Noname manuscript No.
(will be inserted by the editor)

Generation of Search-able PDF of the Chemical Equations segmented from Document Images

Anubhab Majumdar · Prerana Jana · Sekhar Mandal

Received: date / Accepted: date

Abstract PDF format of scanned document images is not searchable. OCR tries to remedy this adversity by converting images or PDF files into editable and searchable data, but it has its own limitations in presence of equations - both mathematical and chemical. OCR system for mathematical document images is already a major research area and has provided successful result. However, chemical equation segmentation has been a less ventured road. In this paper we present a novel method for automated generation of searchable PDF format of segmented chemical equations from scanned document images by performing chemical symbol recognition and auto-correction of chemical compounds. We use existing OCR systems, pattern recognition, contextual data analysis and a standard LaTeX package to generate the chemical equation in searchable PDF format. The effectiveness of the proposed method is demonstrated by testing it on 234 document images.

Keywords Chemical equations, mathematical symbols, morphological operation.

1 Introduction

Text-based keywords are used for retrieving documents from WWW using search engines like Google. A large number of documents are being digitized today for the purpose of archival analysis, transmission and browsing. The existing OCR systems show high accuracy in interpreting text portions, but

Anubhab Majumdar
IIEST Shibpur India
E-mail: anubhabmajumdar93@gmail.com

Prerana Jana
IIEST Shibpur India
E-mail: anubhabmajumdar93@gmail.com

fail to properly process other components like graphics, half-tones, chemical and mathematical equations.

A few studies [3], [4], [5] are directed toward math-symbol or math equation recognition assuming that the math-zones are already marked. We, on the other hand, contend that a better approach is to segment the chemical equations from the mixed material thereby helping the future OCR activity to focus its processing only on specific content.

There are some methods that are used to reconstruct chemical molecules form scanned imge. They have used chemical datasets. Algorri et al. [7, **?**] proposed a system that reconstruct chemical molecules from scanned document. They have used connected component analysis and their own vectorisation algorithm for character recognition. Connected components that are not recognised by the OCR engine, are used to produce a graph of vectors. A rule based approach reconstructs the molecule description from the vector graph and the character information. ChemReader [9] starts with connected component. Alphanumerics are recognised using using the GOCR open source OCR tool. Graphical components are identified using Hough transforms, corner detection and other bespoke algorithms.

In this paper we propose fully automated segmentation and detection technique of chemical equations present in heterogeneous document images followed by generation of LaTeX file of the segmented equations.

2 Proposed Work

The proposed method consists of four distinct steps and they are as follows: (i) Segmentation of displayed equations; (ii) Identification of chemical equations; (iii) Auto correction of chemical compounds and the equation; and (iv) Generation of chemical equation in search-able PDF format.

2.1 Segmentation of displayed equations

A skew free heterogeneous binary image is the input of the proposed algorithm. The tables and graphics are extracted out from the heterogeneous documents using the methods described in [2] and [1]. The rest of the document contains normal text lines and displayed equations if any. Our main motivation is to classify chemical and non-chemical equations, we restrict our study to displayed equations only.

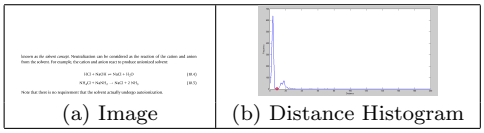
Major steps of segmentation of displayed equations (DE) are - (i) Text line segmentation; (ii) Blob formation using morphological tools; and (iii) DE zone extraction.

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

2.1.1 Text line segmentation

As the document image is skew free, the horizontal projection profile of the document page is taken to segment the text line.

2.1.2 Creation of word blobs



(a) Image (b) Distance Histogram

Fig. 1: A document page and its distance histogram.

This is done by coalescing the characters in a word using morphological closing operation. Such character coalescing process depends on the accuracy in detecting the normal character gap. The component analysis is done first. Let C_a and C_b be the two consecutive connected components in a text line. L_b is the left most x coordinate of C_b and R_a is the right most x coordinate of C_a. A distance function D is defined as $D = L_b - R_a$. The histogram of D is obtained (see Fig. 1). The blob formation requires information on inter-word gap. The distance histogram is a multi-modal histogram. The first peak is corresponding to the character gaps. Our intention is to find out character gaps in running texts of a document page so that we can combine the consecutive characters into a single blob. Hence, we consider the upper boundary (l) of the first hump as the length of structuring element. Morphological close operation with a structuring element of size ($l \times 1$) will form the blobs. The result of blob formation is shown in Fig. 2.

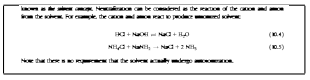


Fig. 2: Blob formation of the image shown in Fig. 1(a).

2.1.3 DE zone extraction

We have considered the set of operators that is commonly used both in chemical equations as well as mathematical equations to fulfil our aim to classify displayed zones containing chemical and non-chemical equations. After blob formation, small component like dots of *i and j* are eliminated on the basis of area. The region, corresponding to each blob is considered from the original image and the number of connected component(s) present in that region

is counted. If the number of components is more than one that blob is not an operator and is removed from the blob image. The remaining components in the blob image are operators along with some alphanumerics like a, A, (, etc. The logical AND operation is performed between the blob image and the original image. The Euler number of the operators that we have considered is 1 (one) and based on this feature some of the alphanumerics are discarded. This image is denoted by I_s. Width (w) and height (h) of each component in I_s are determined. The components are divided into two classes based on the ratio of w, h; (i) *thin class* ($\frac{w}{h} > 2$) and (ii) *regular class* ($0.8 < \frac{w}{h} < 1.3$). Some instances of the *thin* and *regular classes* are (-, $\rightarrow$, $\leftarrow$, $\leftrightarrow$, $\rightharpoonup$, $\rightharpoondown$, $\sim$, etc.) and (+, $\bullet$, X $\times$, etc) respectively.

The number of *on pixels*(N_p) and number of *off pixels* (N_w) within each bounding box of the thin components are calculated. Morphological opening operation is performed on *thin class* with a line like structuring element (SE) of length $\frac{w}{2}$. If the output of the opening operation has a single component, then thinning is performed on the corresponding original component. Next, the number of end points of each thinned component is checked, if the number of end points is two and $\frac{N_p}{N_w} > 0.8$, then the component under test is a '-' sign. If the number of end points is greater than two, then element is an arrow. We need to identify the direction of the arrowhead for its correct representation in LaTeX file. The direction of arrowhead is identified by measuring the height of the arrow element near its two ends.

From the *regular class* our objective is to identify the '+' sign and it is detected using the following rules:
(i) Horizontal and Vertical Projection Profile of each component is considered; (ii) Ratio of $maxProj$, D and ratio of $maxIndex$, (M_l) are obtained where maxProj is the maximum horizontal or vertical projection value, maxIndex is the index of maxProj closest to M_l and D is width (height) for horizontal (vertical) profile. Here, M_l is y (x) coordinate of the middle point for horizontal (vertical) profile. If $0.9 < \frac{maxProj}{D} \leq 1$, $0.9 < \frac{maxIndex}{M_l} \leq 1.2$ and $\frac{N_w}{N_p} > 3$, then the component under test is a '+' sign. Where N_w and N_p are number of *on pixels* and *off pixels* along the diagonal direction of the bounding box of the component. All the threshold values used here are selected based on our empirical study on 234 images.

To detect '=' or '$\rightleftharpoons$' one extra step is required. For each '-' and right-arrow symbol, a rectangular mask is placed below the symbol to check if there is '-' and left-arrow respectively within the mask; if present, they are considered to form either an '=' or '$\rightleftharpoons$' sign. Let the length of the symbol be l. The area of the mask is $(l \times l/2)$.

The horizontal line separating the numerator and the denominator is identified as its length is greater than the median length of the operators. Two windows are placed above and below the separating line to merge all the components within the windows with the separating line to form a single logical line. Otherwise, they would be treated as three consecutive text lines and we will not be able to associate the intermediate math-symbols (+, -, =) to a

single expression. The area of the window is (length of the separating line)× (twice the median width of the text lines).

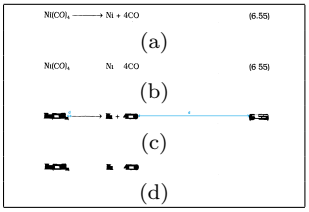


Fig. 3: The output of H-RLSA on portion of an image (a) a part of an image; (b) same part without operators; (c) result of H-RLSA operation; (d) after equation number removal.

Initially, all the text lines consisting at least one operator are considered candidate displayed equations (CDE). The operators are eliminated from CDE. The upper boundary (u_v) of the second hump of distance histogram (Fig. 1(b)) is obtained which represents the word gaps in the text line. For each CDE zone Run Length Smoothing Algorithm in horizontal direction (H-RLSA) is carried out. If the distance between two neighbouring components is less than u_v, it means they belong to a same word and are merged by H-RLSA. H-RLSA has a similar effect as of dilation of black areas in horizontal direction. The characters in a word are dilated and coalesced to the other characters of the same word. The output of H-RLSA is shown in Fig. 3.

Equation numbers are common in the displayed equation zones. These numbers have to be removed because for each CDE we have counted the number of operators and corresponding other components in the output of H-RLSA. If the number of components $\leq 2\times$ number of operators, then the CDE is considered displayed equation; otherwise some embedded formulae/equations may exist in the line. To eliminate the equation number from the output of H-RLSA the operators are moved to the output of H-RLSA and the component analysis is done. From both ends distance (d) (see Fig. 3(c)) between the first two consecutive components is measured and if $d > 5\times\ u_v$, then the first component from the end is considered the equation number and is removed. (see Fig. 3(d).

2.2 Identification of chemical equations

Each segment of a displayed equation is divided into three zones; namely upper zone, middle zone and lower zone (see Fig. 4). To identify the three zones of a DE zone, uppermost and lowermost co-ordinates of each connected component below the same DE zone are also obtained. The median of uppermost coordinate, and median of lowermost co-ordinate of such components in DE 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 DE zone. Similarly, the median of uppermost co-ordinate of the components in the DE zone generates a horizontal line. This horizontal line, called top line, separates the middle and upper zones of the DE zone.

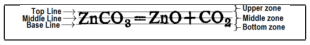


Fig. 4: Different zones of DE equation

The subscripts in a DE 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 DE zone we have detected the subscripts and superscripts and are separated from the DE zone. The operators are also separated from DE zone.

Now, each displayed equation is an input to an OCR of MATLAB R2014a. The OCR returns each DE zone as a text string. We made a dictionary out of all the elements in the periodic table. An important observation is that an element always starts with a capital letter. Using this property, an element can be expressed by a regular expression [A-Z][a-z]*. It means an element's symbol starts with an upper case letter and may or may not have one or more lower case letters. We have designed a parser to extract the sub-string matching the regular expression mentioned above with the following grammar:
$start \rightarrow capital.follow$
$follow \rightarrow small.follow| \in$
$capital \rightarrow A|B| \ldots |X|Y|Z$
$small \rightarrow a|b| \ldots |x|y|z$
Each of the substring returned by the parser is matched against the aforesaid dictionary and if it is a positive match then that substring is considered as a symbol of the chemical element. Let us consider, the number of substrings extracted from the OCR output by the parser is n and the number of positive matches the aforementioned dictionary is m. If m:n ratio is more than a threshold value β then this DE is considered a Chemical Equation. This threshold (β) is set to 0.7 by running our algorithm on our dataset containing 1390 displayed equations. The reason for the ratio not being 1 are (i) Limitations of OCR; (ii) Touching and broken characters.

The $\uparrow$ and $\downarrow$ are frequently used in chemical equation to represent the state of compounds and are thus important to detect. Let each chemical equation be denoted by C_e. Each C_e zone is AND with I_s which produces an output I_o. I_o may contain operators, single character elements and ($\uparrow$, $\downarrow$). The operators are removed from I_o as they are already identified. For the rest of the components in I_o, we have checked whether the component is the starting character of the chemical equation, if not, then its immediate left neighbour in C_e is checked. If the left neighbour is not an operator, then then component under test is $\uparrow$ or $\downarrow$. The arrowhead direction is determined in the same way as right/left arrow in 2.1.3.

2.3 Auto correction of chemical compounds and the equation

Next, auto correction is performed based on the chemical context as existing OCRs can not produce perfect result in case of chemical equations. The output of H-RLSA (see Fig. 3 (d)) is taken as input here. Each character within a word blob is an input to the the OCR and the corresponding output is stored in a cell and these cells form a string, $S_{chemical}$ for each word blob. For each superscript and subscript, '^' and '_' are inserted before them respectively in $S_{chemical}$.

First, an error map is created based on the observation of OCR outputs of 280 chemical equations consisting of 1022 compounds (see Table 1). Next, this table is stored into a hash map H where the key is the OCR output and its value is the possible input set. For example, if '8' is an erroneous OCR output for inputs 'g', '3'and 'a' (see Table 1) then, in the hash map H, key is '8' and its corresponding value is [g, 3, a] .

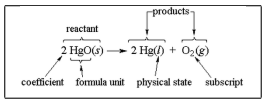


Fig. 5: Different components of a chemical equation.

Each chemical compound in any equation (see Fig. 5) has the following format- [Coefficient]$^{[0,1]}$[Formula Unit][state]$^{[0,1]}$. Auto correction of the OCR output corresponding to each word blob includes the following steps - (i) Coefficient extraction; (ii) State separation; (iii)Auto correction of the formula unit; and (iv) Auto correction of the entire equation using Context Table.

2.3.1 Coefficient Extraction

Coefficient extraction is done by matching its regular expression $[2-9]^{+}[0-9]^{*}$ at the beginning of $\mathrm{S}_{chemical}$ as it has numerical values.

2.3.2 State separation

There are 4 physical states of a chemical compound which are represented by '(s)', '(g)', '(l)' and '(aq)'. To detect the physical state of the compound, regular expression [(][A-Za-z]$^{[1,2]}$[)]] is used and the checking starts from the end of $\mathrm{S}_{chemical}$. The matched substring (S) is extracted from $\mathrm{S}_{chemical}$ and Algorithm 1 is run. In this algorithm, S and H are taken as input and all possible $Combinations$ of corrected OCR output is produced by $GetAllCombinations$ (See Fig. 6). These $Combinations$ are compared with 's', 'g', 'l' and 'aq'. If no match is found, the substring extracted from $\mathrm{S}_{chemical}$ is a radical, not a state; else, we separate the state from the compound.

2.3.3 Auto correction of the formula unit

After extracting coefficient and state, only the formula unit is left in $\mathrm{S}_{chemical}$. The algorithm for autocorrection of each formula unit is done in two steps (See Algorithm 1 and 2). First, Algorithm 1 is performed on the formula unit. Next, the output of Algorithm 1($Combinations$) are matched against a nearly exhaustive list of all molecules, chemical compounds, radicals and atoms namely $ChemList$ collected from Wikipedia If a perfect match is found, that match is considered as the $Corrected$ formula unit (See Fig. 7). But in case of no match, we go for longest common substring(s) (LCS) match ($SubMatch$).If there is only one $SubMatch$ then the corresponding formula unit in $ChemList$ is considered as the $Corrected$ formula unit; else the $SubMatch$es are considered as $PossibleFormulaUnit$s.

2.3.4 Auto correction of the entire equation using Context Table

Here, we have all the possible formula units and try to find out the $FinalEquation$ in the context of the equation itself. Algorithm 3 takes all $Corrected$ and $PossibleFormulaUnit$s and returns the $FinalEquation$ by forming the Context Table. Chemical equations have

Table 1: Part of the Error list

correct input	all observed outputs given by OCR
g	8 S
O	0
3	8 'E s w
a	3. 21 8 El 8.
l	1 I
s	S

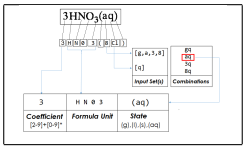


Fig. 6: Extracting the formula unit, numeric coefficient and physical state from a chemical compound.

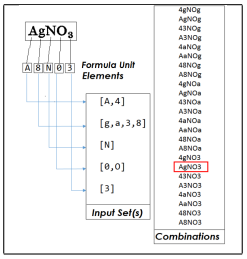


Fig. 7: Example of autocorrection of a formula unit.

the same periodic elements in the left hand side, called *Reactants* as that in the right hand side, called *Products*. All the periodic elements follow the regular expression [A-Z][a-z]*. So, for each $PossibleFormulaUnit$, the set of periodic elements in the *Reactants*, P_R and in the *Products*, P_P are computed and stored in the $ContextTable$. When the set difference of P_R and P_P in the table is empty, that $PossibleFormulaUnit$ is considered as *Corrected* and included in the $FinalEquation$ (See Fig. 8). But if the above condition comes true for multiple possibilities, we cannot decide which of the possible formula units are actually in the original equation.The algorithm shows multiple $FinalEquation$s. This is considered an *ERROR* case. An example of formation of context table is demonstrated in Fig. 8. The co-effiecient and state (if any) are added with their correspodning *Corrected* formula unit. The $FinalEquation$ is then converted to LaTeX format using *mhchem* package.

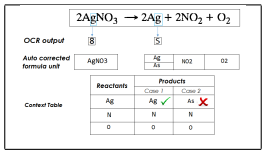


Fig. 8: Formation of context table.

Algorithm 1 Get All Combinations from the Error Hash Map

```
1: procedure GetAllCombinations(S,H)
2:     for all element(s) ∈ S do
3:         InputSet(s) ← H.Get(element)
4:         if InputSet(s) is NULL then
5:             RETURN                                   ▷ Not in Error Map
6:         else
7:             if length(element) = 1 then
8:                 InputSets = ∪ [element]
                ▷ Element might be correct output but still in the error list for other inputs
9:             else
10:                Ignore
                                  ▷ Input is one character, output length > 1 means error
11:            end if
12:        end if
13:    end for
14:    Combinations ← CartesianProduct(InputSets)
15:    Return Combinations
16: end procedure
```

Algorithm 2 Find Match between ChemList and Combinations derived from Algorithm 1

```
1: procedure FindMatch(ChemList,Combinations)
2:     for all Combinations do
3:         match with ChemList
4:     end for
5:     if #(Match Found) = 1 then
6:         Corrected ← Match
7:         Return Corrected
8:     else if #(Match Found) = 0 then
9:         SubMatch ← LCS(Combinations, ChemList)
                                          ▷ LCS : Longest Common Substring
10:        if NumberOf(SubMatch)=1 then
11:            Corrected ← SubMatch
12:            Return Corrected
13:        else
14:            PossibleFormulaUnit(s) ← SubMatch
15:            Return PossibleFormulaUnit(s)
16:        end if
17:    else
18:        PossibleFormulaUnits ← Matches
19:        Return PossibleFormulaUnit(s)
                                                  ▷ Multiple matches
20:    end if
21: end procedure
```

3 Experimental Result

We have implemented our algorithm in MATLAB 8.3.0.532 (R2014a) in a PC (Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz running Windows 8). The proposed method has been tested on a dataset consisting 234 document pages. Out of 234 pages 50 pages are taken from ICDAR 2013 Math-zone segmentation datasets and other document pages are scanned from different Mathematics and Chemistry books. The summary of the experimental results

Algorithm 3 Auto-Correction of the entire equation using chemical context table

procedure GETFINALEQN($PossibleFormulaUnit$,$Corrected$)
  Include all $Corrected$ units in $FinalEquation$
  $Count \leftarrow 0$
  for every $PossibleFormulaUnit$ **do**
    compute(P_R)
      ▷ P_R : Set of periodic elements in Reactants
    compute(P_P)
      ▷ P_P : Set of periodic elements in Products
    if $P_R - P_P = \emptyset$ **then**
      $Corrected \leftarrow PossibleFormulaUnit$
      Include that in the $FinalEquation$
      $Count \leftarrow Count + 1$
    end if
  end for
  if $Count \geq 2$ **then**
    Multiple $Corrected$ compounds
    Multiple $FinalEquations$
      ▷ $ERROR$
  end if
  Return $FinalEquation(s)$
end procedure

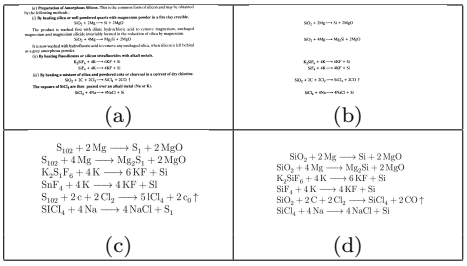


Fig. 9: Experimental result; (a) Input image; (b) Segmented Chemical equation; (c) OCR output in PDF format; (d) Output of the proposed method.

is shown in Table 2. Out of 3406 chemical compounds in the test dataset, 114 were partially corrected and 52 could not be corrected at all. The overall accuracy of autocorrection is 95.12%. These results are quite encouraging. See the sample image (Fig. 9(a)). Corresponding segmented displayed chemical equations are shown in Fig. 9(b). Fig. 9(c) shows the direct OCR output where 'i' has been wrongly identified as 'l', 'I' and '1' (for Si in all the lines of Fig. 9(c)). Similarly 'O' results in '0' (line 1,2,3). 'S' sometimes is detected as '5' (line 5). Our auto correction algorithm remedies these issues. Fig. 9(d) demonstrates the effect of our auto correction algorithm. This algorithm is targeted towards chemical equation with linear representation. It fails when the chemical equation contains some text such as *and*, *or* etc between two displayed equation in the same line. Also, when the chemical compound is written in formats such as $(Na_2SiO_3)_n$, only Na_2SiO_3 is detected based on $ChemList$. Some equations have conditions (pressure, temperature) written over the arrows. We have not ventured in that yet. But the error case that has been mentioned in Algorithm 3 has very less probability of occurrence. Hence it is ignored.

Table 2: Summary of Experimental Results

#Total DEs	1390
Operator recognition	98.83%
DE segmentation accuracy	98.63%
Chemical DE Classification Accuracy	98.83%
Auto correction accuracy	95.12%

4 Conclusion

We have presented an automated chemical equation segmentation and chemical context based auto correction system that is able to provide the exact searchable format of linear chemical equations in any document image. The experimental results demonstrate the efficiency of our proposed method. This work leads to several research avenues. Chemical context horizon can be widened. Auto correction on non-linear or bond structure representations of chemical equations could be ventured in.

References

1. S. P. Chowdhury, S. Mandal, A. K. Das, and B. Chanda, *Segmentation of Text and Graphics from Document Images*, In Proc. of ICDAR, pp. 619623,2007.
2. S. Mandal, S. P. Chowdhury, A. K. Das, and B. Chanda, *A simple and effective table detection system from Document Images*, IJDAR, Vol. 8(2), 172182, 2006.
3. D. Blostein and A. Grabavec, *Recognition of Mathematical Notation*, Handbook of Character Recognition and document Image Analysis, 577–582, 1997.
4. K-F. Chan and D-Y. Yeung, *Mathematical Expression Recognition: A Survey*, IJDAR, Vol. 3, no: 1, 315, 2000.
5. U. Garain and B. B. Chaudhuri *An OCR of Printed mathematical Expressions, Digital Document Processing*, Ed: B. B. Chaudhuri, Advances in pattern Recognition, 235259 , 2007.
6. A. Fujiyoshi, M. Suzuki, S. Uchid, *Grammatical Verification for Mathematical Formula Recognition Based on Context-Free Tree Grammar*, Mathematics in Computer Science, 279–298, 2010.
7. M. E. Algorri, M. Zimmermann, C. M. Friedrich, S. Akle, and M. Hofmann-Apitius, *Reconstruction of chemical molecules from images*, In Proc. 29th Annual International IEEE Conference on Engineering in Medicine and Biology Society, 4609–4612, 2007.
8. M. E. Algorri, M. Zimmermann, and M. Hofmann-Apitius, *Automatic recognition of chemical images*, In pro. Eighth Mexican International Conference on Current Trends in Computer Science, 41–46, 2007.
9. J. Park, G. R. Rosania, K. A. Shedden, M. Nguyen, N. Lyu, and K. Saitou, *Automated extraction of chemical structure information from digital raster images*, Chemistry Central journal, vol. 3(1), 2009.
10. A. K. Jain, J, Mao, K. M. Mohiuddin *Artificial Neural Network: A Tutorial*, Computer (Volume:29 , Issue: 3), Mar 1996