International Journal of Grid and Distributed Computing
Vol. 11, No. 11 (2018), pp.13-28
http://dx.doi.org/10.14257/ijgdc.2018.11.11.02

Improved Results of an OCR via NLP using MapReduce Framework

Amine El Hadi[1], Rachid El Ayachi[2], Mohammed Erritali[3] and Mohamed Baslam[4]

[1,2,3,4]Laboratory of Information Processing and Decision Support, Faculty of Sciences and Technics, Sultan Moulay Slimane University, Beni Mellal, Morocco
[1]elhadi.amine@gmail.com, [2]rachid.elayachi@usms.ma, [3]m.erritali@usms.ma, [4]baslam.med@gmail.com

Abstract

This work deals with the orthographic correction of the text that appears in many areas of natural language processing. Optical Character Recognition (OCR) systems are able to convert printed images to written text, but their results still need to be improved. In this context, several correction techniques were presented and their numerical results were discussed. A comparative study between the technique of the noisy channel model and the proposed technique is carried out. These techniques are applied in a Big Data system using the Hadoop framework working with the MapReduce programming model and the HDFS file system to improve the results obtained by the OCR system. The numerical results obtained and presented showed the efficiency and the performance of the technique adopted.

Keywords: *Natural Language Processing (NLP), Spell Correction, Automated Processing, Optical Character Recognition, Tesseract, Big Data, Hadoop, MapReduce, HDFS*

1. Introduction

Information processing techniques are currently undergoing a very active development in connection with computer science and have a growing potential in the field of human-machine interaction. The writing will remain one of the major foundations of civilizations and the mode of excellence of conservation and transmission of knowledge. The recognition of writing (RW) is a vast field that constitutes a subset of the systems of the recognition of the forms (RF). These systems are the first step of a process of comprehension of our universe in the global framework of the man-machine communication. The recognition of handwriting or print is still a subject of research and experimentation, the problem is not yet fully solved although we know how to achieve fairly high rates in some applications for which the vocabulary is limited either the cast is unique or limited in number.

There are several areas for which handwriting recognition is applied with some success: Optical Character Recognition (OCR) systems, automatic mail sorting, automatic administrative file processing, survey forms, or the registration of bank checks. In some countries, they have not yet computerized all the documents of the citizens, it is not easy to rewrite all the information of the citizens word by word in the machines, at this level we see the interest of the OCR systems, which are our field of study in this article, these OCR systems that aim to transform a physical document into a digital document [1]. For example, one may want to search for content or process the information contained in these documents, hence the interest of extracting their content. But when it comes to converting scanned documents from image mode to optical character recognition (OCR) text mode, it

Received (August 12, 2018), Review Result (October 1, 2018), Accepted (October 15, 2018)

ISSN: 2005-4262 IJGDC

does not always guarantee documents that match the originals. What constitutes our problem in this article, is to draw our objective which is to make a correction on the results obtained by an optical character recognition system (OCR).

The rest of the paper is structured as follows: Section 3 describes the general organization of a recognition system. In the Section 4 and 5, we present the general OCR system and our proposed system structure which contains at most one result correction module, we describe the approaches of correction of the adopted texts which are two in number; the first is based on the noisy channel model, while the second is based on our proposed method. In the Section 6 we explain the big data challenges by Hadoop. Then in Section 7 a comparative study is mentioned which is based on a set of numerical results, this study shows the performance and the success of the proposed character recognition system including the correction module. Finally, the conclusion is shown in Section 8.

2. Methodology

Character recognition is not an easy task; there are several problems that can occur that makes the role of OCR more complicated. Among these problems are:

- The quality of the document: a document photocopied several times is more difficult to process than the original copy. Writing can become degraded with missing parts of the text.
- Printing: The type of printer used can affect the quality. An ink-jet printer can introduce ink tasks and a spreading of characters, a laser printer can generate lines or backgrounds...
- Acquisition: the adopted acquisition techniques can generate defects (inclination, noise...).
- Separation: Some documents contain text and graphics, the OCR must be able to locate them.
- Tilt correction: For tilted text, the angle of tilt must be detected and corrected.
- Overlap: It is difficult to segment overlapping characters.

3. General Organization of a Recognition System

The recognition of documents is typically considered as a recognition problem of [3]. Generally, the character recognition system encompasses a set of steps that successively execute Figure 1.

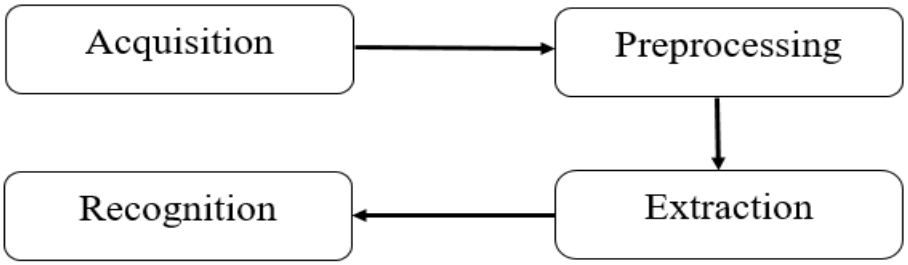

Figure 1. Document Recognition System

4. OCR System Example

Tesseract was developed as proprietary software in the Hewlett Packard laboratories in Bristol, England and Greeley, Colorado, between 1985 and 1994, with some

modifications made in 1996 to the port to Windows and some migration from C to C + + in 1998. Much of the code was written in C, then a little more was written in C ++. Then, all the code was converted to compile with a C ++ compiler. Very little work was done in the following decade. It was then published in open source in 2005 by Hewlett Packard and the University of Nevada, Las Vegas (UNLV). The development of Tesseract has been sponsored by Google since 2006.

Tesseract is considered today one of the most powerful open source optical character recognition software, but of course, the recognition rate of this software remains to be improved by the researchers.

For this purpose, this project of end of studies, is to find a solution to improve the result of Tesseract, based on printed images of the English language.

In the next parts we will quote the different approaches found to correct the result of an OCR, but these results remain non-performing, and give results close to the results of Tesseract, and in the end we present a solution that gives of the best results by making a comparison with the different approaches found in the literature.

5. Proposed System Structure

In this work we worked with Tesseract as an optical character recognition application, based on the results obtained by Tesseract, we begin our objective to correct these results in order to improve them, they are suitable with the content of the image introduced at the acquisition phase.

To summarize, our system encompasses these different steps that are successively executed Figure 2.

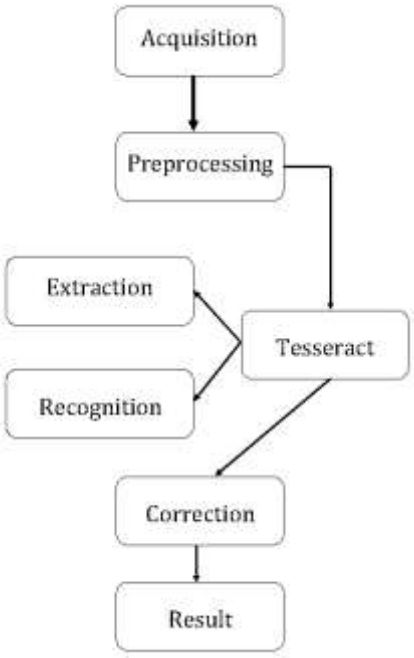

Figure 2. Correction System for Documents Recognition

5.1. Acquisition

Acquisition is the first phase in a document recognition system, this phase allows the conversion of the paper document (text) in the form of a digital image (bitmap). This step is important because it is concerned with the preparation of the documents to be entered, the choice and the parameterization of the input equipment (scanner), as well as the format of storage of the images.

5.2. Preprocessing

The pre-processing is the second phase of a recognition system, it's applied after the acquisition phase, it consists in preparing the data from the sensor to the next phase.

Before passing the image to the next step and introduce the OCR (Tesseract) system, some functions must be applied to the image obtained in the acquisition phase. This phase includes:

- Gaussian blur filter.
- Sobel filter.
- Binarization with the Otsu method.
- Morphological closure.

5.2.1. Gaussian filter

A Gaussian blur is the result of the blurring of an image by a Gaussian function. It reduces the image noise and reduces details. The visual effect of this blur technique is a smooth blur resembling that of looking at the image through a translucent screen.

The equation of a Gaussian function in two dimensions is as follows:

$$G(x,y) = \frac{1}{2\Pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

Figure 3 and 4 illustrates the application result of the Gaussian filter.

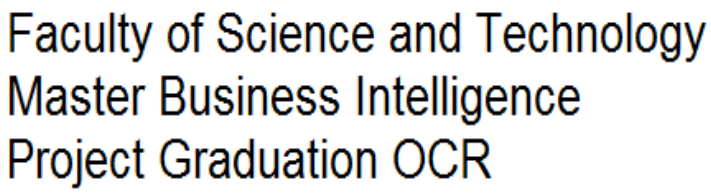

Figure 3. Before Gaussian Application

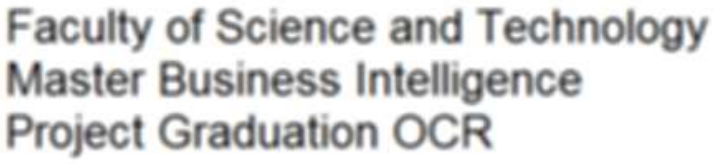

Figure 4. After Gaussian Application

5.2.2. Sobel Filter

The Sobel filter uses convolution matrices. The matrix (usually of size 3x3) undergoes a convolution with the image to calculate approximations of the horizontal and vertical derivatives. Let A be the source image, G_x and G_y two images which in each point contain approximations of the horizontal and vertical derivative of each point. These images are calculated as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \quad (2)$$

And

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \quad (3)$$

In each point the approximations of the horizontal and vertical gradients can be combined as follows to obtain an approximation of the gradient norm in order to create the final image:

$$G = \sqrt{G_x^2 + G_y^2} \quad (4)$$

An example of how to use the Sobel filter is shown in Figure 6.

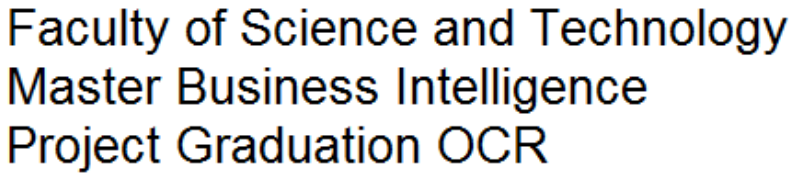

Figure 5. Before Sobel filter application

Figure 6. After Sobel Filter Application

5.2.3. Binarization with Otsu Method

Otsu method is a traditional thresholding method which belongs to the global thresholding techniques**Error! Reference source not found.**.

Figure 8 shows the result of binarization by applying the Otsu method.

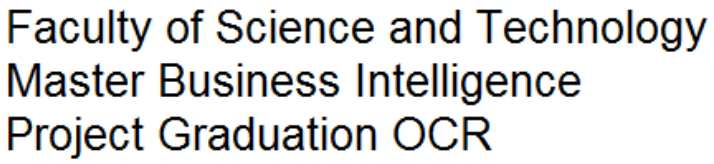

Figure 7. Before Binarization

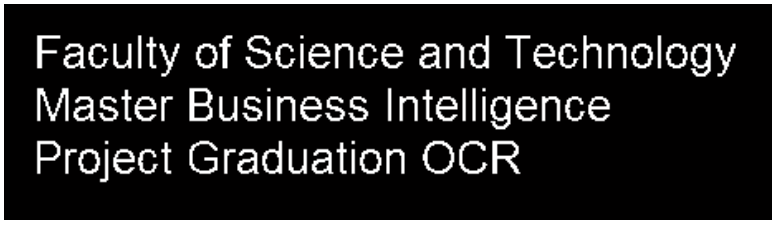

Figure 8. After Binarization

5.2.4. Morphological Closure

The set mathematical morphology treats binary images and calls upon the theory of sets. It uses a set S of center x, of known geometry and size, called a structuring element.

As in convolution, we have a large input image and a small size mask and we compute at each point the convolution by browsing the input image to have the output image.

The chosen structuring element is moved so that its center x passes through all the positions in the binary image to be analyzed. The closure (Figure 9) is a composite operator that combines a dilation followed by erosion.

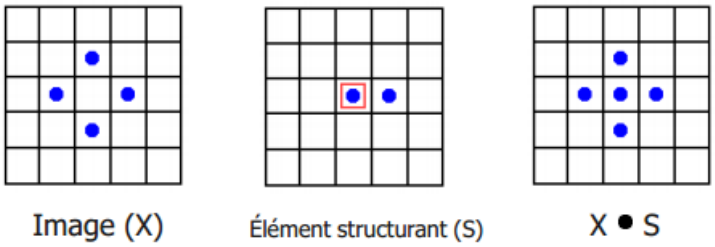

Figure 9. Morphological Closing

An example of execution of the morphological closure is shown in Figure 10.

Figure 10. Example of Morphological Closing

5.3. Tesseract

In this project, we worked with the Tesseract OCR system that was presented in Section 4. Several errors may occur with this system**Error! Reference source not found.** , namely:

- Segmentation errors.
- Character recognition errors.
- Word recognition errors.

5.4. OCR Correction

Correcting the result of an OCR requires the use of NLP approaches. In this section, we present two approaches. The first is called noisy channel model which has shown its effectiveness in the literature. On the other hand, the second is an approach proposed to improve the result.

5.4.1. Noisy Channel Model

The noisy channel model is used in spell checkers, speech recognition and machine translation. This model was chosen because it is the most powerful model in this field of word correction, and we found several research based on this model (Kissos and Dershowitz **Error! Reference source not found.** Tong and Evans**Error! Reference source not found.**). In this model, the goal is to find the word

from a word where the letters have been scrambled in one way or another, this model proposes a list of candidates for the wrong word, and classifies them according to a probability to choose the best candidate that can be the correct word of the wrong word **Error! Reference source not found.**.

- **Generating Candidates:**

The first step in correcting a word is to find words on a dictionary that differs from the wrong word by a single insertion, deletion, substitution, or transposition, calculating the editing distance of Damerau-Levenshtein between the erroneous word and the words of the dictionary.

Consider, for example, the word "acress", the first step is to generate candidates in the table below,

Table 1. Words within Distance = 1 of Acress

Error	Candidate	Correct Letter	Error Letter	Type
acress	actress	t	-	Deletion
acress	cress	-	a	Insertion
acress	caress	ca	ac	Transposition
acress	access	c	r	Substitution
acress	across	o	e	Substitution
acress	acres	-	s	Insertion
acress	acres	-	s	Insertion

From this table, it can be seen that several correct dictionary candidates can be generated from the wrong word, so that the correct word "actress" could be transformed by the noisy channel by removing the t from Position 2 of the word "acress", the same for the correct word "cress" (cress in French) which could be transformed by adding the a to the position 0.

- **Classification of Candidates:**

We define the probability of a word c in the corpus by the maximum likelihood estimation**Error! Reference source not found.**:

$$p(c) = \frac{freq(c)}{N} \tag{5}$$

With *freq(c)* is the number of occurrences of the word c in the corpus.

And N is the total number of words in the corpus**Error! Reference source not found.**.

The correct word is chosen from the calculation of the score of each candidate c, using the following formula:

$$\hat{c} = \underset{c}{argmax}\ \ p(c/t) \tag{6}$$

Using the Bayes rule, the probability $p(c/t)$ in formula below is replaced by the following expression:

$$p(c/t) = \frac{p(t/c)\,p(c)}{p(t)} \tag{7}$$

Which gives us:

$$\hat{c} = \underset{c}{argmax}\{p(c/t)\}$$
$$= \underset{c}{argmax}\{\frac{p(t/c).p(c)}{p(t)}\}$$
$$= \underset{c}{argmax}\{p(t/c).p(c)\}. \quad (8)$$

It is clear that the term p(t) does not intervene in the new formula, since it is considered constant.

The conditional probabilities, $p(t/c)$, are computed from four confusion matrix **Error! Reference source not found.**, where:

- $del[x, y]$ The number of times the xy characters (in the correct word) were typed as x in the test database.
- $add[x, y]$ The number of times x has been typed as xy.
- $sub[x, y]$ The number of times there was typed as x.
- $rev[x, y]$ The number of times that xy has been typed as yx.

The probabilities are estimated from these matrix by dividing the number of times xy and x appears in the test database, respectively, by [x, y] or chars[x].

$$p(t \mid c) = \begin{cases} \frac{del[c_{p-1}, c_p]}{chars[c_{p-1}, c_p]} & Si\ \ suppression \\ \frac{add[c_{p-1}, t_p]}{chars[c_{p-1}]} & Si\ \ insertion \\ \frac{Sub[t_p, c_p]}{chars[c_p]} & Si\ \ substitution \\ \frac{rev[c_p, c_{p+1}]}{chars[c_p, c_{p+1}]} & Si\ \ transposition \end{cases} \quad (9)$$

Where c_p is the p^{th} character of c, and similarly t_p is the p^{th} character of t.

Problem: The result may be wrong. So he needs another check.

Solution: In the processed sentence, one calculates the probability of the words close to the result obtained in the first phase of treatment.

How to calculate p (w): Let W be a sentence composed of n words $w_1, w_2, w_3, \ldots, w_n$, According to the rule of derivation, we obtain,

$$p(w_1, w_2, w_3, \ldots, w_n) = p(w_1)p(w_2/w_1)\ldots p(w_n/w_1, \ldots, w_{n-1})$$
$$= \prod_i p(w_i/w_1, w_2, \ldots, w_{i-1}) \quad (10)$$

From the Markov property **Error! Reference source not found.** the formula 10 becomes,

$$p(w_1, w_2, w_3, \ldots, w_n) = \prod_i p(w_i / w_{i-1}), \forall i \in 1,2,\ldots,n \quad (11)$$

• The maximum likelihood estimate [8]:

$$p(w_i / w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})} \quad (12)$$

So, the sentence that contains the correct word is the one that has the greatest probability.

5.4.2. Proposed Approach

The goal of this new algorithm is the same as the first algorithm (noisy channel model), ie it try to find the correct word from a scrambled and erroneous word. This algorithm comes to optimize the results of the first algorithm, as we have already seen that the algorithm of noisy channel depends on the four transformations (insertion, suppression, transposition and substitution), whereas this new algorithm depends just on the suppression phase.

Dictionary Generation:

This new algorithm has as its first phase the generation of a new dictionary, it takes all the words of the dictionary, and generates for each word all the words with a distance = 1 based on the distance of Damerau-Levenshtein **Error! Reference source not found. Error! Reference source not found.** (delete operation only), and stores the original word and the generated words in the new dictionary.

The following example shows the generated terms by the Damerau-Levenshtein distance based on the delete operation only.

Hello $\Rightarrow$ ello, hllo, helo, hell

Table 2. Main Dictionary

Original word	Frequency
Word 1	f_1
Word 2	f_2
...	...
Word n	f_n

Table 3. New Dictionary

Generated Words	Original words		
New Word 1	Word 1	Word 6	Word 4
New Word 2	Word 15	Word 11	Word 8
...	...	...	...
New Word M	Word 51	Word 78	Word 7

The table above (Table 3) presents the new dictionary that was built using the main dictionary (Table 2) and the suppression technique. Each line of the new dictionary contains the new word generated followed by the original word (source) of the new word,

i.e., from all these original words one can extract the new word generated by an application of the suppression technique.

Spell Correction:

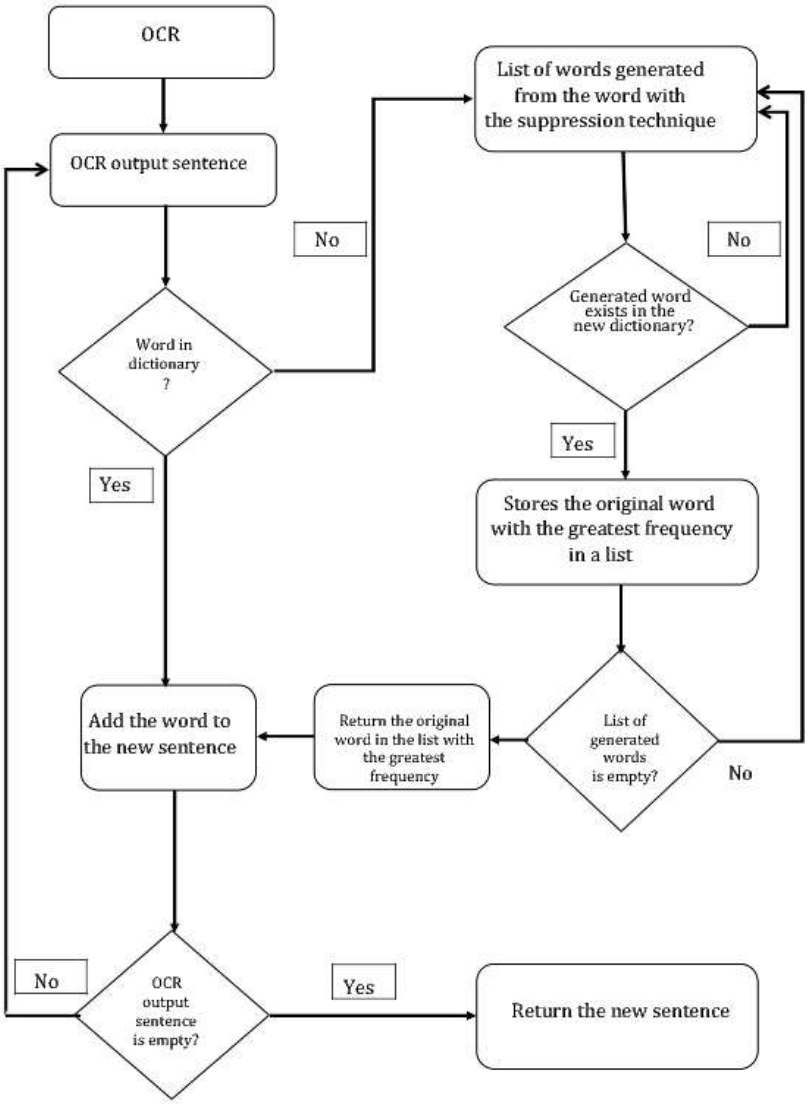

Figure 11. New Structure of the Proposed System

The algorithm of the proposed orthographic correction system (Figure 11) performs a verification of all the words of the output of the OCR system, and applied to each word not found in the main dictionary the technique of suppression, and for each generated word, the algorithm search the new dictionary, if it exists, the algorithm returns the original word that has the highest frequency in the main dictionary. And if we have several generated words (from the wrong word) existing in the new dictionary, the algorithm returns the original word which is the correct word of the erroneous word, from the list of all the original words returned for each word Generated and has the highest frequency in this list in the main dictionary.

6. Big Data Challenges by Hadoop

This is the world of Big Data, Web logs, Internet texts and Documents, Internet search indexing, Internet of Things (IoT), Cloud Computing... These all includes Big Data for making more strategic decisions and taking the full advantage of available information, it is required to process Big Data efficiently**Error! Reference source not found.**.

In this article we decided to work with big data in the case if we have a lot of documents to check, so the experiment is to run our MapReduce algorithm with a number variable of documents in input(in the corpus stored in HDFS) and get the correction of all documents fast and at the same time.

6.1. Hadoop Clustering

Hadoop is an open source Apache software framework that evaluates gigabytes or petabytes of structured or unstructured data and transforms it more manageable for applications to work with this large data**Error! Reference source not found.**. The core components of Hadoop are HDFS and MapReduce. HDFS is basically used to store large data sets and MapReduce is used to process such large data sets.

6.2. Hadoop Distributed File System (HDFS) Architecture

The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks**Error! Reference source not found.**. HDFS uses a write-once, read-many model that breaks data into blocks that it spreads across many nodes for fault tolerance and high performance.

HDFS stores file system metadata and application data separately on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes.

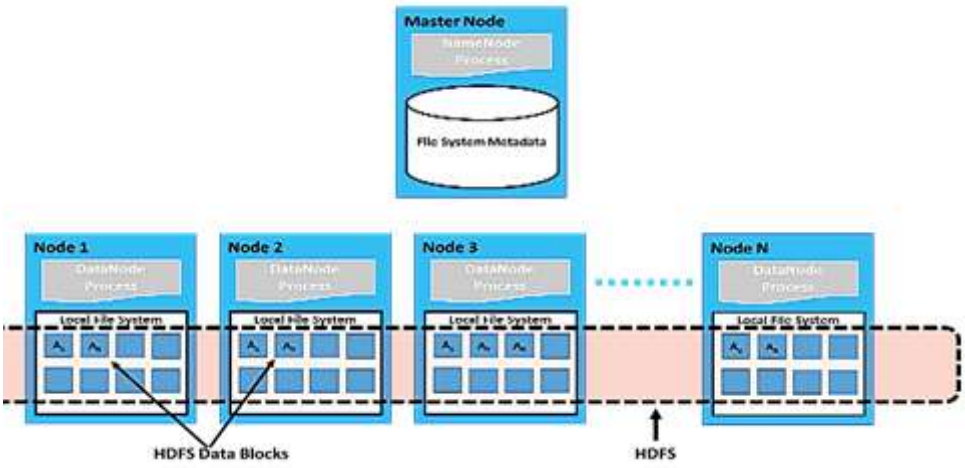

Figure 12. The High-Level Overview of the Structure of HDFS

NameNode: the node that controls the HDFS. It is responsible for serving any component that needs access to files on the HDFS. It is also responsible for ensuring fault-tolerance on HDFS. Usually, fault-tolerance is achieved by replicating the files over different nodes.

DataNode: this node is part of HDFS and holds the files that are put on the HDFS. Usually these nodes also work as TaskTracker. JobTracker tries to allocate work to nodes such files accesses are local, as much as possible.

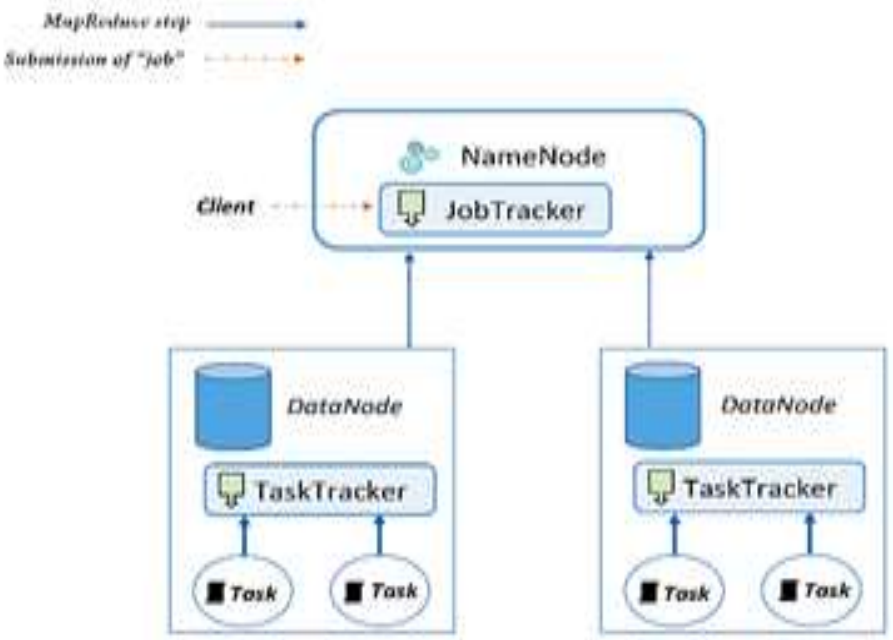

Figure 13. The Interaction between HDFS and MapReduce Job

At the level of the NameNode, the JobTracker is respon- sible for the management of resources that is the control of the DataNodes in cluster. It manages the entire duration of the life of a job. The TaskTracker has responsibilities more simple, namely launch the tasks in the order provided by the JobTracker and periodically give a status of progress of the task to the JobTracker.

6.3. MapReduce Programming Model

MapReduce is a programming model and an associated implementation for processing and managing large data sets with a parallel, distributed algorithm on a cluster.

MapReduce divides into three parts: Map, Shuffle and Sort, and Reduce. A Map part of MapReduce job splits the input datasets into independent chunks. The independent chunks are processed in a completely parallel manner using Map task. Then the Reduce function merged these values to form a possibly smaller set of values. That is, the Reduce function filtered the Map output and produces the results with respect to the key of the Map phases**Error! Reference source not found.**.

In our proposed algorithm runs on two consecutive MapReduce phases, the first to build an indexing phase and the second is to do the orthographic correction, as our design MapReduce shown in the following figure:

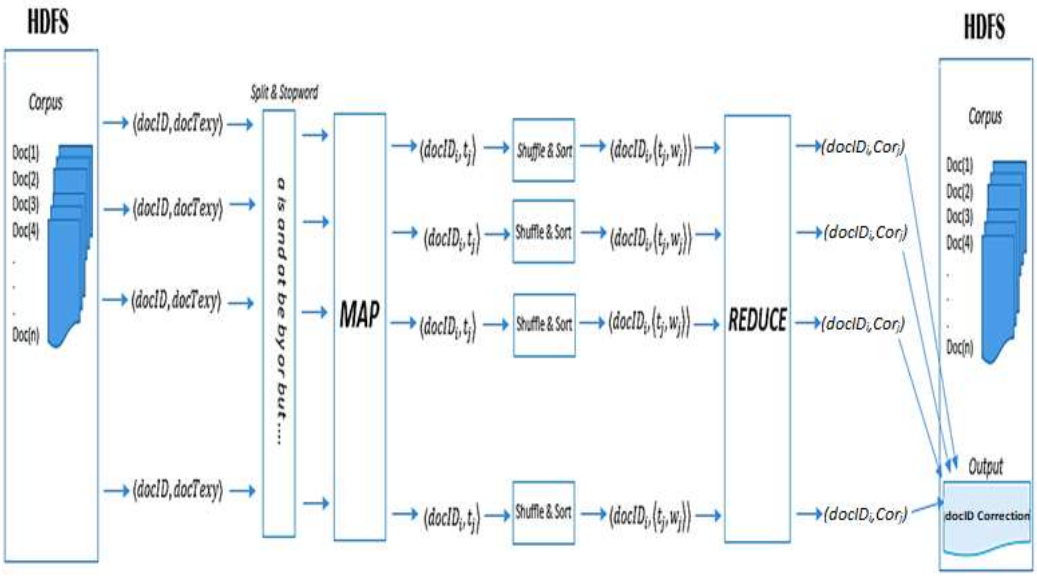

Figure 14. The Process of our MapReduce Model

• Document indexing: given a corpus, for each term of document, the mapper emits the document ID as the key, and his words as the value. The shuffle phase of MapReduce, groups these words by a collection of the values of each document, and delivers these inverted lists to the reducers, that write them to blocks.

• Orthographic correction: In this step, Reduce takes the output of the Map function and do the spell checking then the orthographic correction for each collection of values of each document and the query.

7. MapReduce Algorithm Proposed

We present here our MapReduce algorithm to do the orthographic correction using our approach:

Class Mapper
  Method Map(Docid,term)
    For each element ∈ (Docid,term)
      Write(Docid,term)
    End for
Class Reducer
  Method Reduce(Docid,List(term))
    For each n ∈ List(term)
      F=Correction(n)
      Write(Docid,F)
    End for

Mapper maps input key/value pairs to a set of intermediate key/value pairs. In our case we have the key is the Docid which is the input file name which contains the term.

Maps are the individual tasks that transform input records into intermediate records. The transformed intermediate records do not need to be of the same type as the input records. A given input pair may map to zero or many output pairs.

Reducer reduces a set of intermediate values which share a key to a smaller set of values.

In our algorithm the Reducer take a list of terms which are in a input file, then apply the correction of each term then give the result of all terms in each input file.

8. Numerical Results

This section is devoted to numerical results. Digital tests will include printed text images. The implementation of all the codes, was carried out using an Android application that we developed under Android Studio, then we moved to Cloudera to apply the orthographic correction using MapReduce framework to the result of the OCR obtained by the android application.

To test the performance of the algorithms, tests were carried out on different text images, which contained almost 1000 words, and we obtained the results shown in the table below,

Table 4. Test Results of Algorithms

	Recognition rate
OCR	73.21%
Noisy Channel Model	78.27%
Proposed approach	87.27%

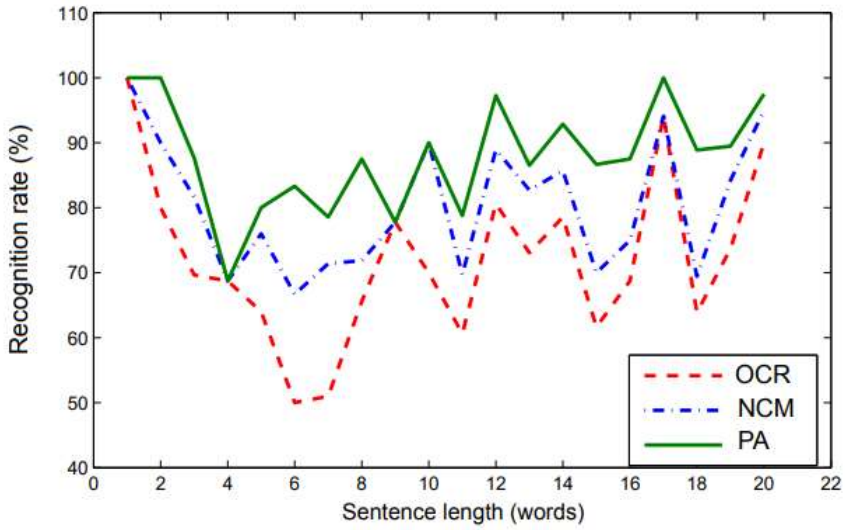

Figure 15. Change in the Recognition Rate, Compared to the Sentence Length for OCR, NCM and PA

From the obtained results, we see that the Tesseract OCR system has a recognition rate that does not exceed 73.21%, which validates our hypothesis that this system remains to be improved. To improve those results, the noisy channel model was applied to the results obtained by the Tesseract OCR system, and a correction of 5% was obtained, so with this model, Recognition rate up to 78.27%, which gives the effort to look for a solution for a better improve these results, and this is reflected in the results obtained using our proposed algorithm, which gives a recognition rate equal to 87.27%, Which is the best result when comparing with the noisy channel model **Error! Reference source not found.**.

9. Conclusion

In this work, we have presented the proposed OCR system and describes its modules which are: acquisition, pre-processing, segmentation, recognition, and post-processing that allows the correction of the results concluded by the OCR system Tesseract. Also, we have detailed the approaches developed for each module.

The objective was making a correction on the results obtained by the Tesseract OCR system using MapReduce framework, for this we presented the approach of the noisy channel model which allowed us to have an improvement, but not Which leads us to propose and present our approach which gives better results than the noisy channel model with a recognition rate equal to 87.27%.

References

[1] S. Shastry, G. Gunasheela, T. Dutt, D. S. Vinay and S. R. Rupanagudi, "i" A novel algorithm for optical character recognition (OCR), 2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), Kottayam, **(2013)**, pp. 389-393.
[2] K. Church, W. Gale and M. A. Kernighan, "spelling correction program based on a noisy channel model", Coling, Helsinki, Finland, **(1990)**.
[3] M. Fakir, B. Bouikhalene, R. El Ayachi and K. Moro, "On the recognition of tifinaghe scripts", JATIT, vol. 20, no. 2, **(2010)**, pp. 61-66.
[4] N. Senthilkumaran and S. Vaithegi, "Image segmentation by using thresholding techniques for medicalimages", Computer Science and Engineering: An International Journal (CSEIJ), vol. 6, no. 1, **(2016)**.
[5] S. Agneand and M. Rogger, "Benchmarking of document page segmentation", Part of the IST/SPIE Conference on Document Recognition and Retrieval VII, San Jose, California, **(2000)**, pp. 165-171.
[6] I. Kissos and N. Dershowitz, "OCR error correction using character correction and feature-based word classification", Document Analysis Systems (DAS), 2016 12th IAPR Workshop, **(2016)**.
[7] D. A. Evans and X. Tong, "A statistical approach to automatic ocr error correction in context", Proceedings of the Fourth Workshop on Very Large Corpora, **(1996)**.
[8] G. E. Box and G. C. Tiao, "Bayesian inference in statistical analysis", John Wiley and Sons, vol. 40, **(2011)**.
[9] S. Shakya and R. Santana, "An EDA based on local Markov property and Gibbs sampling", Proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM, **(2008)**.
[10] F. J. Damerau, "A technique for computer detection and correction of spelling errors", Communications of the Association for Computing Machinery, **(1964)**, pp. 171-176.
[11] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", Soviet Physics Doklady, **(1966)**, pp. 707-710.
[12] H. Bagheri and A. Abdullah Shaltooki, "Big Data: Challenges, Opportunities and Cloud Based Solutions", International Journal of Electrical and Computer Engineering (IJECE), vol. 5, no. 2, **(2015)** April, pp. 340-343.
[13] T. White, "Hadoop: The definitive guide", O'Reilly Media, Inc, **(2012)**.
[14] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System", **(2010)**.
[15] Y. B. Reddy, "Document Identification with MapReduce Framework", Data Analytics: The Third International Conference on Data Analytics, **(2014)**.

