A reply to Moro et alia's claim that "LLMs can produce 'impossible' languages"

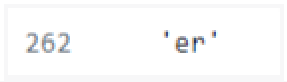
Stela Manova
manova.stela@gmail.com

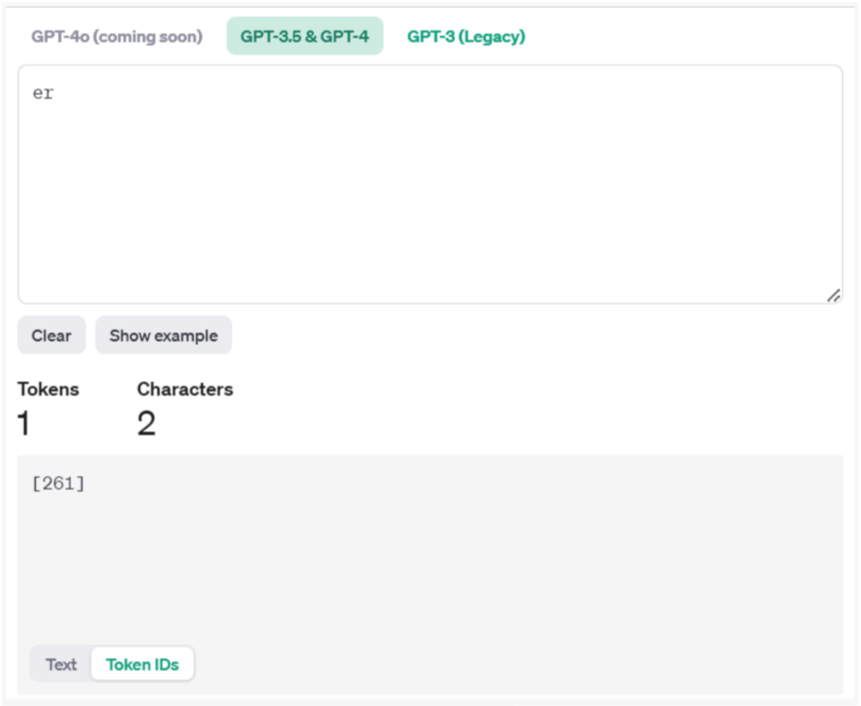
In two recent short texts, Moro, Greco & Cappa (2023), published as a *viewpoint* in *Cortex*, and Bolhuis, Crain, Fong & Moro (2024), published under *correspondence* in *Nature* (see also lingbuzz/008093 and lingbuzz/008092, respectively), the authors claim that LLMs can produce 'impossible' languages. In neither of the pieces a specific LLM able to generate 'impossible' languages is mentioned. Thus two questions arise:

Question 1

Why don't Moro et alia make public the set of characters and tokens used by the LLM that managed to produce an 'impossible' language? And since in order to train a LLM one needs a large amount of text, it would also be interesting to learn about the origin of the large 'impossible' language data used in the LLM training?

Let me illustrate the type of information I require from Moro et alia: A LLM such as ChatGPT works not with words but with tokens, more precisely with token IDs, see below. ChatGPT operates with a set of 100k tokens, a list of all tokens can be found at the following GitHub page: https://github.com/kaisugi/gpt4_vocab_list/blob/main/cl100k_base_vocab_list.txt. These tokens are extracted from Internet texts. Additionally, the list of tokens can be verified with the ChatGPT interactive tokenizer, available at: https://platform.openai.com/tokenizer. For example, *-er* / "er" is #262 in the list of tokens (Figure 1) and has the token ID `[261]` in the tokenizer (Figure 2). Due to the fact that counting in computers starts from `[0]`, the numbering in the two sources differs in a unit (the list of tokens follows the human way of counting and starts from #1). Nevertheless, the list and the tokenizer coincide. Thus, *A* / "A" is #33 in the list and `[32]` in the tokenizer, *linguistic* / " linguistic" is #65768 in the list and `[65767]` in the tokenizer, etc. A smaller ID indicates a higher frequency of the item, i.e. *A* `[32]` is much more frequent than *linguistic* `[65767]`. The 100k list of tokens contains 100256 entries because the first 256 items serve for initialization of the vocabulary, i.e. for the building of the 100k tokens.


```
262     'er'
```
Figure 1: *-er* in the list of tokens



```
er
```

Tokens 1 Characters 2

```
[261]
```

Figure 2: -*er* in the interactive tokenizer

For tokenization, ChatGPT uses `tiktoken` (https://github.com/openai/tiktoken), which involves Byte Pair Encoding (BPE). Tokenization makes possible the representation of a large amount of text with a small set of subword units (tokens). An easy to understand illustration of the BPE logic can be found at: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp/; the most important steps of the algorithm are the following:

Text corpus: **"ab", "bc", "bcd", "cde"** (i.e. the corpus consists of four words)

Step 1: Initialize the vocabulary (i.e. establish all characters (bytes) used in the corpus)
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters (in the same position)
The most frequent pair in the initial position is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain "bc" (i.e. of "bc" itself):

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

Step 3d: Add the new subword unit to the vocabulary
Add "bc" to the vocabulary:

Vocabulary = {"a", "b", "c", "d", "e", "bc"}

Repeat steps 3a-3d until the desired vocabulary size is reached.

It is hard to imagine the set of tokens used by a LLM generating 'impossible' languages, as well as how such tokens should be extracted and combined. ChatGPT extracts tokens from large linear sequences of text in existing (i.e. possible) languages and based on statistical information about the co-occurrences of the tokens in these sequences selects the next token when generating language, which thus makes language a large linear sequence of tokens.

Question 2

If there exist impossible languages that can serve for testing the nature of the language competence of LLMs, why don't Moro et alia discuss what would happen if a child is exposed only to an 'impossible' language as L1 input?

Conclusions

An experiment verifying whether a LLM can produce 'impossible' languages is impossible, at least for the moment: It is unreasonable to invest millions of dollars in training a LLM to generate something useless.

An experiment on whether a child exposed only to an 'impossible' first language would acquire this language cannot be run either, but for a different reason: It is unethical.

Does this situation justify Moro et alia's claim that LLMs can produce 'impossible' languages? I do not think so.