Report for the assignment 2 - Tokenizer analysis for compression ratio

Problem statement

One primary task of the tokenizer is to compress the compress and encode it into a smaller tokens that can represent the entire corpus in a reasonable way. This encoding of the corpus into sub-words is known as tokenization. After the subwords are found, they are converted into numerical form, and is known as token ids. The objective of this report is to analyze the ratio between the length of the corpus and the length of the vocuabulary and analyze and see the trending of the compression ratios across different tokenizers and different languages.

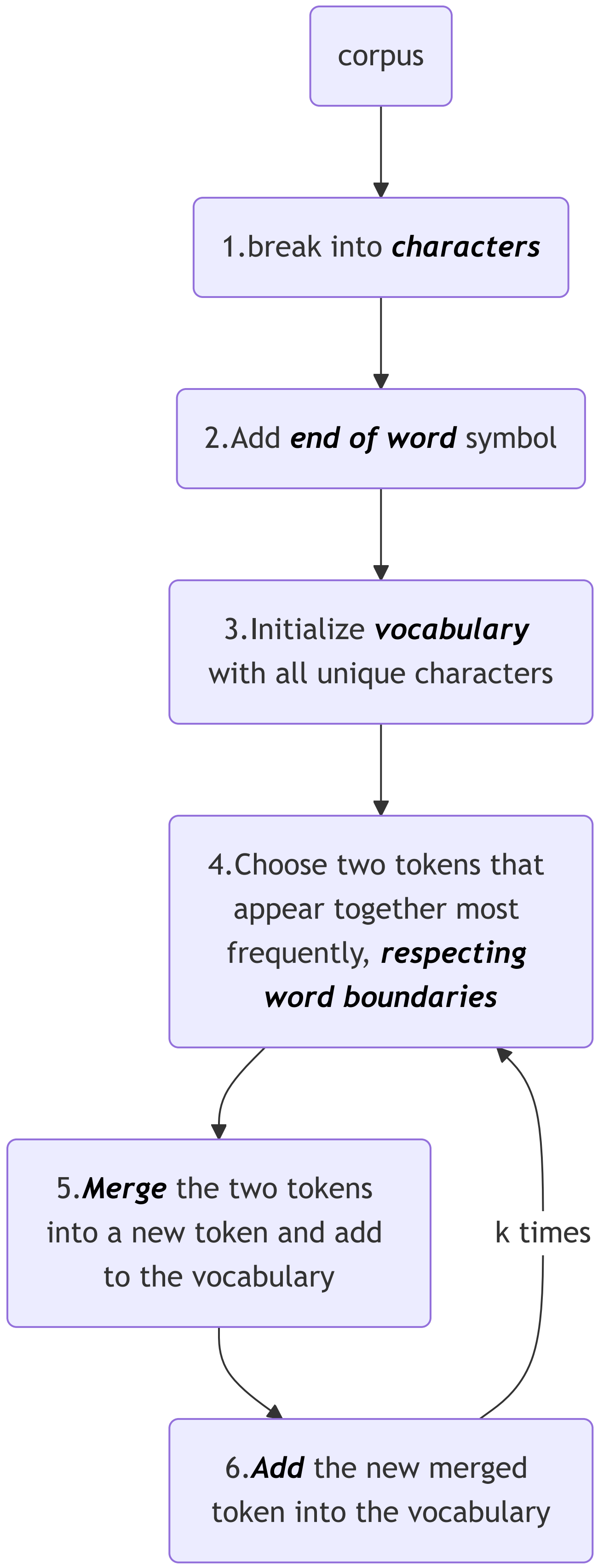
Methodology and approach used

Implementation of BPE from scratch

Algorithm

The tokenization algorithms like BPE (Byte Pair Encoding) identify sub-words which are most prevalent in the corpus. This way most commonly available tokens (units of text) are identified that is extent in the corpus data with which the model is trained. And those are considered separate tokens. Tokens are merged and new bigger tokens are created accordingly based on what is most commonly available.

BPE is a method to build vocabulary by generated tokens from the corpus.
This is used in GPT-2, GPT-4, Llama2, BERT, etc.


corpus
1.break into *characters*
2.Add *end of word* symbol
3.Initialize *vocabulary* with all unique characters
4.Choose two tokens that appear together most frequently, *respecting word boundaries*
5.*Merge* the two tokens into a new token and add to the vocabulary
k times
6.*Add* the new merged token into the vocabulary

Implementation of GPT tokenizers

A python library named `tiktoken` is imported to use the tokenizers in the GPT models - `"gpt2", "gpt-3.5-turbo", "gpt-4"`

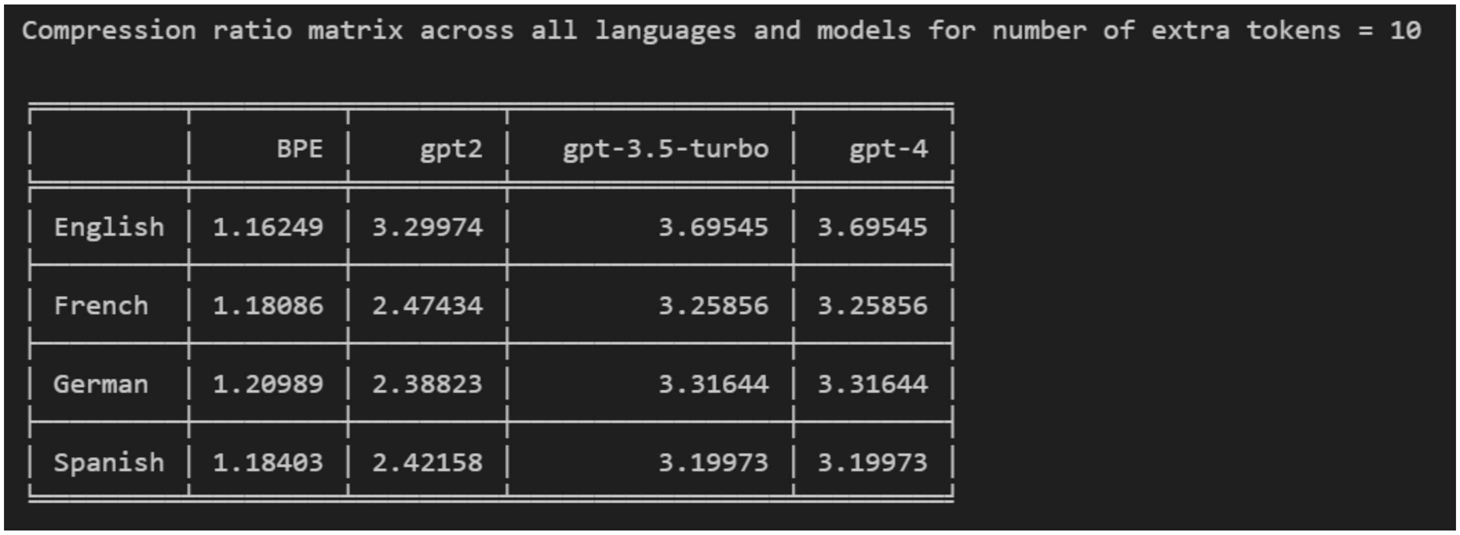
```
import tiktoken
tokenizer = tiktoken.encoding_for_model("gpt2")
token_ids = tokenizer.encode(text)
```

Utilities developed

1. `get_file_text(file_path)` : get the texts in a corpus file.
2. `get_character_tokens(file_path)` : divide the corpus text into characters.
3. `get_stats()` : generates a dictionary of pairs of two adjacent tokens and count of those tokens.
4. `merge()` and `merge_token_ids()` : merge the two tokens which are together for most number of times.
5. `print_compression_ratio()` : print the compression rations and the length of corpus and the vocabulary ids
6. `draw_bar_graph` : draw bar graph for the compression ratios across the different tokenizers and languages.
7. `get_token_ids` : convert character tokens into
8. `compute_comp_ratio_matrix()` : compute the matrix for the compression ration across different languages and tokenization strategies.
9. `visualize_comp_ratios_matrix()` : draw the bar graphs for the different comparisons of the compression ratios and the tokenization schemes.
10. `print_comp_ratios_matrix()` : print the table of compression ratios in tabular format.
11. `heatmap()` : draw a heatmap for the compression ratios across different tokenization schemes.

Analysis and findings

Compression ratios across tokenizers and languages

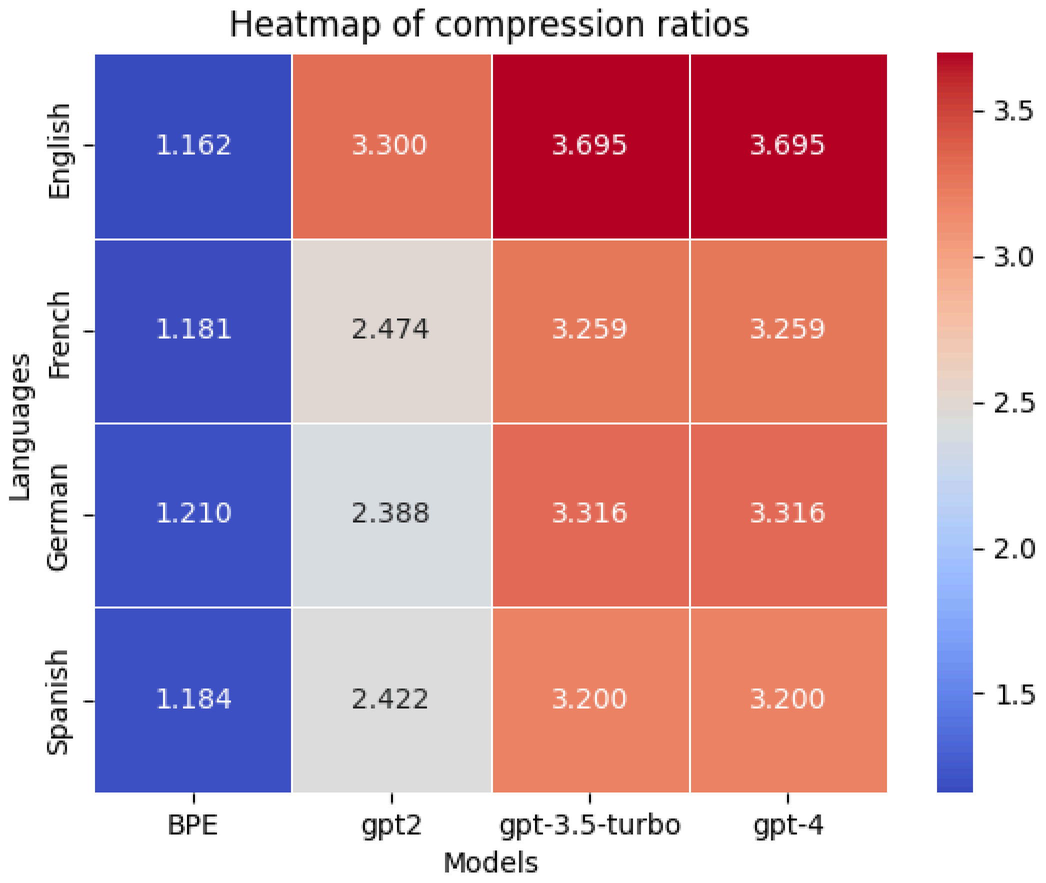


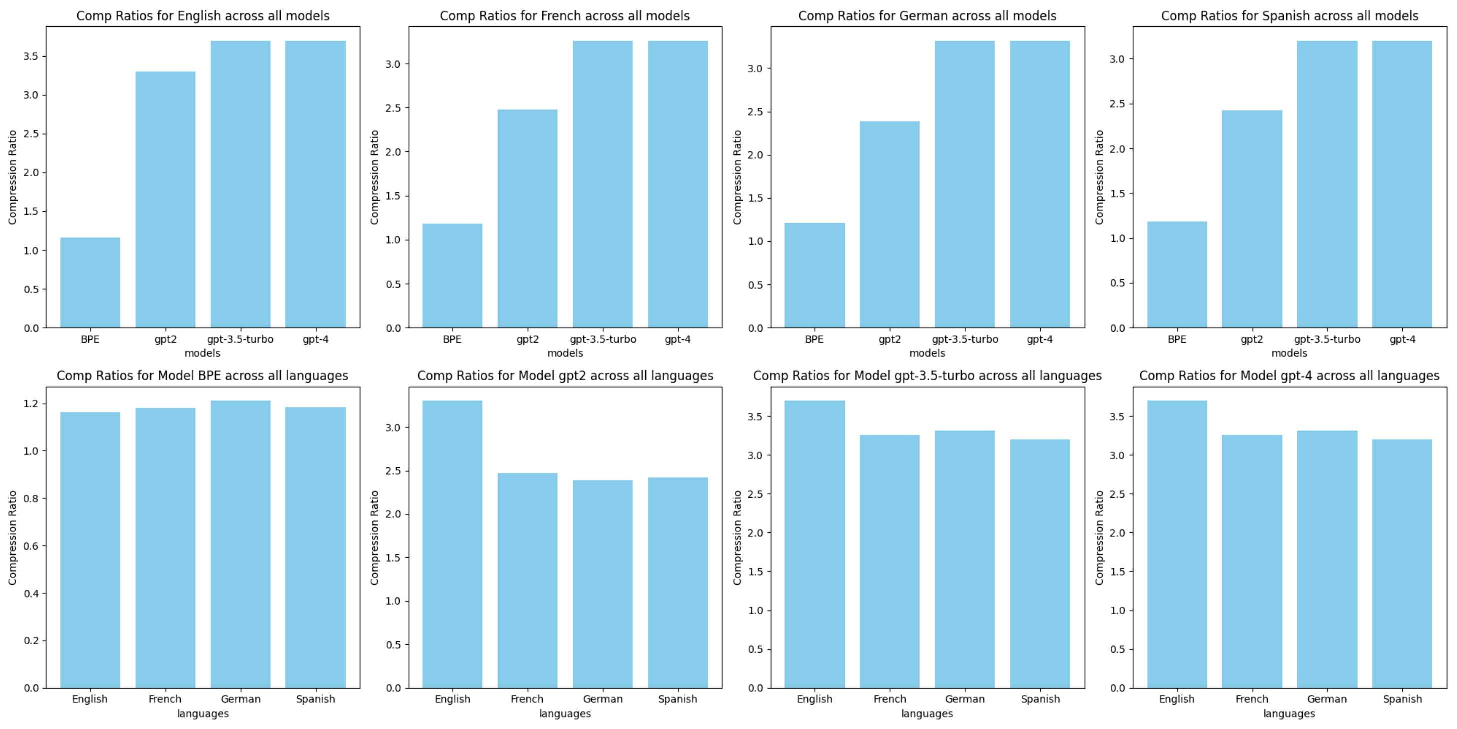
```
Compression ratio matrix across all languages and models for number of extra tokens = 10

╒═════════╤═════════╤═════════╤═══════════════╤═════════╕
│         │     BPE │    gpt2 │   gpt-3.5-turbo │   gpt-4 │
╞═════════╪═════════╪═════════╪═══════════════╪═════════╡
│ English │ 1.16249 │ 3.29974 │       3.69545 │ 3.69545 │
├─────────┼─────────┼─────────┼───────────────┼─────────┤
│ French  │ 1.18086 │ 2.47434 │       3.25856 │ 3.25856 │
├─────────┼─────────┼─────────┼───────────────┼─────────┤
│ German  │ 1.20989 │ 2.38823 │       3.31644 │ 3.31644 │
├─────────┼─────────┼─────────┼───────────────┼─────────┤
│ Spanish │ 1.18403 │ 2.42158 │       3.19973 │ 3.19973 │
╘═════════╧═════════╧═════════╧═══════════════╧═════════╛
```

1. It was found that there are no difference in the compression ratio across GPT 3.5 and GPT 4
2. Compression ratio of English is higher than other languages in the order of 1/10th.
3. Languages in the order of compression ratio from highest to lowest is as follows - English >> German >> French >> Spanish.

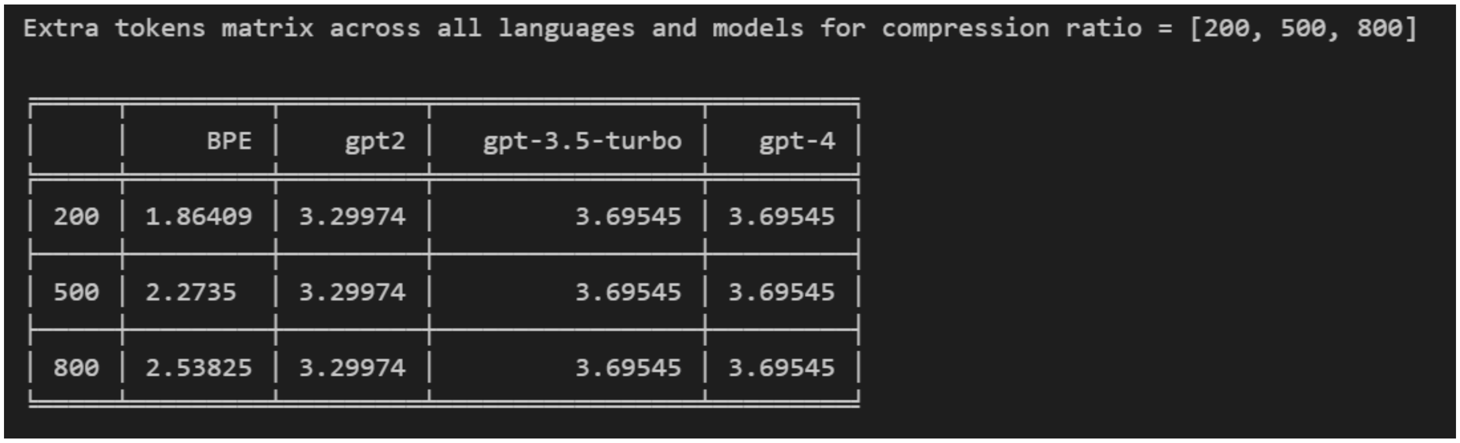
Comparison of the compression ratios across languages and tokenization algorithms





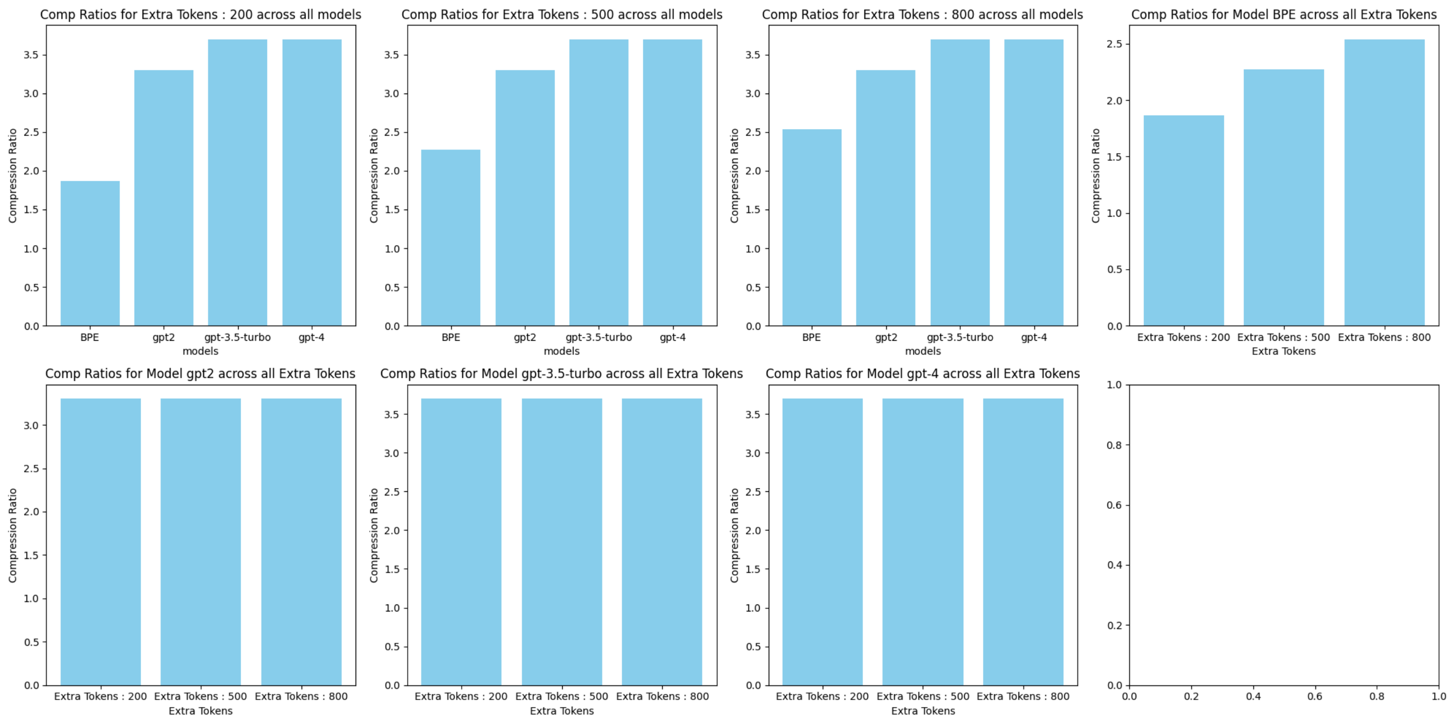
Comparison of compression ratios based on extra tokens being used for different number of merges

Taking merges / extra tokens of 200, 500 and 800



```
Extra tokens matrix across all languages and models for compression ratio = [200, 500, 800]

╒═════╤═════════╤═════════╤═════════════════╤═════════╕
│     │     BPE │    gpt2 │   gpt-3.5-turbo │   gpt-4 │
╞═════╪═════════╪═════════╪═════════════════╪═════════╡
│ 200 │ 1.86409 │ 3.29974 │         3.69545 │ 3.69545 │
├─────┼─────────┼─────────┼─────────────────┼─────────┤
│ 500 │ 2.2735  │ 3.29974 │         3.69545 │ 3.69545 │
├─────┼─────────┼─────────┼─────────────────┼─────────┤
│ 800 │ 2.53825 │ 3.29974 │         3.69545 │ 3.69545 │
╘═════╧═════════╧═════════╧═════════════════╧═════════╛
```



1. After 200 merges GPT tokenizers do not show any improvements on compression ratio!
2. The compression ratio of BPE is typically lower than GPT tokenizers.
3. With increase in number of extra tokens the compression ratio typically does not change for a given tokenizer.

Code base

Assignment code base