Report

Tappan Ajmera, Saurabh Parekh

Section 1: Design

The parser we created uses graph-based approach. As data [3] consists of handwritten formulas, relations between each symbol can be described by an edge and a formula can be represented as a graph. Inspired by work done in [1], we decided to go with this approach as the results are promising. One of the major advantage of graph-based approach is that a promising accuracy is attained even without defining grammar. Hence, we chose a graph-based approach.

To create a graph from a given set of symbols, line of sight (LOS) algorithm is used. A line of sight graph is constructed based on whether an unobstructed line can be drawn from the centre of one symbol to the convex hull of another symbol [1]. The graph created using LOS is a directed graph. Once graph is created, a relationship classifier is used which gives probability for a relationship between two symbols. The probability of a relationship class with max probability chosen. These probabilities are used as edge weight for the graph. Edmonds algorithm is applied to this graph. Edmond's algorithm is modified to find *maximum* spanning tree in the graph. This maximum spanning tree is the path of parsing.

The relationship classifier is trained on geometric features between two symbols. These features include distance between bounding box centres, distance between the average centres, vertical and horizontal distance, writing curvature and writing slope. The idea behind using these features was the angles and distances between the consecutive symbols will give discriminating information about the spatial relationship between the symbols. For example. The vertical distance and the Euclidean distance between the bounding box of one symbol above another will be different as compared to a symbol being on the right of the current symbol. Similarly, the writing curvatures and writing angles for the super script and subscript will help differentiating these two relations.

Section 2: Pre-processing and Features

Pre-processing:

1) Duplicate filtering:
    a. This removes duplicate points from the strokes, as they are not useful.
2) Normalization:
    a. All the x and y coordinates are normalized to the range of 0 and 1.
    $X_i = (X_i - X_{min}) / X_{max} - X_{min}$
    $Y_i = (Y_i - Y_{min}) / Y_{max} - Y_{min}$
3) Smoothing:
    a. The stroke is smoothened by taking the average of the current point, previous point and next point in a stroke.

Features:

We have used geometric features to find symbol pair relationship. The feature used are distance between bounding box centres, distance between the average centres, vertical and horizontal distance, writing curvature and writing slope.

1) To calculate **distance between bounding box centres**, the maximum and minimum x and y co-ordinates are found. Then using these points, the centre of the bounding box is calculated. Centre of bounding box is calculated as

$$X_c = (X_{max} + X_{min})/2$$

$$Y_c = (Y_{max} + Y_{min})/2$$

Once the bounding box centre of the two symbols are calculated. We calculate the distance between them using Euclidean distance formula.

$$\text{Euclidean Distance} = \sqrt{\left(x_{C_1} - x_{C_2}\right)^2 + \left(y_{C_1} - y_{C_2}\right)^2}$$

Where $x_{C_1}, y_{C_1} and\ x_{C_2}, y_{C_2}$ are the centres of two different bounding box.

2) Another similar features which we use is **distance between average centres**. The centre is calculated by finding the average of the x and y co-ordinates and then calculating the distance between the average centres of the two symbols. Average centre is calculated as:

$$X_{\text{average centre}} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Y_{\text{average centre}} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

The maximal point pair distance is the maximum Euclidean distance two points, one from each symbol.

3) **Horizontal distance** is calculated by finding the difference between the X co-ordinates of the averaged centres and **Vertical distance** is calculated by calculating the difference between the Y-coordinates of the averaged centres of successive symbols.

4) **Writing slope** is calculated by calculating the angle between the horizontal and the line connecting the first symbol of the next symbol and the last point of the current slope.

$$\theta = \tan^{-1}((Y_{next} - Y_{current})/(X_{next} - X_{current}))$$

5) **Writing curvature** is calculated by finding the angle formed when the lines constructed using the first point of next symbol and last point of the current and next symbol is intersected. The angle is calculated as:

$$\theta = \tan^{-1}((m_2 - m_1)/(1 + m_2 m_1))$$

Where m_1 and m_2 are slopes of current and next symbol. The slope is calculated using the formula:

$$\text{Slope} = (Y_1 - Y_2)/(X_1 - X_2)$$

where X_1, Y_1 and X_2, Y_2 are the first and last point of the current and next symbols respectively.

Section 3: Parser

The parsing algorithm is created using a graph-based approach. The output space of this algorithm is a MST in which vertices represents symbols and the edge between them represents the relation between them.

Parsing Algorithm [1]:

1) Take strokes as input
2) Run segmentation to segment symbols and create a symbol list
3) Create symbol to index mapping dictionary D
4) Create NxN adjacency matrix G and initialize with 0
5) Create Line-of-sight graph
    a. For every symbol S, find its bounding box center and initialize an unblocked angle range UAR [0,2π] do
        Sort other symbols (OS) based on closest point pair distance
        For each symbol in sorted symbols do
            1. Calculate angle range its convex hull can block: BAR
            2. Check the visibility angle range VAR = overlap in BAR and UAR
                If bounding box center of S can see another symbol
                Then
                i. Add an edge from S to OS i.e add 1 in (i,j) where i is index of symbol S and j is index of symbol OS in the adjacency matrix G
                ii. UAR = UAR – VAR
            3. end for
    b. end for
6) Create NxN matrix G1 to store relationship labels
7) For i = 0 to N:
    a. For j = 0 to N:
        i. If G[i][j] == 1
            1. Retrieve symbols using D
            2. Classify relationship between symbols at i and j
            3. G[i][j] = class probability
            4. G1[i][j] = class label
8) Apply Edmond's algorithm to create maximum spanning tree (MST) from G
9) To recover the relation parse MST
    a. For every symbol pair (i,j) that has edge between them
    b. Get label from G1[i][j]

Algorithm for calculating BAR [1]:

For every node (X_n, Y_n) in convex hull do

a. Construct vector V from centroid (X_0, Y_0) to the node. $V1 = (X_n-X_0, Y_n-Y_0)$
b. Calculate the angle Θ between V1 and horizontal vector V2 = (1,0)

$$\Theta = \arccos((V1 \cdot V2)/(||V1|||V2||) \text{ if } Y_n \geq Y_0$$

$$\Theta = 2 \times \pi - \arccos((V1 \cdot V2)/(||V1|||V2||) \text{ if } Y_n \leq Y_0$$

End for

Return BAR = [min Θ, max Θ]

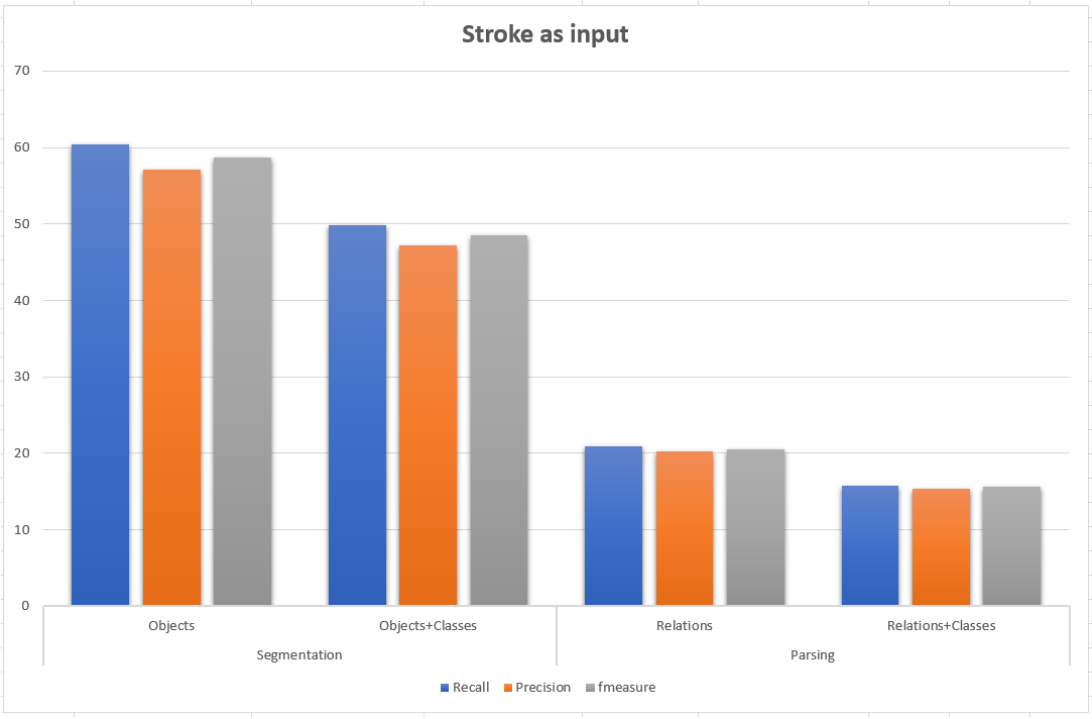
For both our parsers, the algorithm remains the same as the input to the parser is same that is segmented symbols.

The time complexity of our algorithm is $O(N^3)$ where N is the number of symbols. In step 3, for creating line of sight graph, for each symbol we sort all other symbols. The time complexity of sorting is $O(N \log N)$ and there are N symbols, so time complexity of step 3 is $O(N^2 \log N)$. Step 4 runs Edmond's algorithm. The time complexity in Edmonds algorithm is given as $O(|E| \times |V|)$ where E is number of edges and V is number of vertices. In our case, V = N as every vertices represents a symbol. In worst case, the graph is fully connected then $E = |V| \times |V-1|$ and hence the complexity becomes $O(|V|^2 \times |V|) = O(V^3) = O(N^3)$. Hence our time complexity is $O(N^3)$ where N is the number of symbols.

Section 4: Results and Discussion

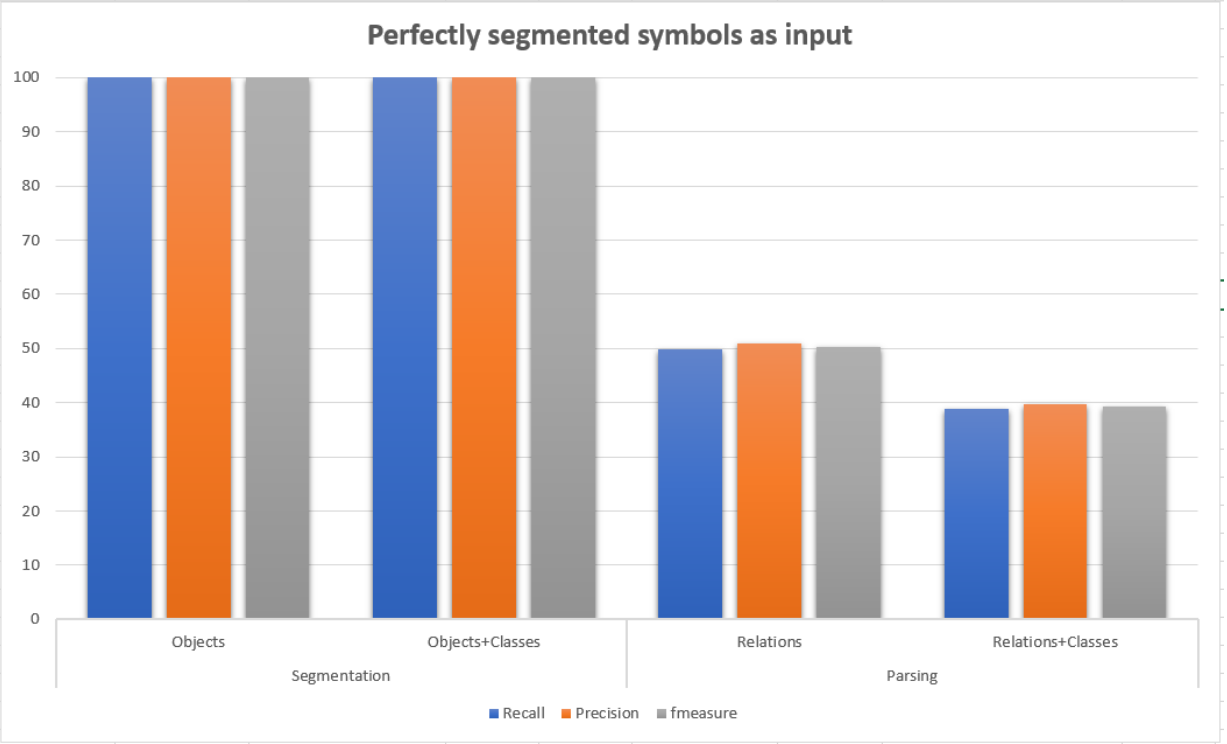
Result for **strokes** as input

	Segmentation		Parsing	
	Objects	Objects+Classes	Relations	Relations+Classes
Recall	60.43	49.85	20.9	15.83
Precision	57.16	47.15	20.27	15.35
fmeasure	58.75	48.46	20.58	15.58



Results for **perfectly segmented** symbols

	Segmentation		Parsing	
	Objects	Objects+Classes	Relations	Relations+Classes
Recall	100	100	49.78	38.91
Precision	100	100	50.88	39.76
fmeasure	100	100	50.33	39.33



For parser with strokes as input. The top error is that parser is unable to find relationship between the two symbols. The reason for such kind of error is that relationship classifier is misclassifying the relationship between two symbols. Another kind of error is for symbol like '-', '1', '=' (symbols with one or two stokes) is that they are merged to form a single symbol and thus no relationship is predicted between them. For example, '-' and '1' are merged to form '+'. However, in symbol level parser as the symbols are perfectly segmented such error are eliminated. Most of the time the relationship classifier misclassifies the 'above' relationship as 'right'.

For parser with symbol as input. The error that parser is unable to find any relationship between two symbols still exists but the misclassification of the relationship for symbols with one or two strokes has reduced comparatively.

The pattern we saw for both the parsers is that the type of parsing errors is same in both the parsers if the error caused by segmentation is not considered.

To improve our parsing model further, shape context features can be used. The shape context features can be found using Parzen window as in [1].

References

[1] L. Hu, “Features and Algorithms for Visual Parsing of Handwritten Mathematical Expressions,” *Thesis*, May 2016.

[2] “scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/

[3] “CROHME 2016.” [Online]. Available: http://ivc.univ-nantes.fr/CROHME/datasets.php.