In [4]:
```python
# 1. Importing the Necessary Modules

import numpy as np
import nltk
from collections import Counter, defaultdict
from nltk.corpus import stopwords as nltk_stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

In [5]:
```python
# 2. Download resources

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-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!
```

Out[5]: True

In [6]:
```python
# 3. Sample Document

text = """
I am learning natural language processing
Natural language processing is the important module of subject artificial intell
This domain has seen many recent advancements in terms of its execution
"""
```

In [7]:
```python
# 4. Tokenization

tokens = word_tokenize(text.lower())
stop_words = set(nltk_stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

In [8]:
```python
# 5. Preprocessing

filtered_tokens = [lemmatizer.lemmatize(word) for word in tokens if word.isalnum
```

In [9]:
```python
# 6. Generate bigrams

def generate_n_grams(tokens, n):
    return [tuple(tokens[i:i + n]) for i in range(len(tokens) - n + 1)]

bigrams = generate_n_grams(filtered_tokens, 2)
```

In [10]:
```python
# 7. Train model (bigram)

def train_grams(n_grams):
    model = defaultdict(Counter)
    for ngram in n_grams:
```

```
        prefix = ngram[:-1]
        next_word = ngram[-1]
        model[prefix][next_word] += 1
    return model

model = train_grams(bigrams)
```

In [11]:
```
# 8. Predict with probabilities

def predict_next_word(model, prefix_words):
    prefix = tuple(prefix_words.split())
    if prefix in model:
        total = sum(model[prefix].values())
        return [(word, round(count / total, 3)) for word, count in model[prefix]
    else:
        return "No Prediction"
```

In [12]:
```
# 9. Predictions for various seed phrases

seeds = ["natural", "language", "artificial", "processing", "subject"]
print("\nPredictions:")
print(f"{'Seed Phrase':<15}{'Predicted Word':<20}{'Probability'}")
print("-" * 50)

for seed in seeds:
    prediction = predict_next_word(model, seed)
    if prediction != "No Prediction":
        for word, prob in prediction:
            print(f"{seed:<15}{word:<20}{prob}")
    else:
        print(f"{seed:<15}{'No Prediction':<20}{'-'}")
```

```
Predictions:
Seed Phrase    Predicted Word      Probability
--------------------------------------------------
natural        language            1.0
language       processing          1.0
artificial     intelligence        1.0
processing     natural             0.5
processing     important           0.5
subject        artificial          1.0
```