In [1]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in c:\programdata\anaconda3\lib\site-pack
ages (3.5)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\lib\site-pack
ages (from nltk) (4.50.2)
Requirement already satisfied: joblib in c:\programdata\anaconda3\lib\site-pa
ckages (from nltk) (0.17.0)
Requirement already satisfied: regex in c:\programdata\anaconda3\lib\site-pac
kages (from nltk) (2020.10.15)
Requirement already satisfied: click in c:\programdata\anaconda3\lib\site-pac
kages (from nltk) (7.1.2)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import nltk
nltk.download('all')
```

```
[nltk_data]    | Downloading package conll2002 to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package conll2007 to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]    |   Package conll2007 is already up-to-date!
[nltk_data]    | Downloading package crubadan to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]    |   Package crubadan is already up-to-date!
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]    |   Package dependency_treebank is already up-to-date!
[nltk_data]    | Downloading package dolch to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]    |   Package dolch is already up-to-date!
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]    |   Package europarl_raw is already up-to-date!
[nltk_data]    | Downloading package extended_omw to
[nltk_data]    |     C:\Users\Admin\AppData\Roaming\nltk_data...
```

In [3]:
```
import re

def generate_ngrams(text, n):
    words = re.findall(r'\w+', text.lower())
    ngrams = []

    for i in range(len(words) - n + 1):
        ngram = tuple(words[i:i + n])
        ngrams.append(ngram)

    return ngrams
```

In [4]:
```
text = "This is an example of n-gram generation. We will use it for bigrams an
n = 2
bigrams = generate_ngrams(text, n)
print("Bigrams:", bigrams)
```

```
Bigrams: [('this', 'is'), ('is', 'an'), ('an', 'example'), ('example', 'of'),
('of', 'n'), ('n', 'gram'), ('gram', 'generation'), ('generation', 'we'), ('w
e', 'will'), ('will', 'use'), ('use', 'it'), ('it', 'for'), ('for', 'bigram
s'), ('bigrams', 'and'), ('and', 'trigrams')]
```

In [5]:
```
unigrams = generate_ngrams(text, 1)
print("Unigrams:", unigrams)
```

```
Unigrams: [('this',), ('is',), ('an',), ('example',), ('of',), ('n',), ('gra
m',), ('generation',), ('we',), ('will',), ('use',), ('it',), ('for',), ('big
rams',), ('and',), ('trigrams',)]
```

In [6]:
```
trigrams = generate_ngrams(text, 3)
print("Trigrams:", trigrams)
```

```
Trigrams: [('this', 'is', 'an'), ('is', 'an', 'example'), ('an', 'example',
'of'), ('example', 'of', 'n'), ('of', 'n', 'gram'), ('n', 'gram', 'generatio
n'), ('gram', 'generation', 'we'), ('generation', 'we', 'will'), ('we', 'wil
l', 'use'), ('will', 'use', 'it'), ('use', 'it', 'for'), ('it', 'for', 'bigra
ms'), ('for', 'bigrams', 'and'), ('bigrams', 'and', 'trigrams')]
```

In [ ]:
```

```