Day - 6 (Sun, 20th-OCT)

19 October 2024 22:14

Could you add to our last session, a discussion of what to expect on the Assessment Test? In this course, there have been a vast number of detailed topics, more than could be covered in the 30 question test. Could you provide some help on what topics should receive the most attention when studying?

Benefit of LangChain, Output Parser, Embedding, Vector Store
Advantage of Fine-Tuning, Pre-Trained Model
Parameter-Efficient Fine-Tuning, Appropriate Scenario
Best Practice, Pre-Trained Model
Purpose, Reinforcement Learning with Human Feedback, RLHF

Generative AI Challenges, Text Generation
Chatbot, Multiple Languages

Self-Consistency, Generated Data
Few-Shot Prompting, Task Improvement

Autoencoder, Image Generation
Prompt Template, Key Feature

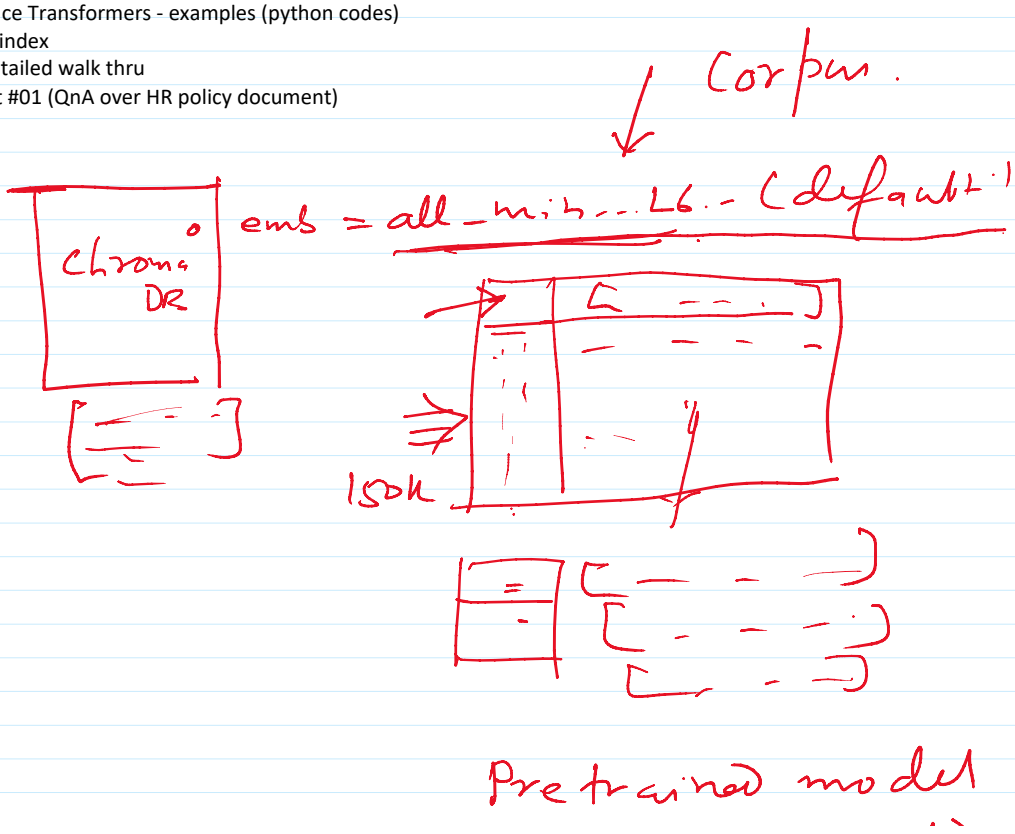
Github link for codes and session notes :

https://github.com/gridflowai/SL-GenAI-course-4/tree/main

Topics for today

- Vector Databases
  - Detailed walk thru of vector DB (ChromaDB) -- In progress
  - Limitations with vdbs --- choice (storage, retrieval) (next week)

- Detailed walk thru on OpenAI Embeddings (python codes)
- Overview on Transformer Architecture (overview)
- Tokenization in Transformers (overview)
- Sentence Transformers - examples (python codes)
- Llama-index
  - Detailed walk thru
- Project #01 (QnA over HR policy document)



Pretrained model (Bert).
tools

Tokenization in LM

Tokenization – key points

01

a fundamental text preprocessing technique

02

primary purpose is to break down a piece of text, typically a sentence or document, into smaller units called tokens.

03

Tokens are the individual elements or building blocks of text, which are usually words or subword units.

methods for tokenization in transformers

Byte-Pair Encoding (BPE)

- Divides words into subword units based on frequency of co-occurrence of byte pairs.
- Implemented in tokenizers like BertTokenizer and OpenAIGPTTokenizer.

WordPiece

- Similar to BPE but segments words into smaller units based on a language model's vocabulary.
- Used in tokenizers such as DistilBertTokenizer and RoBERTaTokenizer.

SentencePiece

- Provides an unsupervised text tokenization method that can tokenize text into subwords or even whole words.
- Applied in tokenizers like BartTokenizer and MBartTokenizer.

Byte Pair Encoding (BPE)

Byte Pair Encoding (BPE) is a data compression technique that has found applications beyond compression

Specially, in the field of natural language processing (NLP) for sub word tokenization.

BPE is used to progressively merge the most frequent pairs of characters or sub word units in a given corpus until a **desired** vocabulary size is reached.

Illustration of how Byte Pair Encoding works

let's say our corpus uses these five words: "hug", "pug", "pun", "bun", "hugs"

base vocabulary (sorted) will then be ["b", "g", "h", "n", "p", "s", "u"].

For real-world cases, that base vocabulary will contain all the ASCII characters, at the very least, and probably some Unicode characters as well.

If an example you are tokenizing uses a character that is not in the training corpus, that character will be converted to the unknown token.

Frequencies

let's assume the words had the following frequencies:

- ("hug", 10),
- ("pug", 5),
- ("pun", 12),
- ("bun", 4),
- ("hugs", 5)

meaning

- "hug" was present 10 times in the corpus,
- "pug" 5 times,
- "pun" 12 times,
- "bun" 4 times, and
- "hugs" 5 times.

Training the tokenizer

- splitting each word into characters (the ones that form our initial vocabulary) so we can see each word as a list of tokens:
  - ("h" "u" "g", 10),
  - ("p" "u" "g", 5),
  - ("p" "u" "n", 12),
  - ("b" "u" "n", 4),
  - ("h" "u" "g" "s", 5)
- pair ("h", "u") is present in the words "hug" and "hugs", so 15 times total in the corpus.
- ("u", "g"), which is present in "hug", "pug", and "hugs", for a grand total of 20 times in the vocabulary.
- Merging Most Frequent Pair: Identify the most frequent pair of consecutive characters (or subwords) in the current dataset. Merge this pair into a single token and add it to the vocabulary.

Merge

- the first merge rule learned by the tokenizer is ("u", "g") -> "ug", which means that "ug" will be added to the vocabulary, and the pair should be merged in all the words of the corpus.
- At the end of this stage, the vocabulary and corpus look like this:
  - Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug"]
  - Corpus: ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

Merge

- the pair ("h", "ug"), for instance (present 15 times in the corpus).
- most frequent pair at this stage is ("u", "n"), however, present 16 times in the corpus,
-
- So the second merge rule learned is ("u", "n") -> "un".
- Adding that to the vocabulary and merging all existing occurrences leads us to:
  - Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un"]
  - Corpus: ("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

… next

- Now the most frequent pair is ("h", "ug")
- So, we learn the merge rule ("h", "ug") -> "hug"
- which gives us our first three-letter token.
- After the merge, the corpus looks like this:

Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

Corpus: ("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

we continue like this until we reach the desired vocabulary size.

BPE Tokenization algorithm

Let's take the example we used during training, with the three merge rules learned:
("u", "g") -> "ug"
("u", "n") -> "un"
("h", "ug") -> "hug"

The word "bug" will be tokenized as ["b", "ug"]. "mug", however, will be tokenized as ["[UNK]", "ug"] since the letter "m" was not in the base vocabulary.

Likewise, the word "thug" will be tokenized as ["[UNK]", "hug"]: the letter "t" is not in the base vocabulary.

transformer models which use BPE

T5 (Text-to-Text Transfer Transformer)

- T5, developed by Google, uses SentencePiece, which is another variant of BPE, for subword tokenization.

ALBERT

- ALBERT, an optimized version of BERT, also uses a variant of BPE for subword tokenization.

DistilBERT (Distill BERT):

- DistilBERT, a distilled version of BERT, uses a variant of BPE for subword tokenization.

ERNIE (Enhanced Representation through kNowledge Integration):

- ERNIE, developed by Baidu, uses a variant of BPE for subword tokenization.

Advantages of BPE

Flexibility in Vocabulary Handling:
Because of subword units, allows the model to effectively handle out-of-vocabulary terms and morphological variations.

Adaptability to Multiple Languages:
BPE is language-agnostic

Efficient Handling of Rare Words:
BPE excels in capturing subword units, which makes it effective in representing and processing rare words or words not present in the training vocabulary.

Improves Generalization:
BPE aids in generalizing

1. build your own LM → ✓
2. Foundation model + Δ (own data)
   → Fine tuning.
   tokenizes → expen. → tokens →

Limitations of BPE

Computational Complexity
- training of BPE involves iteratively merging the most frequent pairs of subword units, which can be computationally expensive and time-consuming, especially for large datasets.

Vocabulary Size
- BPE may lead to a large vocabulary size, especially when considering subword units.

Loss of Interpretability
- Subword tokenization may result in tokens that do not directly correspond to meaningful linguistic units, leading to a potential loss of interpretability compared to word-level tokenization.

Example 1: Standard Words

- For common words that appear frequently in the training data:
  - Input: "Turing was a genius."
  - Tokenized Output: ["Turing", "was", "a", "genius", "."]

Example 2: Unseen Words or Rare Words

- Words that the model hasn't seen or that are less common might be broken into sub words.
- Input: "I visited Antananarivo."
- Assuming "Antananarivo" (the capital city of Madagascar) is not a common word in the training corpus:
- **Tokenized Output**: ["I", "visited", "Ant", "##an", "##an", "##ar", "##ivo", "."]
- *the "##" indicates that the subsequent piece is a continuation of the previous token.*

Example 4: Very Rare or Gibberish Words

Input: "I love xxyzzyx."

Assuming "xxyzzyx" is gibberish and the model hasn't seen anything like it:

Tokenized Output: ["I", "love", "x", "##x", "##y", "##zz", "##yx", "."]

Example 5: Words from Other Languages

- Input: "The samurai were warriors."
- Assuming "samurai" is not a common word in the training corpus (though in many actual corpora, it probably would be):
- Tokenized Output: ["The", "sam", "##ura", "##i", "were", "warriors", "."]

Tokenizers used

Byte-Pair Encoding (BPE)	WordPiece
• **BertTokenizer**: Tokenizer used for models like BERT • **OpenAIGPTTokenizer**: Tokenizer for the OpenAI GPT (Generative Pre-trained Transformer).	• **DistilBertTokenizer**: Tokenizer for DistilBERT, a distilled version of BERT. • **RoBERTaTokenizer**: Tokenizer for RoBERTa (Robustly Optimized BERT Approach).