Video Link :- https://youtu.be/zduSFxRajkE?list=PLAqhIrjkxbuWI23v9cThsA9GvCAUhRvKZ

10/11/24

Tokenizer - Andrej Karpathy

tokenizing :-
- converting every possible character into a token (Integer)
  - Using an embedding table

(Ex) - if there are 65 possible tokens, then in the embedding table there are 65 tables. These rows are trainable parameters (using back propagation) - these vectors are fed into transformers.

Tokenization → is the process for translating strings or texts into sequence of tokens and Vice Versa

"Tik tokenizer . vercel app" → live tokenizer count

→ encodings are better than Unicodes.
(doesn't change num) (they change as bytes)

UTF 8, UTF 16, UTF 32
- most common
- Variable length
- backward compatable for ASCII encoding

UTF 32 → fixed length
has multiple zeros and not desirable.

→ Byte Pair Encoding Algorithm → allows to compress the byte sequences of the UTF-8 encodings

→ The byte encoding can be directly used in the transformers but the attention calculations will be very large → it will be tokenization free. [Good]

Algorithm :-

① find pair of tokens that occur frequently
② Replace that pair with a single new token
③ Repeat the step ① and ② again till the least pairs are found.

Picture [upload]

```
aaabdaaabac
```

The byte pair "aa" occurs most often, so it will be replaced by a byte that is not used in the data, such as "Z". Now there is the following data and replacement table:

```
ZabdZabac
Z=aa
```

Then the process is repeated with byte pair "ab", replacing it with "Y":

```
ZYdZYac
Y=ab
Z=aa
```

The only literal byte pair left occurs only once, and the encoding might stop here. Alternatively, the process could continue with recursive byte pair encoding, replacing "ZY" with "X":

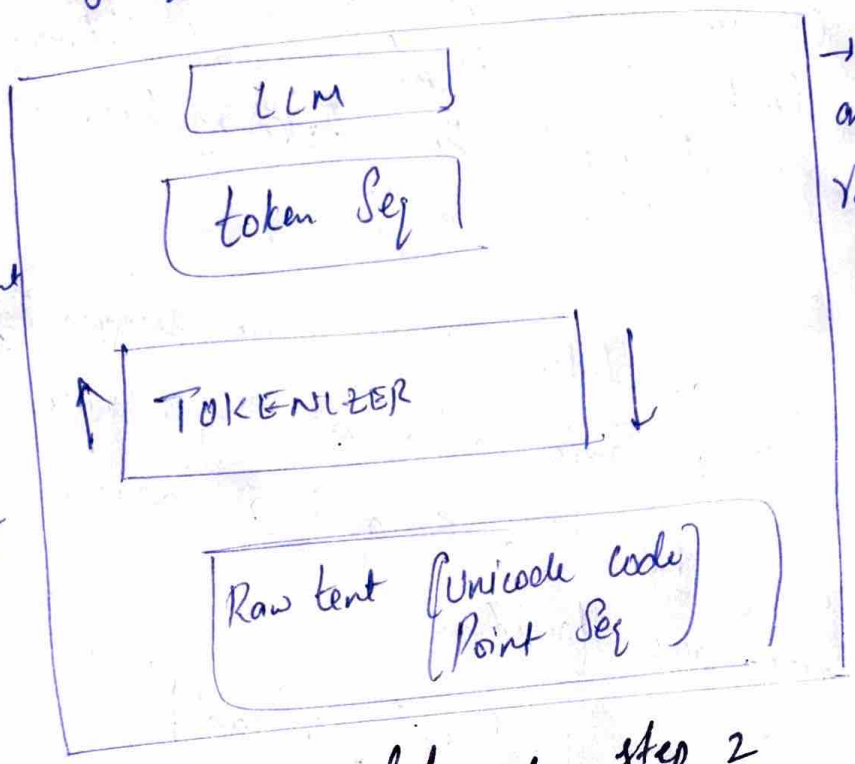
```
XdXac
X=ZY
Y=ab
Z=aa
```

This data cannot be compressed further by byte pair encoding because there are no pairs of bytes that occur more than once.

To decompress the data, simply perform the replacements in the reverse order.

→ tokenizer is a completely seperate object from the large language model it self

① → tokenizer is completely seperate independent module from LLM
→ has its own training dataset of text on which BPE algorithm is applied



② → translates back and forth btw raw text and sequence of tokens

→ language model is trained later as step 2

decoding :-

Vocab : mapping / dictionary in Python
→ mapping id to bytes object of that token.
→ [0, 255]
→ tokens at this stage are Raw bytes. need to decode those using UTF 8

Text → Alphabet

Encoding :-

1. get raw bytes of the tokens after merged [BPE encoded]
2. get the pairs with min merges

(CODE the tutorial) TODO

BPE → Practical middle ground between character and word level LM, that interpolate between word level inputs for frequent symbol sequences and character level inputs for in-frequent sequences.

→ BPE operates on unicode points not byte sequences

→ Tokenizer can do both encode and decode

GPT2 - enforce few rules so that some parts of the vocabulary will not get merged.

[Tokenization BPE algorithm modification]

written that rules in the Encoder class under "self.pat = Regex code"

regex → extension of re

→ It will take the pattern and try to match with the strings (left to right), will get all occurances and put in a list.

→ tokenization in GPT works differently for uppercase and lowercase

tiktoken - OpenAI's official tokenization library

- Changed the regex in GPT 4 compared to GPT 2 (equivalent, executes faster)

Vocab Size → 50K to 100K
(GPT2) (GPT4)

<|endoftext|> Special Token's.

These special tokens are fully a single Token

<|IMStart|> , <|IM_END|>

"cl100k_base" → OpenAI encoding

→ Transformer model surgery is required when the special tokens are added to the model. Increase the vocab size

ToDo **Exercise** :- minbpe exercise in Karpathy's GitHub

BPE Algorithm | Karpathy/minbpe (training a tokenizer)

https://github.com/hrithiksagar/minbpe-exercise-tokenization

link :- ---

SentencePiece -- Can train and inference BPE tokenizations
(Used in Llama, mistral)

- Runs BPE on Unicode Points Directly.
- encodes with Utf8 and then encodes the raw bytes instead

(Q) Why can't Vocab size = infinite?

(A) (1) · token embedding table is going to grow
· linear layer is going to grow
· lot of computation will be going on

(2) · Because we have more parameters, we are going to train these parameters
· model will not have time to think per some number of characters in text.

Solid GoldMagikarp :- observed that there are few chunks of tokens are very weird.

↓

reddit user [who used to post a lot]

Undefined never before seen data appeared in forward pass, but never trained in backward pass

the output when asked about those tokens lead to weird outputs seen as hallucinations. (trigger words)