1. *Two Best Paths*: Devise a two-pass algorithm to find the *two most likely paths* through the HMM trellis. Specifically, let the first pass be the Viterbi algorithm, and devise a second pass over the same trellis to find the second most likely path *knowing* the most likely path. Assume if needed that there are no cycles made up entirely of null arcs.

   (a) Describe your second pass in the same way the Viterbi algorithm is described in Chapter 2 (§2.4, page 22).

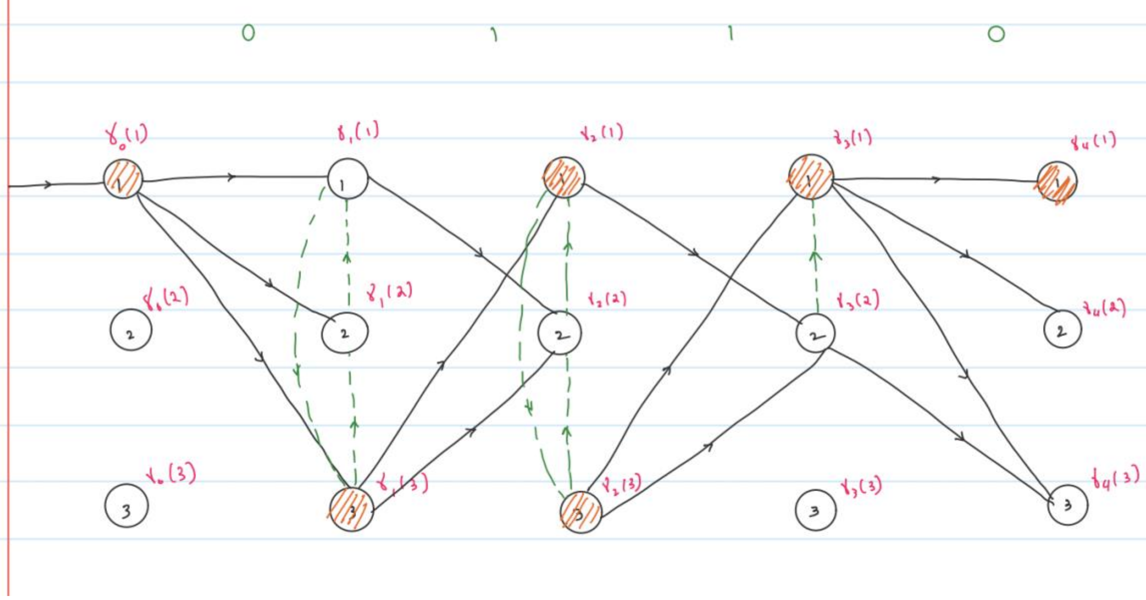
   (b) Redraw the trellis of Problem 1(e) in Homework #2, run your algorithm on it by hand, and color the two most likely paths.

   Hint: The answer expected here is different from the N-best algorithm described in Section 5.6: here, you assume in the second pass that you already know the best path, so there is no need to consider the two highest γ's in *every* state of the trellis. Why? Discuss.

→ we found the best path with viterbi Algorithm

→ the best path is colored in Orange in the trellis

→ Here I have also added all the paths which are possible for the the question " Y "



⇒ These are my Gamma values from HW2

$$\gamma_1 = \begin{pmatrix} 1/2 \\ 1/6 \times \frac{1}{2} \\ 1/6 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.0833 \\ 0.16667 \end{pmatrix}$$

$$\gamma_2 = \begin{pmatrix} 0.125 \\ 0.04166 \\ 0.02083 \end{pmatrix}$$

$$\gamma_3 = \begin{pmatrix} 1/72 \\ 1/96 \\ 1/108 \end{pmatrix} = \begin{pmatrix} 0.0138 \\ 0.0104 \\ 0.00925 \end{pmatrix}$$

$$\gamma_4 = \begin{pmatrix} 1/128 \\ 1/768 \\ 1/384 \end{pmatrix} = \begin{pmatrix} 0.0078125 \\ 0.00130208 \\ 0.0026641 \end{pmatrix}$$

→ Now i will find the second best path

→ Keeping all the calculation's in the fraction form
→ taking all the calculation's from the previous HW
→ entering the fractions directly

$$\delta_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\delta_1 = \begin{bmatrix} 1/2 \ , \ 1/36 \\ 1/12 \\ 1/6 \ , \ 1/12 \ , \ 1/36 \end{bmatrix}$$

$$= \begin{bmatrix} 1/2 \\ 1/12 \\ 1/6 \end{bmatrix} \longrightarrow \text{for first best path}$$

$$= \begin{bmatrix} 1/36 \\ \\ 1/2 \end{bmatrix}$$ $\longrightarrow$ for second path

$$f_2 = \begin{bmatrix} 1/8, & 1/72 \\ 1/24, & 1/24 \\ 1/72, & 1/48, & 1/54 \end{bmatrix}$$

$$= \begin{bmatrix} 1/8 \\ 1/24 \\ 1/48 \end{bmatrix}$$ $\rightarrow$ first best path

$$= \begin{bmatrix} 1/72 \\ 1/24 \\ 1/54 \end{bmatrix}$$ $\longrightarrow$ for second

$$f_3 = \begin{bmatrix} 1/64, & 1/120 & \\ 1/96, & 1/190 & \\ 1/380, & 1/108, & 1/217 \end{bmatrix}$$

$$= \begin{bmatrix} 1/64 \\ 1/96 \\ 1/108 \end{bmatrix} \longrightarrow 0 \text{ because no path from there}$$

$\longrightarrow$ first

for this we have to consider previous history also.

We shd take max of both

$$= \begin{bmatrix} \max[1/70, 1/150], & 1/120 \\ 1/192 & \\ 0 & \end{bmatrix}$$

$$= \begin{bmatrix} 1/70 \\ 1/192 \\ 0 \end{bmatrix} \longrightarrow \text{second}$$

$$\gamma_4 = \begin{bmatrix} 1/128 , & 1/500 \\ 1/756 , & 1/3K \\ 1/384 , & 1/864 \end{bmatrix}$$

$$= \begin{bmatrix} 1/128 \\ 1/756 \\ 1/384 \end{bmatrix} \rightarrow \text{first}$$

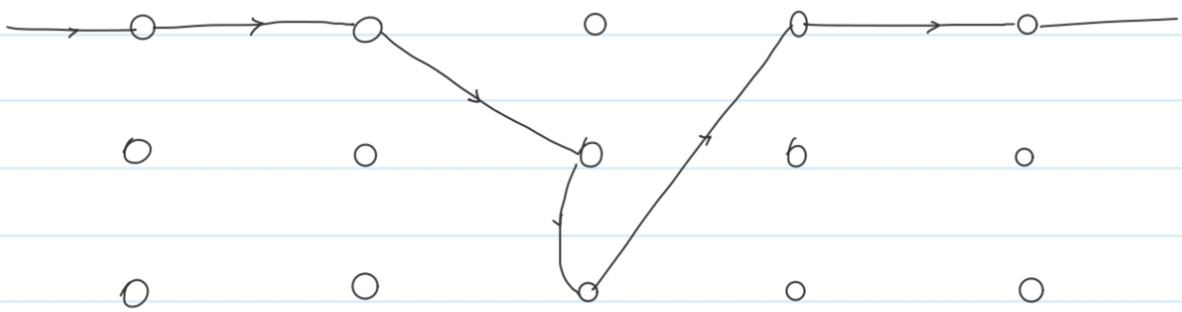
$$= \begin{bmatrix} 1/144 , & 1/550 \\ 1/2300 , & 1/1536 \end{bmatrix} = \begin{bmatrix} 1/44 \\ 1/2300 \end{bmatrix} \rightarrow \text{second}$$

$\gamma_4(1)$ (second path) $> \gamma_4(3) > \gamma_4(2) > \gamma_4(2)$ [second path]

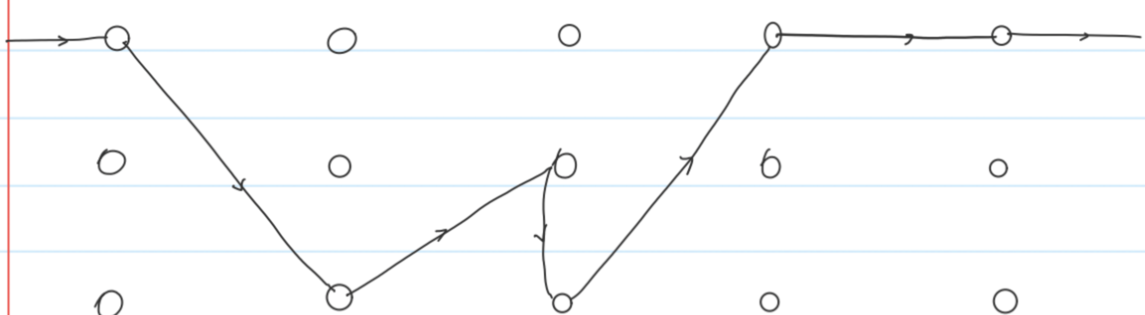
$\downarrow$

because of this we have 2 options

2 paths



we can also consider this



My Algorithm for 2nd pass:-

1. Set $\gamma_0(s_0) = 1$ for the start state s_0, and $\gamma_0(s) = 0$ for all other states $s \neq s_0$.

2. Use the following equation to compute $\gamma_1(s)$ for all states s in the first column of the trellis:

$$\gamma_1(s) = \max[p(y_1, s|s_0) \times \gamma_0(s_0)]$$

Since $\gamma_0(s_0) = 1$ and $\gamma_0(s') = 0$ for all $s' \neq s_0$, this simplifies to:

$$\gamma_1(s) = p(y_1, s|s_0)$$

3. Compute $\gamma_2(s)$ for all states s in the second column of the trellis, avoiding the most likely path found in the first pass:

$$\gamma_2(s) = \max[p(y_2, s|s') \times \gamma_1(s')]$$

where the maximization is taken over all transitions from states s' in the first column to state s in the second column, excluding the transition corresponding to the most likely path from the first pass.

Purge all transitions from states s' in the first column to states s in the second column for which $\gamma_2(s) > p(y_2, s|s') \times \gamma_1(s')$. If more than one transition into state s remains, select (arbitrarily) one to keep and purge the rest.

4. In general, compute $\gamma_i(s)$ for all states s in the ith column of the trellis, avoiding the most likely path found in the first pass:

$$\gamma_i(s) = \max[p(y_i, s|s') \times \gamma_{i-1}(s')]$$

where the maximization is taken over all transitions from states s' in the (i-1)th column to state s in the ith column, excluding the transition corresponding to the most likely path from the first pass.

Purge all transitions from states s' in the (i-1)th column to states s in the ith column for which $\gamma_i(s) > p(y_i, s|s') \times \gamma_{i-1}(s')$. Then, purge all but one of the remaining transitions into state s.

5. Find the state s in the trellis's kth column for which $\gamma_k(s)$ is maximal, excluding the state corresponding to the most likely path from the first pass. In the purged trellis, trace back from this state s to the initial state s_0 in the 0th column along the remaining transitions. The states $s_1, s_2, \ldots, s_k = s$ encountered along this path constitute the second most likely state sequence.

→ the only difference I have made is it ignore first path and purge the remaining

2. *Back-off Bigram Decoding Graph*: In theory, the bigram decoding graph of Figure 5.2 has N *language-model* (LM) states along the rightmost column, and N^2 null arcs to represent $P(w \mid v)$ for every possible bigram $\langle v, w\rangle \in \mathcal{V} \times \mathcal{V}$, where $N = |\mathcal{V}|$. This makes the complexity of Viterbi decoding prohibitive in practice: $|\mathcal{V}| =$ 100k-400k words is not uncommon.

   In practice, we can reduce this complexity considerably. Consider a back-off bigram model

$$P(w \mid v) \quad = \quad \begin{cases} \alpha_v f(w \mid v) & \text{if } C(v, w) > 0 \\ \beta_v P(w) & \text{otherwise,} \end{cases}$$

   where β_v is chosen using, say, the Good-Turing formula, and α_v ensures that $\sum_w P(w \mid v) = 1$ for each word v. For such a model, one need not draw N outgoing arcs from the LM state $e(v)$ for each v, but only draw

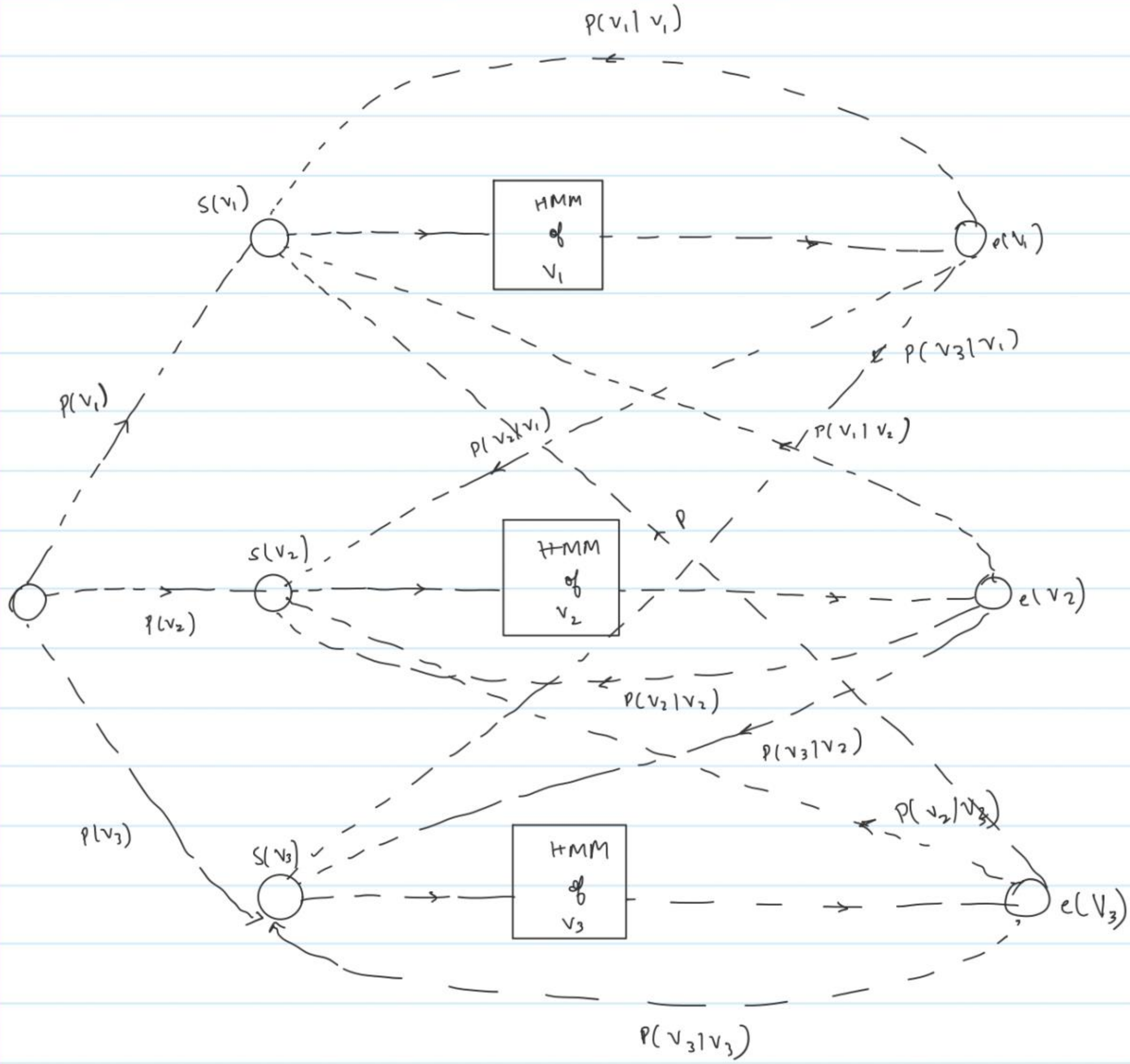
   - as many arcs $e(v) \rightarrow s(w)$ with probability $P(w \mid v)$ as there are w with $C(v, w) > 0$,
   - 1 null arc $e(v) \rightarrow e(\phi)$ with "probability" β_v to a new state $e(\phi)$ shared by all v, and
   - N null arcs $e(\phi) \rightarrow s(w)$ with probability $P(w)$, one arc for each $w \in \mathcal{V}$.

   We will analyze and understand this decoding graph in this problem.

   (a) Discuss how this construction assigns language model probabilities to unseen bigrams, e.g. when $\langle w_{i-1}, w_i\rangle$ has $C(w_{i-1}, w_i) = 0$.

Let's consider just 3 words only and then work on the model

So we know if I have 3 words
"cat", "bat", "mat"
then my bigrams from this
is 9,



P(v1 | v1)
S(v1)
HMM of v1
e(v1)
P(v3 | v1)
P(v1)
P(v2 | v1)
P(v1 | v2)
P
S(v2)
HMM of v2
e(v2)
P(v2)
P(v2 | v2)
P(v3 | v2)
P(v3)
P(v2 | v3)
S(v3)
HMM of v3
e(v3)
P(v3 | v3)

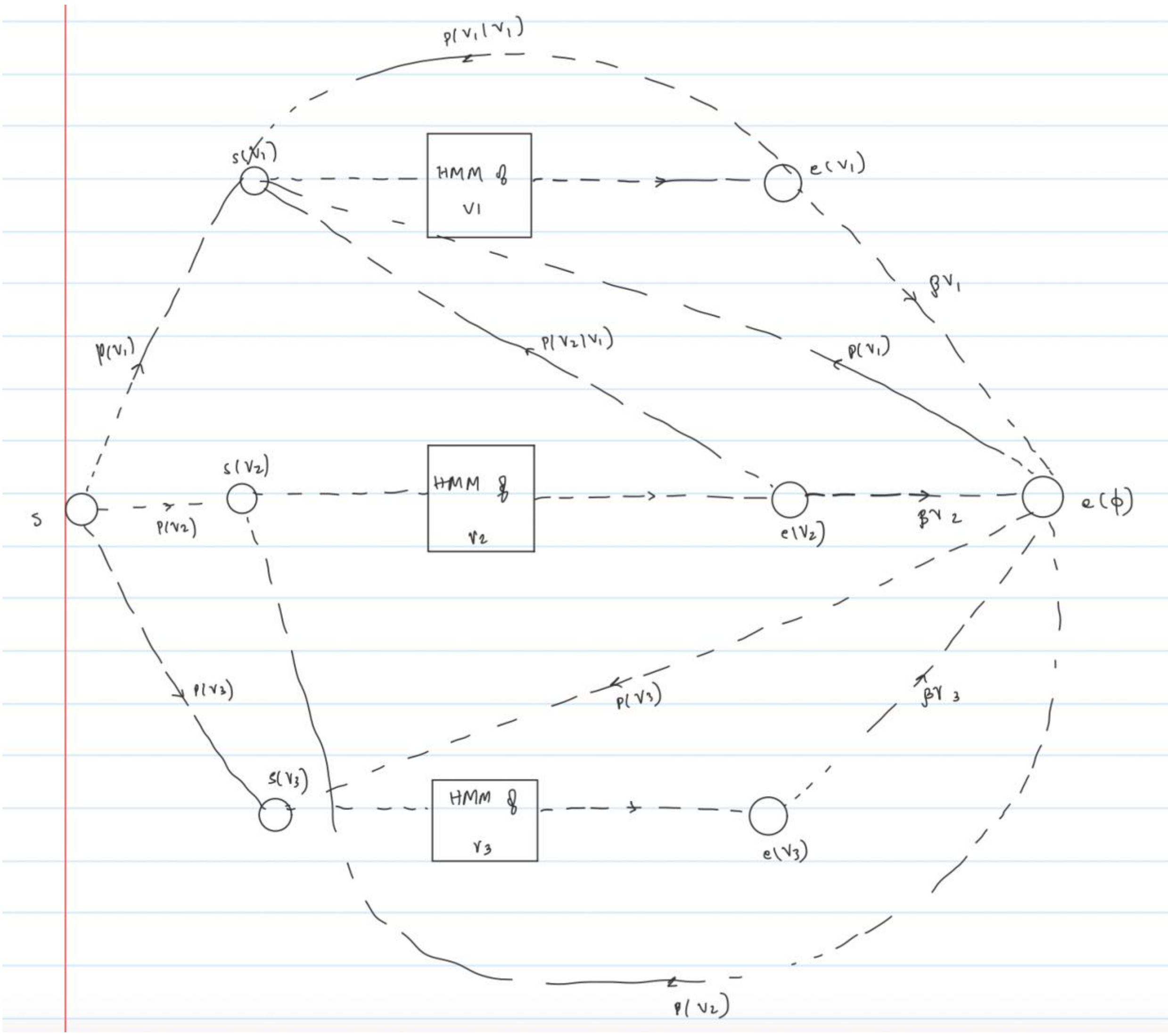
This is from the textbook & has a total of N^2 null arcs

$N = 3$ so $N^2 = 9$

Now we have to consider Backoff :-

so out of N^2 bigrams let's consider that we have 9 bigrams

So we only observe few bigrams in the data


p(v1 | v1)
s(v1)
HMM of
v1
e(v1)
βv1
p(v1)
p(v2|v1)
p(v1)
s
p(v2)
s(v2)
HMM of
v2
e(v2)
βv2
e(φ)
p(v3)
p(v3)
βv3
s(v3)
HMM of
v3
e(v3)
p(v2)

here in the above graph as you can see

I only considered 2

Lets count the total no of null arcs

here

$2N + \varnothing$

$2 \times 3 + 2 = 8$ null arcs

b) compared to N^2 in the bigram there is

$2N + \varnothing$ in the backoff

for ex :- $N = 10$

$N^2 = 100$

$\varnothing = 2$

$2N + \varnothing = 20 + 2 = 22$

reduction

prevents from drawing unnecessary paths

→ N null arcs form $e(v) \rightarrow e(\phi)$

These have "probabilities" βr.

→ There are N null arcs of

$e(\phi) \rightarrow s(w)$, these have

unigram probability $p(w)$

c) There are

i) of $\mathcal{S}$: 4

N = 3

$N^2 = 9$

$2N + \mathcal{S} = 10$ → this is greater so $$2N + \phi < N^2$$

$$\phi < N^2 - 2N$$

This shd be less than $N^2 - 2N$

only then its useful or else no

its worse

ii) for ϕ bigrams there are 2

paths

remember we have $e(\phi)$ here

so

$P(v_2 | v_2) \rightarrow \quad e(v_1) \longrightarrow s(v_1)$

and

$e(v_1) \longrightarrow e(\phi) \longrightarrow s(v_1)$

this is very problematic when we want to get the best path.

Note:-

Note :-

```
import nltk
from nltk.tokenize import word_tokenize
from collections import Counter
def read_text(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read()
    return text
# Read the text data
text_a = read_text('textA.txt')
text_b = read_text('textB.txt')
# Combine Text A and Text B
combined_text = text_a + " " + text_b
# Tokenize the corpus into words
words = word_tokenize(combined_text)
# Determine the size N of the word vocabulary
word_vocabulary_size = len(set(words))
# Tokenize Text A into words
words_text_a = word_tokenize(text_a)
# Count the number of seen bigrams in Text A
bigrams_text_a = list(nltk.bigrams(words_text_a))
seen_bigrams_count = sum(1 for bigram in bigrams_text_a if bigram in
Counter(nltk.bigrams(words)))
print("Size of word vocabulary (N):", word_vocabulary_size)
print("Number of seen bigrams in Text A:", seen_bigrams_count)
# Size of the full bigram graph (Figure 5.2)
full_bigram_graph_size = word_vocabulary_size ** 2
# Size of the back-off bigram graph
backoff_bigram_graph_size = 2 * word_vocabulary_size + seen_bigrams_count
print(f"\nSize of the full bigram graph (Figure 5.2): {full_bigram_graph_size}")
print(f"Size of the back-off bigram graph: {backoff_bigram_graph_size}")
```

```
(base) lavanya@lavanyas-mbp Project1 % python3 bigramm.py
Size of word vocabulary (N): 1732
Number of seen bigrams in Text A: 5064

Size of the full bigram graph (Figure 5.2): 2999824
Size of the back-off bigram graph: 8528
```

- The size of the back-off bigram graph (8528) is significantly smaller than the size of the full bigram graph (2999824). This reduction in size is achieved by representing the unseen bigrams more compactly using the back-off weights and unigram probabilities, rather than explicitly storing all possible bigram transitions.
- The size of the full bigram graph grows quadratically with the vocabulary size N (N^2 = 1732^2 = 2999824). This can become prohibitively large for larger vocabularies, making the graph construction and Viterbi decoding computationally expensive or even infeasible. In contrast, the size of the back-off bigram graph grows linearly with the vocabulary size (2N + Q = 2 * 1732 + 5064 = 8528), making it much more scalable for larger vocabularies.
- The back-off bigram graph provides an efficient way to represent and handle unseen bigrams (those not present in the training data) by using the back-off weights and unigram probabilities. This allows the language model to assign non-zero probabilities to unseen bigrams, which is essential for practical applications.