In [1]:
```
# Notes:
```

a) Tokenization: the process of breaking down a piece of texts (sentence, paragraph, document) into a bunch of discrete components 'tokens' using a tokenizer algorithm. We can split text sequences into individual characters, text sequences to individual words/spaces, etc.

b) We can decompose rare (lower frquency) words into meaningful sub words. Tokens allow for efficient processing and reduced complexity, improved understanding and capturing nuances, enhanced prediction capabilities, flexibility in handling unfamiliar words.

c) 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. Each character can be represented by 1-4 bytes, and ASCII characters are encoded using a single byte allowing for smooth transition.

d) 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) 2) Find the most frequent occurring pair of symbols from the current vocabulary 3)Add this to the vocabulary and size of vocabulary increases by one 4) Repeat steps (2) and (3) till the defined number of tokens are built or no new combination of symbols exist with required frequency

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

Step 1: Initial Vocabulary 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)

Step 2: Get the most frequent pair Most frequent: ("u", "g") -> "ug"

Step 3: New Vocabulary New Corpus ("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"]

Step 4: Repeat Step 2 and 3 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" 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"]

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" New Corpus ("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5) New Vocabulary ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

In [3]:
```
# Code:
```

In [8]:
```
'Aikyam Lab (Sanskrit: ऐक्यम्; meaning oneness)!'
```

Out[8]:
```
'Aikyam Lab (Sanskrit: ऐक्यम्; meaning oneness)!'
```

In [10]:
```
ord('ऐ')   # takes a single Unicode character as input and returns
           # its corresponding integer Unicode code point
```

Out[10]:
```
2320
```

In [13]:
```
# ord("Aikyam Lab (Sanskrit: ऐक्यम्; meaning oneness)!")
# leads to an error - can only be used to find coding of single character.
# Instead do the following:
```

In [14]:
```
[ord(s) for s in "Aikyam Lab (Sanskrit: ऐक्यम्; meaning oneness)!"]
```

Out[14]:
```
[65,
 105,
 107,
 121,
 97,
 109,
 32,
 76,
 97,
 98,
 32,
 40,
 83,
 97,
 110,
 115,
 107,
 114,
 105,
 116,
 58,
 32,
 2320,
 2325,
 2381,
 2351,
 2350,
 2381,
 59,
 32,
 109,
 101,
 97,
 110,
 105,
 110,
 103,
 32,
 111,
 110,
 101,
 110,
 101,
 115,
 115,
 41,
 33]
```

In [15]:
```
"Aikyam Lab (Sanskrit: ऐक्यम्; meaning oneness)!".encode('utf-8')
```

Out[15]:
```
b'Aikyam Lab (Sanskrit: \xe0\xa4\x90\xe0\xa4\x95\xe0\xa5\x8d\xe0\xa4\xaf\xe0\xa4\xae\xe0\xa5\x8d; meaning oneness)!'
```

In [18]:
```
## Let's train a Character-Level tokenizer for the Harry Potter dataset!
## (See associated pycharm doc)
```

In [ ]: