges has been analyzed.
- For high – temperature applications, we have generated some polymers which conventional polymers might fail at.
- Similarly, for low - temperature applications also, we have generated polymers based on their Tg values, and these polymers have a better ability to function in these applications, in which regular polymers may fail.
- Overall, by proper optimization of the given data to the model and training it, we can obtain diverse properties of current polymers and use them for new polymer generation, which will

be used in various applications in real life.

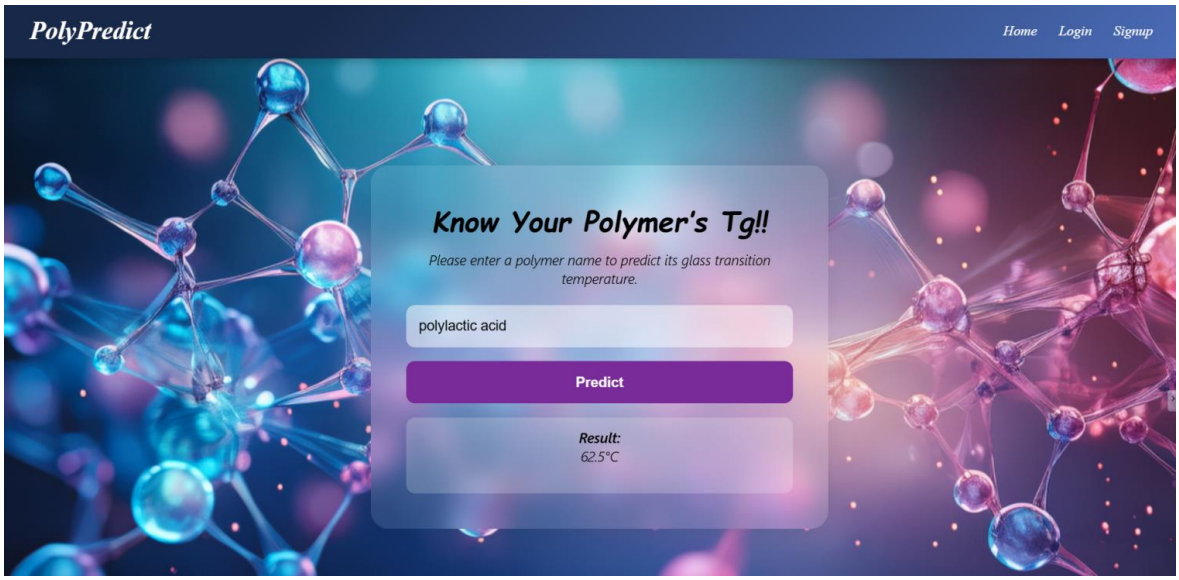

Figure 3 Result Page

CHAPTER-5

CONCLUSION:

In this project, by the usage of fine-tuned language models, we have been able to predict and generate new polymer structures with useful properties, with a primary focus on sustainability. The projectfacilitates a path for the proposal of new polymer structures, optimizing their properties, and using them in support for the development of sustainable materials. It has also streamlined polymer design process, thereby reducing the need for physical experimentation of polymer discovery, and thus saving a lot of time and resources in the process.

With the success of this project, we can also derive the point that machine learning, if used in a productive way, can be a huge boon to the manufacturing and chemical industries by the construction and manufacturing of appliances and materials with these new polymers, and thus improving their performance. New research can also be conducted on the new materials and their properties, and their results can be implemented for various purposes contributing to industrial development.

In conclusion, this project is a small step in attempting to implement sustainability in industries in order to protect the environment and provide new solutions for product enhancement.

REFERENCES:

1. Prediction of the glass transition temperature of polymers H.G. Weyland, P.J. Hoftyzer, D.W. Van Krevelen
2. Machine learning discovery of high-temperature polymers Lei Tao,1,3 Guang Chen,1,3 and Ying Li1,2,4, * 1Department
3. Hergenrother, P.M. (2003). The use, design, synthesis, and properties of high performance/high temperature polymers: an overview. High Perform. Polym. 15, 3–45.
4. Batra, R., Song, L., and Ramprasad, R. (2020). Emerging materials intelligence ecosystems propelled by machine learning. Nat. Rev. Mater. 1–24.
5. Choi, J., Yu, S., Yang, S., and Cho, M. (2011). The glass transition and thermoelastic behavior of epoxy-based nanocomposites: a molecular dynamics study. Polymer 52, 5197–5203.
6. Palomba, D., Vazquez, G.E., and Dı´az, M.F. (2012). Novel descriptors from main and side chains of high-molecular-weight polymers applied to prediction of glass transition temperatures. J. Mol. Graph. Model. 38, 137–147.
7. Kim, C., Chandrasekaran, A., Jha, A., and Ramprasad, R. (2019). Activelearning and materials design: the example of high glass transition temperature polymers. MRS Commun. 9, 860–866.
8. McKenna, G.B. (2020). Looking at the glass transition: challenges of extreme time scales and other interesting problems. Rubber Chem. Technol. 93, 79–120.
9. Advances of machine learning in materials science: Ideas and techniques Sue Sin Chong1,* , Yi Sheng Ng1,*, Hui-Qiong Wang1,2,† , Jin-Cheng Zheng1,2,‡
10. A market-oriented database design for critical material research Ruby T. Nguyen1,*, Ange-Lionel Toba1 , Michael H. Severson1 , Ethan M. Woodbury2 , Austin R. Carey3 and D. Devin Imholte1
11. Yu, K.Q., Li, Z.S., and Sun, J. (2001). Polymer structures and glass transition: a molecular dynamics simulation study. Macromol. Theory Simul. 10, 624–633.
12. Machine learning for advanced energy materials Yun Liu, Oladapo Christopher Esan, Zhefei Pan, Liang An∗

13. Gibbs, J.H., and DiMarzio, E.A. (1958). Nature of the glass transition and the glassy state. J. Chem. Phys. 28, 373-383.

14. Attention Is All You Need Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

15. PoLyInfo: Polymer Database for polymeric materials design Shingo Otsuka∗, Isao Kuwajima†, Junko Hosoya†, Yibin Xu† and Masayoshi Yamazaki† ∗Department of Information and Computer Sciences, Kanagawa Institute of Technology, Kanagawa, Japan

16. POLYIE: A Dataset of Information Extraction from Polymer Material Scientific Literature Jerry Junyang Cheung1∗ , Yuchen Zhuang1∗ , Yinghao Li1 , Pranav Shetty2 , Wantian Zhao1 , Sanjeev Grampurohit1 ,Rampi Ramprasad2 , Chao Zhang1

17. Machine Learning Strategy for Accelerated Design of Polymer Dielectrics Arun Mannodi-Kanakkithodi1, Ghanshyam Pilania2, Tran Doan Huan1, Turab Lookman3 & Rampi Ramprasad