Generative AI overview and architecture

Significance

- Generative AI refers to deep learning models that can generate various types of content such as text, images, audio, 3D objects and music based on the data they were trained on.
- These models understand the relationship between words and phrases and generate contextually relevant text. eg. (next narrative of the story, language translation) . eg., GPT(Generative Pre-trained Transformer).
- These models can generate images from text input and seed images like DALL-E ,GAN and diffusion model.
- You can use these models to generate natural sounding speech and text to speech synthesis. eg., WaveNet.
- Generative AI applications include content creation(creation of articles, blog posts, marketing materials, create visuals and videos for entertainment and advertising),condense long documents and articles, translation, chatbots & VA, analyse data and suggest creative solutions for complex problems.
- Future applications - enhancing personalised recommendations, drug discovery, smart homes ,autonomous vehicles.
- Generative AI has specific applications in industries such as healthcare, finance, gaming, and IT.

Generative AI Architectures and Models

Types of Generative AI models:

- **Recurrent Neural Networks (RNNs):** RNNs use sequential(order matters) or time series data and a loop based design (that remembers previous inputs that influence current I/O & O/P)for training.To fine-tune RNNs, adjustments to its weights and structure might be necessary to align it with specific tasks or datasets. Used in NLP, translation, speech recognition, and image captioning.
- **Transformers:** Transformers utilize the self-attention mechanism to focus on the most important parts of the information. Employs feedback mechanisms to improve accuracy.The selective focus on different parts of the input sequence allows the model to concentrate on specific segments concurrently, enabling parallelization for efficient training. Applications in human like text responses - Chatbots, Virtual assistants etc.
- **Generative Adversarial Networks(GANs):** GANs consist of a generator(creates fake samples and sends to discriminator) and a discriminator(checks their authenticity by comparing them with real samples and provides a probabilty score for authenticity). This adversarial process continues like a friendly competition, with the generator striving to make things look real and the discriminator learning to distinguish between real and fake, both entities improving their respective outputs.used for image and video generation(deepfakes that are close to real life)
- **Variational AutoEncoders(VAEs):** VAEs operate on an encoder decoder framework and create samples based on similar characteristics. Encoder condense the input data and sends to decoder which tries to recreate the original data thereby can produce a range of possible outputs for a given input. Applications in art and creative design , generating new fashion designs based on existing style.
- **Diffusion models:** Diffusion models generate creative images by learning to remove noise and reconstruct distorted examples, relying on statistical properties of its training data. Applications in restoring old distorted images

Generative AI for NLP

- Generative AI architectures enable machines to comprehend human language and generate responses you cannot distinguish from what a human generates.
- Generative AI Evolution - started with rule based systems(follow predefined linguistic rules) —> machine learning(focusing on statistical methods)—> deep learning(uses neural networks trained on extensive datasets) — > Transformers
- These enhanced capabilities has led to advancements in machine translation (precise and context aware conversions), chatbot conversation (more natural and humanlike with a degree of empathy and personalization), sentiment analysis(grasp subtle language expressions offering deeper insights) , and text summarization (recognition of the core meaning and significance of text).
- LLMs are foundation models that use AI in deep learning with vast data set of size running into petabytes with have billions of parameters that can be used for fine tuning.
- examples of LLMs are
    - GPT(Generative Pre-trained Transformer)-acts as decoder - chatbots
    - BERT(Bidirectional encoder representation from transformers) - only encoder- understands context of word in sentence - sentiment analysis and Q/A
    - BART (Bidirectional and autoregressive transformers) & T5(text to text transfer transformer) - encoder(for contextual understanding), decoder(generate text)- versatile tasks
- GPT vs ChatGpt
    - text generation vs conversations
    - Supervised learning vs supervised & reinforcement learning
    - No human feedback vs RLHF (reinforcement learning from human feedback)
- You can pre-train LLMs for generic purposes, and then fine tune with a much smaller data set.

Basics of AI Hallucinations

- AI hallucinations refer to an AI model generating output presented as accurate but seen as unrealistic, inaccurate, irrelevant, or nonsensical by humans.
- You can prevent the problems caused by AI hallucinations through:
    - Eliminating any bias in the training data and perform extensive training with high-quality data
    - Avoiding manipulation of training data
    - Ongoing evaluation and improvement of the models
    - Fine-tuning on domain-specific data
    - Being vigilant.These models are trained on vast amounts of data and learn statistical patterns, but they lack semantic understanding or comprehension **like human beings.**
    - Ensuring human oversight
    - Providing additional context in the prompt

Overview of Libraries and Tools

- Libraries and tools that you can use to develop NLP applications using generative AI - PyTorch, TensorFlow, Hugging Face, LangChain, and Pydantic.
- PyTorch is an open source deep learning framework. It is a Python-based library well-known for its ease of use, flexibility, and dynamic computation graphs.
- TensorFlow is an open-source framework for machine learning and deep learning. It provides tools and libraries to facilitate the development and deployment of machine learning models.

- The tight integration of TensorFlow with Keras provides a user-friendly high-level neural networks API, facilitating rapid prototyping and building and training deep learning models.
- Hugging Face is a platform that offers an open-source library with pretrained models and tools to streamline the process of training and fine-tuning generative AI models. It offers libraries such as Transformers, Datasets, and Tokenizers.
- LangChain is an open-source framework that helps streamline AI application development using LLMs. It provides tools for designing effective prompts
- Pydantic is a Python library that helps you streamline data handling. It ensures the accuracy of data types and formats before an application processes them.

Data preparation for LLMs

Tokenizer

- The process of breaking a sentence into smaller pieces, or tokens, is called tokenization. The tokens help the model understand the text better. The program that breaks down text into individual tokens is called a tokenizer.
- **Tokenization methods:**
    - **word based**: The text is divided into individual words, each word considered a token. **Advantages** - preserves the semantic meaning. **Disadvantages -** treating each word as a token significantly increases the model's overall vocabulary(disadvantage because of reasons like it uses more memory , complexity ,captures noise etc)
    - **character based:** It involves splitting text into individual characters. **Advantages** - the vocabularies are small(only includes number of letters and characters in the language as opposed to number words in the previous approach).**Disadvantages -** no semantic meaning , increased input dimensionality(number of tokens for a given sentence) and computational needs.
    - **subword based:** frequently used words can remain unsplit while infrequent words are broken down into meaningful sub words.It combines the advantages of word based and character based tokenization. various algorithms for implementing subword based tokenization.
        - **wordpiece:** It evaluates the benefits and drawbacks of splitting and merging two symbols(basic units can be characters, word parts, words etc) to ensure its decisions are valuable. eg.,Bert tokenizer from transformers library splits the word “tokenization” as [token,##ization] where ## indicates the word should be attached to the previous word without a space.
        - **unigram:** It breaks text into smaller pieces. It begins with a large list of possibilities and iteratively narrows down based on how frequently they appear in the text.
        - **sentencepiece:** It segments text into manageable parts and assigns unique IDs.
- XLNetTokenizer from transformers library uses unigram and sentencepiece.”IBM taught me tokenization” split as [_ibm,_taught,_me,_token,ization], where tokens with preceding _ means it is appearing for the first time and with a space at the start.
- pytorch uses torchtext library for tokenization. **build_vocab_from_iterator** function creates a vocabulary from these tokens that the model can understand.It assigns each token in the vocabulary a unique index represented by an integer. The model then uses these indices to map words in the vocabulary.
- **NLTK & Spacy** python libraries for NLP has tokenizers that consider for example words like unicorn and unicorns as different leading to issues during NLP tasks. In Spacy You can add special tokens, such as BOS at the beginning and EOS at the end of a tokenized sentence or <pad> to add padding.

Tokenizer libraries

- **nltk** natural language toolkit, will be employed for data management tasks. It offers comprehensive tools and resources for processing natural language text, making it a valuable choice for tasks such as text preprocessing and analysis.
- **spaCy** is an open-source software library for advanced natural language processing in Python. spaCy is renowned for its speed and accuracy in processing large volumes of text data.
- **BertTokenizer** is part of the Hugging Face Transformers library, a popular library for working with state-of-the-art pre-trained language models. BertTokenizer is specifically designed for tokenizing text according to the BERT model's specifications.
- **XLNetTokenizer** is another component of the Hugging Face Transformers library. It is tailored for tokenizing text in alignment with the XLNet model's requirements.
- **torchtext** It is part of the PyTorch ecosystem, to handle various natural language processing tasks. It simplifies the process of working with text data and provides functionalities for data preprocessing, tokenization, vocabulary management, and batching.

Data Loaders

- **Purpose of Data Loaders**: It's challenging to manually load and shuffle huge datasets to train the underlying language model. To manage this process efficiently, a data loader helps in preparing and loading data. Data loaders can output data in batches and shuffle the data.
- **Integration with PyTorch**: PyTorch has a dedicated data loader class that you can use to **enable batching and shuffling**, which is essential for training neural networks effectively.It allows **on-the-fly pre-processing** for memory optimization by loading only the required data during training. Data loaders seamlessly integrate with the **PyTorch training pipeline**, making it easier to train and evaluate models. They simplify **data augmentation and pre-processing**, allowing you to apply various transformations to the input data
- **Dataset**: A dataset consists of samples and labels, typically divided into training(to train the model), validation(to tweak and validate the model), and test sets(to assess the model's performance in real-world scenarios)
- **Custom Dataset Creation**: You can create a custom dataset by inheriting from torch.utils.data.Dataset, implementing methods like __init__, __len__, and __getitem__.
- **Batching and Shuffling**: Data loaders can output data in batches and shuffle the data to prevent learning patterns based on order. Done using torch.utils.data.DataLoader
- **Transformations in pytorch**: Data loaders facilitate various transformations, such as tokenization(**tokenizer**), numericalizing(**build_vocab_from_iterator)**, padding(**pad_sequence**), and converting data into tensors.
- **pad_sequence(batch,batch_first=True,padding_value=0):** batch_first = true means 1st dimension in output tensor is **batch size** and 2nd dimension is **sequence size . S**o tensor will be 2d array (B*S). In false case, vice versa 2d array (S*B)
- To keep the original dataset untouched, you can take care of data transformation in the custom collate function. A collate function is employed in the context of data loading and batching in machine learning, particularly when dealing with variable-length data, such as sequences (e.g., text, time series, and sequences of events). Its primary purpose is to prepare and format individual data samples (examples) into batches that machine learning models can efficiently process

Data quality and diversity for effective LLM training

- **Data Quality**:
    - Refers to the **accuracy, consistency, and completeness** of the dataset.
    - Practices to ensure high quality include:
        - **Noise Reduction**: Removing irrelevant data like typos to enhance focus on significant patterns.
        - **Consistency Checks**: Regularly verifying data to prevent conflicting /outdated information.eg names / technical terms consistently used
        - **Labeling Quality**: Ensuring accurate labeling to avoid misleading the model.
- **Diverse Representation**:
    - A diverse dataset enhances inclusivity and reduces biases.
    - Achieving diversity involves:
        - **Inclusion of Varied Demographics**: Incorporating text from different demographic groups to improve global applicability.
        - **Balanced Data Sources**: Using a mix of sources like news, social media, and literature.
        - **Regional and Linguistic Variety**: Including datasets from various regions and languages to enhance accuracy in multilingual contexts and better supporting translation tasks.
- **Regular Updates**: Updates help capture new vocabulary and adapt to changing cultural norms which is essential for relevance and accuracy.
- **Ethical Considerations**:
    - **Data privacy**: Use anonymized data to protect personal information, especially in datasets containing sensitive or identifiable information.
    - **Fair representation**: Ensure the inclusion of marginalized voices to avoid bias that can reinforce societal inequalities.
    - **Transparency in data sources**: Disclose data sources used for model training. This transparency fosters user trust and allows for understanding the foundation of the model's knowledge.