NLP Assignment Solution

SECTION 1: Implementing Bag-of-Words

---

1. Vocabulary Construction:
Corpus:
- Post A: "cats are cute and funny"
- Post B: "dogs are funny animals"
- Post C: "cats and dogs rarely get along"

Unique Vocabulary (sorted alphabetically):
['along', 'and', 'animals', 'are', 'cats', 'cute', 'dogs', 'funny', 'get', 'rarely']

2. Dimension of the Vector:
The dimension is equal to the size of the vocabulary.
Dimension = 10

3. Matrix Construction:
Count Vectors (based on the sorted vocabulary):
- Post A: [0, 1, 0, 1, 1, 1, 0, 1, 0, 0]
- Post B: [0, 0, 1, 1, 0, 0, 1, 1, 0, 0]
- Post C: [1, 1, 0, 0, 1, 0, 1, 0, 1, 1]

Stacked Matrix Shape: (3, 10)

4. Matrix Interpretation:
- The Rows (3) represent the individual documents (Post A, Post B, Post C).
- The Columns (10) represent the unique words in the global vocabulary.
- Each cell (i, j) represents the frequency of word j in document i.

SECTION 2: Designing a simple model for classification

---=---

1. Probability Expressions:
Given P(cat) = p.
>Probability of a post of length 10 having NO cat=: (1 - p)^10
>Probability of a post of length 10 having AT LEAST one "cat"= 1 - (1 - p)^10
>General expression for post of length L being cat-type
  P(cat-type) = 1 - (1 - p)^L

2. Probability Vector Calculation:
Total words in corpus = 5 (A) + 4 (B) + 6 (C) = 15 words.
Frequency of "cats" in corpus = 2 (once in A, once in C).

Probability P = 2 / 15 ≈ 0.133

Theoretical probability of Post A (Length L=5) being cat-type:
P(Post A is cat-type) = 1 - (1 - 2/15)^5
P(Post A is cat-type) ≈ 1 - (0.487)
P(Post A is cat-type) ≈ 0.513 (or 51.3%)

3. Conditional Probability (Bayes):
Let A = Post contains cute
Let B = Post is cat-type

From the corpus:
- Cat-type posts are Post A and Post C (Total = 2).
- Post A contains "cute". Post C does not.

P(A|B) [Probability it contains "cute" given it is cat-type]:
= (Number of cat-type posts with "cute") / (Total cat-type posts)
= 1 / 2 = 0.5
In this corpus, there is a 50% chance that a discussion about cats involves them being cute.

Using Bayes Theorem ,we find P(B|A) [Probability it is cat-type given it contains "cute"]:
P(B|A) = (P(A|B) * P(B)) / P(A)

- P(A|B) = 0.5
- P(B) [Prior probability of being cat-type] = 2 posts out of 3 = 2/3
- P(A) [Prior probability of containing "cute"] = 1 post out of 3 = 1/3

P(B|A) = (0.5 * 0.666...) / 0.333...
P(B|A) = 1.0

If a post contains "cute", it is 100% certain to be a cat-type post in this specific corpus.

SECTION 3: Designing a simple model for Optimising Upvotes

---

1. Maximizing Upvotes U(L):
Function: U(L) = (-1/20)L^2 + 3L

First Derivative U'(L):
U'(L) = (-2/20)L + 3
U'(L) = -0.1L + 3

Set U'(L) = 0 to find critical point:
0 = -0.1L + 3
0.1L = 3
L = 30

Second Derivative Test U''(L):
U''(L) = -0.1
Since U''(L) is negative, the function is concave down, confirming L = 30 is a Maximum.

Optimum length = 30 words.

2. Optimizing G(L, p):
G(L,p) = [1 - (1-p)^L] * [(-1/20)L^2 + 3L]

Suggestion for finding optimum without derivatives:

Since the function is complex and non-linear, and analytical derivatives might be difficult to solve for roots, we can use Numerical Optimization methods such as:

1. Gradient Descent: Iteratively moving in the direction of the steepest ascent.

2. Grid Search: Since L must be an integer , we can evaluate the function at discrete intervals of L to find the peak.

I also plotted a graph to portray the upvote function,the expected upvote function for the cat type post and cat +cut type post.
It is given in the folder

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def U(L):
    """
    Hypothetical upvotes based purely on length L.
    Formula: -1/20 * L^2 + 3L
    """
    return -1/20 * L**2 + 3 * L

def G_single_type(L, p):# for cat type
    prob_existence = 1 - (1 - p)**L
    return prob_existence * U(L)

def G_two_types(L, p1, p2):# for cat+cute type
    prob_1 = 1 - (1 - p1)**L
    prob_2 = 1 - (1 - p2)**L
    prob_both = prob_1 * prob_2
    return prob_both * U(L)



L = np.linspace(0, 65, 600)


total_words = 15
p_cat = 2 / total_words
p_cute = 1 / total_words


u_values = U(L)
g_cat_values = G_single_type(L, p_cat)
g_cat_cute_values = G_two_types(L, p_cat, p_cute)

plt.figure(figsize=(12, 7))

# Plot U(L) - The theoretical ceiling
plt.plot(L, u_values, label='U(L): Potential Max Upvotes',
         color='blue', linestyle='--', alpha=0.6)

# Plot G(L) for "cat"
plt.plot(L, g_cat_values, label=f'G(L): "cat" only (p={p_cat:.3f})',
         color='green', linewidth=2)

# Plot G(L) for "cat" + "cute"
```

```
plt.plot(L, g_cat_cute_values, label=f'G(L): "cat" + "cute"
(p_cute={p_cute:.3f})',
         color='red', linewidth=2)


idx_max_u = np.argmax(u_values)
idx_max_cat = np.argmax(g_cat_values)
idx_max_cc = np.argmax(g_cat_cute_values)

plt.scatter(L[idx_max_u], u_values[idx_max_u], color='blue', zorder=5)
plt.scatter(L[idx_max_cat], g_cat_values[idx_max_cat], color='green',
zorder=5)
plt.scatter(L[idx_max_cc], g_cat_cute_values[idx_max_cc], color='red',
zorder=5)

plt.text(L[idx_max_u], u_values[idx_max_u]+1,
         f'Max U\nL={L[idx_max_u]:.1f}', color='blue', ha='center')
plt.text(L[idx_max_cat], g_cat_values[idx_max_cat]-4,
         f'Max "cat"\nL={L[idx_max_cat]:.1f}', color='green',
ha='center')
plt.text(L[idx_max_cc], g_cat_cute_values[idx_max_cc]-4,
         f'Max "cat+cute"\nL={L[idx_max_cc]:.1f}', color='red',
ha='center')


plt.title('Optimization of Post Length: Generic vs. Specific Content
Constraints')
plt.xlabel('Post Length (Number of Words)')
plt.ylabel('Expected Upvotes')
plt.legend()
plt.grid(True, alpha=0.3)
plt.axhline(0, color='black', linewidth=0.5)

plt.show()
```

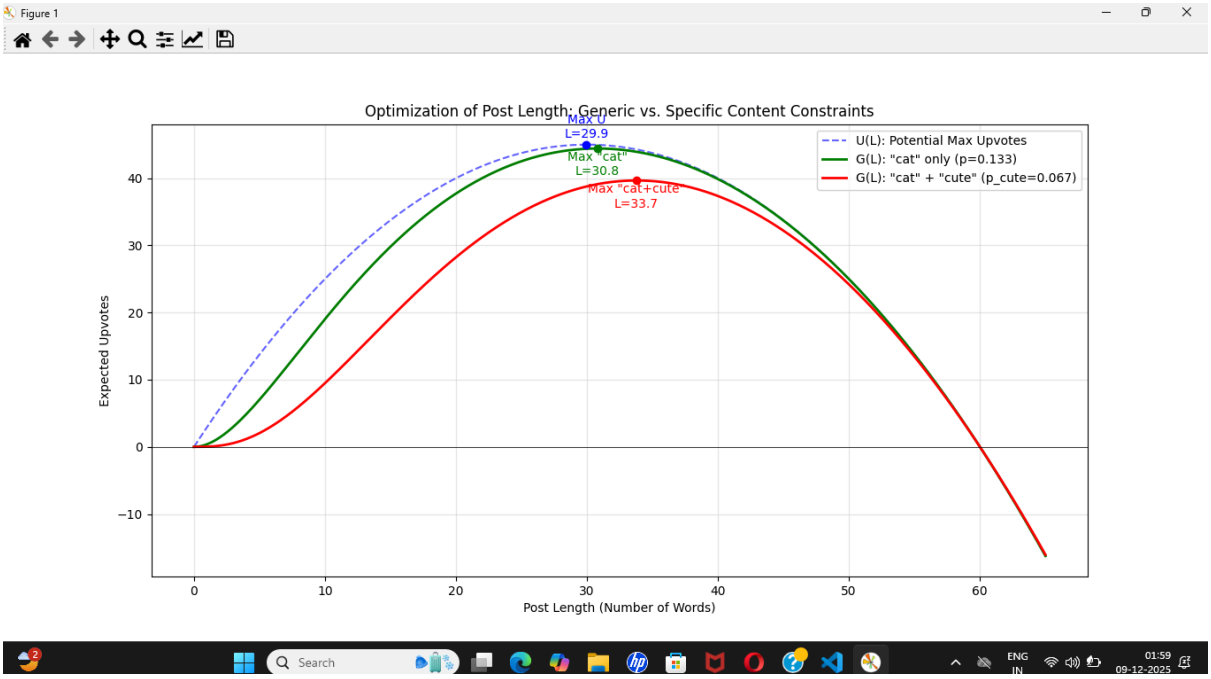
Figure 1
Optimization of Post Length: Generic vs. Specific Content Constraints
Max U
L=29.9
Max "cat"
L=30.8
Max "cat+cute"
L=33.7
U(L): Potential Max Upvotes
G(L): "cat" only (p=0.133)
G(L): "cat" + "cute" (p_cute=0.067)
Expected Upvotes
Post Length (Number of Words)
40
30
20
10
0
−10
0
10
20
30
40
50
60
Search
ENG
IN
01:59
09-12-2025