Evaluation of Optical Character Recognition (OCR) accuracy: Supervised and Unsupervised techniques

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF,

Master of Technology in *Cryptology and Security*

By

Niladri Banerjee

MTech Cryptology and Security, ISI Kolkata; Data Science Intern, iManage

Under the guidance of,

Dr. Clarisse Magarreiro

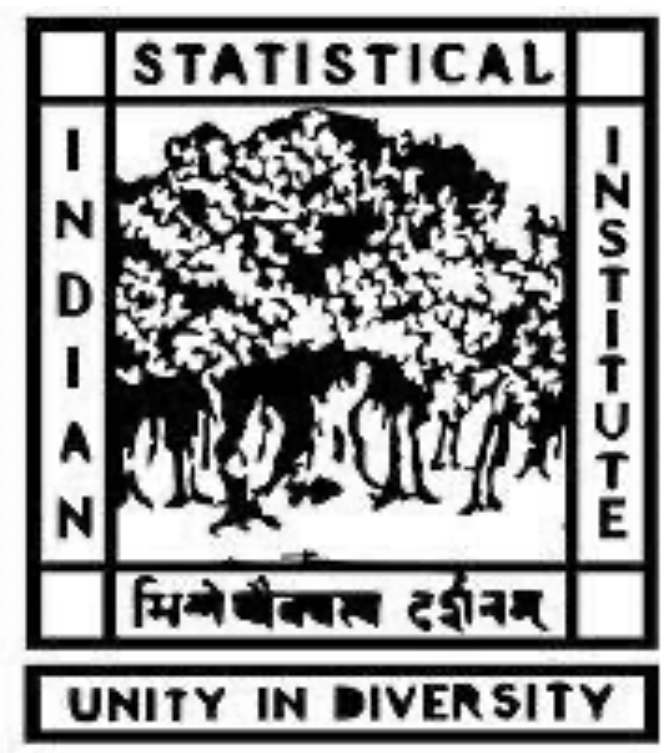
Data Scientist, iManage

Dr. Rakesh Kumar

Sr. Manager, Data Science, iManage

Dr. Anisur Rahaman Molla

Assistant Professor, Indian Statistical Institute Kolkata



Indian Statistical Institute, Kolkata

July 2021

Acknowledgements:

I am grateful to my guide, Dr Anisur Rahaman Molla, Assistant Professor, Indian Statistical Institute, Kolkata, for his invaluable advice and guidance. I would like to show my highest gratitude to the Data Science team of iManage, specially to my guide Dr. Clarisse Magarreiro and Dr. Rakesh Kumar, for their continuous support and valuable suggestions, which not only helped me completing the project successfully but also gave me an idea to grow further in the Data Science industry and helped me a lot to learn several new concepts. I would also like to thank all the professors of Indian Statistical Institute, Kolkata for help me acquire knowledge in several topics. I am thankful to placement cell for providing the opportunity to work with iManage. I also extend my heartfelt thanks to my family and well-wishers.

Niladri Banerjee

MTech Cryptology & Security

Indian Statistical Institute, Kolkata

Pin. 700108, India

Abstract:

This work's aim is to find an efficient method to measure the Optical Character Recognition (OCR) accuracy in the absence of the ground truth text. To successfully obtain the desired result, initially we have tried some efficient supervised (in the presence of the ground truth text) accuracy measuring techniques. Then we tried some unsupervised (in the absence of the ground truth text) techniques, which is the final goal of our project, and compare their performance with respect to the previously obtained supervised techniques. Our final project goal is to provide an efficient unsupervised accuracy measuring technique which can help us to automate the document analysis process.

Index:

1. Introduction:

OCR is the process of converting non editable texts (i.e., pdf, images) into editable ones (text format). This a technology which is being used broadly in current days. The biggest companies in the world (Google, Amazon) are not only using this tool, but also developing their own model for better result. Here we are not going to discuss about how OCR is done (that is done using deep learning tools like CNN, NLP), but focus will be on the comparison between several OCR accuracy measuring indices. This step is crucial to make an informed decision on the best OCR accuracy measure to use when ground truth (i.e., the source text file) of a document is not available.

In the organization iManage, text obtained from an OCR engine is fundamental for their applications. Document classification, information retrieval or Named Entity Recognition are examples of processes that rely on text. For that purpose, several non-editable documents need to be converted into the editable form to make the searching process through characters easy. It should be noted that in any OCR process several wrong character conversions will occur and OCR engine we are using is not an exception on that. The performance of the applications will inevitably be influenced by the accuracy of the OCR process. So, before considering improvements to the OCR process it is essential to assess the OCR quality.

OCR's accuracy depends upon several constraints. One major observation was that the font style affects the OCR quality badly. Also handwritten digits, historical fonts affect the OCR quality. So, in this project mostly our focus will be finding an 'efficient' accuracy measure, with less computational complexity.

1.1 Our Contribution:

In this project we will evaluate accuracy measures in the presence of the ground truth text, i.e., the original raw text file. Next, we will assess efficient methods for the case where the ground truth is missing. Our plan is to compare the performance between methods that determine accuracy in the absence of the ground truth with respect to the methods in the presence of the ground truth.

For the case where ground truth is available, we have found some efficient methods. First one is Jaccard index. There are two more methods depend on an interesting and innovative idea, the recursive alignment methods [1] [3] [4] [5] [7]: Hidden Markov Model probabilistic approach (HMM) and the Recursive Text Alignment Scheme (RETAS). These were some supervised methods which rely on text-to-text evaluation. There are some other processes in the supervised case, xml-to-xml evaluation, and text-to-xml evaluation, mentioned in the paper by Romain Karpinski, Devashish Lohani, Abdel Belaid [5]. They also mentioned about one method called ZoneMapAltCnt [5] [8]. But it was easy for us to work with the text-to-text approach only, so we didn't focus on these methods.

The Jaccard index is a method which deals with the very simple mathematical calculations, such as, union and intersection. However, it lacks the positional information of the words. The recursive alignment method takes care of this, by subdividing the text into smaller chunk of texts. Over those smaller chunks of texts, we run the HMM and RETAS algo. HMM [3] is a probabilistic model which keeps track about the relation between the original and the OCRed text by the help of one hidden sequence. This method involves optimizations to get these values. On the other hand, RETAS [1] method does a character level checking w.r.to *edit distance*.

Finally, we are going to devise a method to perform the measurements in the absence of the ground truth. Initially, we started dictionary lookup method [6] [7], i.e., checking whether the words in the OCRed text are meaningful or not by searching them in a dictionary. Later we started working with other methods because there was some problem in dictionary lookup method. As it lacks positional information like the Jaccard index and there are problems with named entities too. Language model can be its one possible solution, but it can be costly with respect to the computational complexity, which may harm the main goal of our project, as we have mentioned earlier, this project is a part of a bigger project, which should not take long time to get finished. Hence, it was not a good idea to move forward with this process. So, in searching for some better algorithm we found a method based on confidence score (will be discussed in the section 2.4), which was provided by the

OCR engine. This can be a good proxy against the dictionary lookup method. We have tested many of the above-mentioned algorithms over several text files. Let us start our discussion with the OCR conversion method.

1.2 OCR conversion process:

The OCR engine we are using in this project, provides us two different extraction methods of the OCRed text.

The first is text extraction method, which on an input of a pdf or image outputs the corresponding full text OCRed output, which totally reorders the paragraphs. So, this was not useful for us.

On the other hand, the second method, namely, the docstream method outputs a character wise detailed output corresponding to the same input as the other method. In this method, not only the characters are given but also their positional information and the other information like bold, italics, confidence score (the probability that a recognition variant is correct) are given in this method too. In this method we can accept the output in 3 different formats: text, html, stream. We will mostly work with the text format output. For ease of calculation, we store this output inside a csv file, so that, it's different columns will contain different attributes, such as, Character, font size, font style, confidence score etc. We will concatenate the characters depending upon some special constraints to get the full text.

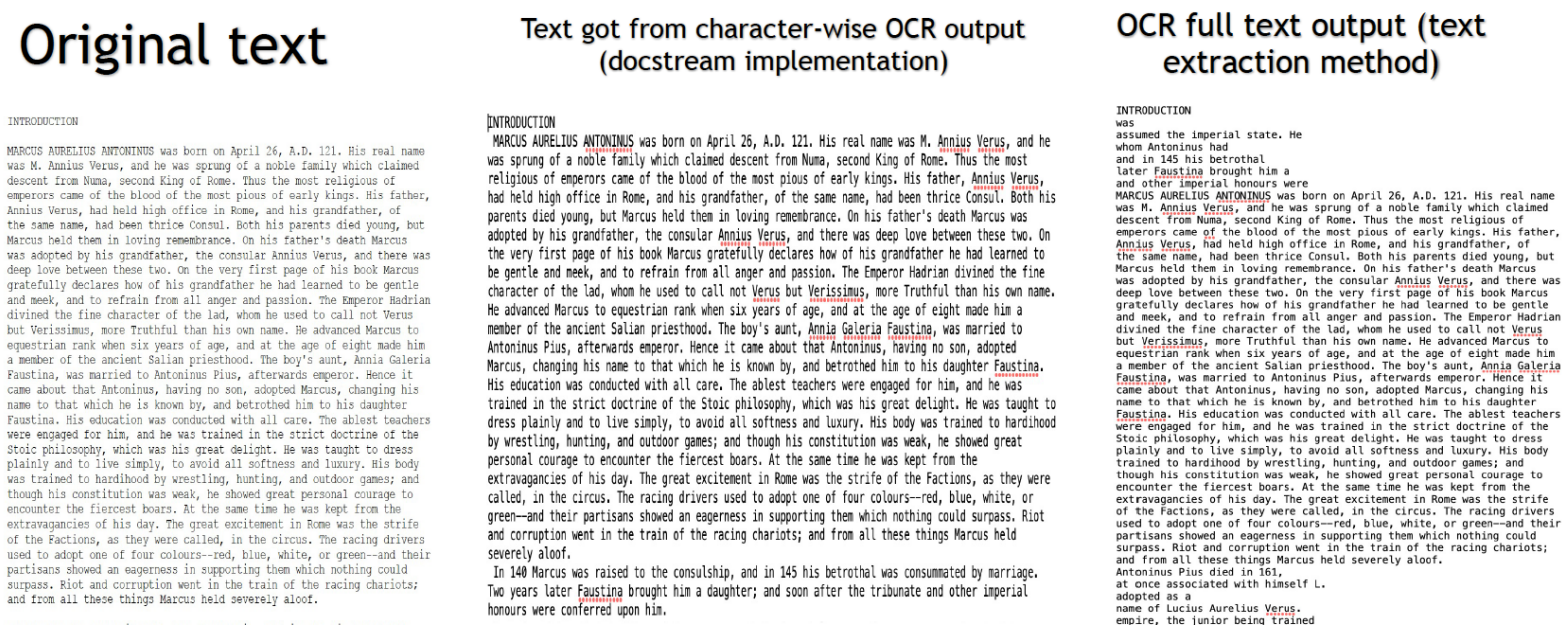


Figure 1: Comparison between the Original and the two files obtained via two different type of OCR conversion for a same file

2. Accuracy Measures (supervised and unsupervised):

We are going to discuss about some accuracy measure techniques in this section. Starting with some supervised techniques we will move into the unsupervised techniques, as in real world scenario we may not have the ground truth text file always. The purpose of starting with the supervised techniques is nothing but to compare the efficiency of the unsupervised techniques with respect to the supervised ones.

2.1 Jaccard Index:

It is a supervised method, i.e., it is calculated in the presence of the ground truth (GT) text. Vaguely speaking, Jaccard index is basically the proportion of the area of overlap over the area of the union. If we call the ground truth text as GT and the OCRed text as OCR then this method will first split both the texts in terms of the words and then will take set of both the lists; say, the sets are GTset and OCRset then, Jaccard index will be,

$$J(GTset, OCRset) = |GTset \cap OCRset| \div |GTset \cup OCRset|$$

$$= \frac{|GTset \cap OCRset|}{|GTset| + |OCRset| - |GTset \cap OCRset|}$$

This method is efficient with respect to computational complexity. But the problem appears due to the set formation. For this not only the positional information gets lost but also it ignores the other important attributes such as the confidence score. So, we then focus on some techniques which consider the positional information. Even if we take weight count for each word still there could be some problems. For example, suppose in a text file the word ‘man’ present exactly 3 times. Suppose due to the OCR error one ‘man’ word has been changed into ‘men’ and some other word, say, ‘main’ changed into ‘man’. Then, in that document, total number of the word ‘man’ remains constant. Hence in this method we will get an accuracy of 100% corresponding to the word ‘man’, but which is not correct.

2.2 Alignment Methods:

As mentioned earlier, we are going to discuss about two different methods, rely on the alignment method, viz., RETAS method and HMM method. Let us discuss about the alignment technique first.

This technique is also a supervised accuracy measuring technique. Hence, the input files are the OCRed, and a GT text files. On input of these two files, we first search for *Anchor Words* following this technique. Anchor words are basically the common unique word from both the texts. In algorithmic perspective,

1. Search for unique words in the GT text file
2. For each unique word in GT checks over the OCRed text file whether it is a unique word in that file or not
3. If yes, then mark it as Anchor word
4. If it does not present in the OCRed text, then search for the next word
5. Otherwise, if the word exists more than one time in the OCRed text then checks for the neighbours of the word for each of its occurrence in the OCRed texts and returns Anchor word output for matching of the neighbouring words.

After finding the Anchor words this technique divides the whole text with respect to these Anchor words, i.e., for each consecutive Anchor words A and B, take the text portion in between A and B. Do it for all the Anchor words. Now repeat the whole process over the smaller text segments, stop until no Anchor word left inside a text segment, or the length of the text segment is smaller than a certain threshold (typically 200, used in the paper by R. Manmatha [1]). Finally, we will do the accuracy checking over the smallest chunk of texts.

There are several advantages of these methods. One is obviously the positional information is taken care in this method, because of the smaller divisions. The other thing is the time complexity. Because of these divisions the complexity got reduced. These alignment techniques are efficient to measure the accuracy of the OCR.

Now the obvious questions may be if there will not be enough number of unique words in the texts then the chunks may not be small, therefore we may do our analysis over a large sized text. But the thing is that one analysis from the paper by R.

Manmatha [1] says, a book of 500 words per page contains typically 10 to 15 unique word per page. So, frequency of getting unique words is high.

2.2.1 HMM:

HMM or Hidden Markov Model is a probabilistic algorithm. This method tries to construct a position sequence depending upon the given OCRed text sequence and GT text sequence. Suppose $O =< o_1, o_2, \dots, o_n >$ be the OCRed text sequence, $G = < g_1, g_2, \dots, g_m >$ be the ground truth and $S =< s_1, s_2, \dots, s_n >$ be the hidden position sequence, where $s_i = j$ implies ith word or character in the O corresponds to the jth word or character in G. The HMM-based alignment model estimates the joint probability of the OCR sequence and the hidden position sequence $\mathrm{P}(\mathrm{O}, \mathrm{S})$ as:

$$\mathrm{P}(\mathrm{O}, \mathrm{S}) = \prod_{i=1}^{n} \mathrm{P}(s_i|s_{i-1})\mathrm{P}(o_i|s_i)$$

Here, $\mathrm{P}(s_i|s_{i-1})$ is the *transition probability*, i.e., the probability of a successful transition from the state s_{i-1} to the state s_i in the ground truth text and $\mathrm{P}(o_i|s_i)$ is called the *generative probability*, which is the probability of generating OCR term o_i from the ground truth term at the position s_i. The definitions of these terms are given in the paper by Feng and Manmatha [3].

Then our goal will be to maximize $\mathrm{P}(\mathrm{O}, \mathrm{S})$ with respect to S, i.e.,

$$\tilde{S} = \arg\max_{S} \mathrm{P}(\mathrm{O}, \mathrm{S})$$

Using the Viterbi algorithm [9] the authors Feng and Manmatha, determine the most likely state sequence $\tilde{S}$ through decoding over the OCR sequence. So, by solving the last equation given, we get a sequence of positions in the ground truth with the same length as the OCR output sequence. For each OCR term, the assigned position value indicates the ground truth term from which it is generated.

2.2.2 RETAS:

This method is an *edit distance*-based method. Edit distance is the minimum number of edits required to obtain a word from a given word. Edits can be of three

types, viz., *Insertion, Deletion* and *Substitution*. Insertion is adding a character at any place of the string 1 to get the string 2, while deletion is removing, and substitution is replacement of one character. The cost of these operations is 1, 1 and 2 respectively. For example, the word ‘spring’ is of distance 2 from the word ‘ring’ (2 deletions from spring), the word ‘art’ is of distance 1 from the word ‘are’ (1 substitution). Previously Rice proposed an idea of edit distance-based accuracy measurement with the help of Ukkonen’s Algorithm [2]. This method was efficient in smaller texts than the larger ones. So here REATS method uses this algorithm after making the chunk of the text smaller. This method checks the edit distances between the words sequentially over the smallest chunks of texts and with respect to the edit distance they align the whole text. If edit distance is 0 then that is considered as correctly OCRed word. Otherwise, they put a ‘@’ or a null value in the place of wrongly OCRed characters. On an input of a OCRed and GT file it outputs a comparison-based output file which consists of detailed character-wise information, which will be discussed later.

2.3 Dictionary Lookup Method:

This is the first unsupervised method we are going to discuss about, here the ground truth text is absent. This method is very simple, but little bad with respect to the time complexity. This method simply for all word in the OCRed document search it in a dictionary or a text file with a rich vocabulary. It recognizes a word as wrongly OCRed if it is not in that dictionary.

Now, there are several problems in this method. For example, suppose in the ground truth text there is a word ‘main’ somewhere. But due to OCR the word has been changed into ‘man’. Now both the two words will be there in the dictionary and as a result this method will identify the wrongly OCRed ‘man’ word as a correct one, which will affect the accuracy.

There is one more crucial problem in this method, i.e., named entity recognition. There can be hundreds of named entities which may not be present in the dictionary. For those words even after being correctly recognized, this method will classify those

as wrongly classified. Now the thing is even after having this type of major problems the overall performance of this method was up to the mark. The comparisons are given in the next section.

2.4 Confidence Score based Accuracy measure:

In the dictionary lookup method, we observed several problems. Previously we have mentioned that there is an attribute called confidence score is given in the output of the OCR engine. So, we are trying to use those confidence scores in our accuracy measurement analysis, as its computational complexity is not high like the language model.

The confidence score is a value between 0 to 100, which represents the probability of confidence. Basically, the OCR engines outputs a character via a classification model. This confidence score is the percentage of the character to be correct after OCR. We took an average value over the confidence scores to get the accuracy in the method 1. And in the other method we ignore all non-alphabets. In this process we took the characters between two non-alphabets as a word, calculated its average confidence score and finally output the average confidence scores of all the words as the accuracy measure. We took the RETAS method output to check the efficiency of this method. From that analysis we developed the later method of accuracy measure. We got some impressive results in this method, which is in the next section.

3. Experimental Results:

In this section we are going to show some sample outputs we got in several steps. Firstly, I am going to show one typical output we got in the time of OCR by the above mentioned docstream method (section 1.2) of the OCR engine.

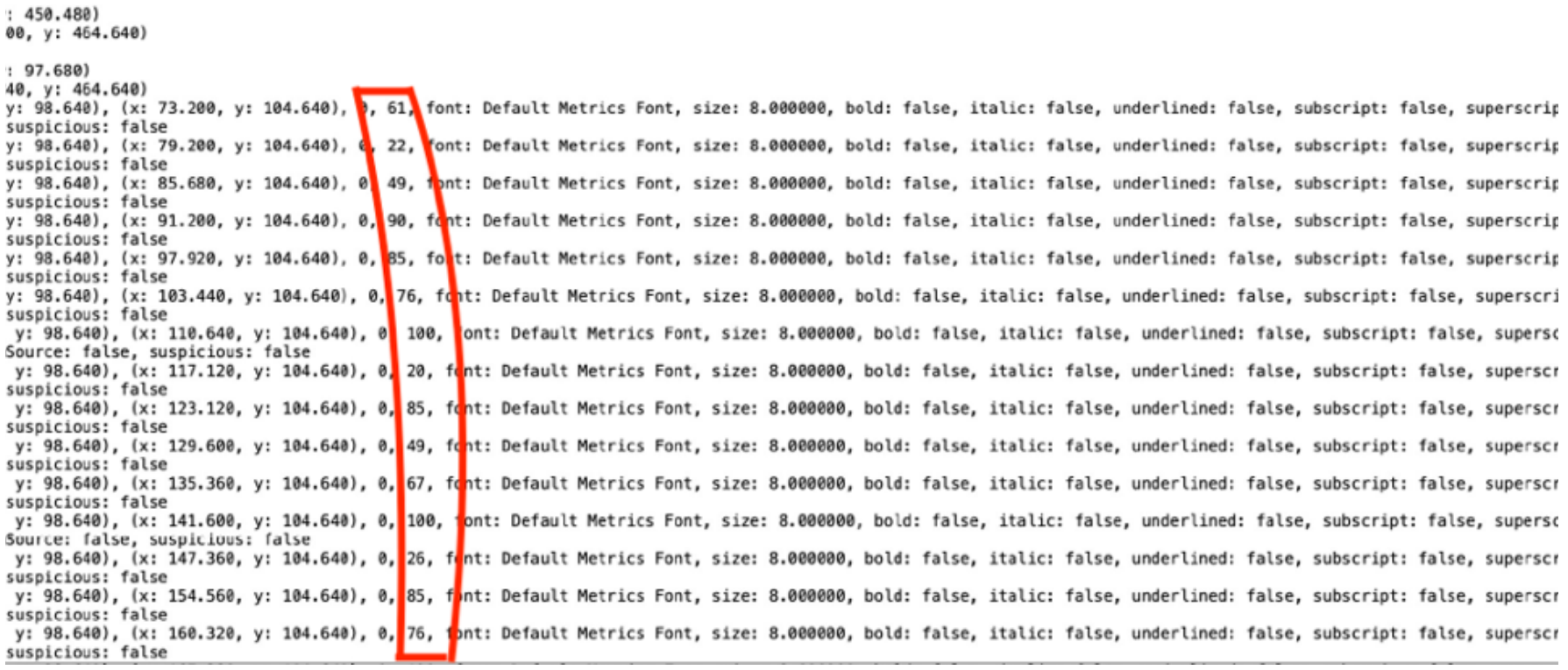
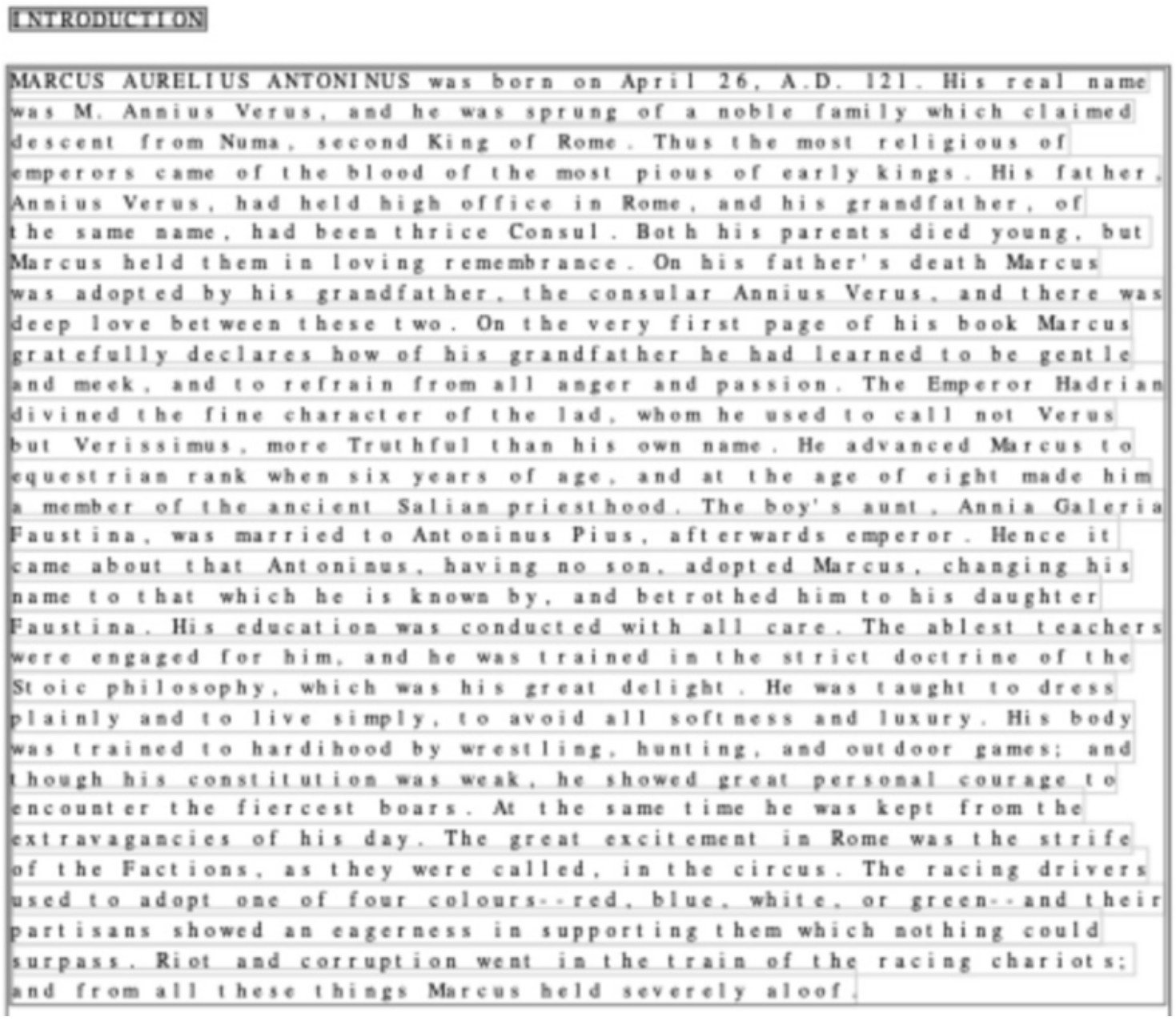


Figure 2: This is an output file of the OCR Engine using text extension. The red marked column is the Confidence score.



INTRODUCTION

MARCUS AURELIUS ANTONINUS was born on April 26, A.D. 121. His real name was M. Annius Verus, and he was sprung of a noble family which claimed descent from Numa, second King of Rome. Thus the most religious of emperors came of the blood of the most pious of early kings. His father, Annius Verus, had held high office in Rome, and his grandfather, of the same name, had been thrice Consul. Both his parents died young, but Marcus held them in loving remembrance. On his father's death Marcus was adopted by his grandfather, the consular Annius Verus, and there was deep love between these two. On the very first page of his book Marcus gratefully declares how of his grandfather he had learned to be gentle and meek, and to refrain from all anger and passion. The Emperor Hadrian divined the fine character of the lad, whom he used to call not Verus but Verissimus, more Truthful than his own name. He advanced Marcus to equestrian rank when six years of age, and at the age of eight made him a member of the ancient Salian priesthood. The boy's aunt, Annia Galeria Faustina, was married to Antoninus Pius, afterwards emperor. Hence it came about that Antoninus, having no son, adopted Marcus, changing his name to that which he is known by, and betrothed him to his daughter Faustina. His education was conducted with all care. The ablest teachers were engaged for him, and he was trained in the strict doctrine of the Stoic philosophy, which was his great delight. He was taught to dress plainly and to live simply, to avoid all softness and luxury. His body was trained to hardihood by wrestling, hunting, and outdoor games; and though his constitution was weak, he showed great personal courage to encounter the fiercest boars. At the same time he was kept from the extravagancies of his day. The great excitement in Rome was the strife of the Factions, as they were called, in the circus. The racing drivers used to adopt one of four colours--red, blue, white, or green--and their partisans showed an eagerness in supporting them which nothing could surpass. Riot and corruption went in the train of the racing chariots; and from all these things Marcus held severely aloof.

Figure 3: Sample output file using html extension

In the figure 3, this is one sample html type output obtained from the OCR engine. In figure 2 the marked column is the confidence score.

Now, as we previously told, the OCR accuracy has a huge dependency on the font style. Here one sample output is shown below:

Font Style	Courier New	Pristina	Brush Script MT	Freestyle Script	Zapfino
Accuracy in terms of Jaccard index	0.999599176976	0.649189615938	0.737535265926	0.565994770102	0.0355824578611

Table 1: Comparison between the Jaccard indices using different fonts for a same text file.

It can be shown that how simply changing the font style can affect a text's OCR accuracy.

Now let us start with RETAS method as our 1st method, i.e., Jaccard index is too easy to calculate. We got a repository [11] for the RETAS method from [1]. Here one typical output file of the RETAS method is given below:

```
OCR:    The in          Project  Gutenberg  eBook  Grey  Wethers  by  V  Victoria  SackvilleWest  This  eBook  is  for  the  use  of  anyone  anywhere
GT :    The in          Project  Gutenberg  eBook  Grey  Wethers  by  V  Victoria  SackvilleWest  This  eBook  is  for  the  use  of  anyone  anywhere

OCR:    the You         United   States     and    most  other    parts  of  the     world          at    no     cost  and  with  almost  no  restrictions  whatsoever
GT :    the You         United   States     and    most  other    parts  of  the     world          at    no     cost  and  with  almost  no  restrictions  whatsoever

OCR:    may this        copy     it         give   it    away    or     reuse  it      under         the   terms  of    the  Project  Gutenberg  License  included  with
GT :    may this        copy     it         give   it    away    or     reuse  it      under         the   terms  of    the  Project  Gutenberg  License  included  with

OCR:    eBook the       or laws  online     at     wwwgutenbergorg  If  you  are     not           located  in  the  United  States  youll  have  to  check
GT :    eBook the       or laws  online     at     wwwgutenbergorg  If  you  are     not           located  in  the  United  States  youll  have  to  check

OCR:    of Victoria     the      country    where  you   are      located  before  using  this  ebook  Title  Grey  Wethers  A  Romantic  Novel  Author  V
GT :    of Victoria     the      country    where  you   are      located  before  using  this  ebook  Title  Grey  Wethers  A  Romantic  Novel  Author  V

OCR:    SackvilleWest Etext  Release  Date  April  eBook  Language  English  Character  set  encoding  UTF  START  OF  THE  PROJECT  GUTENBERG  EBOOK  GREY  WETHERS
GT :    SackvilleWest Etext  Release  Date  April  eBook  Language  English  Character  set  encoding  UTF  START  OF  THE  PROJECT  GUTENBERG  EBOOK  GREY  WETHERS
```

Figure 4: Sample output file of the RETAS method

For the dictionary lookup method, already there was an algorithm at iManage. We have done a little modification to that algorithm. The algorithm first converts everything into small letters and then drop every character except alphabets and spaces and finally tokenize those into words. We just modified two things, one is in the time of text processing and the other in the time of tokenization. In the previous algo some null value was being considered, for these two changes that problem got resolved.

So, we have worked with these four methods; two supervised, Jaccard Index & RETAS, and two unsupervised, Dictionary Lookup & Confidence score based approach. We have done the whole experiment in 3 ways. Initially, we have done the experiments

using files downloaded from project Gutenberg [10] website. Then, we have synthetically generated some input texts and have done the same experiments over these files too. Finally, we take a combination of both of the files and do the same experiments. The main reason behind using these synthetic input or synthetic output methods is to create some documents with 'bad' OCR accuracy, as most of the real world documents have an OCR accuracy around 80% to 100%.

We also have done another experiment using synthetic output documents. The main difference between synthetic input based method and the synthetic output based method is in the first one we will try to generate fake GT text and in the latter case we will try to generate fake OCRed text data. Let us start our discussion with the synthetic output based method.

3.1 Synthetic Output based approach:

The main goal of the project was to find an efficient OCR accuracy measuring method or ensure the efficiency of the previous one (Dictionary Lookup). Here in this synthetic output based approach we tried to do that comparison; but in the absence of the OCR engine. Basically, for a given

GT document we tried to generate its corresponding OCRed text file. Now, we are going to discuss the algorithm to generate the files. This algorithm can be broken down into two parts; namely, the text generation and the confidence score generation. We are going to discuss these for each of the files.

- Text generation:

  For each files:

  → Pick a number from 0 to 1 (say α) (100α will be the expected accuracy for that document)

  → For each word in the text

  → Draw a random number between 0 and 1 (say β)

  → If $\beta < \alpha$: then predict the word as correctly OCRed

  → Else: predict as incorrectly OCRed *

→ Keep non-alphanumeric values unchanged and output the result as OCRed text corresponding to the given GT text

* Types of incorrect OCRs:

1. Wrong but same length with correct word
2. Wrong prediction by breaking a word
3. Combination of 1 and 2

We will randomly do any of the 3 methods.

- <u>Confidence Score Generation:</u>

We will do this in two major steps:

1. Observing the distribution of the word-wise confidence scores both for the correctly predicted and incorrectly OCRed words.
2. Generation of the confidence score per word for the synthetic output files depending upon the word was OCRed correctly or not.

For (1) we need the output files of RETAS method for some previously tested files. We divided the files into train and test files in 5:1 ratio. We will follow the following algorithm to generate the distribution:

1. Create 2 list (or dataframe or set) True and False.
2. Input the output file performed by the RETAS method with True/False labelling corresponding to each word.
3. Take the average confidence scores corresponding to each word which will be calculated using the confidence scores per letter and append that in the corresponding dictionary.

Now, for the step (2), i.e., the generation of the confidence score we will 1st check for each word whether the words are correctly OCRed or not. Depending upon that 'True' or 'False' labelling we will draw randomly a confidence score from the corresponding list (or set or dataframe). Thus, for all word we will do the same. Now, we have trained and tested these over some documents. These graphs are given below.

Original Confidence Score **Synthetically Generated Confidence Score**

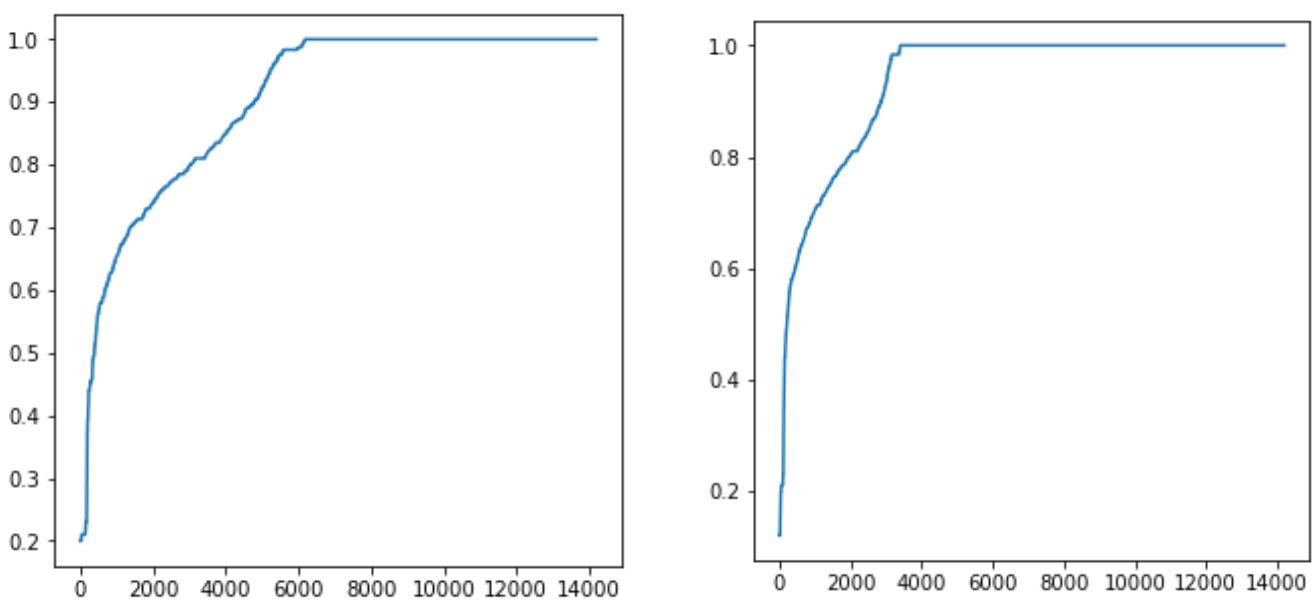


Figure 5.1: Distribution of the Confidence Score per word for all the words in the documents

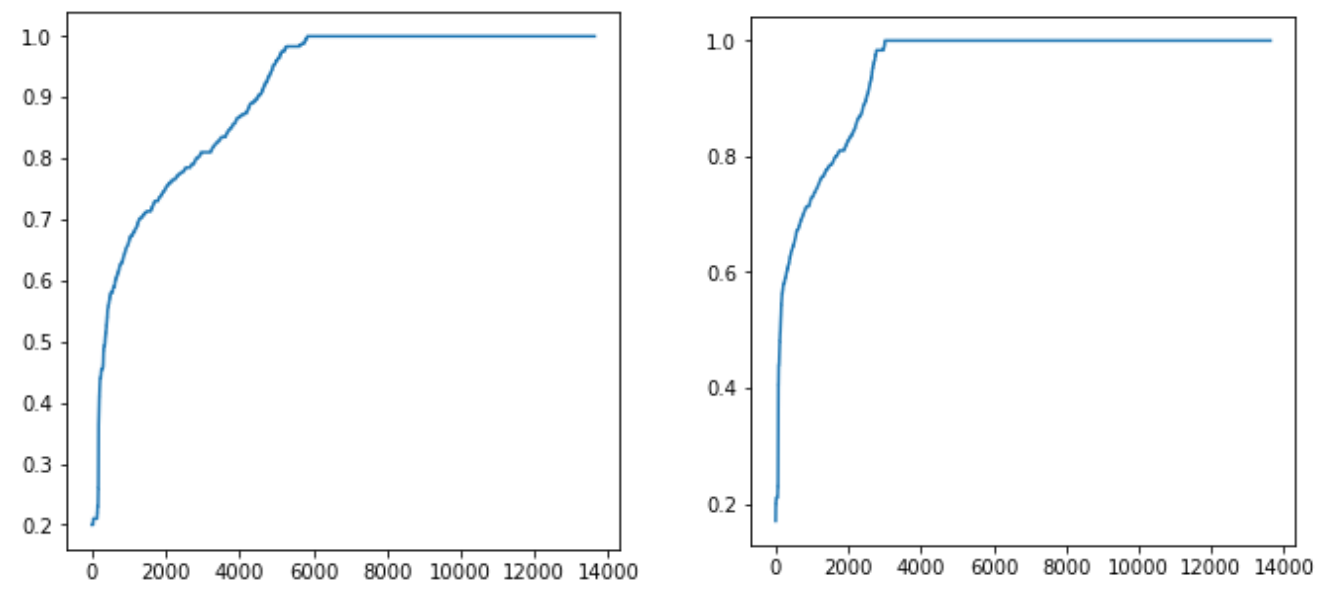


Figure 5.2: Distribution of the Confidence Score per word for the correctly OCRed words in the documents

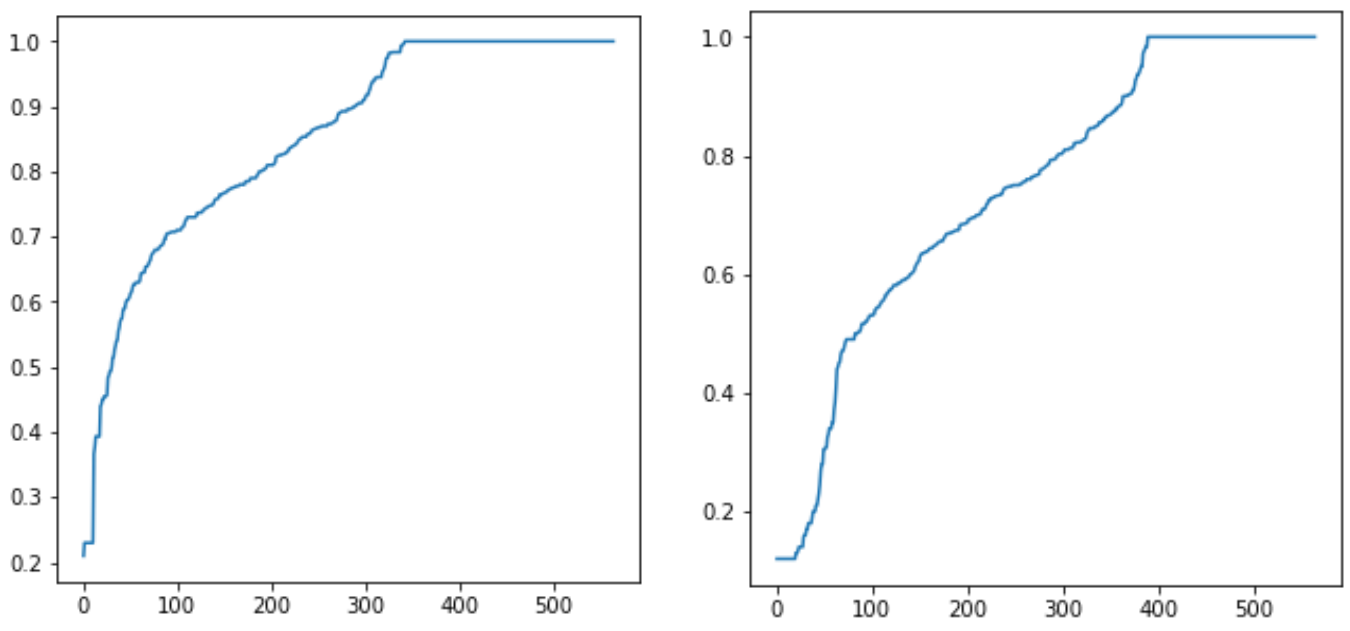


Figure 5.3: Distribution of the Confidence Score per word for the incorrectly OCRed words in the documents

Here in the diagrams given above in the left-side diagram at each level those are the diagrams for the words from the real documents and in the right-side those are the documents from the documents with synthetically generated confidence scores for the documents from the test set. The y-axis denotes the accuracy score (a number from 0 to 1) and the x-axis is the number of words.

Now, we can see the generated confidence score was quite similarly distributed like the real-world documents. So, depending on these generated synthetic output datasets we tried to analyze the unsupervised accuracy measuring methods.

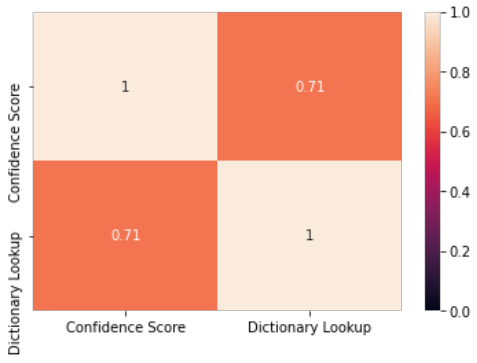


Figure 6.1

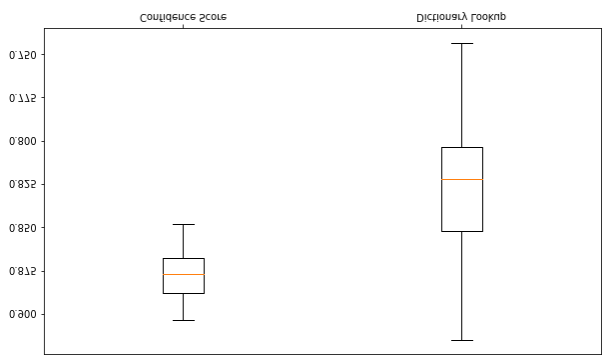


Figure 6.2

Correlation Heatmap and boxplots for the two unsupervised methods using synthetically generated output files. For the boxplot in the y-axis the accuracy is given (a value from 0 to 1) and the x-axis, the methods

Now, the problem with this method is that, in this method the OCR engine is not involved. So, we look for some better method to generate synthetic data, i.e., the synthetic input method. In this method we try generating some input file which should perform badly in the time of OCR conversion. But before discussing this method let us discuss the results, we got using real text documents collected from Project Gutenberg website [10].

3.2 Analysis using text data collected from online resources:

We used python library *BeautifulSoup* from *bs4* to automate the process of downloading. To be precise we didn't download the files rather just copy the text from the website. After that we have used python library *FPDF* to convert these into pdf files. We have used the font style *Arial* for the pdfs. After getting the pdfs and the

text files we automate the process of pdf to text conversion. For this conversion we use a CentOS virtual machine. So, we used *os.system()* call to perform the command line arguments. But there was a little problem in it, as every time even after a successful conversion the OCR engine ended up with a segmentation fault. We used the integer output given by the OCR engine after a successful run to tackle this problem. Finally, we used all the four accuracy measuring methods (including both old and new methods of dictionary lookup) over these files. Here there are the results we got:

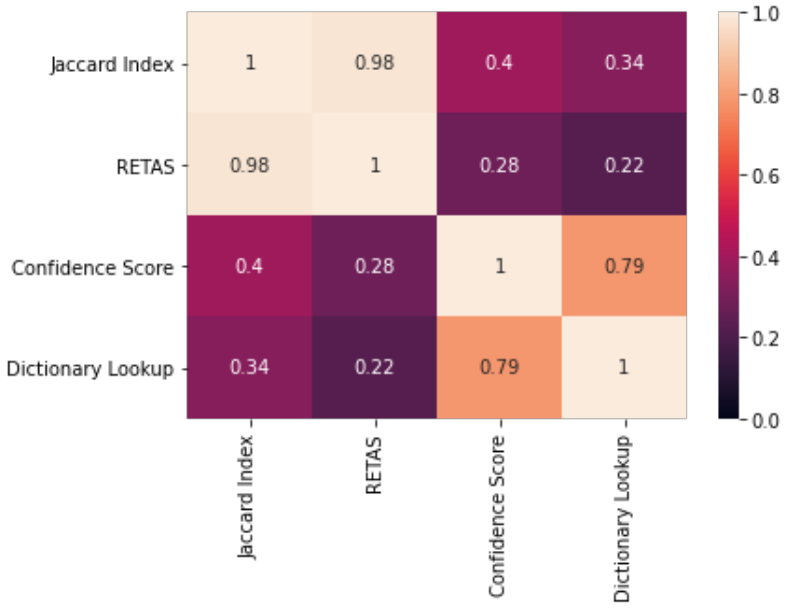


Figure 7.1

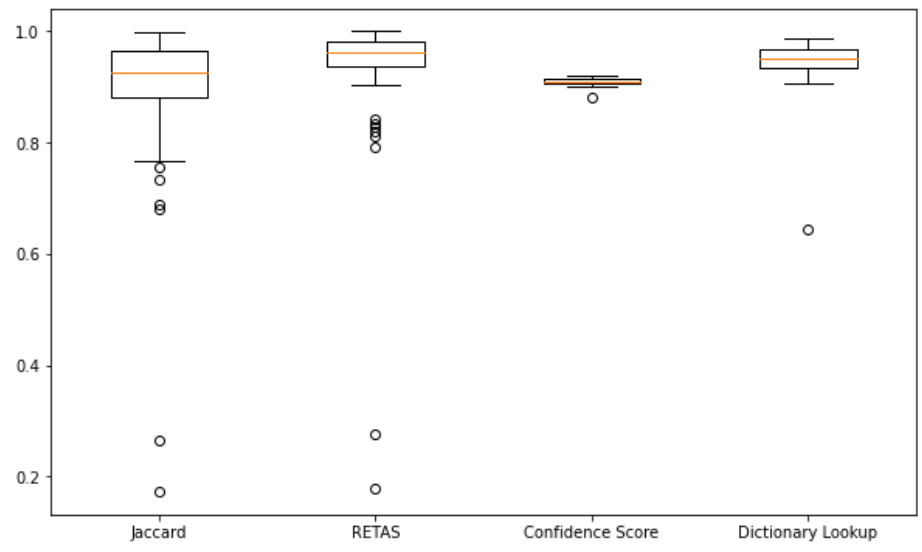


Figure 7.2

Correlation Heatmap and boxplots for the two supervised and two unsupervised methods using files downloaded from the project Gutenberg website. For the boxplot in the y-axis the accuracy is given (a value from 0 to 1) and the x-axis, all the four (two supervised and two unsupervised) methods.

Here in the figure 7.1 we can see the correlation coefficients are not so good. That is because there are some outliers in the data, which is clearly visible in the figure 7.2. And this is one of the main reasons behind exploring the method based on synthetically generated text files. So, let us start discussion on this.

3.3 Synthetic Input based approach:

There are two main reasons behind using this particular approach. These are:

1. The number of ‘real’ documents were not sufficient enough to conclude any results.

2. The reason already been told in the section 3.2. We need to manage the proportion of outliers.

One possible solution can be to remove those outliers, but instead of doing that we want to add some data files with 'low' accuracy score. In this section, we will discuss the case using synthetically generated text files first, and then finally, we do the same using both synthetic input and real text documents.

In the generation process of synthetic input text we will follow the similar algorithm like the synthetic output one. The basic idea is to create two dataframe (or, other datatypes like list) one containing the correctly predicted words and the other containing the incorrectly predicted words and then appending those in a certain proportion to get the text files.

The main intuition behind this approach is: the word which previously have been OCRed correctly will certainly have a high probability to be OCRed correctly again and the same thing goes for the incorrectly predicted words too. So, using the labelling from the RETAS method's output file we want to do the job. Now, let us discuss the algorithm of generating the files.

For file in files_to_be_generated:

↦ Pick a number from 0 to 1 (say α)

↦ Generation of the text

↦ Draw a random number between 0 and 1 (say β)

↦ If $\beta < \alpha$:

↦ Pick a word randomly from the dataframe of correctly OCRed words

↦ Else:

↦ Pick a word randomly from the dataframe of incorrectly OCRed words

↦ Put a space between every words

Thus, we can get a text file with $100\alpha\%$ words from the correctly OCRed collections. Notice that we are not using any non-alphanumeric characters here. In this generation we want to generate files with OCR conversion accuracy around $100\alpha\%$. One obvious question may be asked that why we should need β. The thing is α divides

the interval into two parts $[0, \alpha)$ and $[\alpha, 1]$. Probability of getting a number from the 1st interval, which is the interval corresponding to the correctly predicted words, is $100\alpha\%$, which is the expected accuracy. In this whole process β is used for selecting the interval.

Once the generation is done, we will output the text-format file and convert those into pdfs using python module FPDF. After getting both the text and pdf format documents we will do the same thing we did in the section 3.2. Now let us look into the results we got in this method.

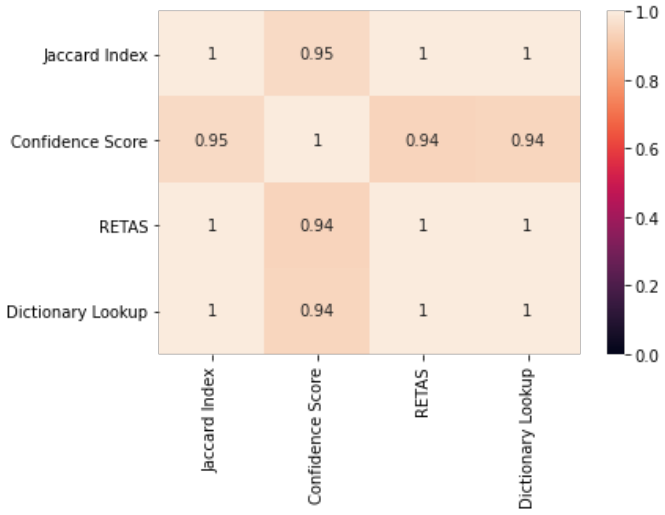


Figure 8.1

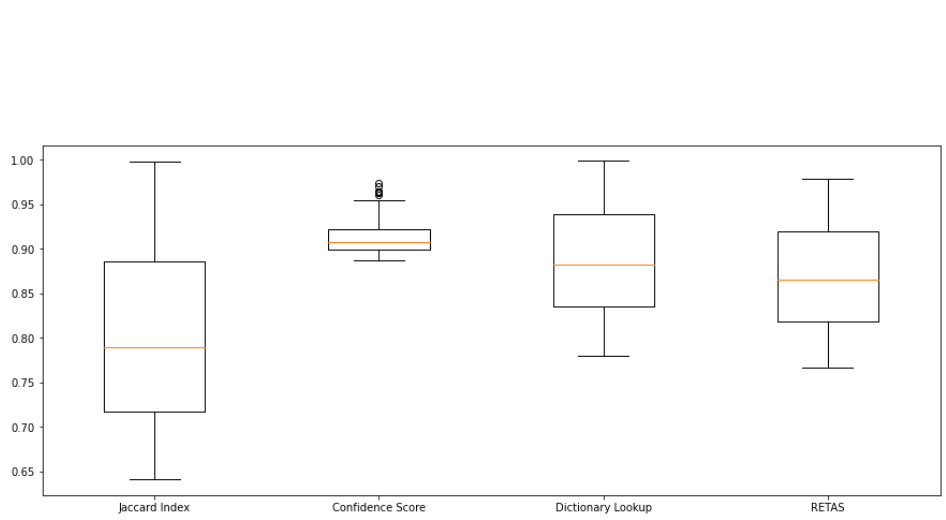


Figure 8.2

Correlation Heatmap and boxplots for the two supervised and two unsupervised methods using synthetically generated input text files only. For the boxplot in the y-axis the accuracy is given (a value from 0 to 1) and the x-axis, all the four (two supervised and two unsupervised) methods.

These results were looking good, i.e., synthetic input method can be a good proxy for the real datafiles. So finally, we added some of these synthetically generated files with all the real datafiles. We took synthetic input datafiles with RETAS accuracy only above 80%, as we already have the real datafiles for this range. Here are those results.

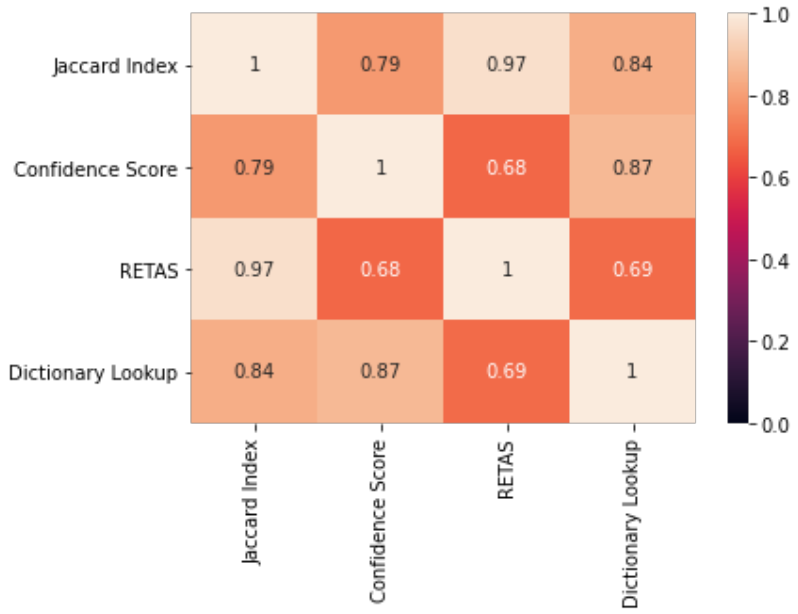


Figure 9.1

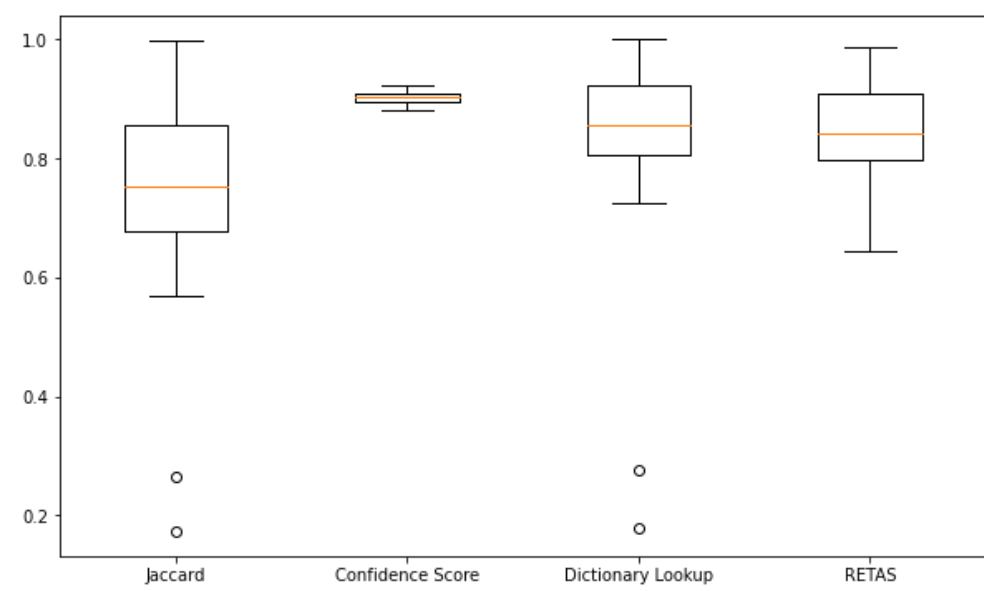


Figure 9.2

Correlation Heatmap and boxplots for the two supervised and two unsupervised methods using both synthetically generated input text files and the files downloaded from project Gutenberg website. For the boxplot in the y-axis the accuracy is given (a value from 0 to 1) and the x-axis, all the four (two supervised and two unsupervised) methods.

Now even after using the synthetic input documents there are several outliers in the and that affects the correlation too. We have used some outlier removal technique (IsolationForest, with hyper-parameter contamination) to observe how good the result look like without outliers. Here are the results depending upon several values of the hyper-parameter contamination.

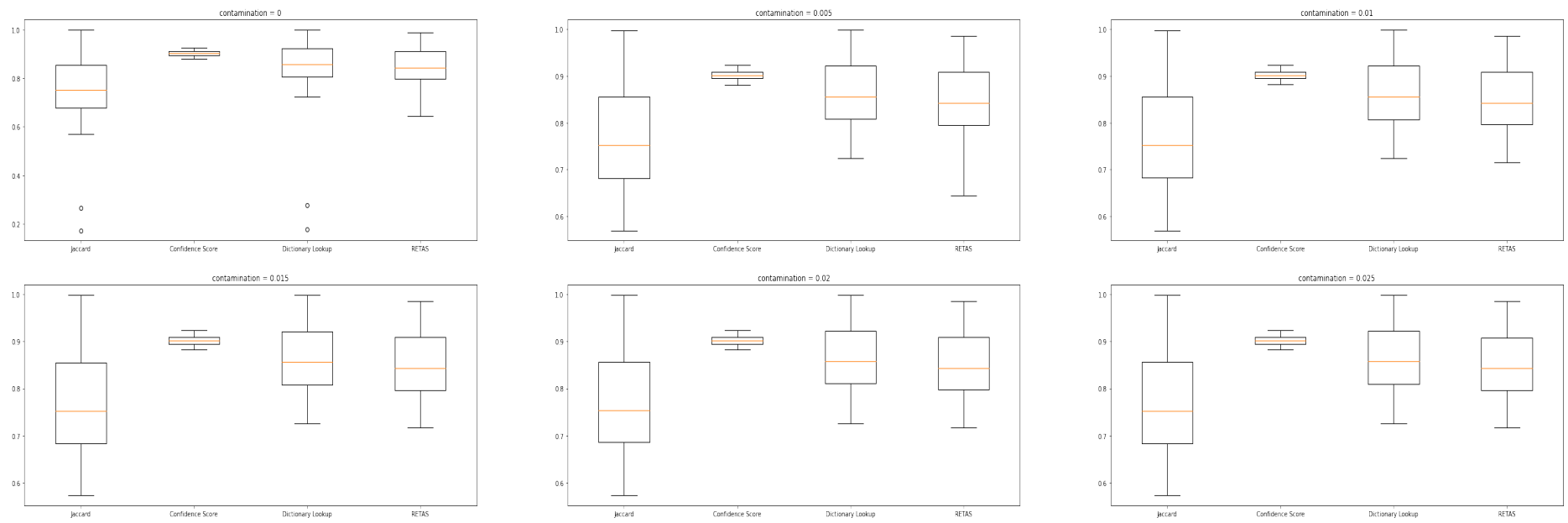


Figure 10.1

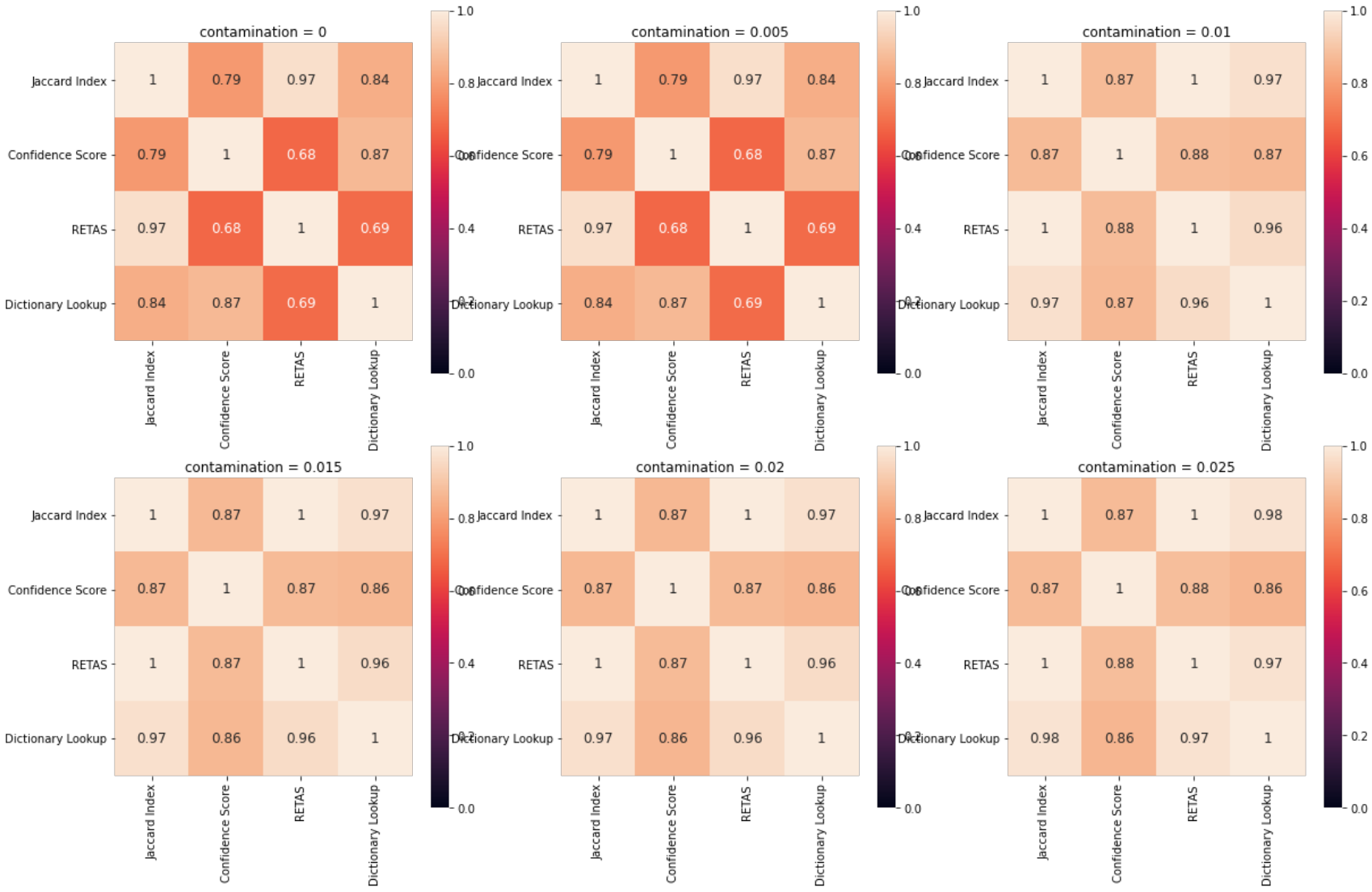


Figure 10.2

Boxplots and correlation heatmaps for the above mentioned four accuracy methods for several values of the hyperparameter contamination using the outlier detection method isolation forest

These results were not only better than the previous one, but also it ensures us that the unsupervised methods correlate well with the supervised methods. Specially the Dictionary lookup method. The confidence score-based approach correlates well with the other methods, but one major problem with this approach is that, in most of the cases the confidence score takes a value around 80 to 100; that is why even after correlating well with the other methods this method returns a very high value comparing to the other methods. As a result, the boxplot is so dense comparing to the others in this method. One possible solution can be fixing some threshold for this, for example, 80% in RETAS ~ 92% in this method.

Finally, we tried not to eliminate the outliers, i.e., if we can generate some datafiles with accuracy ranging from 0% to 60% then the problem will be solved. To do this we used a trick, inspiring by the results of Table 1. That means, we changed the font to get some low accuracy pdfs. In this method we used add_font() function of FPDF module. We replaced half of the synthetic input documents from each range, [10,

20), [20, 30), [30, 40), … [90, 100]. Here the proportion of this replacement is given below.

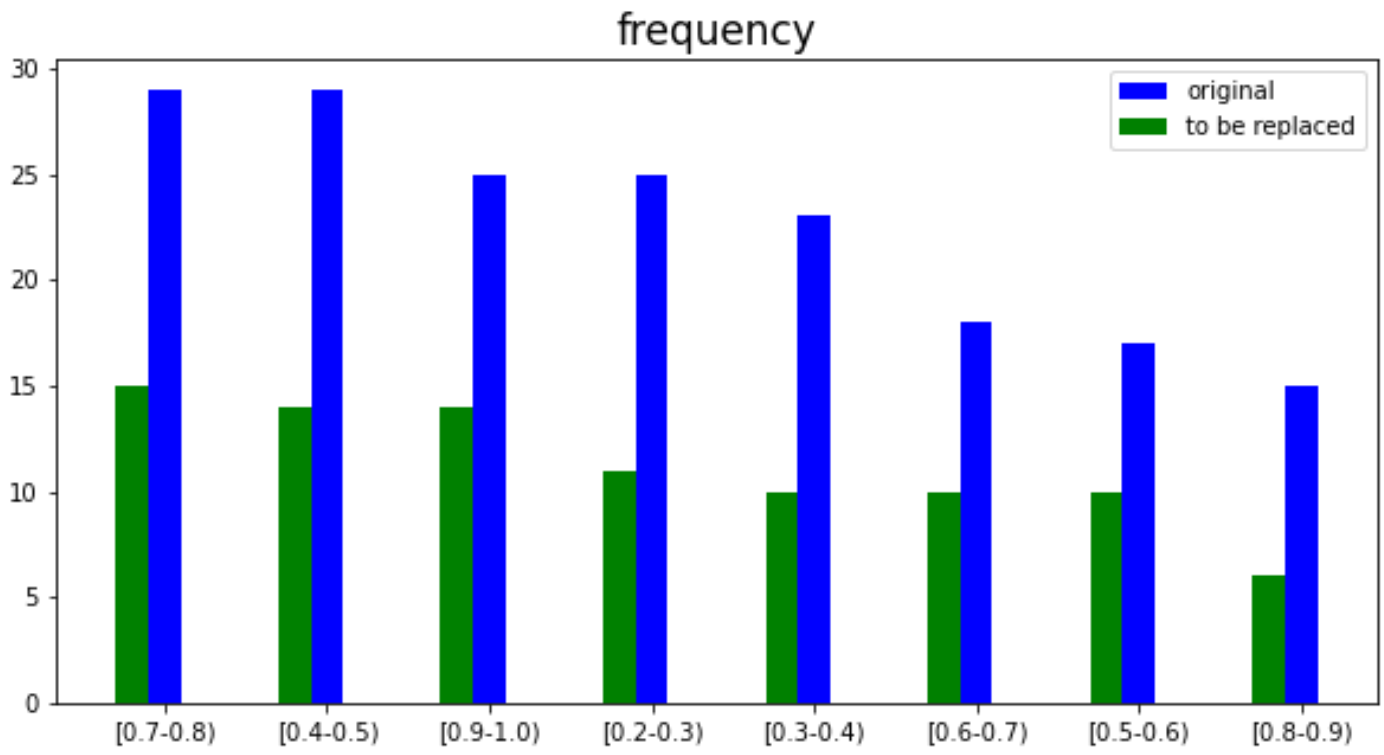


Figure 11: Comparison between the number of total files and the number of files to be replaced by the files with different fonts, where x-axis denotes the accuracy ranges and y axis denoted the number of files in that range

We replaced these texts with documents using 10 different fonts and using different proportion of words taken from both the text sets (correct and incorrect). Once the replacement is done, we started running all the four accuracy measuring methods over these files again. After doing these we measured the correlation, RMSE, R2 scores between all the four methods. Also plotted the boxplot. If we look at the box plot (figure 12.2) then we can see that there are still some outliers, but the number of values less than 20% accuracy is much higher than the previous one. That obviously help us in the analysis. Notice that, in every other boxplot graphs the graph corresponding to the confidence score is very dense except this one. That is because while converting scripted font documents the OCR Engine getting confused between the letters, hence giving us a very bad confidence score. That is why the boxplot for the confidence score-based accuracy score is surprisingly much wider in terms of its range.

Figure 12.1

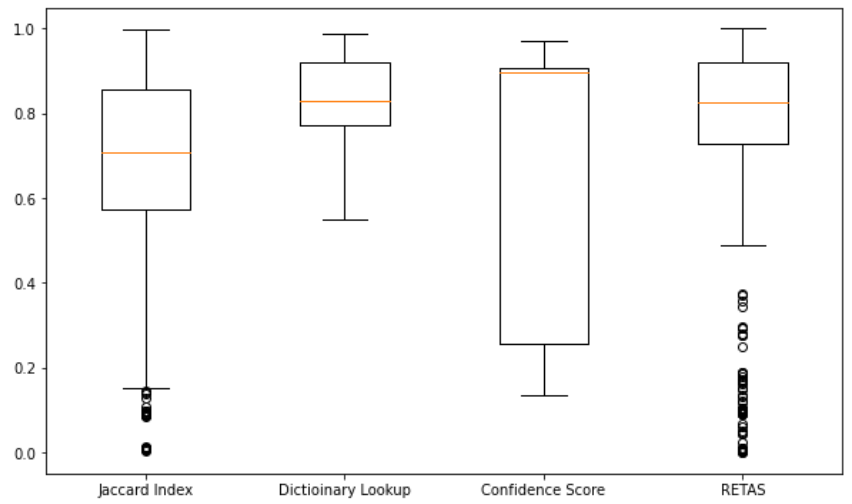


Figure 12.2

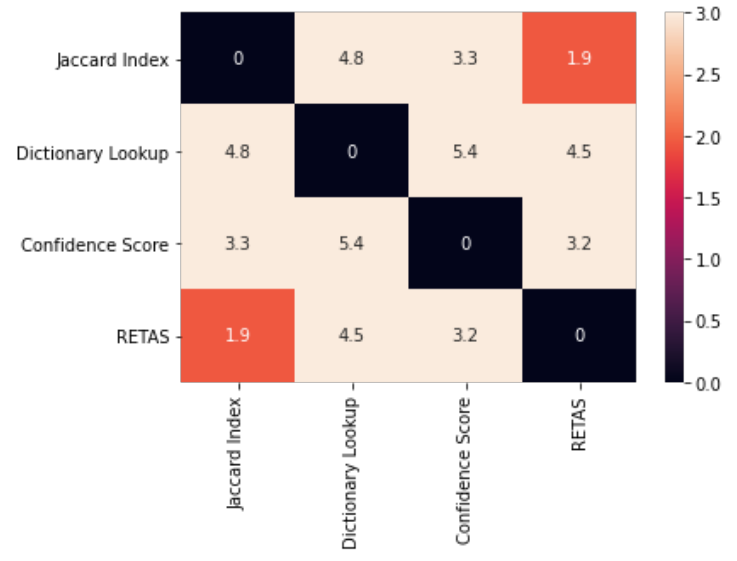


Figure 12.3

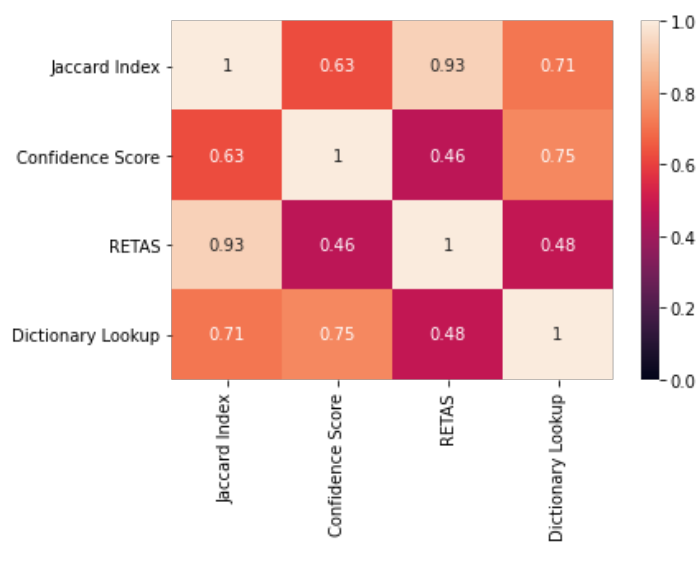


Figure 12.4

Correlation Heatmap (12.1), boxplot (12.2), RMSE (12.3) and R2 score (12.4) for the two supervised and two unsupervised methods using both synthetically generated input files, files with scripted fonts and the files downloaded from the project Gutenberg website. For the boxplot in the y-axis the accuracy is given (a value from 0 to 1) and the x-axis, all the four (two supervised and two unsupervised) methods.

We also have plotted the scatter plot between the 4 methods, that means 6 scatter plots in total. In those scatter plots the comparison between the accuracy scores for all the methods are given.

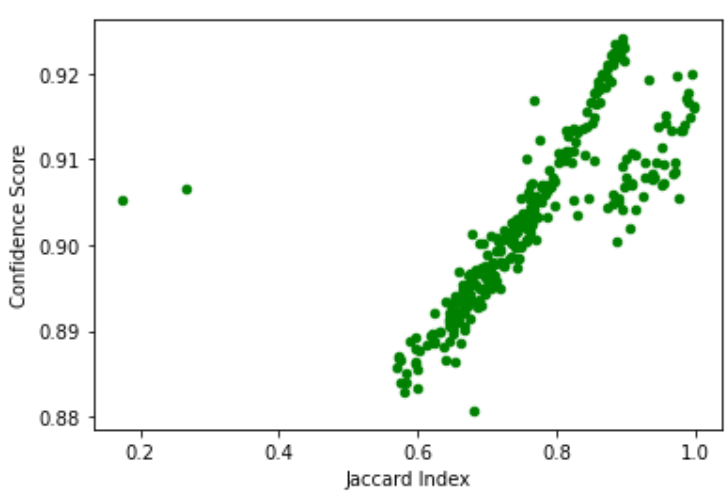


Figure 13.1: Jaccard index vs Confidence Score based approach

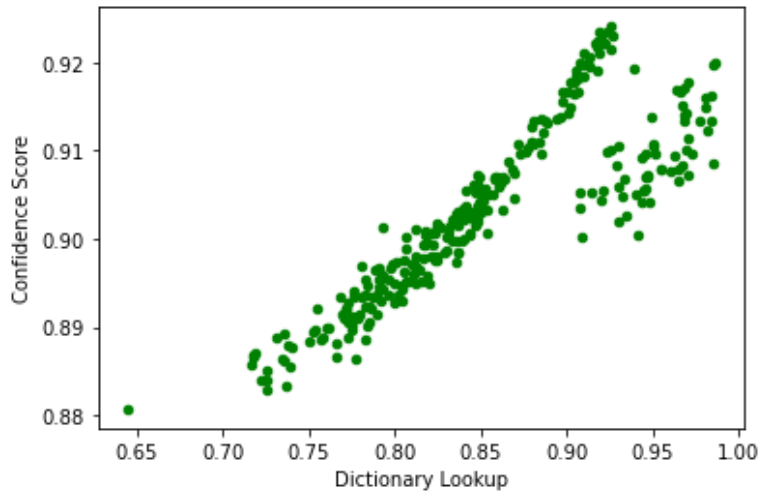


Figure 13.2: Dictionary Lookup vs Confidence Score based approach

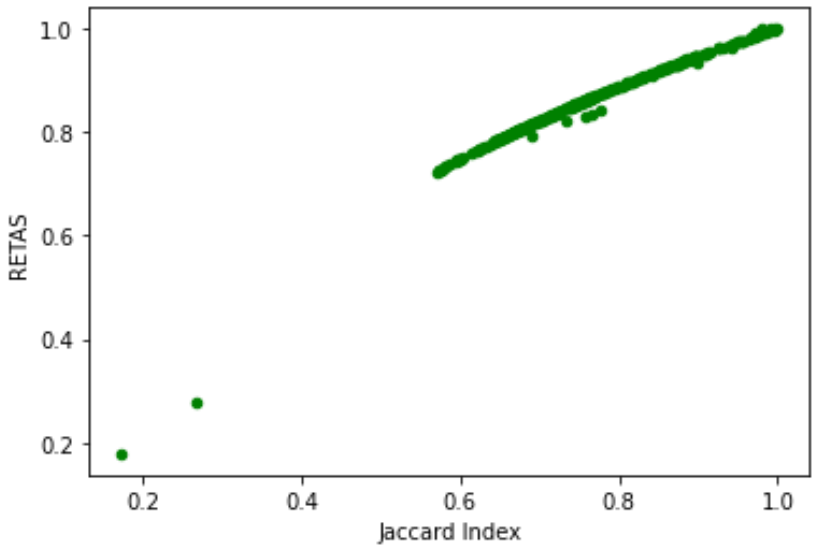


Figure 13.3: Jaccard index vs RETAS method

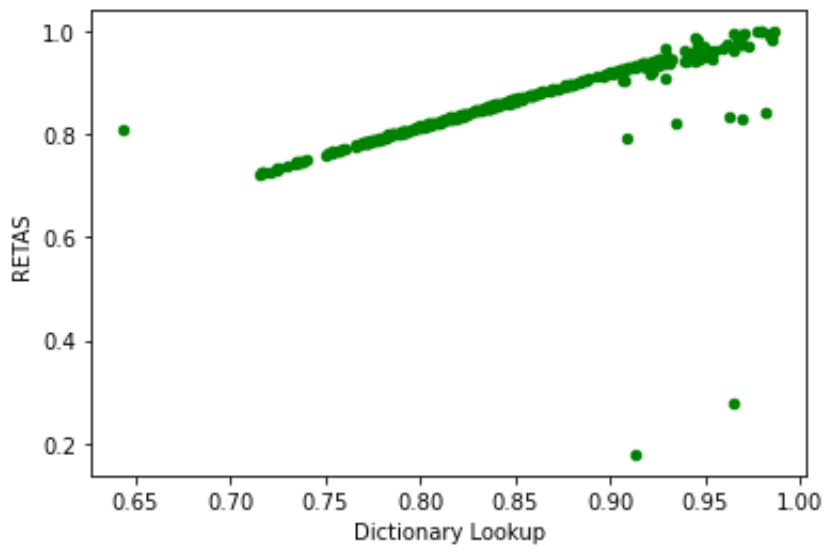


Figure 13.4: RETAS method vs Dictionary Lookup method

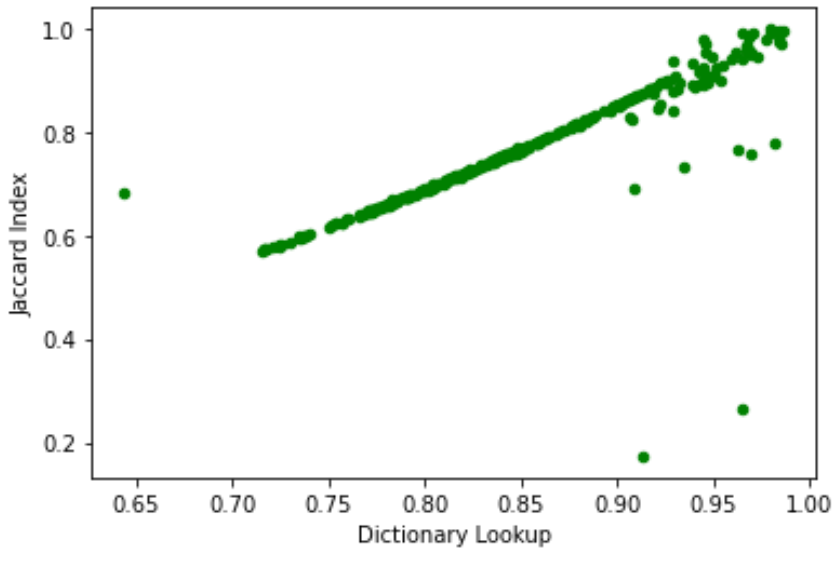


Figure 13.5: Jaccard index vs Dictionary Lookup method

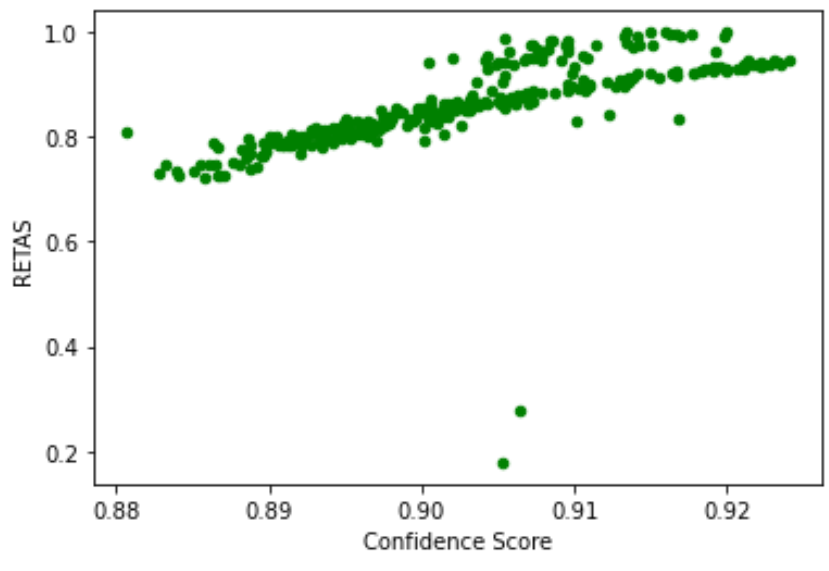


Figure 13.6: RETAS method vs Confidence Score based approach

Pairwise comparison of the accuracy values' distribution among all the 4 methods via scatter plots.

3.4 Analysis using real life scanned documents:

So far, we have done all the analysis either using the synthetically generated data or, some text-file downloaded from some website. But, for the final evaluation of the newly proposed confidence score-based approach or the previously used Dictionary Lookup method, some real-life documents were needed. For this we have used some files already present in iManage. These documents were mostly scanned images. Also, there was some signatures, dates in handwritten fonts. We have taken around 50 such type of documents. We have checked the accuracy scores for each documents using both the unsupervised accuracy measures.

In this evaluation process we did not have any ground truth text. So, we have done a manual checking and assigned a rating as per our preference out of 10 for each file.

Basically, for each scanned documents we arbitrarily have chosen some 2 or 3 pages and compare the corresponding page for the OCRed text file. Considering the fact that this evaluation may be biased, which is the biggest disadvantage of this manual checking; we also have found some advantages too. The very first advantage is the fact that we at least have some proxy to the ground truth. The other advantage was actually some interesting observations, which we have made during this manual checking. In spite of giving us nearly 100% accuracy for the printed non-scripted normal fonts, the OCR engine performs horribly for the handwritten digits. The other observation was that for the low-quality scanned images (hazy picture or some unnecessary marks in the pdf) the OCR engine gives us very low 'confidence score' even for the correctly recognized characters. As a result, our newly approached confidence score-based accuracy measure fails badly for these documents. Here I'm providing the correlation coefficients and the boxplots corresponding to these. Also, follow the Appendix (section 6) for the detailed table corresponding to these scanned documents.

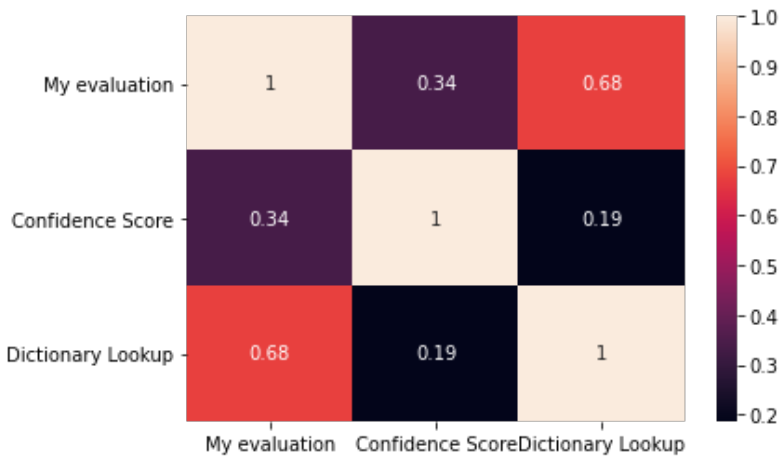


Figure 14.1: Correlation heatmap for confidence score-based accuracy measure, dictionary lookup method and OCR quality rating given by me for scanned documents

Figure 14.2: Boxplots for confidence score-based accuracy measure, dictionary lookup method and OCR quality rating given by me for scanned documents

4. Conclusion:

In search of an efficient algorithm for the unsupervised accuracy measuring technique, the method we found based on Confidence score can be a very good proxy against the dictionary lookup method already present inside the company.

Initially in the testing phase we notice that one problem with the confidence score is whenever the font style of the pdf is like the typical printed format font, not like scripted font or handwritten digits, then the confidence scores for each character lies around 80 to 100%. As a result, the accuracy seems high comparing to the other methods and its boxplots become dense most of the cases. But at the same time, it correlates well with the other accuracy measuring methods. So, fixing a threshold can resolve the problem.

After considering the low accuracy documents via synthetic input method, the method dictionary lookup and Confidence score both gave us a correlation above 70% in all the cases. The main advantage of the confidence score-based approach is we can fetch our required information directly from the output file our OCR Engine (preferably in .csv format), unlike the dictionary lookup method. In the dictionary lookup method text formation and tokenization of words is needed before checking. Even the checking itself is a time-consuming method comparing to the confidence score-based approach.

But when we consider the real world scanned documents, the scenario got changed. The method for evaluating OCR quality using confidence scores produced by the OCR engine yield a strong correlation neither with the dictionary lookup method, nor with the accuracy scores given by us by manual inspection. Probably this problem arose because of the document picture quality. But whatever the problem is, these results help us to conclude that this method is not good enough comparing to the dictionary lookup method, as it correlates better with the scores given by us manually, that is in simple words, dictionary lookup performed better for real-life scanned documents.

Finally, I want to conclude that, in spite of this poor correlation values I am still sure about the fact that the confidence score can be a really good proxy for the dictionary lookup method, which definitely will be more efficient than the dictionary

lookup in terms of the computational complexity. Initially, we tried to use median, mode, GM instead of mean (AM), but accuracy scores obtained from all these methods have a high correlation (95%) with the method explained in this project. So, our final conclusion is that *confidence score can be used in accuracy measurement but with some different approach*, which requires further research.

5. Bibliography:

1. **A Fast Alignment Scheme for Automatic OCR Evaluation of Books** by *Ismet Zeki Yalniz, R. Manmatha,* Multimedia Indexing and Retrieval Group, Dept. of Computer Science, University of Massachusetts.
2. **Measuring the accuracy of page-reading systems** by *Stephen Vincent Rice*, University of Nevada, Las Vega.
3. **A Hierarchical, HMM based Automatic Evaluation of OCR Accuracy for a Digital Library of Books** by *Shaolei Feng and R. Manmatha*, Computer Science Department, University of Massachusetts, Amherst.
4. **A generic approach for OCR performance evaluation** by *A. Belaïd and L. Pierron.*
5. **Metrics for Complete Evaluation of OCR Performance** by *Romain Karpinski, Devashish Lohani, Abdel Belaid*.
6. **Assessing the Impact of OCR Quality on Downstream NLP Tasks** by *Daniel van Strien, Kaspar Beelen, Mariona Coll Ardanuy, Kasra Hosseini, Barbara McGillivray, Giovanni Colavizza*.
7. **Automatic quality evaluation and (semi-) automatic improvement of OCR models for historical printings** by *Uwe Springmann · Florian Fink · Klaus U. Schulz*.
8. **ZoneMapAlt: An alternative to the ZoneMap metric for zone segmentation and classification** by *Romain Karpinski, Abdel Belaid*.
9. **Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm** by *Andrew J. Viterbi*.
10. **Project Gutenberg Website**, http://www.gutenberg.org 2011.
11. **RETAS method repository,** https://github.com/Early-Modern-OCR/RETAS.

6. Appendix:

This is the table of the scanned documents used to evaluate the performance of the two unsupervised accuracy measuring methods (Confidence score based accuracy measuring method in column 3 and Dictionary Lookup method in column 4). In column 2 the accuracy rating (out of 10) given by manual evaluation is there. In the correlation and boxplot calculation we transformed these values out of 1 instead of 10.

Filename	Manual Evaluation	Confidence Score	Dictionary Lookup
file01	10	0.980327127	0.992424978
file02	8	0.648290348	0.98214857
file03	9	0.956337246	0.986263335
file04	10	0.926827821	0.992468462
file05	9	0.836073035	0.985192351
file06	9	0.940471223	0.987251402
file07	10	0.980456206	0.992453107
file08	9	0.985240164	0.979654501
file09	8	0.870989639	0.953535177
file10	9	0.847800648	0.959615581
file11	9	0.812905561	0.967689048
file13	10	0.973620899	0.991226819
file15	9	0.956082992	0.985557769
file16	8	0.485856164	0.975984932
file17	9	0.892498635	0.975535684
file18	10	0.978154464	0.986168313
file19	10	0.783051549	0.99123506
file20	9	0.906417042	0.977149075
file21	8	0.82442636	0.954450435
file22	9	0.638171728	0.987480714
file23	8	0.668211588	0.964593809
file24	9	0.627219124	0.978441352
file26	9	0.883809263	0.958711479
file27	8	0.902921526	0.963208502
file28	9	0.986444172	0.985742727
file29	10	0.413059553	0.991772763
file30	9	0.978413432	0.985276796
file31	9	0.651442126	0.984913793
file32	8	0.588968884	0.958221701
file33	9	0.951716902	0.979841173
file34	9	0.859702395	0.97795668
file35	8	0.459272959	0.938186382
file36	10	0.866232768	0.99154334
file37	9	0.74152156	0.989981666
file38	9	0.818444358	0.992875424
file39	9	0.756090875	0.988804071
file40	9	0.725221065	0.992689613
file41	9	0.946890715	0.982369824
file42	9	0.311077744	0.988047809
file43	9	0.993372993	0.968798066
file44	9	0.794622355	0.951055231
file45	9	0.995054048	0.987853403
file46	10	0.872043542	0.987962167
file47	9	0.495809897	0.978995757
file48	10	0.983768411	0.993015307
file49	9	0.666142686	0.97892198