Integer Encoding

- **Definition:** The process of assigning a unique integer index to each distinct word in a corpus.
- **Vocabulary (word set):** The set of all unique words in the corpus.
- **Example:** If you have a corpus with 5,000 unique words, you would assign integers from 1 to 5,000 to each word.
- book: 150
- dog: 171
- love: 192
- books: 212
-

Process of assigning integer indices to words

- **Tokenization:** Split the text into individual words (tokens).
- **Vocabulary Creation:** Create a set of all unique words in the corpus.
- **Frequency Calculation:** Count the frequency of each word in the corpus.
- **Sorting:** Sort the words in the vocabulary by frequency, typically in descending order (most frequent first).
- **Index Assignment:** Assign a unique integer index to each word, starting from a chosen index (usually 0 or 1) and incrementing for each subsequent word in the sorted vocabulary.
- **OOV Handling:** Assign a specific index to out-of-vocabulary (OOV) words (words not in the vocabulary).
- **Mapping Storage:** Store the mapping between words and their assigned integer indices (e.g., in a dictionary).

Step 1:

```
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords


raw_text = "A barber is a person. a barber is good person. a
barber is huge person. he Knew A Secret! The Secret He Kept
is huge secret. Huge secret. His barber kept his word. a
barber kept his word. His barber kept his secret. But
keeping and keeping such a huge secret to himself was
driving the barber crazy. the barber went up a huge
mountain."
```

Step 2: sentence tokenization

```
sentences = sent_tokenize(raw_text)
print(sentences)
```

```
['A barber is a person.', 'a barber is good person.', 'a
barber is huge person.', 'he Knew A Secret!', 'The Secret He
Kept is huge secret.', 'Huge secret.', 'His barber kept his
word.', 'a barber kept his word.', 'His barber kept his
secret.', 'But keeping and keeping such a huge secret to
himself was driving the barber crazy.', 'the barber went up
a huge mountain.']
```

Step 3: Cleansing and Nomalization

```
vocab = {}
preprocessed_sentences = []
stop_words = set(stopwords.words('english'))
```

1 vocab = {}: An empty dictionary called vocab is created. This dictionary will store the words in the corpus as keys and their frequencies as values.
2 preprocessed_sentences = []: An empty list called preprocessed_sentences is created. This list will store the preprocessed sentences, where each sentence is a list of tokens (words).
3 stop_words = set(stopwords.words('english')): A set of English stop words is created using nltk.corpus.stopwords. Converting it to a set makes checking for stop words more efficient.

Looping through Sentences:

1 for sentence in sentences:: The code iterates through each sentence in a list called sentences (presumably, this list contains the original sentences from your text data).

Processing each sentence:

1 tokenized_sentence = word_tokenize(sentence): The current sentence is tokenized into individual words using the word_tokenize function (likely from NLTK). The result is stored in the tokenized_sentence list.
2 result = []: An empty list called result is created. This list will store the words that remain after preprocessing the current sentence.
3 for word in tokenized_sentence:: The code iterates through each word (token) in the tokenized_sentence list.

Word-level processing

1 word = word.lower(): The current word is converted to lowercase. This helps to reduce the vocabulary size by treating words like "The" and "the" as the same.
2 if word not in stop_words:: The code checks if the word is a stop word (i.e., if it's present in the stop_words set). If the word is a stop word, it's skipped (not added to the result list).
3 if len(word) > 2:: The code checks if the length of the word is greater than 2 characters. If the word has 2 or fewer characters, it's skipped. This helps to remove short, potentially meaningless words or punctuation.
4 result.append(word): If the word passes both the stop word check and the length check, it's appended to the result list.
5 if word not in vocab:: The code checks if the word is already in the vocab dictionary. If it's not in the dictionary, it means this is the first time the word has been encountered.
6 vocab[word] = 0: If the word is not in the vocab dictionary, it's added to the dictionary with an initial frequency count of 0.
7 vocab[word] += 1: The frequency count for the current word in the vocab dictionary is incremented by 1.

```
for sentence in sentences:
tokenized_sentence = word_tokenize(sentence)
result = []

for word in tokenized_sentence:
word = word.lower()
if word not in stop_words:
if len(word) > 2:
result.append(word)
if word not in vocab:
vocab[word] = 0
vocab[word] += 1
preprocessed_sentences.append(result)
print(preprocessed_sentences)
```

1 Tokenizes sentences into words.
2 Converts words to lowercase.
3 Removes stop words.
4 Removes words with a length of 2 or less.
5 Creates a vocabulary and counts the frequency of each word.
6 Stores the preprocessed sentences in a list.

Output

```
[['barber', 'person'], ['barber', 'good', 'person'], ['barber',
'huge', 'person'], ['knew', 'secret'], ['secret', 'kept', 'huge',
'secret'], ['huge', 'secret'], ['barber', 'kept', 'word'], ['barber',
'kept', 'word'], ['barber', 'kept', 'secret'], ['keeping', 'keeping',
'huge', 'secret', 'driving', 'barber', 'crazy'], ['barber', 'went',
'huge', 'mountain']]
```

Print('word set :' , vocab)

Output

word set : {'barber': 8, 'person': 3, 'good': 1, 'huge': 5, 'knew': 1, 'secret': 6, 'kept': 4, 'word': 2, 'keeping': 2, 'driving': 1, 'crazy': 1, 'went': 1, 'mountain': 1}

```
print(vocab["barber"])
```

Output

8