Lecture No. 1

■ Summary

- The tutorial is focused on understanding how LLM (Large Language Model) works, specifica lly GPT (Google Gemini Cloud).

- It explains how GPT generates content based on user input and pre-training data.

- It emphasizes that the generative ability of GPT is based on various types of pre-traini ng data like internet data, books, conversations, historical data, etc.

- The tutorial concludes by comparing GPT to standard search engines and explaining its ge nerative nature.

■ Key Terms and Concepts

- LLM (Large Language Model)

- GPT (Google Gemini Cloud)

- Generative word

- Indexing of data

- Pre-training in the context of AI models

- Generative nature of AI language models

- Basis for generating next sequence

- Training data sources (internet, books, conversations, historical data)

■ Review Questions

1. What is the generative word and why is it important in the context of AI language model s?

2. How does GPT's generative nature compare to traditional search engines like Google and Bing?

3. What does pre-training data refer to in the context of AI language models?

4. Explain how the generative nature of GPT enables it to respond to user input.

5. Why is the basis for generating the next sequence important in understanding the functi

oning of an AI language model?

■ Summary

- The video discusses the Transformer architecture and its applications in deep learning t asks.

- The Transformer architecture can generate an output based on given input data.

- The speaker highlights a specific example of using the Transformer architecture for tran slations.

- The video emphasizes the transformer's ability to predict or transform text and image da ta.

■ Key Terms and Concepts

- Transformer

- Generative nature

- Neural network

- GPT and LLM transformer models

- Input sequences and output predictions

- Transformer in Google Translate

■ Review Questions

1. What is the main function of the Transformer architecture?

2. Explain the concept of the generative nature of the Transformer model.

3. Can you provide an example of how the Transformer can be used for language translation?

4. How is the concept of input sequences and output predictions related to the Transformer architecture?

5. What are the key applications of the Transformer architecture in deep learning?

■ Summary

- The video demonstrates how a transformer works in predicting the next word in a sequence

.

- The process involves appending the input, passing it through the transformer, and gettin g the next predicted word.

- Encoding and tokenization are crucial steps in the process.

■ Key Terms and Concepts

- Transformer

- Input sequence

- Next word prediction

- Tokenization

- Encoding

- Number conversion

- Master data

- Tokenize

- Embeddings

■ Review Questions

1. What are the key steps involved in the transformer prediction process?

2. Can you explain the concept of tokenization and its role in the prediction?

3. Why is number conversion important in the context of the transformer?

4. What is the significance of master data in the process of tokenization?

5. How does tokenization help in converting inputs to a particular number?

■ Summary

- The tutorial explains how to perform tokenization using Python's Transformers library

- It covers the process of creating and running a tokenizer for a particular model

■ Key Terms and Concepts

- Tokenization

- Pretrained model

- Python's Transformers library

- Auto tokenizer

- Google Agama model

■ Review Questions

1. What is tokenization and why is it used?

2. What function is used to import a pretrained model in the Python Transformers library?

3. How can you create a custom tokenizer for a specific model?

■ Summary

- Acquiring Hugging Face token

- Providing the token for proper functioning

- Loading of the tokenizer for a particular model

- The process of tokenization and its impact on interactions

- Tokenization process based on vocabulary size

- Vector embeddings and their role in providing a numerical representation of data

- Understanding vector embeddings of words using images of a cat and a dog

- How to plot and visualize vector embeddings on a graph

■ Key Terms and Concepts

- Hugging Face token

- Tokenization

- Vocabulary size

- Vector embeddings

- Semantic meaning

- 3D graph visualization

- Numerical representation of data

- Edge Vector Embeddings

■ Review Questions

1. How do you acquire a Hugging Face token, and why is it important?

2. What is the role and importance of loading the tokenizer for a particular model?

3. Explain the relationship between vocabulary size and the tokenization of words.

4. How do vector embeddings help in creating a numerical representation of data?

5. Describe how vector embeddings are used to understand the semantic meaning of words.

■ Summary

- Basic example of setting colors and calculating distance between objects

- Demonstrates likelihood through direction and probability

- Explains the relationship between words and semantic meanings

- Describes the formation of semantic embeddings and how to convert and use vector embeddi
ngs

- Suggests sourcing vector embeddings from OpenAI

■ Key Terms and Concepts

- Semantic meanings

- Vector embeddings

- OpenAI

- Probability

- Objects direction

■ Review Questions

1. What is the basic example of setting colors and calculating the distance between object

s?

2. How is likelihood demonstrated through direction and probability?

3. What is the relationship between words and semantic meanings?

4. Explain the formation of semantic embeddings and how to use vector embeddings.

5. Where can vector embeddings be sourced from according to the video?

■ Summary

- The code is simple and involves creating embeddings from input text using a particular m odel.

- The return embeddings are available for use with dimensions defaulting to 1536 for a par ticular model.

- The dimension parameters can be tweaked to reduce the dimension without losing the repre sentation properties.

- The length of the embeddings can be seen, and the vector embeddings are ready with seman tic meaning.

- Step one of the transformer is completed, and the next step is to be followed.

■ Key Terms and Concepts

- Embeddings

- Input text

- Model

- Return embeddings

- Dimensions

- Vector embeddings

- Transformer

■ Review Questions

1. What are the steps involved in creating embeddings from input text?

2. How can the dimension parameters be tweaked to reduce the dimension without losing repr esentation properties?

3. What does it mean when the vector embeddings are ready with semantic meaning?

4. What is the significance of completing step one of the transformer process?

■ Summary

- Positional encoding is a method used in natural language processing for creating embeddi ngs that store the position of words in a sentence

- Positional encoding allows the model to make use of the order of the sequence

- It adds information about the positions of tokens to the input vector embeddings, creati ng a new matrix with positions for every token

■ Key Terms and Concepts

- Positional encoding

- Token-by-token vector embeddings

- Sequence

- Cosine and sign formulas

- Magical matrix

- Recurrence and convocation

- Relative and absolute positions

■ Review Questions

1. What is the purpose of positional encoding in natural language processing?

2. How can position information be added to the input vector embeddings?

3. What is the significance of the magical matrix in positional encoding?

4. Explain the impact of positional encoding on the model's ability to make use of the seq uence order.

5. Why is it important to inject information about the relative and absolute positions of

tokens in the sequence?

■ Summary

- Positional encoding
- Multi-Head Attention
- Self Attention Model
- Transformer Architecture
- Vector Embeddings
- Contextual meaning in tokenization
- Financial Banking System

■ Key Terms and Concepts

1. Positional encoding
2. Multi-Head Attention
3. Self Attention Model
4. Transformer Architecture
5. Vector Embeddings
6. Contextual meaning
7. Tokenization
8. Financial Banking System

■ Review Questions

1. What is the purpose of positional encoding in transformer architecture?
2. How does the multi-head attention mechanism work?
3. Explain the role of the self-attention model in tokenization.
4. What is the significance of the vector embeddings in managing contextual meaning?
5. How does the traditional financial banking system relate to the concepts discussed in t

he video?

■ Summary

- Self-attention and multi-head attention are the core concepts

- Self-attention is based on vectors talking to each other

- To implement self-attention, an original matrix's transpose is divided by its square roo
t of dimensions

- Multi-head attention processes different sequences concurrently to understand everything

- Contextual understanding is at the heart of self-attention

■ Key Terms and Concepts

- Self-attention

- Multi-head attention

- Transpose

- Vectors

- Dimension

- Contextual understanding

■ Review Questions

1. How is self-attention implemented?

2. What is the purpose of multi-head attention?

3. What role does contextual understanding play in self-attention?

■ Summary:

- Tokenizing, encoding, and creating vector embeddings from input

- Using positional encodings to understand word positions

- Utilizing multihead attention to combine contexts

- Normalization layer and feed forward layer

- Repeated process to refine tokens
- Output prediction and new matrix generation
- Decoding tokens for human readability
- LLM working process
- User input, encoding, and obtaining output
- Sending output to the decoder with a start selection
- How to start with an initial input

■ Key Terms and Concepts:

- Tokenizing
- Encoding
- Vector embeddings
- Positional encodings
- Multihead attention
- Normalization layer
- Feed forward layer
- Output prediction
- New matrix generation
- Decoding tokens
- LLM (Language Learning Model)
- Encoder and decoder architecture
- Start of string or score selection

■ Review Questions:

1. What are the key steps involved in creating vector embeddings from input data?
2. How are positional encodings used to understand word positions in a sequence?

3. Describe the process of using multihead attention to combine contextual information.

4. What are the main components involved in the LLM working process?

5. Explain the significance of the start selection process when sending output to the deco der.

■ Summary

- The process of training a decoder in a neural network involves giving initial input and predicting an output, then calculating the loss and updating the weights to improve the ne xt output.

- The training phase includes an expected output from the user, and an output prediction b y the decoder.

- An inferencing phase occurs where the decoder provides an output, and the model calculat es the loss, followed by backpropagation to update the weights.

■ Key Terms and Concepts

- Encoder and decoder in a neural network

- Training phase and inferencing phase

- Cross-entrophy loss

- Backpropagation

- Neural network layers and weights

■ Review Questions

1. What are the key phases involved in training a decoder in a neural network?

2. Explain the process of updating the weights in the backpropagation phase.

3. What is the role of cross-entrophy loss in the training process?

■ Summary

- Back Propagation occurs with the loss

- When in inferencing mode, the model doesn't interfere

- During training, input outputs the next token and predicts it

- Model outputs the next token based on the input

- Back propagation starts from the output and goes back

- The linear and soft max functions are used for probability calculation

■ Key Terms and Concepts

- Back Propagation

- Inferencing mode

- Predictive modeling

- Soft Max

- Linear function

- Probability calculation

■ Review Questions

1. What is the purpose of back propagation in machine learning?

2. What happens when a model is in inferencing mode?

3. How does a machine learning model predict the next token based on the input?

4. What is the role of the linear and soft max functions in probability calculation in mac hine learning?

5. Explain the concept of back propagation in the context of training a machine learning m odel.

■ Summary

- Soft Max function is responsible for picking up the probability distribution

- Building a tokenizer and loading a model using Python code

- Obtaining input IDs for further processing

■ Key Terms and Concepts

- Soft Max function

- Tokenizer

- PyTorch framework

- Input IDs

- Model loading and downloading

- Python code snippets for tokenization and model loading

■ Review Questions

1. What is the role of the Soft Max function in probability distribution?

2. How is a tokenizer built in Python, and what is its purpose?

3. Explain the process of loading a model using PyTorch and obtaining input IDs.

4. How does the Python code aid in model loading and tokenization?

5. What are the key considerations when working with input IDs?

■ Summary:

- The video is a tutorial on using Python to add a string of numbers.

- The presenter demonstrates generating output for a given model and decoding it.

- The TensorFlow self-attention language model is discussed, emphasizing positional encodi ng, self-attention, and embedding.

- The process of using input and output tokens, and the difference between training and fi ne-tuning the model are also covered in the tutorial.

■ Key Terms and Concepts:

- Tokenize

- Return tensor

- Generative output

- Decoder

- Self-attention

- Positional encoding

- Embeddings

- Input and output tokens

- Language model training and fine-tuning

■ Review Questions:

1. What is the basic process for adding a string of numbers in Python?

2. How is the generative output from the model achieved in the tutorial?

3. What are the key concepts associated with the TensorFlow self-attention language model?

4. Explain the significance of input and output tokens in language modeling.

5. Differentiate between training and fine-tuning a language model.

■ Summary

- Inferencing fuzz and training fuzz are two types of fuzz.

- Training fuzz has a back inside the propagation layer for returning the model.

- Weights should be updated to make predictions and influence predictions.

- Back propagation is not used in influencing.

- Changing the GPT chart is possible by returning to the charge GPT.

- Going deep into the details is recommended for application developers.

- Unless you are a research engineer, you don't need to worry about internal workings.

- Understanding matrix multiplications are important for mathematical calculations.

- Building applications is emphasized, and the deployment of AI on a large scale is mentio ned.

■ Key Terms and Concepts

- Inferencing fuzz

- Training fuzz
- Back inside the propagation layer
- Back propagation
- Influence predictions
- GPT chart
- Charge GPT
- Matrix multiplications
- Deployment of AI on a large scale
- Machine learning content
- Application developer

■ Review Questions

1. What are the two types of fuzz mentioned in the video?
2. Why should weights be updated in the training process?
3. What does back propagation mean in the context of influencing?
4. How can you change the GPT chart?
5. Why is it important for application developers to understand matrix multiplications?
6. What is emphasized in the video when it comes to deploying AI on a large scale?