2. **Byte-Pair Encoding (BPE) Tokenizer**

    1. **Understanding Unicode**

        a. `chr(0)` returns '`b'\x00`"

        b. `__repr__()` shows an escaped, unambiguous form whereas its printed representation outputs the character itself which is actual NULL character(invisible).

        c. When we add it to text, it simply adds an invisible character at that position which may not be visually noticeable.

    2. **Unicode Encodings**

        a. UTF-8 is preferred over UTF-16 or UTF-32 due to its superior compression, universal applicability without out of vocabulary errors, and native compability with byte-stream data.

        b. If we give the input 牛, it breaks and gives an error "UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe7 in position 0: unexpected end of data". This is due to the fact that the decoding is done byte by byte and when the encoded character is multi-byte, it fails to decode it properly.

        c. `b'\xC3\x28`' is invalid in UTF-8 because 0xC3 indicates the start of a 2-byte character, but 0x28 is not a valid continuation byte.

    3. **BPE Training on TinyStories**

        a. It seems that the longest token is 'accomplishment' with a length of 15 bytes. The memory usage is around 158.3 MB and the training took approximately 61.72 seconds. It makes sense that longer tokens are formed from frequently occurring sequences of characters in the dataset, and 'accomplishment' might be a common word in the TinyStories dataset.

        b. Pre-tokenization part took the most time (40 seconds). Finding chunks and their word count seems to be the most time-consuming part of the process. Second highest is merging the tokens based on the pairs found (20 seconds).

    4. **Experiments with tokenizers**

        a. The tokenizer's compression ratio (bytes/token) is around 2.8609 in a sample of 10 documents from TinyStories dataset.

        b. The througput of the tokenizer is approximately 3000 bytes/second. To tokenize the Pile dataset (about 825 GB), it would take around 76,388 hours.

        c. uint16 is appropriate choice for encoding the token IDs since the vocabulary size is 10,000 which fits well within the range of uint16 (0 to $2^{16} - 1 = 65535$).

3. **Transformer Language Model Architecture**

    1. Subsection 1

        a. a

        b. b

    2. Subsection 2

        a. a

        b. b

        c. c