Sai Vasanth Vinay Kudari
UBIT: 50365626

Goal

One way to match characters on a text document is to extract the characters from the document and compare each character with the reference set and return the character which has the highest correlation but this approach has some challenges, we have to transform the reference set image resolution to match the target resolution. Another way to match characters is to generate features and match the feature vectors. I have used the SIFT feature descriptor to extract the features and a custom matching function to match and classify the target character with the highest matching score as the recognised character.

In this project I've explore both the techniques, listing below the steps I've performed in each of the techniques

Enrollment

Image difference

1. Apply thresholding to binarize the image
2. Crop the image so that the resulting image doesn't have any background borders
3. Resize to 64 X 64 resolution and save the image to JSON file

SIFT feature matching

1. Pad the characters with background pixels (white) : I had to do this because SIFT couldn't extract features of characters which don't have any background pixels Eg. I, dot
2. Optionally resize the image if the image is small : SIFT is unable to extract features if the size of the image is low
3. Save the feature descriptors for each character to JSON file

Detection

This step is common to both the approaches, I've used Depth First Search to find the connected components. Initially I have split the image into rows and applied DFS to each row by iterating left to right and top to bottom in order rank the components in english reading order, but this approach might fail if the image is slightly skewed, also the requirement to have it in order was not strict so I've dropped this approach and performed DFS on the whole image.

1. Binarize the image by applying thresholding using Otsu method : Otsu's method gave slightly better results than global thresholding

2. Iterate from left to right and top to bottom to find foreground pixels and explore the path until it covers the whole character and rank the character with a number and increment the number after one character is explored and save the top, left, right and bottom coordinates
3. After getting the components, I've applied **gaussian filter to smoothen and remove any artifacts**
4. Optionally resize if the character is too small for the same reason as above
5. Extract features using SIFT descriptor and save the feature vectors along with bounding box coordinates in a JSON file

Recognition

Image difference

1. For every component subtract each character with the matching source characters and take the mean of the resulting image
2. The character with least minimum score of recognized as the matched character
3. If the minimum score is below set threshold of 30 we mark the character as unknown: This threshold is found by trial and error
4. Save the result in the prescribed format

SIFT feature matching

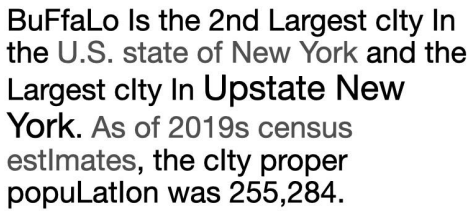
1. For every component each feature vector is matched with every feature vector of matching source characters. Matching between vectors using the below steps

    a. L4 Norm was used to measure the distance between feature vectors : L4 seemed to help when setting threshold to mark the character as unknown over L2 Norm
    b. Find the distance between all the feature vectors and find the ratio between least two distances, if the ratio is less than 0.8 add the minimum distance to the list **else add the average of all the all the distances**: Adding the average distance has improved the accuracy significantly

2. Take the mean of the list of distances obtained from matcher and match the character with least mean distance if its below a set threshold of 194
3. Save the result with matching character in prescribed format

Results

Though image difference technique gave a higher score with all the three test cases when compared to feature extraction technique, I have chosen to submit the latter as it will lead to a higher score



Test case 1

Test case 2

Test case 3

Connected Components

	Test Case 1	Test Case 2	Test Case 3
No. of components	142	278	265

F1 Score

Technique	Test Case 1	Test Case 2	Test Case 3
Image Difference	0.85	0.86	0.98
SIFT Feature Matching	1.00	0.70	0.64