i22-0074

September 29, 2023

1 Question 01

Applying spaces to the non segemented words by word matching mechanism from urdu dictionary

Reading the Urdu dictionary from the first file into a list and Read the sentences from the test word file

[1]:
```
dictionary_filename = 'words.txt'
sentences_filename = 'word_test.txt'
output_filename = 'outputQ1.txt'

dictionary = []
with open(dictionary_filename, 'r', encoding='utf-8') as dict_file:
    for line in dict_file:
        dictionary.append(line.strip())

sentences = []
with open(sentences_filename, 'r', encoding='utf-8') as sentences_file:
    sentences = sentences_file.readlines()
```

Implementing maximum matching algorithm to insert spaces

[2]:
```
segmented_sentences = []

for sentence in sentences:
    sentence = sentence.strip()
    segmented_sentence = []
    i = 0

    while i < len(sentence):
        max_word = ""
        for j in range(i, len(sentence)):
            current_word = sentence[i:j + 1]
            if current_word in dictionary:
                max_word = current_word

        if max_word:
            segmented_sentence.append(max_word)
            i += len(max_word)
```

```
        else:
            segmented_sentence.append(sentence[i])
            i += 1

    segmented_sentences.append(" ".join(segmented_sentence))
```

OUTPUT of Question 01

[3]:
```
with open(output_filename, 'w', encoding='utf-8') as output_file:
    output_file.write("\n".join(segmented_sentences))

print("Segmented sentences have been written to", output_filename)

with open(output_filename, 'r', encoding='utf-8') as output_file:
    contents = output_file.read()

print(contents[0:300])
```

```
Segmented sentences have been written to outputQ1.txt
                                                2 0 2 3


     2 0 2 3
```

2 Question 02

Applying word correctness for the problems: 1. Omitting one letter, e.g., the input word is .“ “ .2 Adding an extra letter, e.g., the input word is .“ “ .3 Mistyping one letter, e.g., the input word is .“ “ .4 Transposing two adjacent letters, e.g., the input word is .“ “

first we implement a function that satisfies the conditions given

:[4]
```
def generate_corrections(word):
    corrections = []

    # Omitting one letter
    for i in range(len(word)):
        omitted_word = word[:i] + word[i+1:]
        if omitted_word in dictionary:
            corrections.append(f"ONE LETTER OMITTED FROM {word} to␣
↪{omitted_word}")

    # Adding an extra letter
    for i in range(len(word) + 1):
        for char in urdu_alphabet:
            added_word = word[:i] + char + word[i:]
            if added_word in dictionary:
```

```
                corrections.append(f"ONE LETTER ADDED TO {word} to␣
↪{added_word}")

    # Mistyping one letter
    for i in range(len(word)):
        for char in urdu_alphabet:
            mistyped_word = word[:i] + char + word[i+1:]
            if mistyped_word in dictionary:
                corrections.append(f"ONE LETTER DIFFERENT FROM {word} to␣
↪{mistyped_word}")

    # Transposing two adjacent letters
    for i in range(len(word) - 1):
        transposed_word = word[:i] + word[i+1] + word[i] + word[i+2:]
        if transposed_word in dictionary:
            corrections.append(f"TWO LETTERS TRANSPOSED IN {word} to␣
↪{transposed_word}")

    corrections.sort()

    if not corrections:
        corrections.append("UNKNOWN")

    return corrections[:3]  # Limit the output to the top 3 corrections
```

OUTPUT of Question 02

[5]:
```
dictionary_filename = 'words.txt'
dictionary = set()

with open(dictionary_filename, 'r', encoding='utf-8') as dict_file:
    for line in dict_file:
        dictionary.add(line.strip())

error_words_filename = 'error_words_urdu.txt'
error_words = []

with open(error_words_filename, 'r', encoding='utf-8') as error_file:
    error_words = [line.strip() for line in error_file]


urdu_alphabet = "                                                              "


for word in error_words:
    possible_corrections = generate_corrections(word)
```

```
    print(word)
    for action in possible_corrections:
        print(action)
    print("\n")
```

```
ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to
ONE LETTER OMITTED FROM     to



ONE LETTER DIFFERENT FROM      to
ONE LETTER OMITTED FROM     to
ONE LETTER OMITTED FROM     to



ONE LETTER OMITTED FROM      to


  _
ONE LETTER OMITTED FROM   _   to   _


  _
ONE LETTER OMITTED FROM    _   to    _


ONE LETTER OMITTED FROM     to


ONE LETTER OMITTED FROM      to


ONE LETTER OMITTED FROM     to


ONE LETTER OMITTED FROM     to


ONE LETTER DIFFERENT FROM    to
```

ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

ONE LETTER OMITTED FROM to

ONE LETTER OMITTED FROM _ to _

ONE LETTER DIFFERENT FROM to
ONE LETTER OMITTED FROM to

ONE LETTER OMITTED FROM to

ONE LETTER OMITTED FROM to

ONE LETTER OMITTED FROM to

ONE LETTER OMITTED FROM _ to _

ONE LETTER OMITTED FROM _ to _

ONE LETTER OMITTED FROM to

ONE LETTER OMITTED FROM _ to _

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to

ONE LETTER DIFFERENT FROM to

ONE LETTER OMITTED FROM to

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

_
ONE LETTER ADDED TO _ to _
ONE LETTER DIFFERENT FROM _ to _
ONE LETTER DIFFERENT FROM _ to _

ONE LETTER ADDED TO to
ONE LETTER OMITTED FROM to
ONE LETTER OMITTED FROM to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER ADDED TO to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER ADDED TO to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER ADDED TO to

UNKNOWN

ONE LETTER ADDED TO to
ONE LETTER OMITTED FROM to

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

ONE LETTER ADDED TO to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

ONE LETTER ADDED TO to
ONE LETTER OMITTED FROM to

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER ADDED TO to

ONE LETTER ADDED TO to
ONE LETTER ADDED TO to
ONE LETTER ADDED TO to

ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

ONE LETTER DIFFERENT FROM to

UNKNOWN

ONE LETTER DIFFERENT FROM to

ONE LETTER DIFFERENT FROM to

UNKNOWN

ONE LETTER DIFFERENT FROM to
ONE LETTER OMITTED FROM to

ONE LETTER ADDED TO to
ONE LETTER DIFFERENT FROM to
ONE LETTER DIFFERENT FROM to

```
ONE LETTER DIFFERENT FROM      to



ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to



  _
ONE LETTER DIFFERENT FROM   _  to   _



ONE LETTER DIFFERENT FROM      to



ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to



   _ _
ONE LETTER DIFFERENT FROM    _ _   to    _ _



ONE LETTER DIFFERENT FROM      to



  _
UNKNOWN



ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to



ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to
```

```
ONE LETTER ADDED TO    to
ONE LETTER ADDED TO    to
ONE LETTER ADDED TO    to



ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to



   _
ONE LETTER DIFFERENT FROM    _   to    _
ONE LETTER DIFFERENT FROM    _   to    _
ONE LETTER DIFFERENT FROM    _   to    _



ONE LETTER ADDED TO     to
ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to



ONE LETTER ADDED TO     to
ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to



ONE LETTER OMITTED FROM     to
TWO LETTERS TRANSPOSED IN     to



ONE LETTER DIFFERENT FROM     to



    _
UNKNOWN



ONE LETTER DIFFERENT FROM     to
TWO LETTERS TRANSPOSED IN     to
```

```
ONE LETTER OMITTED FROM     to
TWO LETTERS TRANSPOSED IN     to



ONE LETTER ADDED TO    to
ONE LETTER ADDED TO    to
ONE LETTER ADDED TO    to



ONE LETTER ADDED TO    to
ONE LETTER ADDED TO    to
ONE LETTER ADDED TO    to


  _
UNKNOWN



UNKNOWN



ONE LETTER OMITTED FROM    to
ONE LETTER OMITTED FROM    to
TWO LETTERS TRANSPOSED IN    to


   _
UNKNOWN


  _
TWO LETTERS TRANSPOSED IN   _   to   _


UNKNOWN



ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to
```

```
ONE LETTER ADDED TO    to
ONE LETTER DIFFERENT FROM    to
ONE LETTER DIFFERENT FROM    to



UNKNOWN



ONE LETTER DIFFERENT FROM     to
ONE LETTER DIFFERENT FROM     to



ONE LETTER ADDED TO  to
ONE LETTER ADDED TO  to
ONE LETTER ADDED TO  to



ONE LETTER ADDED TO  to
ONE LETTER ADDED TO  to
ONE LETTER ADDED TO  to



ONE LETTER ADDED TO  to
ONE LETTER ADDED TO  to
ONE LETTER ADDED TO  to
```

3 Question 03

Apply kernal multiplication on a grey scale image array to detect Boundries and edges present with in the picture

:[6]
```python
from PIL import Image
img = Image.open(r"me.jpg")
display(img)
```

Using built-in PIL librarey to perform the initial operation of converting image in to grey scale array (further task is performed without using any librarey)

[7]:
```
import math
from PIL import Image
from IPython.display import display

image_path = 'me.jpg'
image = Image.open(image_path)
width, height = image.size

# Convert the image to grayscale
grayscale_image = image.convert('L')

# list to store the gradient values
gradient = [[0] * width for _ in range(height)]
```

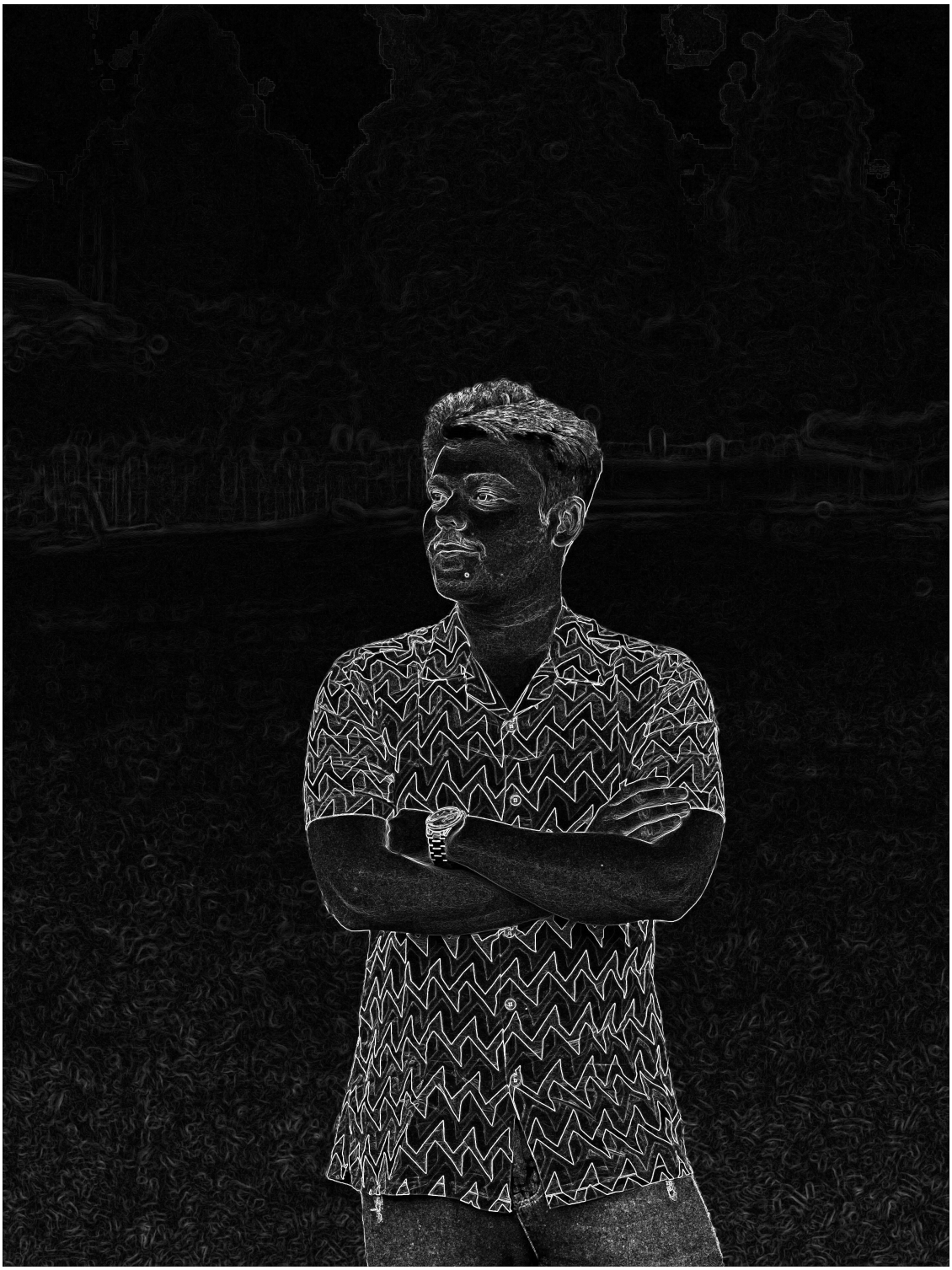
performing kernal multiplication to detect edges and boundries

[8]:
```
Gx = [
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]
]

Gy = [
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1]
]

for y in range(1, height - 1):
    for x in range(1, width - 1):
        vertical_gradient = 0
        horizontal_gradient = 0

        for ky in range(3):
            for kx in range(3):
                # Calculate pixel coordinates in the image (.getpixel is used␣
↪to calculate only the pixel value for grey scale image not for the given␣
↪task)
                px = x - 1 + kx
                py = y - 1 + ky
                vertical_gradient += Gx[ky][kx] * grayscale_image.getpixel((px,␣
↪py))
                horizontal_gradient += Gy[ky][kx] * grayscale_image.
↪getpixel((px, py))


        magnitude = math.sqrt(vertical_gradient**2 + horizontal_gradient**2)
```

```
        # keeping megnitude in range [0, 255]
        magnitude = min(max(0, magnitude), 255)
        gradient[y][x] = magnitude


gradient_image = Image.new('L', (width, height))
for y in range(height):
    for x in range(width):
        pixel_value = int(gradient[y][x])
        gradient_image.putpixel((x, y), pixel_value)


gradient_image.save('edgesimage.jpg')

display(gradient_image)
```