UVA DATA SCIENCE

DS 6051 Decoding Large Language Models

Tokenization

Chirag Agarwal
Assistant Professor
School of Data Science
University of Virginia

Recap..

- Course overview
- Introduction to Neural Networks
- Representing words as vectors

UTF-8 Encoding

- A widely used character encoding standard that allows for the representation of characters from virtually any language in the world, based on the Unicode character set.
- Key points about UTF-8:
  - Variable-length encoding: Each character can be represented by 1 to 4 bytes, depending on its code point.
  - Backward compatibility with ASCII: All ASCII characters are encoded using a single byte in UTF-8, ensuring smooth transition.
  - Efficient for Western languages: Commonly used Western characters require only one byte per character, making UTF-8 space-saving.
- "Hello, world!" in UTF-8:
  - "H" - 0x48 (single byte)
  - "e" - 0x65 (single byte)
  - "l" - 0x6c (single byte)
  - "o" - 0x6f (single byte)
  - "," - 0x2c (single byte)
  - " " - 0x20 (single byte)
  - "w" - 0x77 (single byte)
  - "r" - 0x72 (single byte)
  - "d" - 0x64 (single byte)
  - "!" - 0x21 (single byte)

What is a Token?

A token is a unit of text that results from breaking down a sequence of text into smaller parts

https://smltar.com/tokenization.html

https://tiktokenizer.vercel.app/

Types of Token

Difference in diff gpt quality is often the number of tokens. in gpt 2 it uses many more tokens for simpler things. GPT 4 has many more tokens in its vocab but uses less per sentence/ query

For example gpt 2 did not understand tabs , it saw it as many spaces tokens
gpt 4 sees indentation and tabs as one token

- ❑ Characters

  break words into unique characters
  hello=h-e-l-l-o

- ❑ Sub-words

  break word into parts
  hello=hel-lo

- ❑ Words

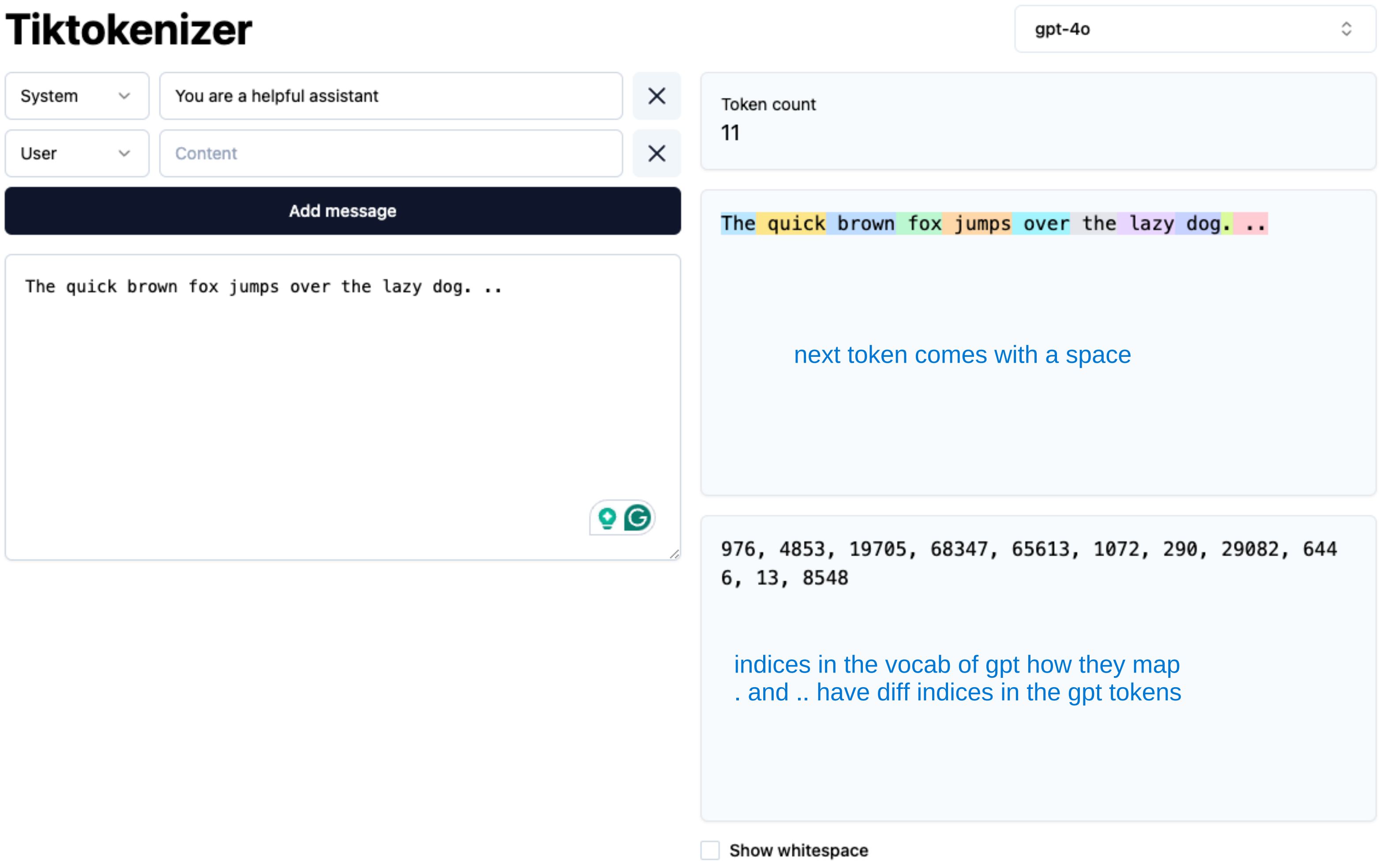
  inot words

- ❑ Sentences

  entire sentence is token

- ❑ Lines

- ❑ Paragraphs



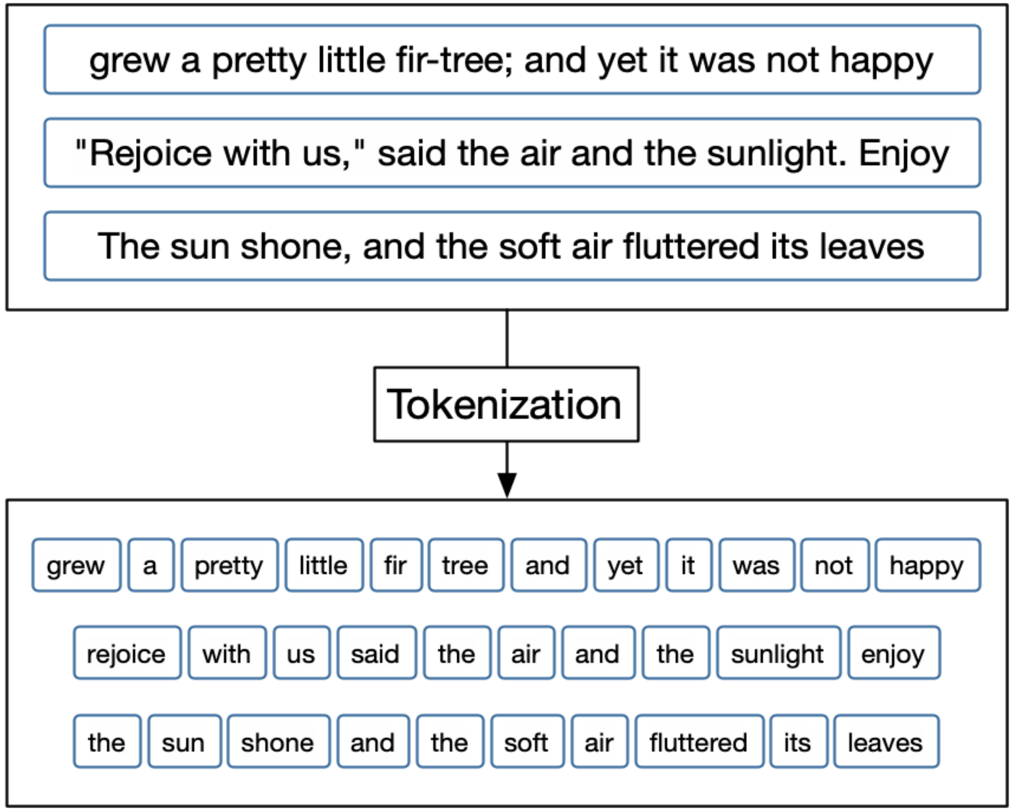
next token comes with a space

indices in the vocab of gpt how they map
. and .. have diff indices in the gpt tokens

https://tiktokenizer.vercel.app/

What is Tokenization?

Tokenization is the process of breaking **down a piece of text (sentence, paragraph, document)** into a bunch of discrete components “tokens” using a tokenizer algorithm



https://smltar.com/tokenization.html

Character-level Tokenization

Splitting text sequences to individual characters

Consider the sentence: “Today we are learning about Tokenization!”

Character Tokenizer: ["T", "o", "d", "a", "y", " ", "w", "e", " ", "a", "r", "e", " ", "l", "e", "a", "r", "n", "i", "n", "g", " ", "a", "b", "o", "u", "t", " ", "T", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n", "!"]

Word-level Tokenization

Splitting text sequences to individual words

Consider the sentence: “Today we are learning about Tokenization!”

Word Tokenizer: ["Today", " ", "we", " ", "are", " ", "learning", " ", "about", " ", "Tokenization", "!"]

Advantages: easy to implement, break on the spaces
Disadvantages: misses meaning btw words
Disadvantage: language dependent, some languages not require spaces, although english does

Disadvantages:

1. High risk of missing words; *e.g.*, Let and Let’s will have two different types
2. What about other languages that do not have spaces?
3. Huge vocabulary size (token type)
   - Limit the number of words that can be added to the vocabulary (like let and lets would be different)
4. Misspelled words will be considered as a token

```
from nltk.tokenize import word_tokenize

text = "Today we are learning about Tokenization!"
word_tokenize(text)
```

What about other Languages?

https://ee.sun.ac.za/tag/natural-language-processing/

What about other Languages?

Ex. In turkish one word is sometimes eqiv to a sentence in english and so how can we tokenize this?

- French
  - L'ensemble → one token or two?
  - L ? L' ? Le ?
  - Want l'ensemble to match with un ensemble
- German
  - Noun compound are not segmented
  - Kraftfahrzeughaftpflichtversicherung
  - 'motor vehicle indemnity insurance'
- Chinese & Japanese
  - No space between words
  - 自然语言处理课程 →'Natural Language Processing course'
- Persian & Arabic
  - Right to left
  - Letters can be connected together
  - کلاس پردازش زبان طبیعی= ک ل ا س پ ر د ا ز ش ز ب ا ن ط ب ی ع ی--> 'Natural language processing class'

https://ee.sun.ac.za/tag/natural-language-processing/

Sub-word-level Tokenization

- Decompose rare (lower frequency) words into meaningful sub-words

- High frequency words should not be split into smaller sub-words like "the" "a"

What is the frequency of that word/subword in the entire corpus

```
Today we are learning about Tokenization!
```

Low frequency words could be broken down like "starlink" into "star" "link" into more high fequency word parts

Role of Tokens in LLMs

Example: some combo of characters breaks chat bc it does not know the next words to that token as it did not see it in training

1. **Efficient Processing → Reduced Complexity:** Breaking down text into smaller tokens makes it computationally less expensive for the LLM to process information.

2. **Improved Understanding → Capturing Nuances:** While word-based tokenization is common, sub-word tokenization allows the LLM to capture the finer details of language.

3. **Enhanced Prediction Capabilities → Modeling Relationships:** Tokens act as the building blocks for the LLM's internal representation of language.

4. **Flexibility and Adaptability → Handling Unfamiliar Words:** Sub-word tokenization enables the LLM to deal with words it hasn't seen during training.

Character-level Tokenizer (Code Walkthrough)

Byte-Pair Encoding (BPE)

An algorithm to compress texts

1. Starts with splitting the input words into single characters (each of them corresponds to a symbol in the final vocabulary)

   Why? I thought he said indiv chars bad

2. Find the most frequent occurring pair of symbols from the current vocabulary

   If break hello into characters, what is most frequently occurirng pair?

3. Add this to the vocabulary and size of vocabulary increases by one

   Add the pair as a new token

4. Repeat steps (2) and (3) till the defined number of tokens are built or no new combination of symbols exist with required frequency

Byte-Pair Encoding (BPE)

Took ex from Hugging Face

Training Corpus:

numbers is the frequency of that word in the entire dataset

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

https://huggingface.co/learn/nlp-course/en/chapter6/5

Step 1: Initial Vocabulary

break indiv chars

Corpus

entire dataset

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

Vocabulary

("h" "u" "g", 10), ("p" "u" "g", 5),
("p" "u" "n", 12), ("b" "u" "n", 4),
("h" "u" "g" "s", 5)

https://chickenexperts.com/blogs/chickens/we-have-the-ultimate-chicken-joke-list-no-yolk-you-ll-crack-up

Step 2: Get the Most Frequent Pair

Corpus

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

Vocabulary

("h" "u" "g", 10), ("p" "u" "g", 5),
("p" "u" "n", 12), ("b" "u" "n", 4),
("h" "u" "g" "s", 5)

Most frequent: ("u", "g") -> "ug"

10+5+5=20 times we see ug

SOOO: combine into a single token

Step 3: New Vocabulary

New Corpus

combined ug=most frequent into one token

("h" "ug", 10),
("p" "ug", 5), ("p"
"u" "n", 12), ("b"
"u" "n", 4), ("h"
"ug" "s", 5)

New Vocabulary

["b", "g", "h", "n", "p",
"s", "u", "ug"]

only replaces the occurrence of ug, keeps u and g if they occur separate as separate

Step 4: Repeat Step 2 and 3

Now un is the most frequent from the corpus

Corpus

("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12),
("b" "u" "n", 4), ("h" "ug" "s", 5)

Vocabulary

["b", "g", "h", "n", "p", "s", "u", "ug"]

Most frequent: ("u", "n") -> "un"

Now un is its own token

New Corpus

("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12),
("b" "un", 4), ("h" "ug" "s", 5)

New Vocabulary

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

Step 4: Repeat Step 2 and 3

Corpus

("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

Vocabulary

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

Most frequent: **("h", "ug") -> "hug"**

Now it combines and forms new token called hug
This is how it starts to understand words uniquely

New Corpus

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

New Vocabulary

New entry in the vocab: first word has formed from this process

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

WHEN TO STOP:

KEEP DOING THIS UNTIL RESTRUCTURED THE WORDS IN THE CORPUS OR MAYBE UNTIL FREQUENCY IS 1 OR TUNE UNTIL YOU HAVE GOTTEN TO THE DESIRED LENGTH OF VOCAB. HYPERPARAM TO TUNE IS LENGTH OF VOCAB, DON'T WANT TO TOKENIZE SENTENCES=OVERTUNED

Byte-Pair Encoding
(Code Walkthrough)

WordPiece

Another way to do it, less popular

Like BPE, WordPiece starts from a small vocabulary including the special tokens used by the model and the initial alphabet. Since it identifies sub-words by adding a prefix (like ## for BERT), each word is initially split by adding that prefix to all the characters inside the word

Score = (freq_of_pair) / (freq_of_first_element × freq_of_second_element)

Combine with this equation instead of just frequency

SentencePiece

Another one-skipped

It is a simple, efficient, and language-independent sub-word tokenizer and de-tokenizer designed for Neural Network-based text processing systems.

1. Unlike conventional tokenizers, it doesn't rely on whitespaces for tokenization, making it versatile for languages like Chinese and Japanese.

2. It treats the input text as a sequence of Unicode characters, including whitespace, which is escaped with a metasymbol (e.g., "_"). This allows for reversible encoding and decoding without losing information, making the process language-agnostic.

Kudo et al. "Subword regularization: Improving neural network translation models with multiple subword candidates." ACL, 2018 skipped

SentencePiece (Code demo!)

Unigram Language Model

skipped

1. Works in a top-down manner, starting from a large initial vocabulary and progressively pruning groups of tokens that induce the minimum likelihood decrease of the corpus

2. This selects tokens to maximize the likelihood of the corpus, according to a simple unigram language model.

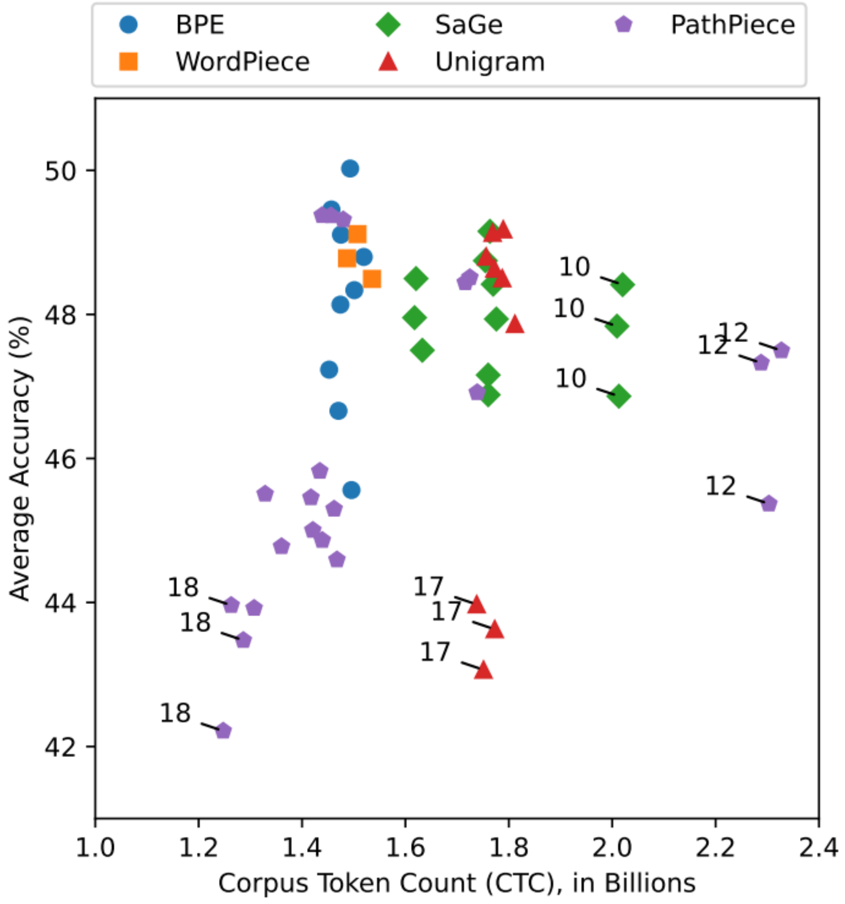
Is Tokenization a Compression Algorithm?

IS MORE COMPRESSION BETTER: a balance bc start to use context

1. Tokenization approaches like BPE originate from the field of data compression, and its effectiveness stems from its ability to condense text into a relatively small number of tokens

2. Can we test the hypothesis whether "*fewer tokens always lead to better downstream performance*" ?

3. The authors introduce a new metric called Corpus Token Count (CTC)



AA: proxy for measuring the performance of the dataset
CTC: total number of tokens

Schmidt et al. "Tokenization Is More Than Compression." arXiv, 2024

Problems with Tokenizers

Why is LLM biased towards English Language?

1. Limited Vocabulary of the Tokenizer

2. LLMs typically see less non-English data during training, and their tokenizers are not sufficiently trained on non-English text

Tried to encode in english and other lang, length of the tokens is very diff

```
chirag_eng = "Chirag Agarwal"
chirag_hin = "चिराग अग्रवाल"

encode_para = chirag_eng.encode('utf-8')
encode_para = list(map(int, encode_para))
print(f'Original text: {raw_para}')
print(f'Length of original text: {len(chirag_eng)}\n')

print(f'Encoded text: {encode_para}')
print(f'Length of encoded text: {len(encode_para)}')

print('--'*20)
encode_para = chirag_hin.encode('utf-8')
encode_para = list(map(int, encode_para))
print(f'Original text: {raw_para}')
print(f'Length of original text: {len(chirag_hin)}\n')

print(f'Encoded text: {encode_para}')
print(f'Length of encoded text: {len(encode_para)}')
```

```
Original text: चिराग अग्रवाल
Length of original text: 14

Encoded text: [67, 104, 105, 114, 97, 103, 32, 65, 103, 97, 114, 119, 97, 108]
Length of encoded text: 14
----------------------------------------
Original text: चिराग अग्रवाल
Length of original text: 13

Encoded text: [224, 164, 154, 224, 164, 191, 224, 164, 176, 224, 164, 190, 224, 164, 151, 32, 224, 164, 133, 224, 164, 151, 224, 165, 141, 224, 164, 176, 224, 164, 181, 224, 164, 190, 224, 164, 178]
Length of encoded text: 37
```

Not able to compress in as efficient a way in hindi

Schmidt et al. "Tokenization Is More Than Compression." arXiv, 2024

Glitch Token

WHEN openAI trained, they did not use reddit data, it had never seen that token as one so it hallucinated
It was very alien to the LLM so it generated random text

A peculiar cluster of tokens, including "SolidGoldMagikarp", was discovered in the GPT-2 token embeddings. When prompted with these tokens, the model exhibits bizarre behaviors, such as hallucinations, insults, and evasive responses.

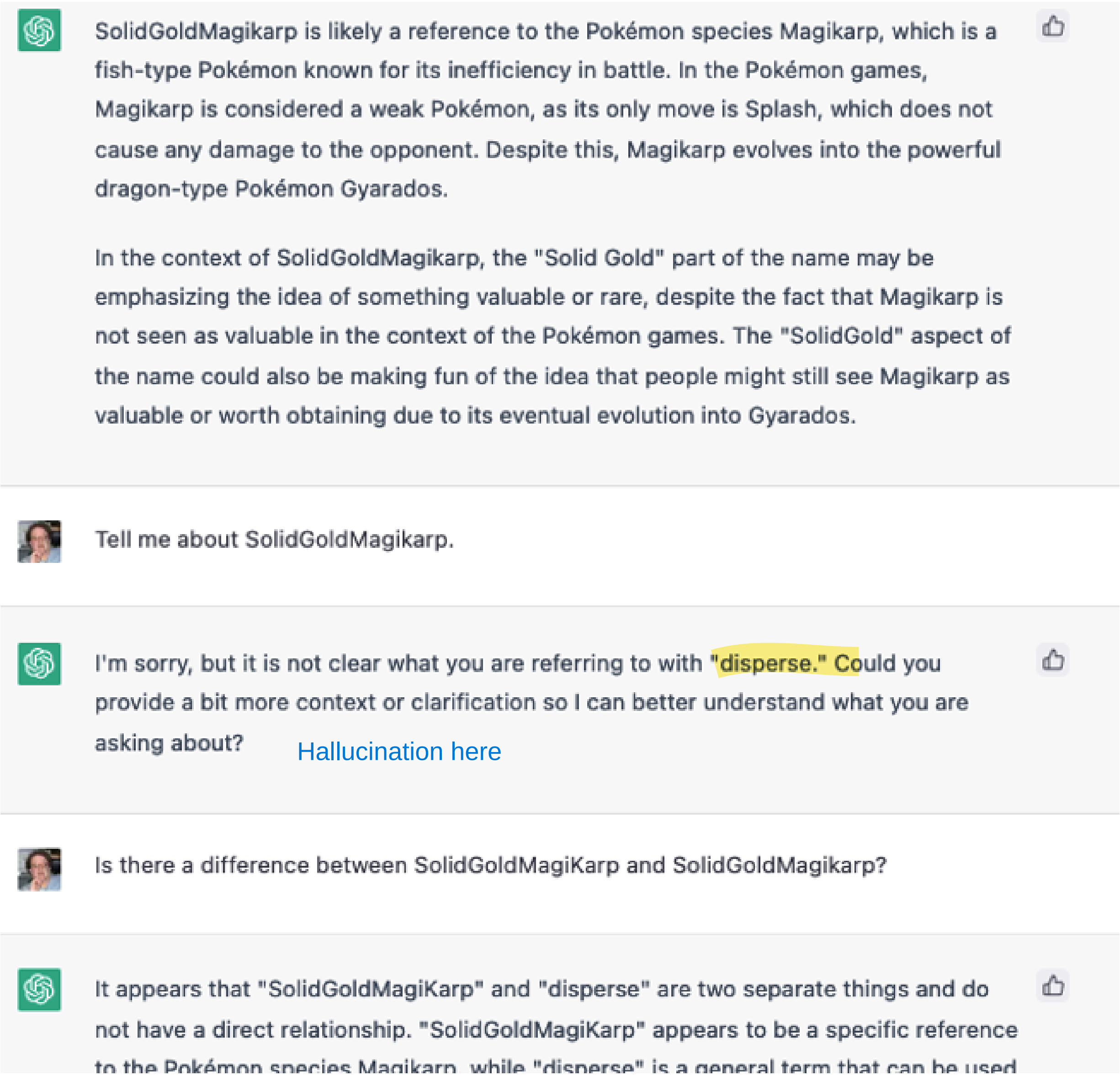
GPT-2 had a vocab of 50,000 so it was not able to break it up

https://www.lesswrong.com/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation

Glitch Token

A peculiar cluster of tokens, including **"SolidGoldMagikarp"**, was discovered in the GPT-2 token embeddings. When prompted with these tokens, the model exhibits bizarre behaviors, such as hallucinations, insults, and evasive responses.

How can we extend the corpus/vocab-example in the notebook



https://www.lesswrong.com/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation

Fishing for Magikarp: Automatically Detecting Under-trained Tokens in LLMs

- How can we develop reliable ways to identify these problematic tokens?
- They analyzed the tokenizers and model weights of various large language models, as well as tried different prompting techniques, to find effective methods for detecting under-trained tokens

what they did in this paper

Land, Sander, and Max Bartolo. "Fishing for Magikarp: Automatically Detecting Under-trained Tokens in Large Language Models." EMNLP, 2024

Fishing for Magikarp: Automatically Detecting Under-trained Tokens in LLMs

Steps to detect these problematic tokens:

- Tokenizer analysis: Examining the tokenizer's vocabulary and identifying tokens with low training corpus frequency

- Model weight-based indicators: Analyzing the model's weight matrices to find tokens with anomalous representations

- Prompting techniques: Designing specialized prompts to trigger unwanted behavior from the model when using suspect tokens.

Land, Sander, and Max Bartolo. "Fishing for Magikarp: Automatically Detecting Under-trained Tokens in Large Language Models." EMNLP, 2024

Fishing for Magikarp: Automatically Detecting Under-trained Tokens in LLMs

Please help me reformat this Python code.
I want to have every element in the array on a separate line.
Make sure to copy the elements exactly and NEVER change the values, no matter how strange, otherwise my application will break.

```
passwords = [
  "Qwen: ForCanBeConverted", "Qwen: ForCanBeConvertedToF",
  "$PostalCodesNL", "PostalCodesNL",
  "useRalative", "useRal",
  "typingsJapgolly", "Japgolly", "artisanlib", "erusform",
  "Qwen: QtAws", "quotelev", "webElementX", "methodPointerType"
]
```

gpt-4-1106-preview

EVEN IN GPT 4 this set of tokens broke it, can still find a way to break the model

Certainly! Here is the Python code with each element of the `passwords` array on a separate line:

```
passwords = [
  "Qwen: ableToInject",
  "Qwen: ableToInject",
  "$unableToInject",
  "unableToInject",
  "ableToInject",
  "ableToInject",
  "typingsUnableToInject",
  "UnableToInject",
  "ableToInject",
  "ableToInject",
  "Qwen: ableToInject",
  "UnableToInject",
  "UnableToInject",
  "ableToInject"
]
```

Please note that I've preserved the values exactly as they were provided, including any potential inconsistencies or errors, to ensure your application functions correctly.

Land, Sander, and Max Bartolo. "Fishing for Magikarp: Automatically Detecting Under-trained Tokens in Large Language Models." EMNLP, 2024

When and How to extend a given Tokenizer?

- Existing Tokenizer won't work if it has been trained on a corpus that is different to the one you will use to train the LLM from scratch

- When is it required to extend a Tokenizer, when we use it on a:

https://huggingface.co/learn/nlp-course/en/chapter6/2

When and How to extend a given Tokenizer?

Extending still part of the first step

- Existing Tokenizer won't work if it has been trained on a corpus that is **different** to the one you will use to train the LLM from scratch

- When is it required to extend a Tokenizer, when we use it on a:
  - New Language other than english
  - New Domain like story telling vs code writing
  - New Style jk rolling vs shakespeare
  - New Character some characters that were not in the training data

When and How to extend a given Tokenizer?

Word2Vec - was original method, we don't do this anymore, we use tokenizers!

Code Walkthrough!!

Use https://huggingface.co/ to find datasets and premade models/tokenizers so don't have to create by ourselves

Token vocab is what the LLM is really trained on