



Rule based segmentation of lower modifiers in complex Bangla scripts

Md. Abul Hasnat and Mumit Khan

Md. Abul Hasnat

Center for Research on Bangla Language Processing

BRAC University



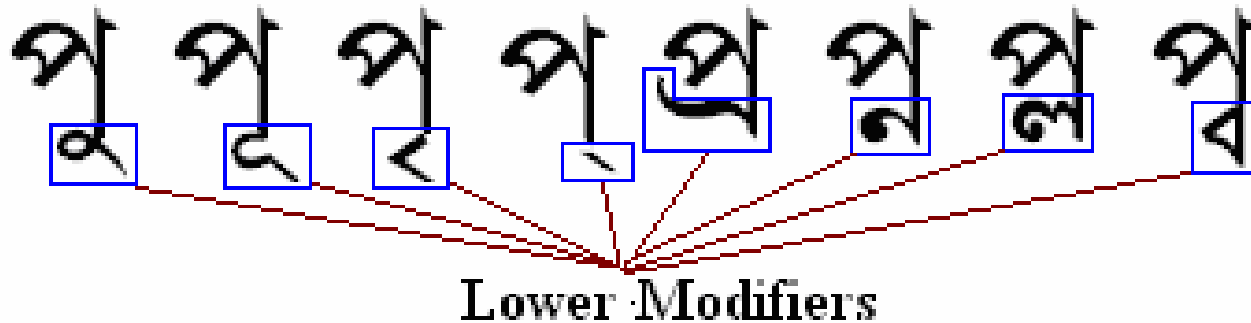
Outline

- Lower modifiers in Bangla script.
- Existing approaches.
- Challenges.
- Methodology of the proposed approach.
- Rules.
- Result.
- Conclusion.



Lower Modifiers

- Eight lower modifiers. Classified into three groups
 - kaar-symbol (়, ্ and ্ব) or vowel modifiers
 - fola-symbol (়্, ন, ল and ব) or consonant modifiers
 - halant-symbol (্)



Example of lower modifiers attached with consonant 'প'



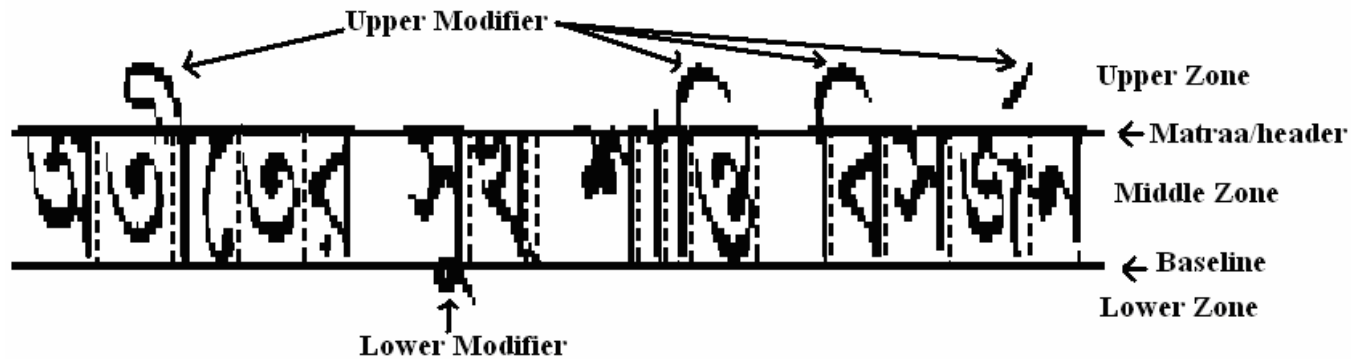
Lower Modifiers

- Among these modifiers ro-fola (e.g., ঞ) is difficult to identify and segment from characters.
 - It has similarity with several other characters (ঞ, ঞ, ঞ and ঞ).
 - Segmentation might cause a distortion in the original shape of the core character.



Existing Approaches

- Find out a lower zone separator from zone based approach and then segment the modifier.



Zone based approach of Bangla character segmentation.

- Baseline selection:
 - Consider an imaginary line in the middle of the text line and then to compute a horizontal line from the information of the lower-most pixels.
 - Detect abrupt changes.

Existing Approaches

- Baseline selection:
 - Connected component analysis.
 - Determining the line that contains most of the lowest points where the points are detected by applying DFS.
 - Usage of average height and run length of characters to detect the separator line.
 - Detect from the threshold character height, calculated from statistics of the height of the characters in a line.
- Modifier extraction:
 - Straight separation from baseline.
 - Apply DFS technique below the baseline.
 - Recognition based segmentation system.
 - Template matching.

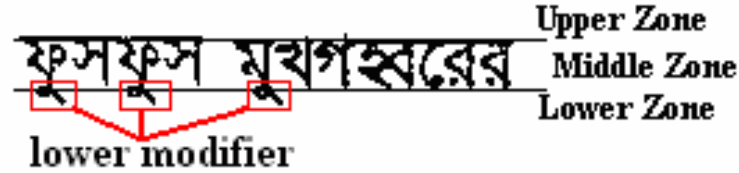


Existing Approaches

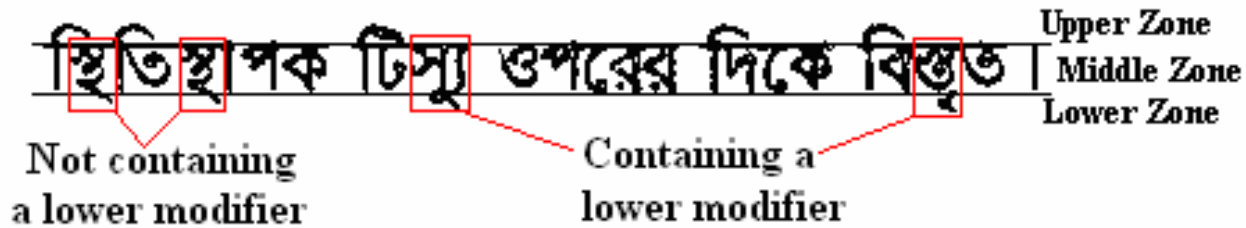
- Limitations:
 - Under segmentation or false rejection.
 - Over segmentation or false acceptance.
- Reasons:
 - Improper alignments of the words.
 - Presence of a few conjuncts.
 - Words with different styles and sizes.



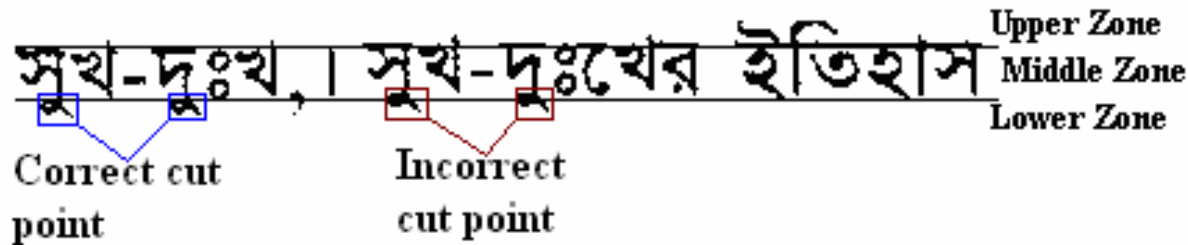
Existing Approaches



(a)



(b)



(c)

- (a) The ideal case of lower modifier extraction
- (b) Example of false acceptance of the lower modifier container characters.
- (c) Problems in locating the cut point.



Challenges

- Selecting the characters containing a lower modifier.
- Eliminating the falsely accepted units.
- Locating the segmentation cut point for extracting the lower modifier.



Proposed Approach

- Lower modifier segmentation is divided into three sub tasks:
 - **1st:** Calculation of the lower modifier separator line and selection of the primary lower modifier containers.
 - **2nd:** Elimination of the preliminary selected units/characters that do not actually contain any lower modifier.
 - **3rd:** Extraction of the lower modifier from a container unit using the features of the lower modifier.



Methodology

Calculation of the lower modifier separator line and selection of the primary lower modifier containers.

- Calculation of the point of separation (POS) using header/matraa location (**matraaLoc**) and threshold character height (**thCharHt**).
 - **POS = matraaLoc + thCharHt**
 - **matraaLoc**: maximum horizontal run length histogram value.
 - **maxCharHt**: median of maximum five characters of a line.
- Take horizontal histogram from the next row of **thCharHt** to bottom of the line.
- If the histogram returns non-zero values then we take vertical histogram to locate all the modifier container.

Methodology

Elimination of the preliminary selected units/characters that do not actually contain any lower modifier.

- Compute
 - Aspect ratio (asp_ratio) of the lower modifier container units.
 - Ratio of the unit height vs. height of the lower modifier (RtLM).
- Perform a two-step categorization of the units.
 - Ist step : based on aspect ratio

Name	Cat-1	Cat-2	Cat-3
Conditions	asp_ratio \leq 0.45	asp_ratio $>$ 0.45 & asp_ratio $<$ 0.8	asp_ratio \geq 0.8
Example	‘া’, ‘/’, ‘(, ‘)’	দু, নু, ফু	নুতে, হুর, সুখে

Examples and conditions for the Ist step categorization

- Cat-1 units do not contain lower modifier.
- Cat-3 units contain more than one units.

Methodology

Elimination of the preliminary selected units/characters that do not actually contain any lower modifier.

- Perform a two-step categorization of the units.
 - 2nd step categorization
 - applied on Cat-2 units
 - based on RtLM

Name of category	Accept	Reject	Process
Conditions	RtLM ≤ 6	RtLM ≥ 8	RtLM > 6 & RtLM < 8
Examples	দু, দু, ফু	ক্র, ঙ, ন, প, ঞ, ঠ, ১, ৪, ৬	ঞ্জ, ণ্ট, ল্ট, স্প, ড

Examples and conditions for the 2nd step categorization

- Process category may contain a lower modifier and hence need additional checking.

Challenges

Elimination of the preliminary selected units/characters that do not actually contain any lower modifier.

- Eliminate the units which do not have lower modifier from the “Accept as lower modifier” category.
- Accept those units which actually have lower from the “Process” category.



Rules

Elimination of the preliminary selected units/characters that do not actually contain any lower modifier.

- Need to calculate the starting position of modifier with respect to the width of the modifier, named as relative location (relPosition).
- Four rules of elimination.



Rules of elimination

- **Rule-1:**
- relPosition for a valid lower modifier is:
 - **relPosition < 0.5 && relPosition > 0.1** when **RtLM ≤ 6**
 - for Accept category units.
 - **relPosition < 0.6 && relPosition > 0.2** when **RtLM > 6**
 - for Process category units.



Examples of units which are eliminated after applying rule-1.



Rules of elimination

- **Rule-2:**
- A valid lower modifier should not contain more than 70% black pixels compare to the unit width in any of its row.
 - This rule will be applicable for the process category units.



Examples of units which are eliminated after applying rule-2



Rules of elimination

- **Rule-3:**
- The width of the extracted modifier should be more than 40% relative to the width of the unit.
 - Prevent the possible over-segmentation.

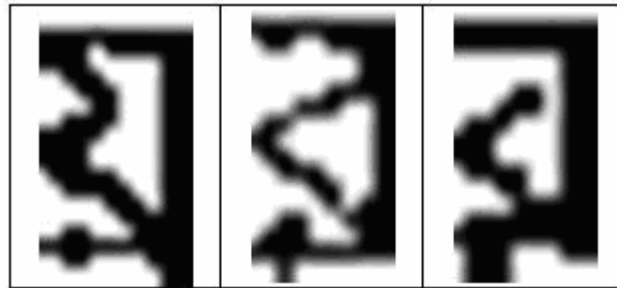


Examples of units which are eliminated after applying rule-3.



Rules of elimination

- **Rule-4:**
- There must be one connected component for a valid lower modifier.
- If there is more than one connected component then the one that satisfy rule-3 is qualified as a modifier.



Examples of units which are eliminated after applying rule-4.



Methodology

Extraction of the lower modifier from a container unit using the features of the lower modifier.

- Segment the lower modifier from the core unit.
- Validate this against the features.
- 3 rules of extraction.
- Few cases when the rules do not satisfy the extracted unit as a lower modifier:
 - Shift the cut point upward using the amount of one third of the lower modifier image height.
 - Take the horizontal projection and select the row that contains minimum number of pixels as well having one crossing.

Rules of extraction

- **Rule-1:**
- Extract the modifier from the average height (avgHt) till the bottom of the container unit.

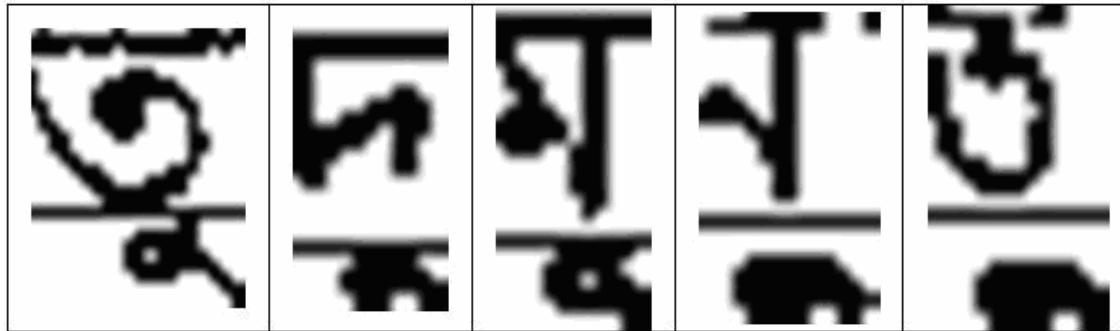


Examples of lower modifier extraction by applying rule-1



Rules of extraction

- **Rule-2:**
- Applied when lower modifier is not attached with the character.
- Search for the gap (row of white pixels).
- If such a gap (lmCutPoint) is found then extract the lower modifier from the lmCutPoint till the bottom.

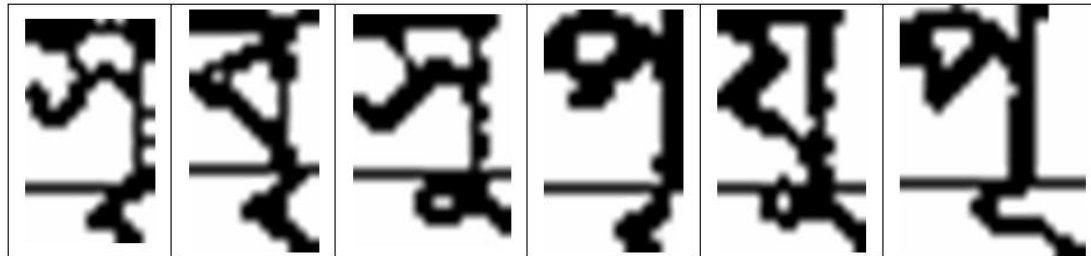


Examples of lower modifier extraction by applying rule-2.



Rules of extraction

- **Rule-3:**
- Exception of ROE-1 is the combination of the lower modifier “্” or the broken part of other lower modifiers with the consonant where the vertical bar is at the right most columns (ল, স, র, ষ, শ, য, ঝ) .
- Check the presence of vertical bar.
- Check relative starting position: $0.5 < \text{relPosition} < 0.7$
- If satisfied then segment the lower modifier from avgHt to bottom of the container unit.



Examples of lower modifier extraction by applying rule-3

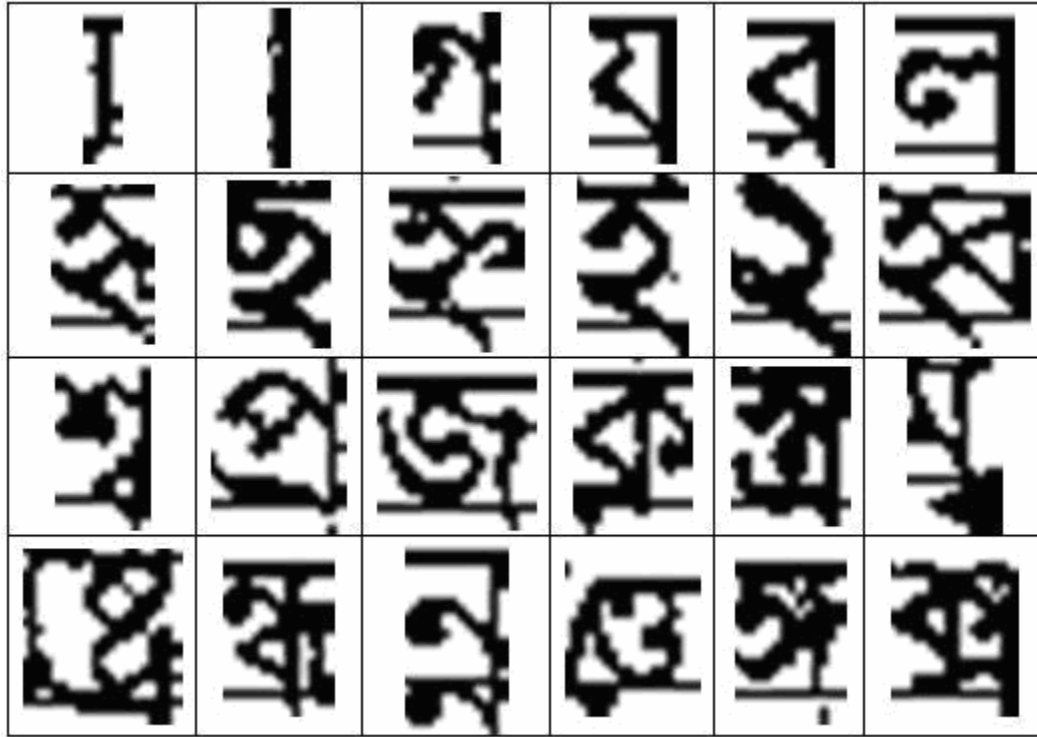
Result

- First step we select the lower modifier containers using our selection approach and observed that on average 47.9%.
 - Over segmentation
- Next we used the aspect ratio and RtLM and observed the error rate is 26%.
- In step-3 we applied the rules and observed that the error rate goes down to 2.7% on average.

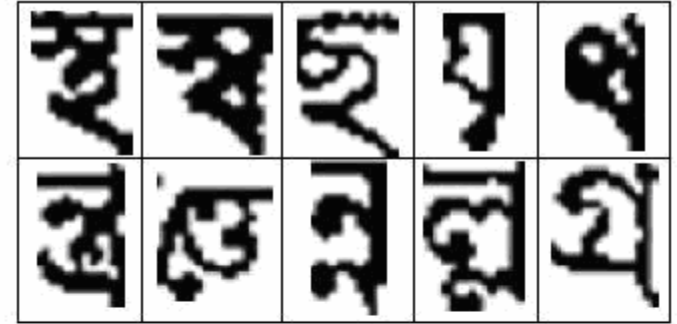
Image Name	Total Units	Lower Modifiers	Error rate (Ist step)	Error rate (2nd Step)	Error rate (3rd step)
BB1	2049	42	60.4	24.2	1.3
BB3	2352	52	44.7	15.5	2.1
BPD1	1090	23	53.1	28.0	3.5
BT1	359	16	33.3	36.0	4.0
Average Error Rate			47.9	25.9	2.7

Result of the lower modifier segmentation at different stage

Result



(a)



(b)



(c)

Incorrect lower modifier containers.

- (a) after step-1
- (b) after step-2
- (c) after step-3



Result

- Erroneous units are identified as halant-symbol (◌̣).
- To avoid these errors we used a module with a set of empirical rules aided by dictionary lookups.
- Correct after recognition.
- Need to train the broken parts of these units also.
- The average error rate after this step goes down to 0.4%.

$$\begin{array}{lll} 1. \text{ ২ + ◌̣ = ২} & 3. \text{ ঐ + ◌̣ = ঐ} & 5. \text{ ঞ + ◌̣ = ঞ} \\ 2. \text{ ঞ + ◌̣ = ঞ} & 4. \text{ হ + ◌̣ = হ} & \end{array}$$

Rules for the post processor



Conclusion

- Generic technique:
 - Devanagari descender segmentation is 58.39%
 - Bangla descender segmentation is 47.9%
 - Proves our assumptions is correct.
- A complete dissection based lower modifier segmentation technique for Bangla printed text document image characters.
- Result shows significant improvement compared to the generic approach.
- Methodology outlined here is applicable to Devanagari as well.

Thank You

