CBOR Compression using BPE Tokenizer

tokenizer-compression

Jaime Jiménez, SCHC, IETF 2024-06-11
On tokenization

- Tokenization is independent of the LLM, a different module with its own training dataset.
- Tokenization of the string "hello world!!!" using a hypothetical larger vocabulary base:

```python
enc = tiktoken.get_encoding("cl100k_base")
print(enc.encode("hello world!!!"))
```

```
[262, 24748, 1917, 12340]
```
Objective

Use Byte Pair Encoding (BPE) to compress arbitrary strings.

- CBOR's limitation is that vendors need to add their own mappings to their specific data syntax in order to add further string compression.
- General-purpose solution for new strings (e.g., addresses, keyboard input, languages,...) it is not a dictionary and it is not a CBOR map.
- Small vocabulary size (30kb) yields already good compression.
- String pattern matching of tokenized string is possible (diff).
- Works on new strings not used for vocabulary training (new vocabulary can be released periodically).
Objective

"Hirsalantie 11, 02420 Kirkkonummi"

becomes

[72, 385, 115, 269, 271, 270, 101, 32, 3351, 44, 32, 1056, 52, 468, 32, 75, 385, 107, 107, 260, 383, 109, 105]
Implementation: train

- **minbpe** for vocabulary training.
- used cbor2 for the cbor encoding/decoding.
- Used brotli and rfcs as input training text.

```bash
poetry run python3 train.py
merge 1/3840: (34, 10) -> 256 (b''\n'') had 9216 occurrences
merge 2/3840: (256, 34) -> 257 (b''\n''') had 9215 occurrences
...
merge 3839/3840: (277, 437) -> 4094 (b''might''') had 2 occurrences
merge 3840/3840: (277, 328) -> 4095 (b''mig''') had 2 occurrences
Training took 223.02 seconds
```
Implementation: run

- Tokenize CBOR Major type 3 strings.

```json
{
    "device_id": "urn:dev:A04B0D56-02A7E3",
    "owner": "John Smith",
    "temperature": 22.5,
    "humidity": 60,
    "address": "Hirsalantie 11, 02420 Kirkkonummi",
    "plus-code": "4GJ7+FJ Kirkkonummi",
    "timestamp": "2024-03-11T10:00:00Z"
}
```
Implementation: output

- Experiments on *Synthetic Data*
- 20–30% data compression.
- **Inference** on more powerful endpoints also possible, LLMs consume tokens.
Food for thought

Tradeoffs

- Benefits of Compression vs Encoded string manipulation
- The Vocabulary Size vs HW Requirements on the endpoint
- The Vocabulary Size vs size of the input strings

Future work?

- Run LLM inference on receiving endpoint.
- Generation of more optimal vocabularies.
- Use better input training texts.
is this useful at IETF?

• ... for SCHC *payload* compression?
• ... as *informational* material?
Related work


- [2] F. Bellard, "NNCP v2: Lossless Data Compression with Transformer".


- [4] sentencepiece encoding. Primary difference being that sentencepiece runs BPE directly on Unicode code points instead of on UTF-8 encoded bytes. It is used by Llama and Mistral.