

---

## Assignment – 1

### N-gram Language Model

---

GitHub link: <https://github.com/SheikhAsiyaNoor/N-gram-Language-Model/> | Roll No.: 423193 | Sheikh Asiya Noor

Date: 03/01/2026

## Objective

To implement a n-gram language model where  $n = 5$  which generates text based on the author's style - Jane Austen; to predict the following words in a sentence from given context words.

## Approach

The programming assignment is implemented by using the built-in libraries re, random and collections (defaultdict, Counter). Uses probability  $P(w_i | w(i-1), w(i-2), w(i-3), w(i-4))$  or simply

$P(\text{target} | \text{context}) = \text{count}(\text{target context}) / \text{count}(\text{context})$ ,

here the context is 4 words for a 5-gram model; to predict the next word. A function 'tokenize' to convert the book text into tokens while maintaining the author's style so that the output sentence is generated in author's style only. A five\_gram\_model function to hold the counts of each 5-gram and then to calculate the prob using the counts stored.

## Code

```
# a five gram language model to predict the following words in a sentence from given context words
```

```
# uses probability  $P(w_i | w(i-1), w(i-2), w(i-3), w(i-4))$  to predict the next word
```

```
# function 'tokenize' to convert the book text into tokens while maintaining the author's style so that the output sentence is generated in author's style only
```

```
# five_gram_model to hold the counts of each 5-gram and then to calculate the prob using the counts stored
```

```
import re
```

```
from collections import defaultdict, Counter
```

```
import random
```

```

def tokenize(text):
    text = re.sub(r'[""]', "'", text)
    text = re.sub(r"[']", '"', text)
    text = re.sub(r'...', '...', text)
    text = re.sub(r'--', '-', text)
    text = re.findall(
        r"\b[a-zA-Z]+(?:[-'][a-zA-Z]+)*\b|[\.!?]+|[,;:][['\"]",
        text
    )
    return [t for t in text] # Consistent Lowercasing for better matching

```

```

global five_gram_counts

```

```

five_gram_counts = defaultdict(Counter)

```

```

def five_gram_model(tokens):

```

```

    for i in range(len(tokens)):
        for n in range(1, 5): # context lengths 1 to 4
            if i >= n:
                context = tuple(tokens[i-n:i])
                target_word = tokens[i]
                five_gram_counts[context][target_word] += 1

```

```

    prob = defaultdict(Counter)
    for c, t in five_gram_counts.items():
        _sum = sum(t.values())
        for w, v in t.items():
            prob[c][w] = v / _sum

```

```

    return five_gram_counts, prob

```

```

def generate_sentence(prob, context, max_length=20):

```

```

    input_context = tokenize(context)
    result = input_context.copy()

```

```

for _ in range(max_length):
    next_word = None
    for n in range(min(4, len(result)), 0, -1):
        ctx = tuple(result[-n:])
        if ctx in prob:
            foll_words = list(prob[ctx].keys())
            foll_word_prob = list(prob[ctx].values())
            next_word = random.choices(foll_words, weights = foll_word_prob)[0]
            break

    if next_word:
        result.append(next_word)
    else:
        break

return ' '.join(result).capitalize()+"..."

```

```
try:
```

```
    with open("book/jane austen - pride and prejudice.txt", "r", encoding="utf-8") as
f:
```

```
    text = f.read()
```

```
tokens = tokenize(text)
```

```
five_gram_counts, prob = five_gram_model(tokens)
```

```
c = 0
```

```
while c<3:
```

```
    input_context = input("Enter the context: ")
```

```
    if len(input_context.split()) < 4:
```

```
        print('Please enter at least 4 context words.')
```

```
        continue
```

```
    print(generate_sentence(prob, input_context))
```

```
    c += 1
```

```
except FileNotFoundError:  
    print("File not found")
```

## Output

Input1: *"Lady Lucas could not..."*

```
PS C:\Users\sheik\OneDrive\Desktop\3RD_YEAR\3RD YEAR (II SEM)\DE1 Natural Language Processing\Google Classroom\five_gram_LM> python .\five_gram_lm.py  
Enter the context: Lady Lucas could not  
Lady lucas could not be insensible of triumph on being able to retort on mrs . bennet the comfort of having a daughter well...
```

Input2: *"This was not very..."*

```
Enter the context: This was not very  
This was not very consoling to mrs . bennet . they had several children . the eldest of them , a sensible , intelligent...
```

Input3: *"If it was not..."*

```
Enter the context: If it was not  
If it was not for the entail , i should not pay him half so much deference . i declare i do not know...
```