Automated Generation of Search-able PDF Format of Chemical Equations from Document Images

Prerana Jana, Anubhab Majumdar, Sekhar Mandal
Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology Shibpur, India
Email: (prerana.jana, anubhabmajumdar93)@gmail.com
sekhar@cs.iiests.ac.in

Bhabatosh Chanda
Electronics and Communication Sciences Unit
Indian Statistical Institute, Kolkata, India
E-mail : chanda@isical.ac.in

***Abstract*—The abstract goes here.**

I. Introduction

Here goes the introduction and previous work

II. Proposed Work

Major steps involved in the automated generation of search-able PDF format are i) Segmentation of displayed chemical equation; ii) Recognition of various chemical symbols present in chemical equations; iii) Optical character recognition of each reactant; iv) Auto correction of reactants and the chemical equation itself; v) Generation of chemical equation in search-able PDF format.

The details of the above steps are given in the following subsections.

A. Segmentation of displayed chemical equation

Segmentation of displayed equations and then extracting only the chemical equations from a document image has been done using the methodology proposed in [1].

1) Text Line Segmentation: We have taken the horizontal projection profile of the document image to segment the text line. The image and its horizontal projection profile is shown in Fig.1.

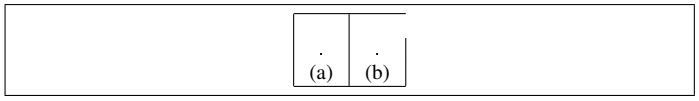


Fig. 1: (a). A scanned document image. (b). Horizontal projection profile.

Each disjoint component of Fig.1b is a line of the document image. This provides the bounding box coordinates of each line which is essential for segmentation.

2) Word blob formation: This requires performing morphological closing operation on the binary document image using a line structural element of suitable length. The length is determined by measuring white space between each adjacent component in the image and plotting the histogram. We will notice one high peak and a low peak separated by a valley which indicates the ideal structural element length. The high peak and low peak corresponds to gap between characters and gap between words respectively because statistically number of characters is much greater than number of words in a document. The histogram is shown in Fig.2. Therefore, the length of structural won't be greater than the gap between two words but certainly more than gap between two letters of a word.

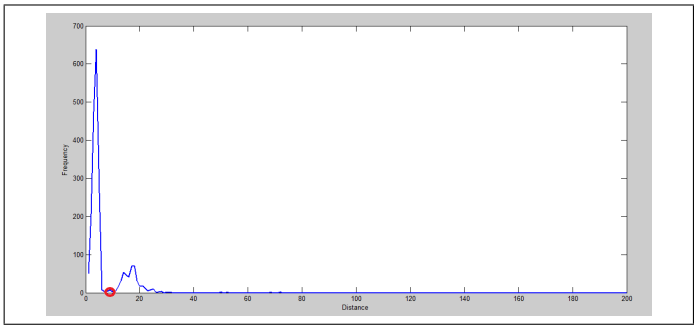


Fig. 2: Histogram of white space between adjacent component. The red circle shows the ideal length of the structural element.

3) Operator identification: After formation of word blob, each component is labeled, the region corresponding to each component is considered and the number of connected component(s) in the region is counted. The chemical symbols we tried to identify are {+, -, $\rightarrow$, $\leftarrow$, $\rightharpoonup$, $\leftharpoondown$, $\uparrow$, $\downarrow$} and their blob will have one connected component only. Also these symbols will have Euler number 1. Using these rules we can filter out most words and few alphanumerics like 4, a, A etc. However along with the desired symbols few other symbols like (,), {, } etc. will remain(see Fig.3a). An one class svm is used to classify the needed symbols from the unwanted(see Fig.3b). The feature vector used are:

- Aspect ratio of each component.
- Density, i.e, $\frac{\#object_pixels}{\#background_pixels}$.
- Horizontal and vertical projection profile for resized image is considered and for each profile the 2^{nd} and 3^{rd} order moment is computed.
- For each profile the location and magnitude global maximum is computed.
- The perimeter of the image.

One class svm is used because the negative class is infinite. Its impossible to determine which character will remain as single character after removal using Euler number.

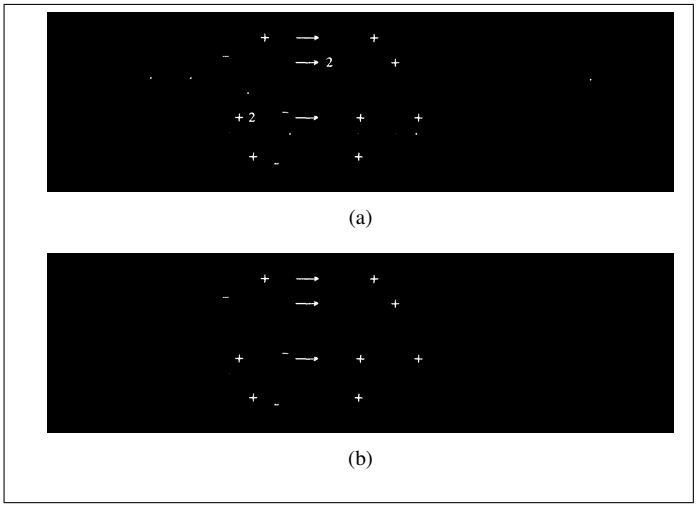


Fig. 3: (a). Single characters after removal using Euler number. (b). After classification using one class svm.

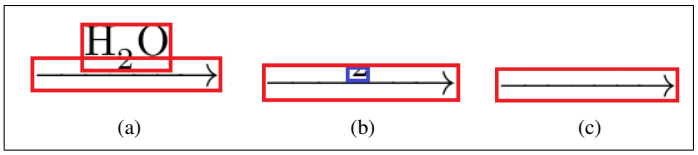
4) Segmentation of Displayed Equations: The lines having at least one symbol identified from the step above are either in line or displayed equation. To determine the displayed equation we use the blobbed image. We count the number of operators and operand in a line and identify a line as Displayed Equation(DE) if #operands $\leq$ 2 x #operands since a operator can have at most 2 operands on its either side.

5) Classification of segmented DE zones: A hash-map of all the periodic element is made. The chemical symbol of all periodic element follows the regular expression [A-Z][a-z]*. For each operand in DE, we take each connected component and input it in OCR. All substrings of a operand matching the regular expression are possible periodic elements and they are matched to the hash table. If $\frac{\#matches}{\#extracted} \geq 0.7$, then we consider that DE as Displayed Chemical Equation(DCE). The reason for the ratio not being 1 are i) Limitations of OCR; ii) Touching and broken characters.

6) Improvements to proposed methodology in [1]: Some improvements on the segmentation error has been attempted. For the reactants over the arrow, the main problem is single character extraction. In [1], component count has been done in the bounding box of one blob. Here, if we find multiple components, each component's bounding box is calculated. If one of the component's bounding box is equal to the bounding box of its blob, we remove the components with smaller bounding boxes (See Fig.4). This is based on the positional context that an arrow with or without reactants over or below it will have the same bounding box as that of its word blob.

B. Recognition of various chemical symbols present in chemical equations

A chemical equation is a way of representing a chemical reaction in symbolic form. Chemical equations consists of reactants separated by myriad chemical symbols. The symbols along with their significance are listed below.

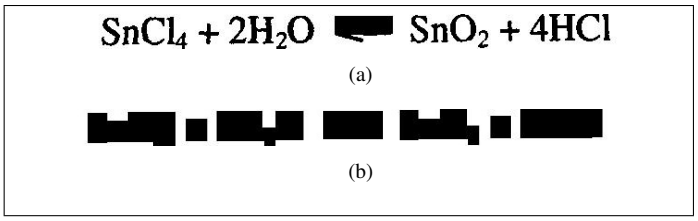


(a) (b) (c)

Fig. 4: (a). Arrow with reactants over it. (b). Bounding Box of the arrow. (c). After removal of smaller components

- + : Separates the reactants
- $\rightarrow$, $\leftarrow$: Separates the reactants from products in irreversible reactions; also denote the direction of reaction
- $\leftrightarrow$, $\rightleftharpoons$: Separates the reactants from products in reversible reactions
- = : Shows stoichiometric equality in chemical equations
- $\uparrow$: Used to denote gaseous compound
- $\downarrow$: Used to denote sediments formed after a reaction

We begin with the extracted displayed chemical equation (DCE) from the step above and run a HRLS algorithm. This results in the coalescing of the chemical compounds into a word blob as shown in Fig.5b.



(a)

(b)

Fig. 5: (a). Original DCE. (b). DCE after closing operation.

All the single components are identified and segregated from the original equation and classified using a decision tree shown in Fig.6.

The function of each node of the decision tree is detailed below.

- #of component : This module counts number of disjoint components in the input symbol.
- Crossing : We measure the number of transitions from object to background pixel and vice versa for each column while moving along the rows and consider the maximum value.
- OCR : The input symbol is run through OCR to positively identify the + symbol. Other symbols return erroneous result.
- Aspect Ratio : Calculates the $\frac{height}{width}$ ratio of input image.
- Distance Transform : The input symbols are closed with a disc structuring element of radius $\frac{m}{2}$ where m is $min(height, width)$ of the input symbol. The distance transform is applied and the location of the global maxima denotes the arrow head; hence determines the direction of the arrow.

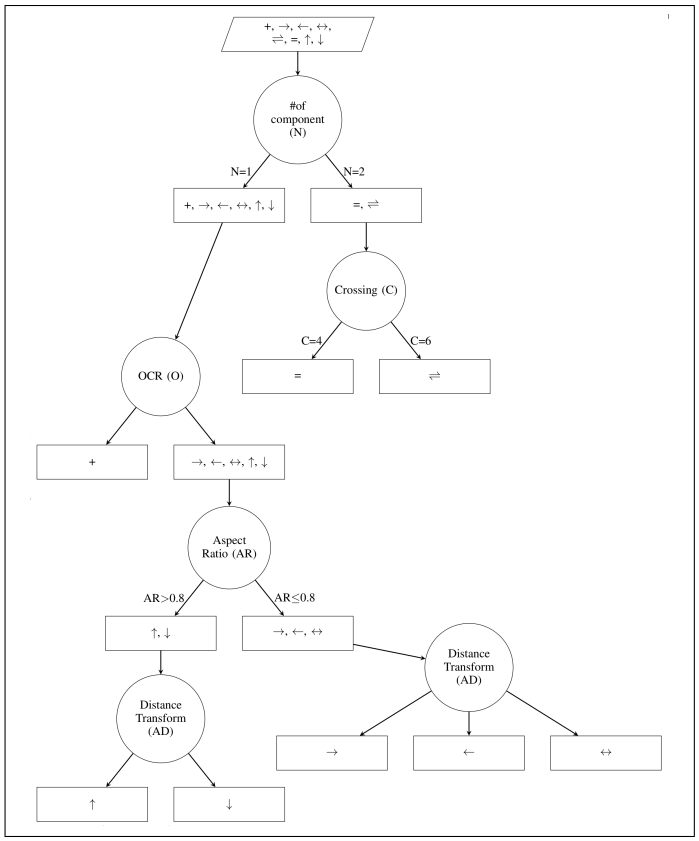


Fig. 6: Decision Tree for Chemical Symbol Classification

Certain chemical element like Carbon, Sulfur etc.(see Fig.7) may also be input to the tree and they get erroneously identified as ↑ or ↓ because characters have AR>0.8.

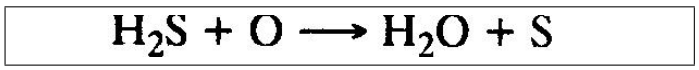


Fig. 7: DCE having element represented by single character

To fix this, we match the symbols classified with the original equation blob image. For each +, →, ←, ↔, ⇌ and = in the classified symbol set, we check its immediate right blob. If the blob has single component in the original equation image, then it must be an single character element because these symbols must be followed by reactants. Also the first blob is checked to see if its a single character element or not. Thus the chemical symbols are successfully segregated and classified from the chemical equation.

C. Optical character recognition of each reactant

After segregating the chemical symbols, rest of the blobs are considered as reactants. Each segment of a displayed chemical equation is divided into three zones; namely upper zone, middle zone and lower zone(see Fig.8). To identify the three zones of a DCE zone, uppermost and lowermost co-ordinates of each connected component below the same DCE zone are also obtained. The median of uppermost coordinate, and median of lowermost co-ordinate of such components in DCE zone are computed. A horizontal line, called the baseline, is drawn through the median of lowermost coordinates of components and this baseline separates the middle zone and lower zone of DCE zone. Similarly, the median of uppermost co-ordinate of the components in the DCE zone generates a horizontal line. This horizontal line, called top line, separates the middle and upper zones of the DCE zone. The subscripts in a DCE zone belong to lower-half of the middle zone and lower zone whereas the superscripts belong to upper zone and upper-half of the middle zone. Based on the location of the components in a DCE zone we have detected the subscripts and superscripts. Each reactant form one word blob. Each character

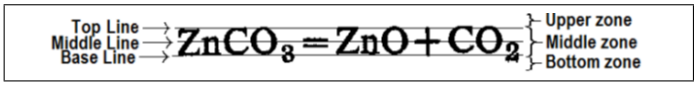


Fig. 8: Presence of different zones in DCE

forming one word blob is run through an OCR engine, Matlab OCR. Each character image is returned by OCR engine in corresponding text format and saved in a string with a space split, $S_{reactant}$ for further processing. For each superscript and subscript, '^' and '_' are inserted before them respectively in $S_{reactant}$ for denoting purposes.

D. Auto correction of reactants in chemical equations

Matlab OCR does not give perfect result for chemical equations which has subscripts and superscripts as well. Hence, to generate the exact equation from the image, auto correction is required in chemical database. First an error map is created based upon our observation on multiple databases (see Table I). Next this table is stored into a hash map where the key is the OCR output and its corresponding value is possible input set. Auto correction is performed based on this error hash map, H.

Each chemical reactant in any equation has the following format- [Coefficient]$^{[0,1]}$[Chemical compound][state]$^{[0,1]}$. Coefficient has numerical values, its regular expression is taken as [2-9]+[0-9]*. The first digit will not be 0 as coefficient cannot start with that. There are 4 states of a chemical compound which are represented by '(s)', '(g)', '(l)' and '(aq)'. To detect if the compound has any state representation, the following regular expression [(][a-z]$^{[1,2]}$[)] is checked in the substring of length 4 extracted from the end of $S_{reactant}$(blank spaces are ignored here). If there is a match, we run the autocorrection algorithm on it and compare the results with 's', 'g', 'l' and 'aq'. If no match is found, the substring extracted from $S_{reactant}$ is a radical, not a state else, we remove the state from the reactant. After extracting coefficient and state, only chemical compound, CC is left in $S_{reactant}$. So, CC is a set containing OCR output of each character of the chemical compound.

The algorithm for autocorrection of each chemical compound is given below. In first step, CC and error map, H are taken as input and all possible combinations of corrected OCR output is produced by $GetAllCombinations$. Next, these $Combinations$ are matched against a nearly exhaustive list of all molecules, chemical compounds, radicals and atoms named $ChemList$. If a perfect match is found, that is considered as the $Corrected$ chemical compound for that CC. But in case of no match, longest common substring(s) (LCS) between $Combinations$ and $ChemList$ are extracted and considered as $SubMatch$. If number of

TABLE I: Error list

correct input	all observed outputs given by OCR
g	8 3 S
n	11 1'1 II 11 X1 Il
O	0
3	8 'E s w
a	3. 3 21 8 El 8.
q	Cl Q
H	1-l 1-1 l-l l-1
2	7 4 Z z
l	1 I
I	l
u	11 U 1l ll 1l
i	1 l I
5	'S
4	A
Z	2
r	l
e	C
s	S

such $SubMatch$(es) is 1 that $SubMatch$ is considered as the $Corrected$ chemical compound for that CC; else they are considered as $PossibleReactants$. All the $Corrected$ compounds are included in the $FinalEquation$.

Algorithm 1 Auto-Correction of each reactant

procedure GETALLCOMBINATIONS(CC,H)
  for all $element(s) \in CC$ **do**
    $InputSet(s) \leftarrow H.Get(element)$
    if $InputSet(s)$ is $NULL$ **then**
      $RETURN$ ▷ Not in Error Map
    else
      if $length(element) = 1$ **then**
        $InputSets = \cup\ [element]$
      end if
    end if
  end for
  $Combinations \leftarrow CartesianProduct(InputSets)$
end procedure

procedure FINDMATCH($ChemList$,$Combinations$)
  for all $Combinations$ **do**
    match with $ChemList$
  end for
  if Match Found **then**
    $Corrected$ = Match
  else
    $SubMatch \leftarrow$ **LCS**($Combinations, ChemList$)
      ▷ LCS : Longest Common Substring
    if NumberOf($SubMatch$)=1 **then**
      $Corrected = SubMatch$
    else
      $PossibleReactants \leftarrow SubMatch$
    end if
  end if
end procedure

Here we have all the possible reactants and try to find out the $FinalEquation$ in the context of the equation itself. Chemical equations have the same elements in the left hand side (LHS) as that in the right hand side (RHS). All the periodic elements follow the regular expression [A-Z][a-z]*. So, for each possiblity, the set of periodic elemnents in the left side of the equation,P_{LHS} and in the right side, P_{RHS} is computed. When the intersection of these two sets are empty, that $PossibleReactant$ is considered as $Corrected$ and included in the $FinalEquation$. But if the above condition comes true for multiple possibilities, we cannot decide which of the possible reactants are actually in the original equation.The algorithm shows multiple $FinalEquation$s. This is considered as an $ERROR$ case.

Algorithm 2 Auto-Correction of equation

procedure GETFINALEQUATION($PossibleReactants$)
  for every possibility **do**
    compute(P_{LHS})
      ▷ P_{LHS} : Set of periodic elements in the L.H.S
    compute(P_{RHS})
      ▷ P_{RHS} : Set of periodic elements in the R.H.S
    if $P_{LHS} \cap P_{RHS} = \emptyset$ **then**
      $Corrected \leftarrow PossibleReactant$
      Include that in the $FinalEquation$
    end if
  end for
  if multiple possibilities get $P_{LHS} \cap P_{RHS} = \emptyset$ **then**
    Multiple $Corrected$ compounds
    Multiple $FinalEquations$
      ▷ $ERROR$
  end if
end procedure

E. Generation of search-able PDF format of chemical equations

The operand list and chemical symbol list is used to form the auto corrected equation. We alternatively put a operand and a symbol starting with operand. Also, for each ↑ or ↓ in the symbol list, we put the next symbol from the list before placing another operand. After the list is formed, we write it in a Tex file with mhchem package. The Tex file is run later to generate the search-able PDF.

III. EXPERIMENTAL RESULT

Here goes the results.

IV. CONCLUSION

The conclusion goes here.

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.