Week-2:Plagiarism Detection using A* Algorithm

Chetankumar kamani(20251603005), Sakariya Devraj (20251603006), Divyesh Dodiya (20251603007), M.Tech Sem-1 IIIT Vadodara

I. Problem Statement

Given two text documents, the task is to align their sentences and detect plagiarism using the A* search algorithm. The alignment should minimize the edit distance (or maximize similarity) between corresponding sentences.

II. Python Code

```
import sys, re
from heapq import heappush, heappop

# --- Removing Punctutation Marks ---

def normalize_the_text(s: str) -> str:
    s = s.lower()
    s = re.sub(r"[^\w\s\.!\?]", " ", s)
    return re.sub(r"\s+", " ", s).strip()

def tokenize(s: str):
    s = re.sub(r"\s*([\.!\?])\s*", r"\1 ", s)
    parts = re.split(r"[\.!\?]\s+", s)
    return [t.strip() for t in parts if t.strip()]

def words(s: str):
    return [t for t in re.split(r"\W+", s) if t]

def compute_edit_distance(a: str, b: str) -> int:
    A = words(a)
    B = words(b)
    m = len(A)
    n = len(B)
    if m == 0:
        return n
    if n == 0:
        return m
    dp = list(range(n + 1))
    #lavenshtein distance calculation
    for i in range(1, m + 1):
        prev = dp[0]
        dp[0] =  i
        for j in range(1, n + 1):
            tmp = dp[j]
            if A[i-1] == B[j-1]:
                cost = 0
            else:
                cost = 1
            # The following two lines were
                incorrectly indented.
            # They must be inside the inner loop
                to work correctly.
            dp[j] = min(dp[j] + 1, dp[j-1] + 1,
                prev + cost)
            prev = tmp
    return dp[-1]



def ned(text_a: str, text_b: str) -> float:
    num_words_a = len(words(text_a))
    num_words_b = len(words(text_b))
    max_length = max(num_words_a, num_words_b)
    if max_length == 0:
        return 0.0
    raw_distance = compute_edit_distance(text_a,
        text_b)
    return raw_distance / max_length


# --- A* function
def a_star_function(SA, SB, skip_penalty=3.0):
    m, n = len(SA), len(SB)
    def h(i, j):
        return abs((m - i) - (n - j)) *
            skip_penalty

    openq = []
    heappush(openq, (h(0,0), 0.0, 0, 0))
    best = {(0,0): 0.0}
    prev = {}

    while openq:
        f, g, i, j = heappop(openq)
        if (i, j) == (m, n):
            steps = []
            cur = (i, j)
            while cur != (0,0):
                p, op = prev[cur]
                steps.append(op)
                cur = p
            return g, list(reversed(steps))

        # SKIP_A
        if i < m:
            ng, s = g + skip_penalty, (i+1, j)
            if ng < best.get(s, 1e18):
                best[s] = ng; prev[s] = ((i,j), ("
                    SKIP_A", i, -1, skip_penalty))
                heappush(openq, (ng + h(*s), ng, *
                    s))
        # SKIP_B
        if j < n:
            ng, s = g + skip_penalty, (i, j+1)
            if ng < best.get(s, 1e18):
                best[s] = ng; prev[s] = ((i,j), ("
                    SKIP_B", -1, j, skip_penalty))
                heappush(openq, (ng + h(*s), ng, *
                    s))
        # ALIGN
        if i < m and j < n:
            c = compute_edit_distance(SA[i], SB[j
                ])
            ng, s = g + c, (i+1, j+1)
            if ng < best.get(s, 1e18):
                best[s] = ng; prev[s] = ((i,j), ("
                    ALIGN", i, j, float(c)))
                heappush(openq, (ng + h(*s), ng, *
                    s))

    return float("inf"), []

def compare_document(fileA, fileB, t_value,
    sp_value):

    SA = tokenize(normalize_the_text(fileA))
    SB = tokenize(normalize_the_text(fileB))
    aligned=[]
    total, steps = a_star_function(SA, SB,
        sp_value)

    for s in steps:
        if s[0]=="ALIGN":
            aligned.append(s)

    plag = []

    print("------------Result---------------")
    print("number of sentence in the file A:", len
        (SA))
    print("number of sentence in the file B:", len
        (SB))
    print("total_path_cost:", total)

    print("\nALIGNMENT")
    for op, i, j, c in steps:
        if op == "ALIGN":
            ne = round(ned(SA[i], SB[j]), 3)
            flag = (ne <= t_value)
            if flag: plag.append(1)
            status = "PLAGIARIZED" if flag else "
                ORIGINAL"
            print(f"[ALIGN] A[{i}]<->B[{j}] cost={
                int(c)} NED={ne} Status: {status}"
                )
            print("    A:", SA[i])
            print("    B:", SB[j])
        elif op == "SKIP_A":
            print(f"[SKIP_A] A[{i}] {SA[i]}")
        else:
            print(f"[SKIP_B] B[{j}] {SB[j]}")

    pr = (sum(plag)/len(aligned)) if aligned else
        0.0
    plagiarism_percentage = round(pr * 100, 2)

    # The documents are "identical" at the
        percentage of aligned content that is not
        plagiarized.
    originality_percentage = round((1.0 - pr) *
        100, 2)

    print("\n------------------SUMMARY
        ------------------")
    print(f"Total number of plagiarized pairs
        found: {sum(plag)} out of {len(aligned)}
        aligned sentences.")
    print(f"The documents are {
        plagiarism_percentage}% plagiarized.")
    print(f"The documents are {
        originality_percentage}% original.")
    print(f"Based on the analysis, the two
        documents are {originality_percentage}%
        identical.")



#Input two files while executing the program at
    the command line.

if len(sys.argv) < 3:
    print("Expectating Files. You have not given
        files.")
    sys.exit(1)

#path of file 1
A_path = sys.argv[1]

#path of file 2
B_path = sys.argv[2]

#Getting the threshold value from the user.
t_value = float(input("Enter threshold value: "))

#Getting the skip-panelty value from the user
sp_value  = float(input("Enter skip-panelty value:
     ")) # Corrected typo in the prompt

#initially checking the length of the file. If the
     file length

#reading the content from the file A
with open(A_path, "r") as f:
    A = f.read()

#reading the content from the file B
with open(B_path, "r") as f:
    B = f.read()

compare_document(A, B, t_value, sp_value)
```

III. Input Files

A. Test Case 1: Identical Documents

T1docA.txt

```
I like Artificial Intelligence. I like IIIT
    Vadodara
I like probability & Statistics. I like IIIT
    Vadodara.
```

T1docB.txt

```
I like Artificial Intelligence.
I like IIIT Vadodara.
I like probability and statistics.
```

B. Test Case 2: Slightly Modified Documents

T2docA.txt

```
I like Artificial Intelligence.
I like IIIT Vadodara.
I like probability and statistics.
```

T2docB.txt

```
I like Artificial Intelligence.
I like IIIT Vadodara.
I like probability and statistics.
```

C. Test Case 3: Completely Different Documents

T3docA.txt

```
My name is Chetan Kamani.
I live in Jamnagar.
I work as a lecturer in Government Polytechnic.
```

T3docB.txt

```
Sorting algorithms arrange data in ascending or
    descending order.
QuickSort uses partitioning and recursion.
Heaps are used for priority queues.
```

D. Test Case 4: Partial Overlap

T4docA.txt

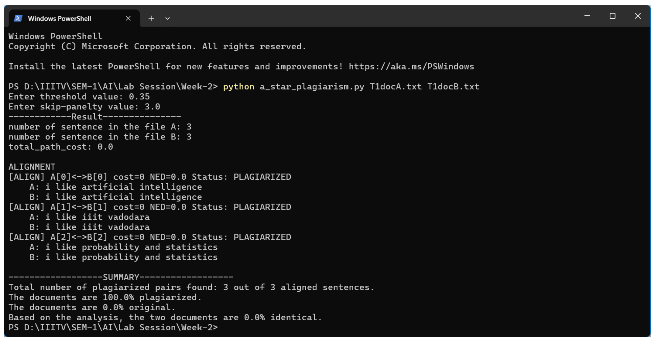
```
My name is Chetan Kamani.
I like IIIT Vadodara.
I enjoy probability and statistics.
Artificial Intelligence is my favorite subject.
I live in Jamnagar.
```

T4docB.txt

```
I like Artificial Intelligence.
I like IIIT Vadodara.
I study probability and statistics.
I live in Jamnagar.
```

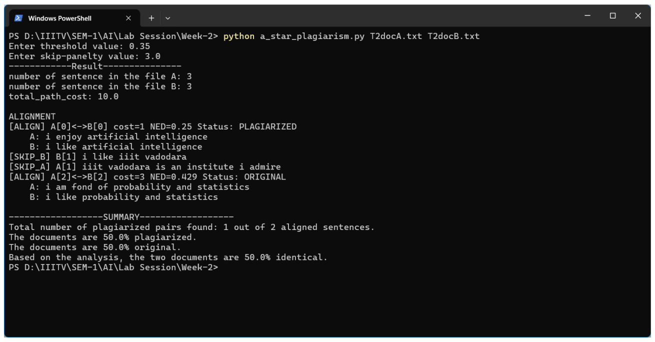
IV. Screenshots of Results

A. Test Case 1



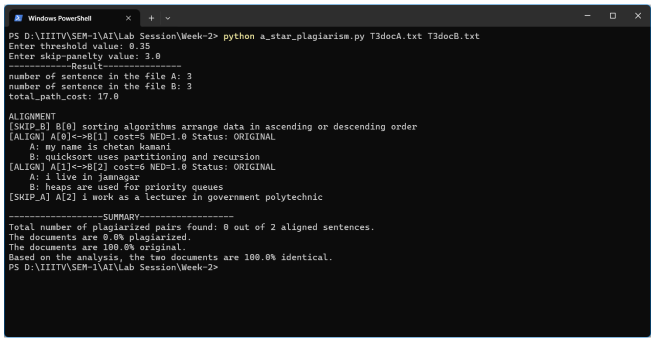
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\IIITV\SEM-1\AI\Lab Session\Week-2> python a_star_plagiarism.py T1docA.txt T1docB.txt
Enter threshold value: 0.35
Enter skip-panelty value: 3.0
------------Result---------------
number of sentence in the file A: 3
number of sentence in the file B: 3
total_path_cost: 0.0

ALIGNMENT
[ALIGN] A[0]<->B[0] cost=0 NED=0.0 Status: PLAGIARIZED
    A: i like artificial intelligence
    B: i like artificial intelligence
[ALIGN] A[1]<->B[1] cost=0 NED=0.0 Status: PLAGIARIZED
    A: i like iiit vadodara
    B: i like iiit vadodara
[ALIGN] A[2]<->B[2] cost=0 NED=0.0 Status: PLAGIARIZED
    A: i like probability and statistics
    B: i like probability and statistics

------------------SUMMARY------------------
Total number of plagiarized pairs found: 3 out of 3 aligned sentences.
The documents are 100.0% plagiarized.
The documents are 0.0% original.
Based on the analysis, the two documents are 0.0% identical.
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2>
```

B. Test Case 2



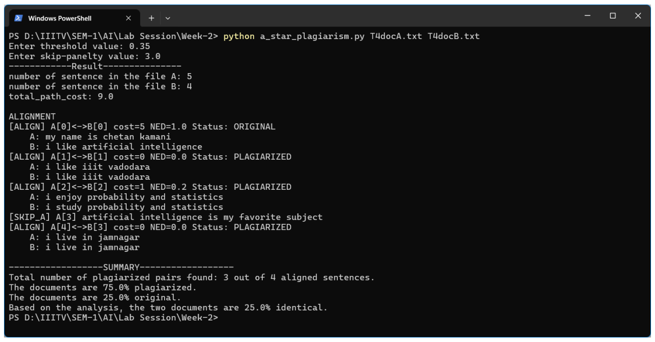
```
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2> python a_star_plagiarism.py T2docA.txt T2docB.txt
Enter threshold value: 0.35
Enter skip-panelty value: 3.0
------------Result---------------
number of sentence in the file A: 3
number of sentence in the file B: 3
total_path_cost: 10.0

ALIGNMENT
[ALIGN] A[0]<->B[0] cost=1 NED=0.25 Status: PLAGIARIZED
    A: i enjoy artificial intelligence
    B: i like artificial intelligence
[SKIP_B] B[1] i like iiit vadodara
[SKIP_A] A[1] iiit vadodara is an institute i admire
[ALIGN] A[2]<->B[2] cost=3 NED=0.429 Status: ORIGINAL
    A: i am fond of probability and statistics
    B: i like probability and statistics

------------------SUMMARY------------------
Total number of plagiarized pairs found: 1 out of 2 aligned sentences.
The documents are 50.0% plagiarized.
The documents are 50.0% original.
Based on the analysis, the two documents are 50.0% identical.
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2>
```

C. Test Case 3



```
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2> python a_star_plagiarism.py T3docA.txt T3docB.txt
Enter threshold value: 0.35
Enter skip-panelty value: 3.0
------------Result---------------
number of sentence in the file A: 3
number of sentence in the file B: 3
total_path_cost: 17.0

ALIGNMENT
[SKIP_B] B[0] sorting algorithms arrange data in ascending or descending order
[ALIGN] A[0]<->B[1] cost=5 NED=1.0 Status: ORIGINAL
    A: my name is chetan kamani
    B: quicksort uses partitioning and recursion
[ALIGN] A[1]<->B[2] cost=6 NED=1.0 Status: ORIGINAL
    A: i live in jamnagar
    B: heaps are used for priority queues
[SKIP_A] A[2] i work as a lecturer in government polytechnic

------------------SUMMARY------------------
Total number of plagiarized pairs found: 0 out of 2 aligned sentences.
The documents are 0.0% plagiarized.
The documents are 100.0% original.
Based on the analysis, the two documents are 100.0% identical.
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2>
```

D. Test Case 4



```
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2> python a_star_plagiarism.py T4docA.txt T4docB.txt
Enter threshold value: 0.35
Enter skip-panelty value: 3.0
------------Result---------------
number of sentence in the file A: 5
number of sentence in the file B: 4
total_path_cost: 9.0

ALIGNMENT
[ALIGN] A[0]<->B[0] cost=5 NED=1.0 Status: ORIGINAL
    A: my name is chetan kamani
    B: i like artificial intelligence
[ALIGN] A[1]<->B[1] cost=0 NED=0.0 Status: PLAGIARIZED
    A: i like iiit vadodara
    B: i like iiit vadodara
[ALIGN] A[2]<->B[2] cost=1 NED=0.2 Status: PLAGIARIZED
    A: i enjoy probability and statistics
    B: i study probability and statistics
[SKIP_A] A[3] artificial intelligence is my favorite subject
[ALIGN] A[4]<->B[3] cost=0 NED=0.0 Status: PLAGIARIZED
    A: i live in jamnagar
    B: i live in jamnagar

------------------SUMMARY------------------
Total number of plagiarized pairs found: 3 out of 4 aligned sentences.
The documents are 75.0% plagiarized.
The documents are 25.0% original.
Based on the analysis, the two documents are 25.0% identical.
PS D:\IIITV\SEM-1\AI\Lab Session\Week-2>
```

V. Code Availability

- The complete source code is available at: GitHub Repository (CS659 – AI Laboratory).
- **GitHub Repository:** https://github.com/ChetanKamani/CS659-LAB-TASK