Tokenization

1. Tokenizing by words
2. Tokenizing by sentence

In [53]:
```
from nltk.tokenize import sent_tokenize, word_tokenize
import string
#nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

In [54]:
```
# Import text that has to be tokenized

example_text = """ChatGPT uses deep learning, a subset of machine learning
to produce humanlike text through transformer neural networks.
The transformer predicts text, including the next word, sentence or paragraph, ba
Training begins with generic data, then moves to more tailored data for a specifi
ChatGPT was trained with online text to learn the human language, and then it use
Human trainers provide conversations and rank the responses. These reward models
To keep training the chatbot, users can upvote or downvote its response by clicki
Users can also provide additional written feedback to improve and fine-tune futur
```

In [55]:
```
# use sent_tkenize to split text into sentences
# text.lower converts text into lower case
sent_tokenize(example_text.lower())
```

Out[55]:
```
['chatgpt uses deep learning, a subset of machine learning \nto produce humanlik
e text through transformer neural networks.',
 "the transformer predicts text, including the next word, sentence or paragraph,
based on its training data's typical sequence.",
 'training begins with generic data, then moves to more tailored data for a spec
ific task.',
 'chatgpt was trained with online text to learn the human language, and then it
used transcripts to learn the basics of conversations.',
 'human trainers provide conversations and rank the responses.',
 'these reward models help determine the best answers.',
 'to keep training the chatbot, users can upvote or downvote its response by cli
cking on thumbs up or thumbs down icons beside the answer.',
 'users can also provide additional written feedback to improve and fine-tune fu
ture dialogue.']
```

In [56]:
```
# use word_tokenize to split text into words
word_tokens = word_tokenize(example_text.lower())
print(word_tokens)
```

```
['chatgpt', 'uses', 'deep', 'learning', ',', 'a', 'subset', 'of', 'machine', 'le
arning', 'to', 'produce', 'humanlike', 'text', 'through', 'transformer', 'neural
', 'networks', '.', 'the', 'transformer', 'predicts', 'text', ',', 'including',
'the', 'next', 'word', ',', 'sentence', 'or', 'paragraph', ',', 'based', 'on', '
its', 'training', 'data', "'s", 'typical', 'sequence', '.', 'training', 'begins'
, 'with', 'generic', 'data', ',', 'then', 'moves', 'to', 'more', 'tailored', 'da
ta', 'for', 'a', 'specific', 'task', '.', 'chatgpt', 'was', 'trained', 'with', '
online', 'text', 'to', 'learn', 'the', 'human', 'language', ',', 'and', 'then',
'it', 'used', 'transcripts', 'to', 'learn', 'the', 'basics', 'of', 'conversation
s', '.', 'human', 'trainers', 'provide', 'conversations', 'and', 'rank', 'the',
'responses', '.', 'these', 'reward', 'models', 'help', 'determine', 'the', 'best
', 'answers', '.', 'to', 'keep', 'training', 'the', 'chatbot', ',', 'users', 'ca
n', 'upvote', 'or', 'downvote', 'its', 'response', 'by', 'clicking', 'on', 'thum
bs', 'up', 'or', 'thumbs', 'down', 'icons', 'beside', 'the', 'answer', '.', 'use
rs', 'can', 'also', 'provide', 'additional', 'written', 'feedback', 'to', 'impro
ve', 'and', 'fine-tune', 'future', 'dialogue', '.']
```

Filtering the stop words

In [57]:
```
# define stopwords
stop_words = set(stopwords.words("english"))
```

In [58]:
```
# filtered list contains words that are not in stop_words
filtered_list = []
for word in word_tokens:
  if word not in stop_words:
    filtered_list.append(word)
```

In [59]:
```
print(filtered_list)
```

```
['chatgpt', 'uses', 'deep', 'learning', ',', 'subset', 'machine', 'learning', 'p
roduce', 'humanlike', 'text', 'transformer', 'neural', 'networks', '.', 'transfo
rmer', 'predicts', 'text', ',', 'including', 'next', 'word', ',', 'sentence', 'p
aragraph', ',', 'based', 'training', 'data', "'s", 'typical', 'sequence', '.', '
training', 'begins', 'generic', 'data', ',', 'moves', 'tailored', 'data', 'speci
fic', 'task', '.', 'chatgpt', 'trained', 'online', 'text', 'learn', 'human', 'la
nguage', ',', 'used', 'transcripts', 'learn', 'basics', 'conversations', '.', 'h
uman', 'trainers', 'provide', 'conversations', 'rank', 'responses', '.', 'reward
', 'models', 'help', 'determine', 'best', 'answers', '.', 'keep', 'training', 'c
hatbot', ',', 'users', 'upvote', 'downvote', 'response', 'clicking', 'thumbs', '
thumbs', 'icons', 'beside', 'answer', '.', 'users', 'also', 'provide', 'addition
al', 'written', 'feedback', 'improve', 'fine-tune', 'future', 'dialogue', '.']
```

Remove punctuation from filtered_list

In [60]:
```
# removing punctuations
filtered_list = list(filter(lambda token: token
                            not in string.punctuation,
                            filtered_list))
```

Lemmatization

In [61]:
```
lemmatizer = WordNetLemmatizer()
```

In [62]:
```
lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_list]
print(lemmatized_words)
```

```
['chatgpt', 'us', 'deep', 'learning', 'subset', 'machine', 'learning', 'produce'
, 'humanlike', 'text', 'transformer', 'neural', 'network', 'transformer', 'predi
cts', 'text', 'including', 'next', 'word', 'sentence', 'paragraph', 'based', 'tr
aining', 'data', "'s", 'typical', 'sequence', 'training', 'begin', 'generic', 'd
ata', 'move', 'tailored', 'data', 'specific', 'task', 'chatgpt', 'trained', 'onl
ine', 'text', 'learn', 'human', 'language', 'used', 'transcript', 'learn', 'basi
c', 'conversation', 'human', 'trainer', 'provide', 'conversation', 'rank', 'resp
onse', 'reward', 'model', 'help', 'determine', 'best', 'answer', 'keep', 'traini
ng', 'chatbot', 'user', 'upvote', 'downvote', 'response', 'clicking', 'thumb', '
thumb', 'icon', 'beside', 'answer', 'user', 'also', 'provide', 'additional', 'wr
itten', 'feedback', 'improve', 'fine-tune', 'future', 'dialogue']
```

POS (Parts of Speech) Tagging

In [63]:
```
nltk.pos_tag(lemmatized_words)
```

Out[63]:
```
[('chatgpt', 'VB'),
 ('us', 'PRP'),
 ('deep', 'JJ'),
 ('learning', 'NN'),
 ('subset', 'VBN'),
 ('machine', 'NN'),
 ('learning', 'VBG'),
 ('produce', 'VBP'),
 ('humanlike', 'JJ'),
 ('text', 'NN'),
 ('transformer', 'NN'),
 ('neural', 'JJ'),
 ('network', 'NN'),
 ('transformer', 'NN'),
 ('predicts', 'NNS'),
 ('text', 'IN'),
 ('including', 'VBG'),
 ('next', 'JJ'),
 ('word', 'NN'),
 ('sentence', 'NN'),
 ('paragraph', 'NN'),
 ('based', 'VBN'),
 ('training', 'VBG'),
 ('data', 'NNS'),
 ("'s", 'POS'),
 ('typical', 'JJ'),
 ('sequence', 'NN'),
 ('training', 'NN'),
 ('begin', 'VB'),
 ('generic', 'JJ'),
 ('data', 'NNS'),
 ('move', 'NN'),
 ('tailored', 'VBN'),
 ('data', 'NNS'),
 ('specific', 'JJ'),
 ('task', 'NN'),
 ('chatgpt', 'NN'),
 ('trained', 'VBD'),
 ('online', 'JJ'),
 ('text', 'NN'),
 ('learn', 'NN'),
```

```
('human', 'JJ'),
('language', 'NN'),
('used', 'VBN'),
('transcript', 'NN'),
('learn', 'NN'),
('basic', 'JJ'),
('conversation', 'NN'),
('human', 'JJ'),
('trainer', 'NN'),
('provide', 'VBP'),
('conversation', 'NN'),
('rank', 'NN'),
('response', 'NN'),
('reward', 'NN'),
('model', 'NN'),
('help', 'NN'),
('determine', 'VB'),
('best', 'JJS'),
('answer', 'NN'),
('keep', 'VB'),
('training', 'NN'),
('chatbot', 'NN'),
('user', 'JJ'),
('upvote', 'JJ'),
('downvote', 'NN'),
('response', 'NN'),
('clicking', 'VBG'),
('thumb', 'JJ'),
('thumb', 'JJ'),
('icon', 'NN'),
('beside', 'NN'),
('answer', 'IN'),
('user', 'NN'),
('also', 'RB'),
('provide', 'VBP'),
('additional', 'JJ'),
('written', 'VBN'),
('feedback', 'JJ'),
('improve', 'VB'),
('fine-tune', 'JJ'),
('future', 'NN'),
('dialogue', 'NN')]
```

In [ ]: