Development of the documents comparison module for an electronic document management system

M A Mikheev[1], P Y Yakimov[1,2]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086
[2]Image Processing Systems Institute of RAS - Branch of the FSRC "Crystallography and Photonics" RAS, Molodogvardejskaya street 151, Samara, Russia, 443001

e-mail: mmasyz@mail.ru, Pavel.y.yakimov@gmail.com

Abstract. The article is devoted to solving the problem of document versions comparison in electronic document management systems. Systems-analogues were considered, the process of comparing text documents was studied. In order to recognize the text on the scanned image, the technology of optical character recognition and its implementation — Tesseract library were chosen. The Myers algorithm is applied to compare received texts. The software implementation of the text document comparison module was implemented using the solutions described above.

1. Introduction

Document management is an important part of any business today. Negotiations, contracts, protocols, agreements must be documented. Documents go through several stages of preparation in most cases, and there is a possibility of replacing or correcting of files by one member without notifying other members. EDMSs with built-in checking mechanisms guarantees the identity of two documents versions in this case.

A variety of data formats as well as a low degree of automation or its complete absence significantly complicates the process of finding differences between documents. As a result, some companies ignore such important process without considering how risky this approach to organization of workflow in a company can be [1].

All the above suggests that the topic chosen by the authors is relevant. Automation of documents and, in particular, the process of comparing versions of documents requires attention and detailed study.

There are a lot of web services and software products for document comparison now. The most common are ABBYY Comparator and Compare Suite. They provide a lot of features for customers. Also, there are software products embedded in enterprise applications and extending its functionality, such as ABBYY ScanDifFinder SDK. But the main drawback all these products is the high coast (up to 40% of the license value).

This paper considers a product that can be used as a component in main application, based on open source libraries and process documents without sending them to the third party.

2. Studying of documents comparison process

The task of text documents comparison is quite extensive. The case of PDF documents comparison is considered in this work. A simplified diagram of this process is presented in Figure 1.

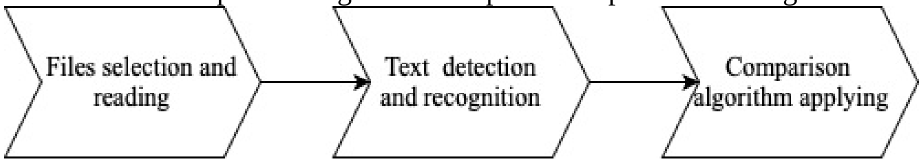

Files selection and reading
Text detection and recognition
Comparison algorithm applying

Figure 1. Document comparison process.

The comparison task is complicated by the fact that the PDF files involved in the process can contain both plain text and scanned copies of documents. Thus, first of all, the text in the image should be detected and recognized, and then the comparison algorithms should be applied.

2.1. Text recognition

Optical Character Recognition (OCR) is a mechanism that converts printed text or handwriting images into digital form for the purpose of further editing, searching, analysis.

OCR has a number of significant problems, including a variety of languages and fonts, distortion of images of characters, size and scale variations of characters. Methods from various computer science disciplines (image processing, pattern classification, natural language processing) are used to solve these problems [2].

The OCR process is a complex activity consisting of several phases performed sequentially.

Image retrieval is the initial stage of OCR, which involves extracting a digital image and transforming it into a suitable form that can be easily processed by a computer. This can involve quantization as well as compression of image [3, 4].

Various preprocessing methods are used to improve the image quality after its receiving: noise removal, thresholding, min and max filters.

At the segmentation stage, the characters in the image are separated for further transmission to the classification. The simplest methods are analysis of related components and projection profiles. However, in situations where characters are overlapped or broken, advanced character segmentation methods are used.

In the next step, the segmented characters are processed for extracting various features that uniquely identify these characters. Then characters are recognized on basis this features. The extracted features should be efficiently computed, minimize intraclass variations and maximize interclass variations.

At the classification stage features of a segmented image are associated with various categories or classes. Some of the approaches of statistical classification are the Bayes classifier, the classifier of the decision tree, the neural network classifier [5], the classifiers of the nearest neighborhoods, etc. [6].

After a character has been classified, various approaches can be used to improve the accuracy of recognition results. One approach is to use several classifiers, the results of which can then be combined. To improve recognition results, you can also perform context analysis [6].

The most popular OCR implementation is the Tesseract engine. Tesseract is an open source library currently maintained by Google. The principle of its work is described in article [7].

Document processing goes according to the traditional step-by-step scheme. At the first, page layouts are defined. After that, text strings are grouped into Blob objects. Strings are selected in the resulting objects, and then they are broken into words by analyzing the length of the spaces between the characters.

The recognition process is carried out in two stages. At the first, the algorithm tries to recognize each word in turn, and passes each successful attempt to the classifier as a training set. The classifier further uses this data when moving through the page at the second stage of recognition, when the algorithm re-works with those words that could not be recognized for the first time.

A distinctive feature of the latest versions of Tesseract is using LSTM (long short-term memory) networks to define page layouts, select lines and individual words. Long short-term memory networks are a type of recurrent network (RNN). However, unlike conventional RNN that contain one layer with the tanh activation function, the LSTM contains 4 layers that interact in a certain way (Figure 2).

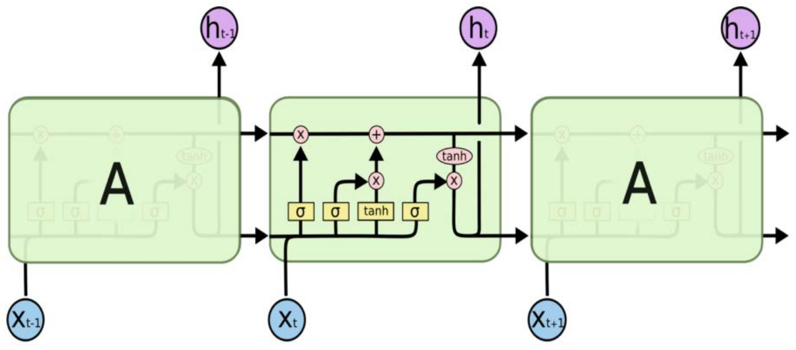


Figure 2. LSTM model.

So, LSTM solves the problem of long-term dependence (loss of ability to bind information due to the large distance between the actual information and the point of its application).

The introduction of LSTM networks in Tesseract has led to a significant improvement in recognition results. The number of errors decreased on 15% [8].

There are other open source software products designed for text recognition. The best results show Tesseract, CuneiForm and OCRopus (table 1). However, with the release of Tesseract 4, which supports LSTM, this library gets preference.

Table 1. The average percent recognition errors.

Library	Errors, %
Tesseract	14,04
Ocropus	31,42
CuneiForm	37,68
Ocrad	50,25
GOCR	64,46

2.2. Simple texts comparison

Detecting differences between words is a fundamental part of text comparison. Formally, the task of determining the differences between two sequences of characters is to find the longest common subsequence or, equivalently, find the minimum script for deleting and inserting characters that transform one sequence into another.

The most popular algorithms for finding the largest common sub-sequence are the Wagner-Fischer algorithm, the Hirschberg's algorithm, and the Hunt-Szymanski algorithm.

The idea of Wagner and Fischer is to consistently estimate the distance between all the longer line prefixes — until the final result is obtained. Intermediate results are calculated iteratively and stored in an array of length $(m + 1) \times (n + 1)$. Thus, the calculation required $O(m \times n)$ time using memory of the same order.

The Hirschberg's algorithm computes LCS in $O(n_1 \times n_2)$ time, but uses memory of $O(n)$, where $n = \min \{n_1, n_2\}$. Saving memory is due to the fact that at any given point in time, only two rows of the array required by dynamic programming are stored in memory. However, this leads to a more sophisticated method of allocating two lines of LCS.

The Hunt-Szymanski algorithm computes LCS in a $O(n \times \log (n))$ time, using memory of $O(n)$, where $n = \max \{n_1, n_2\}$. The time complexity of this algorithm exceeds $O(n \times \log (n))$ only when the rows x_1 and x_2 have the number of matching letters greater than a value of the order of $O(n)$ (for example, when the size of the alphabet is small relative to the length of the rows).

In practice, the calculation of the distance between the lines is of interest only when this distance d is relatively small compared to the length of the lines. In this case the Ukkonen-Myers algorithm, considered in article [9], is preferable. The complexity of the Myers algorithm is approximately proportional to the sum of the lengths of the sequences, and not to the product, as in the Wagner-Fischer algorithm. The same can be said about the memory used by the algorithm $O(n_1 \times n_2)$, where n = min $\{n_1, n_2\}$. Thus, this is the best-known algorithm for solving our problem [10, 11, 12].

3. Development of the document versions comparison module

The development of any system begins with the design stage. Design includes the development of a system model at the logical and physical levels, as well as the choice of software.

3.1. Choice and justification of the software

CUBA Studio 2018.3, developed on the basis of IntelliJ IDEA was chosen as the development environment and Java was chosen as a programming language. There are a lot of open source projects, libraries, frameworks written in Java, which in terms of the task will greatly simplify the work. Java is the foundation for all types of network applications and the universal standard for developing enterprise software.

3.2. Choice of frameworks

The application is based on the CUBA platform. This platform is a high-level Java framework for quickly building enterprise applications with a full-fledged web interface. It abstracts the developer from heterogeneous technologies, allowing him to focus on solving business problems.

The text document processing module is based on the Apache PDFBox library, an open source tool that supports the development of Java applications that create, convert and process PDF documents [13].

For pixel-by-pixel document comparison, the PdfCompare library, built on the basis of Apache PDFBox, is used. It provides the ability to graphically highlight different areas of documents. Apache PDFBox contains many methods designed to manage certain aspects of the system, in particular, memory consumption [14].

Text comparison uses the high-performance Google Diff Match Patch library, which implements the Myers algorithm, which is generally considered the best general-purpose difference algorithm. some optimizations should be made to improve the runtime before such computationally expensive process as a comparison. After that, the remaining text is compared by a difference algorithm, the result is processed, taking into account the semantic features of the text [15].

3.3. Logical database model

It is necessary to develop an application that briefly reflects the subject area we need to demonstrate the work of the document comparison module. In our case, it is a document management flow.

Figure 3 shows the logical model of the database system. The main entities of the system are: “Document type”, “Attachment”, “Document category”. "Document". Entities are connected to each other by one-to-one and many-to-one relationships and are reflected in the object model of the system.

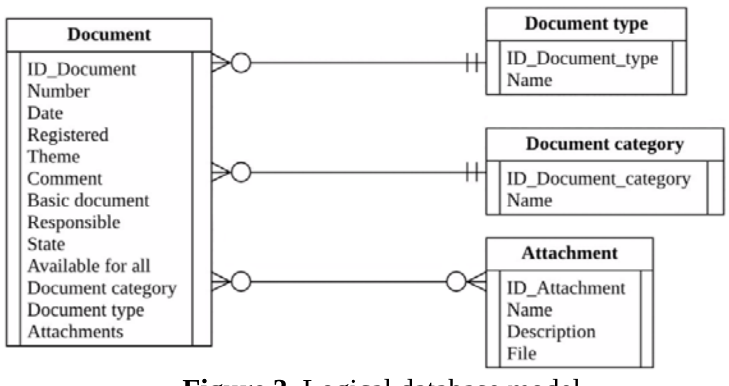


Figure 3. Logical database model.

3.4. Class diagram

The class diagram serves to represent the static structure of the system model in the terminology of the classes of object-oriented programming [16]. Figure 4 shows the class diagram of the system.

DocumentCategory, DocumentKind, DocumentAttachment and Document are entities stored in the database. Work with them is performed using a universal user interface and already implemented operations of creation, deletion and modification.

Special attention should be given to the DocumentEdit document editing screen controller. Its main compareFiles method is called by clicking on the "Compare" button and starts the document comparison mechanism by calling the compareFilesPixelToPixel and CompareFilesText methods of the DocComparisonService service. Services form a component layer that defines a variety of Middleware operations available to the client level of the application. The service interface is located in the global module and is available at the client level and at the Middleware.

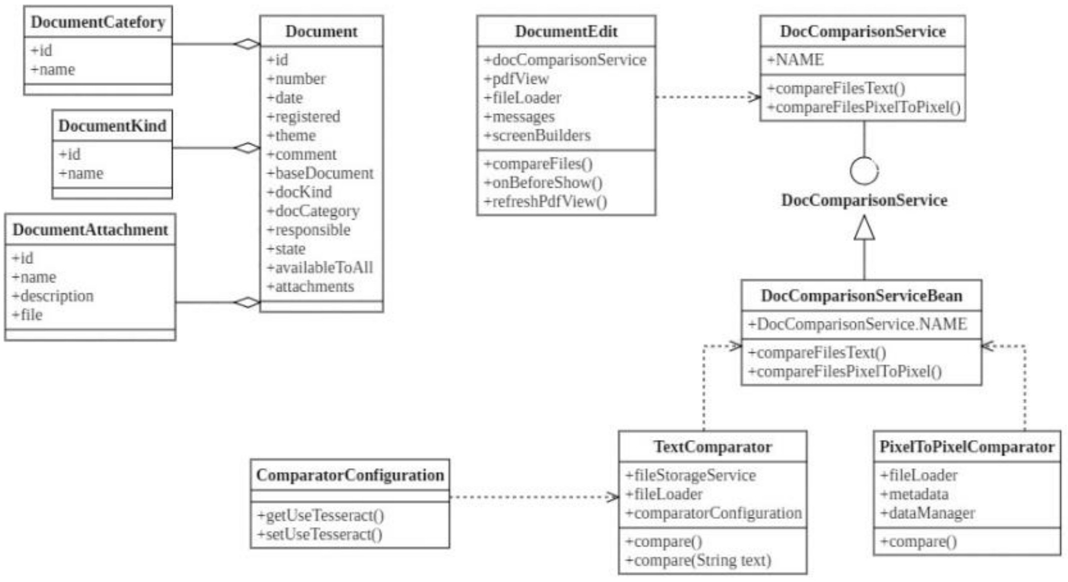
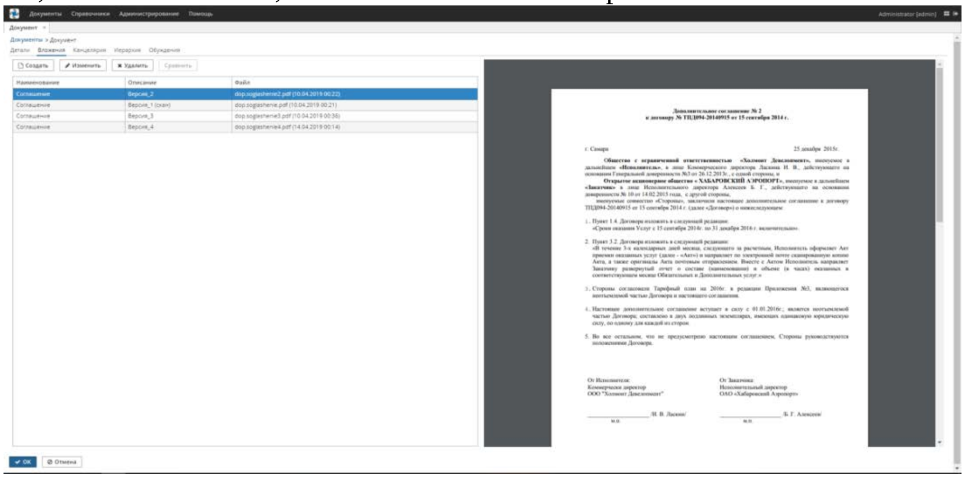


Figure 4. Class diagram of the system.

4. The results of experimental studies of the module

In order to demonstrate the work of the module, an information system that partially implements the functions of electronic document management was developed.

After launching of application and entering personal data, the main system form with the items "Documents", "Reference books", "Administration" and "Help" is shown.



Наименование	Описание	Файл
Соглашение	Версия_2	dop.soglashenie2.pdf (10.04.2019 00:22)
Соглашение	Версия_1 (скан)	dop.soglashenie.pdf (10.04.2019 00:21)
Соглашение	Версия_3	dop.soglashenie3.pdf (10.04.2019 00:38)
Соглашение	Версия_4	dop.soglashenie4.pdf (14.04.2019 00:14)

Дополнительное соглашение № 2
к договору № ТПД094-20140915 от 15 сентября 2014 г.

г. Самара 25 декабря 2015г.

[illegible]

Figure 5. Documents editing form.

The document editing form is a set of tabs "Details", "Attachments", "Office", etc.

The Attachments tab (Figure 5) presents a form containing a table of document versions, the main attributes of which are “Name”, “Description” and “File” — a PDF document attached in the attachment editing screen. A form of a preview document is shown on the right side of the screen.

Pressing the button "Compare" causes the main logic — the comparison of documents. Upon completion of the comparison, the form opens, with the results of pixel-by-pixel and textual comparison (Figure 6).

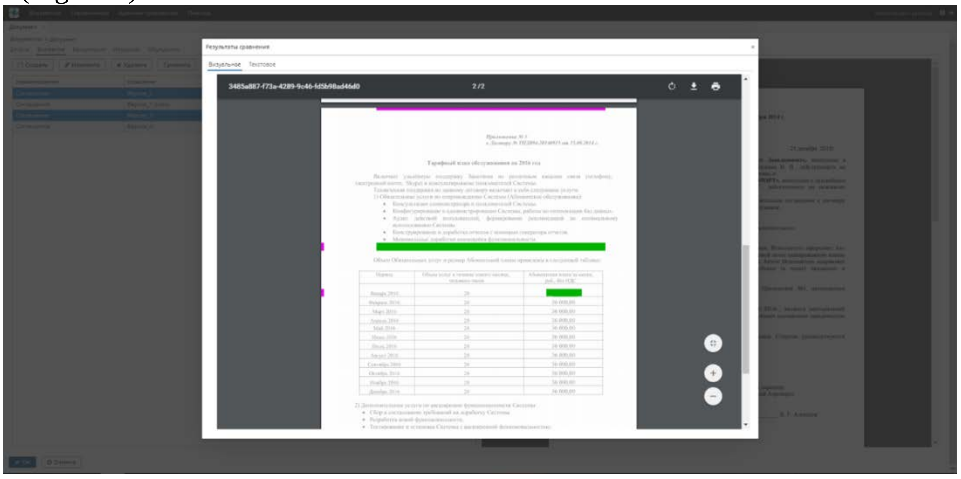


Figure 6. Visual comparison result form.

Differing pixels are marked in red and green. Ignored areas are marked with a yellow background. Deleted pages are highlighted with a red frame and added pages highlighted with a green frame (Figure 7).

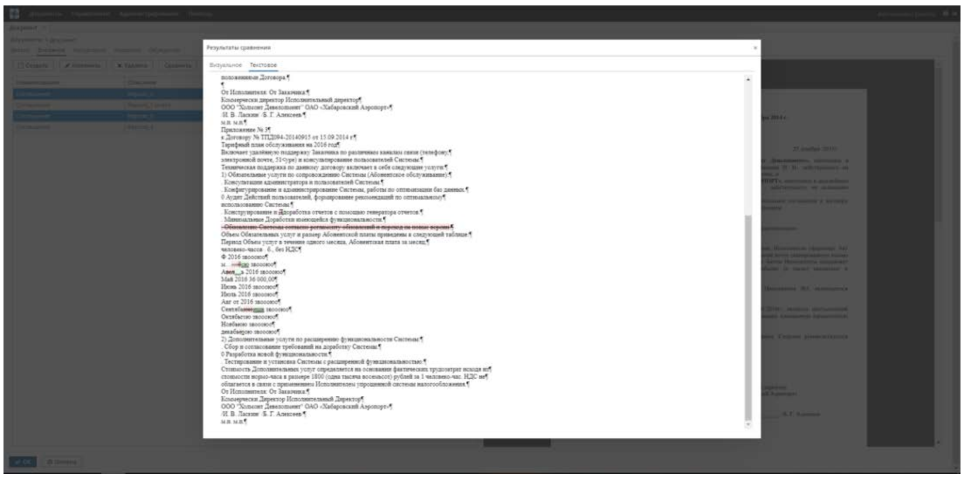


Figure 7. Textual comparison result form.

Added characters are highlighted in green and underlined, deleted characters are highlighted in red and strikethrough.

Several experiments were conducted in order to explore recognition accuracy. 3 pdf-files containing scanned copies of text documents participated in the test. Recognition results are presented in table 2.

Table 2. The result of recognitions.

Experiment’s number	Number of recognized characters / Total number
Experiment 1	1127 / 1148
Experiment 2	1502 / 1531
Experiment 3	964 / 1021

The percentage of correctly recognized characters is more than 98%, which confirms the data declared by the Tesseract's developer. Most of the mistakes are in recognizing punctuation marks and some pairs of similar characters.

The main drawback detected during the experiments process is the incorrect recognition of multi-column text, as well as recognition errors in the presence of pictures and in areas with stamps and signatures. These features should be fixed on the pre-processing stage.

5. Conclusion

As a result of the work done, the task of text documents comparison in the electronic document management system was solved.

The mechanism of optical character recognition on the image for obtaining texts from the scanned image was studied. The most popular libraries that implement OCR were considered, a comparative analysis of the effectiveness of character recognition was conducted. The main approaches to the comparison of texts are investigated, in particular, the Myers algorithm, which is the most suitable in our case, is considered.

The result of the work is the results of researches that substantiate the choice of technologies used, as well as a software product available for use by third-party developers. Currently, the developed module allows to compare text documents in PDF format and returns the result in a form that is easy to read.

6. References

[1] Algorithms and document comparison software analysis URL: http://cad.kpi.ua/attachments/093_2017p_Ishchenko.pdf (16.01.2019)

[2] Islam N, Islam Z and Noor N 2016 A Survey on Optical Character Recognition System *Journal of Information and Communication Technology* **10** 1-4

[3] Lazaro J, Martin J L, and Arias J 2010 Neuro semantic thresholding using OCR software for high precision OCR applications *Image and Vision Computing* **28** 571-578

[4] Lund W B, Kennard D J and Ringger E K 2013 Combining Multiple Thresholding Binarization Values to Improve OCR *Output Document Recognition and Retrieval XX Conf.* **8658** 11

[5] Amosov O S, Ivanov Y S and Zhiganov S V 2017 Human localiztion in video frames using a growing neural gas algorithm and fuzzy inference *Computer Optics* **41(1)** 46-58 DOI: 10.18287/2412-6179-2017-41-1-46-58

[6] Ciresan D C, Ciresan D C, Meier U and Gambardella U 2011 Convolutional neural network committees for handwritten character classification *Int. Conf. on Document Analysis and Recognition* **2** 1135-1139

[7] Smith R 2007 An overview of the Tesseract-OCR engine *Int. Conf. on. IEEE* **2** 629-633

[8] Understanding LSTM Networks URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (17.04.2019)

[9] The Myers diff algorithm url: https://blog.jcoglan.com/2017/02/12/the-myers-diff-algorithm-part-1/ (16.04.2019)

[10] Myers E W 1986 An O(ND) Difference Algorithm and its Variations *Algorithmica* **1** 251

[11] Common subsequences URL: http://algolist.manual.ru/search/lcs/ (16.04.2019)

[12] Smith B 2006 *Methods and algorithms of calculation on strings* (Moscow: Williams publisher) p 485

[13] Apache PDFBox librart URL: https://pdfbox.apache.org (16.04.2019)

[14] PdfCompare library URL: https://github.com/red6/pdfcompare (22.04.2019)

[15] Diff Strategies URL: https://neil.fraser.name/writing/diff (22.04.2019)

[16] Leonenkov A V 2007 *UML 2 Tutorial* (Saint Petersburg: BHV-Petersburg) p 558

Acknowledgements

This work was partly funded by the Russian Federation Ministry of Science and Higher Education within a state contract with the "Crystallography and Photonics" Research Center of the RAS under agreement 007-ГЗ/Ч3363/26.