the code performs **text preprocessing and analysis** on BBC News articles using **Natural Language Processing (NLP)** techniques.
It:

1. Fetches news data from an RSS feed,
2. Extracts and cleans the article text,
3. Tokenizes and analyzes words and sentences,
4. Removes stop words,
5. Applies stemming and lemmatization,
6. Generates bigrams and trigrams.

🔹 Importing Libraries

```
import feedparser
import re
from urllib.request import urlopen
import nltk
from bs4 import BeautifulSoup
```

- **feedparser**: reads and parses RSS feeds (used to get news articles).
- **re**: provides regular expressions for text cleaning (not used much here but helpful).
- **urllib.request.urlopen**: opens URLs to fetch web pages.
- **nltk**: Natural Language Toolkit — used for tokenization, tagging, etc.
- **BeautifulSoup**: extracts readable text from HTML web pages.

🔹 Downloading NLTK Resources

```
nltk.download("punkt")
nltk.download("averaged_perceptron_tagger_eng")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("punkt_tab")
```

These lines download essential NLP datasets:

- **punkt** → for sentence and word tokenization.
- **averaged_perceptron_tagger_eng** → for part-of-speech tagging (POS).
- **stopwords** → list of common words like “the”, “is”, etc.
- **wordnet** → for lemmatization (getting base word forms).
- **punkt_tab** → updated tokenizer resource.

🔹 Tokenizers, Stopwords, and Stemmers

```
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import bigrams, trigrams
```

Imports specific NLP tools:

- **sent_tokenize** → splits text into sentences.
- **word_tokenize** → splits text into words.
- **stopwords** → provides English stopwords list.
- **PorterStemmer** → reduces words to root form (e.g., *running* → *run*).
- **WordNetLemmatizer** → returns meaningful base form (e.g., *better* → *good*).
- **bigrams, trigrams** → creates pairs or triples of consecutive words.

🔹 Step 1: Read RSS Feed

```
rss_url = "http://feeds.bbci.co.uk/news/uk/rss.xml"
feed = feedparser.parse(rss_url)

print("Number of articles:", len(feed.entries))
```

- **rss_url** → BBC UK news RSS feed link.
- **feedparser.parse()** → fetches and parses the feed.
- **feed.entries** → list of articles (each entry has title, link, summary, etc.).
- **len(feed.entries)** → number of articles retrieved.

🔹 Step 2: Display First Article Info

```
first_entry = feed.entries[0]
print("\nArticle Title:", first_entry.title)
print("Link:", first_entry.link)
```

- Takes the **first article** and prints its **title** and **link**.

🔹 Step 3: Extract Full Article Text

```
url = first_entry.link
```

```
html = urlopen(url).read().decode("utf-8")
```

- Opens the article link and reads its **HTML source code**.

Then:

```
soup = BeautifulSoup(html, 'html.parser')
paragraphs = soup.find_all('p')
clean_text = ' '.join([p.get_text() for p in paragraphs])
```

- **BeautifulSoup** parses the HTML.
- **find_all('p')** → finds all <p> paragraph elements.
- **get_text()** → extracts only the text from each paragraph.
- **join(...)** → merges all paragraphs into a single string.

```
print("\nOriginal Text (first 500 characters):\n",
clean_text[:500])
```

- Prints only the first 500 characters for preview.

🔹 Step 4: Tokenization

```
sentences = sent_tokenize(clean_text)
words = word_tokenize(clean_text)
```

- **sent_tokenize** → splits the text into sentences.
- **word_tokenize** → splits text into words and punctuation.

```
print("\nNumber of sentences:", len(sentences))
print("Number of words:", len(words))
```

- Shows how many sentences and words were found.

🔹 Step 5: Part-of-Speech (POS) Tagging

```
pos_tags = nltk.pos_tag(words)
print("\nPOS tagging (first 20):", pos_tags[:20])
```

- **nltk.pos_tag(words)** → assigns a grammatical label (noun, verb, adjective, etc.) to each word.
  Example: `[('The', 'DT'), ('cat', 'NN'), ('sat', 'VBD')]`

🔹 Step 6: Stop Words Removal

```
stop_words = set(stopwords.words("english"))
filtered_words = [w for w in words if w.lower() not in
stop_words and w.isalpha()]
```

- **stop_words** → creates a set of common words to ignore.
- **w.isalpha()** → keeps only alphabetic words (removes punctuation and numbers).
- **filtered_words** → the cleaned list of meaningful words.

```
print("\nAfter Stop Words Removal (first 20):",
filtered_words[:20])
```

🔹 Step 7: Stemming & Lemmatization

```
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

stems = [stemmer.stem(w) for w in filtered_words[:20]]
lemmas = [lemmatizer.lemmatize(w) for w in filtered_words[:20]]
```

- **Stemming** → cuts words to their root form (e.g., *connected* → *connect*).
- **Lemmatization** → finds base dictionary form using grammar (e.g., *better* → *good*).
- Applies both methods on the first 20 filtered words.

```
print("\nStemming (first 20):", stems)
print("Lemmatization (first 20):", lemmas)
```

🔹 Step 8: Generate N-grams

```
print("\nBigrams (first 10):",
list(bigrams(filtered_words))[:10])
print("Trigrams (first 10):",
list(trigrams(filtered_words))[:10])
```

- **Bigrams** → pairs of consecutive words (e.g., `("mental", "health")`).
- **Trigrams** → triples of consecutive words (e.g., `("machine", "learning", "model")`).
- Useful for finding common word patterns.

✅ **In summary:**

Step	Process	Purpose
1	Read RSS Feed	Fetch BBC articles
2	Extract HTML Text	Get article content
3	Tokenization	Split into sentences/words
4	POS Tagging	Identify grammatical roles
5	Stopword Removal	Keep meaningful words
6	Stemming & Lemmatization	Normalize word forms
7	N-grams	Detect word pair/triple patterns

Output

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng
to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already
up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
Number of articles: 28

Article Title: New digital ID will be mandatory to work in the
UK, Starmer says
Link:
https://www.bbc.com/news/articles/cn832y43ql5o?at_medium=RSS&at_
campaign=rss

Original Text (first 500 characters):
 Digital ID will be mandatory in order to work in the UK, as
part of plans to tackle illegal migration. Sir Keir Starmer said
the new digital ID scheme would make it tougher to work in the
UK illegally and offer "countless benefits" to citizens, while
his senior minister Darren Jones said it could be "the bedrock
of the modern state".  However, opposition parties argued the
proposals would not stop people crossing the Channel in small
boats. The prime minister set out his plans in a broader speec

Number of sentences: 60
Number of words: 1506

POS tagging (first 20): [('Digital', 'NNP'), ('ID', 'NNP'),
('will', 'MD'), ('be', 'VB'), ('mandatory', 'JJ'), ('in', 'IN'),
('order', 'NN'), ('to', 'TO'), ('work', 'VB'), ('in', 'IN'),
('the', 'DT'), ('UK', 'NNP'), (',', ','), ('as', 'IN'), ('part',
```

```
'NN'), ('of', 'IN'), ('plans', 'NNS'), ('to', 'TO'), ('tackle',
'VB'), ('illegal', 'JJ')]

After Stop Words Removal (first 20): ['Digital', 'ID',
'mandatory', 'order', 'work', 'UK', 'part', 'plans', 'tackle',
'illegal', 'migration', 'Sir', 'Keir', 'Starmer', 'said', 'new',
'digital', 'ID', 'scheme', 'would']

Stemming (first 20): ['digit', 'id', 'mandatori', 'order',
'work', 'uk', 'part', 'plan', 'tackl', 'illeg', 'migrat', 'sir',
'keir', 'starmer', 'said', 'new', 'digit', 'id', 'scheme',
'would']
Lemmatization (first 20): ['Digital', 'ID', 'mandatory',
'order', 'work', 'UK', 'part', 'plan', 'tackle', 'illegal',
'migration', 'Sir', 'Keir', 'Starmer', 'said', 'new', 'digital',
'ID', 'scheme', 'would']

Bigrams (first 10): [('Digital', 'ID'), ('ID', 'mandatory'),
('mandatory', 'order'), ('order', 'work'), ('work', 'UK'),
('UK', 'part'), ('part', 'plans'), ('plans', 'tackle'),
('tackle', 'illegal'), ('illegal', 'migration')]
Trigrams (first 10): [('Digital', 'ID', 'mandatory'), ('ID',
'mandatory', 'order'), ('mandatory', 'order', 'work'), ('order',
'work', 'UK'), ('work', 'UK', 'part'), ('UK', 'part', 'plans'),
('part', 'plans', 'tackle'), ('plans', 'tackle', 'illegal'),
('tackle', 'illegal', 'migration'), ('illegal', 'migration',
'Sir')]
```

```
[ ]
```