A Simple and Uniform Way to Introduce Complimentary Asynchronous Interaction Models in an Existing Document Analysis System

Joseph CHAZALON and Bertrand COÜASNON
Université Européenne de Bretagne
INSA, IRISA (UMR 6074), Rennes, France
{*joseph.chazalon,bertrand.couasnon*}*@irisa.fr*

Aurélie LEMAITRE
Université Européenne de Bretagne
Univ. de Rennes 2, IRISA (UMR 6074), Rennes, France
aurelie.lemaitre@irisa.fr

***Abstract*—Extracting and indexing meaningful contents from degraded documents, like historical ones, is commonly considered as a challenging problem. Existing analysis systems usually rely on a manual correction of results during a post-processing stage, and cannot make use of external information to adapt their response. This paper presents how an existing document analysis system can be easily adapted to enable an efficient interaction during the analysis stage, and benefit from external information. We identify the minimal architecture required, and we detail the two complimentary interaction models we propose: a directed interaction model which allows to handle cases where errors can be automatically detected; and a spontaneous interaction model which permits to cope with the other cases. Both models are asynchronous to avoid to make the human operator or the system wait for each other during mass document processing. They are based on a common foundation which uses standard exception-like mechanisms to implement error detection, correction and recovery aspects. Our system was tested on several tasks. For the transcription of handwritten words in documents dating from the 18th century, where we were able to diminish the human workload by 28% for an overall recognition rate of 80%.**

***Keywords*-human interaction; asynchronous interaction; interaction models; degraded documents; historical documents**

I. INTRODUCTION

Difficulties encountered during content extraction and indexing in old and degraded documents have at least 2 main causes. *First*, the media is damaged, and the content is altered. Documents are noisy channels which require error-correcting analysis methods. *Second*, both contents and structures in historical collection can vary a lot. Printed and handwritten contents can be mixed, writers can change, vocabulary is outdated or unknown in advance (family names for instance), structures are not constrained by some modern formating system... The models we use tend to be unreliable as unexpected elements will very probably be encountered in the important amounts of pages we process.

Those causes lead to important amounts of errors during the mass processing of documents. Contextual dependencies between elements increase the final amount of manual corrections needed. It is often more efficient to let a human operator manually produce the results instead of correcting all the errors after the processing. Therefore, an analysis system should make use of external information *during* the processing stage to improve its response to *past and future pages* and globally *reduce the amount of human work required*. This interaction must be *asynchronous* to prevent the human and the system from waiting for each other.

This paper details how, using a uniform basis, complimentary interaction models can be easily implemented in an analysis system. In Sec. II, we review existing interactive approaches for document processing and show that two interaction models are necessary. In Sec. III, we present the minimal architecture required to enable those models. In Sec. IV, we focus on how we transformed our page analyzer using a common basis, easy to adapt to other systems, for both models. In Sec. V, we illustrate the capabilities of such system on two experiments we made.

II. RELATED WORK

Few document analysis systems allow some human operator to interact with them during the analysis stage to guide or correct their response. Some systems among those few which illustrate well the challenges we face.

The Edelweiss system [1] proposes a first kind of interaction. It process document pages and progressively produces an XML structure some human operator can edit at some points in the analysis process. On the one hand, it permits to influence the response of the system for the rest of the page processing by virtually transforming any element in the document structure, but on the other hand, it does not reconsider previous intermediary results, and does not validate human-provided information against some document model.

The smartFIX system [2] give a practical view of what motivates the need for interacting with a human operator. It is designed for mass processing of business documents, like medical invoices, and automatically triggers business transactions. It first extracts and recognizes useful elements from documents and gives them a confidence score. For problematic ones, a human correction is needed, and for suspicious ones, a human validation is necessary. While this

approach exhibits two different behaviors for interaction depending on the confidence associated to extracted elements, it limits the modifications made by a human to final results, preventing to correct original errors and to guide the analysis at the right moment.

Nagy and Veeramachaneni explore, in [3], some links between document processing and interaction. In particular, they suggest that *"the operator should not even have to look at data that the system had no trouble in classifying"*, and also that *"every interaction should be utilized by the system to improve subsequent classification"*. There first requirement relies on what they call *"machine-initiated interaction"*, where the human operator reacts to problems which were detected. However, the detection of errors and the localization of their cause cannot always be done automatically, while it is still necessary to replace some parts of the automated analysis by human processing.

The ability to detect errors is the key criterion to decide which interaction model use. We propose to distinguish interaction which is *"directed"* (by automatic error detection) [4], where error detection is possible, and where a question (which requires an answer) can be asked to a human operator; and interaction which is *"spontaneous"* [5] for case were error detection is not possible. For the latter, interaction is *"human-initiated"* as the automated system will have to react to external information, and use it to improve its response on a given page. The following sections present how an *iterative analysis* of pages permits to adapt the system response, and how a common foundation enables *both* directed and spontaneous interaction models. System improvement using machine learning will not be discussed, even if our architecture supports it.

III. Iterative Analysis Architecture for Human Interaction

In order to enable some interaction between a human operator and the document analysis system which could occur during the analysis stage, be asynchronous, and permit to reconsider previous results for the pages processed, we need: i) an iterative analysis of pages; ii) a *visual memory* structure; iii) appropriate software components. This section details those aspects, and shows how important is the design of the page analyzer to make use of external information.

A. Iterative Analysis Overview

As an example, the simplest iterative analysis scenario we use is the one where we first process all the pages of a collection automatically, then we let the human operator asynchronously provide necessary information about pages. Each page is then reprocessed using external information, and this scheme is repeated as many times as needed. At each "iteration", new information about the page being reprocessed is provided.

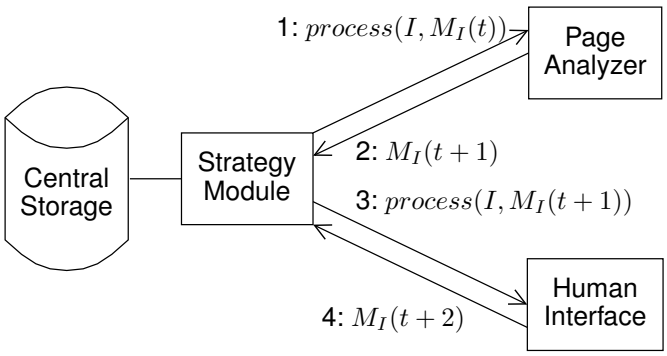

Figure 1. A single step (repeated as many times as needed during the *iterative analysis*) of a possible information exchange about a given image between a page analyzer and a human interface, showing the implied components of the architecture and the *visual memory* M_I associated to the image I. At time t, the strategy modules first invokes the page analyzer on the image I, with the memory $M_I(t)$. The strategy module gathers $M_I(t+1)$ after the analysis, and transfers it to the human interface, which, in turn, produces $M_I(t+2)$ and sends it back.

To enable such approach, both the system and the human operator have to be given the same goal, and the ability to exchange information about the content of the pages with each other, according to a previously established protocol. As interaction must be asynchronous, the system has to be able to store information associated to pages, and therefore it should be able to process them independently from the human operator. The human operator provides as much information as required (*directed* model), and may be able to provide extra elements (*spontaneous* model).

B. Visual Memory

The *"visual memory"* is a persistent data structure associated to each page which is used for information exchange between processes, but also during document analysis in order to reintegrate external information and use it. It is an associative structure that each process can change, and which gives to each data a corresponding shape and position in in its image's referential. In order to enable, during page analysis, the fusion between the current image content and external information, this structure is very strongly integrated in the page analyzer. It permits to guarantee that, at each moment, external data is available and that creation, deletion and modification of elements contained in the visual memory is possible.

As an example, the content associated to some image containing a table of numbers could be composed of geometric objects indicating column types, recognized numbers and their values, suspicious numbers, etc. We note $M_I(t)$ the visual memory related to some image I at time t.

C. Necessary Components

The iterative analysis of document pages is made possible thanks to the following components, illustrated in Fig. 1.

Strategy Module: It manages the invocation of the different processes during the analysis and their communication. This is the abstraction level where the analysis scenario is played, and where collection properties can be used.

Central Database: The central database stores the associated visual memories for each page in the whole collection. It provides a *collection-centric view* about the knowledge about the pages which progressively grows.

Page Analyzer: The page analyzer uses a document model to try to locate and recognize contents from a given image I and its associated visual memory $M_I(t)$. The iterative analysis scheme allows it to transform $M_I(t)$ and return an updated content in $M_I(t+1)$ to the strategy module when done. At each iteration, it *reprocesses the whole image* (we integrated several optimizations to keep some results) and makes use of external information which was asynchronously provided. This permits to produce new structured results, which are validated against the document model. In a directed interaction model, the page analyzer can generate questions indicating its lack of knowledge.

Human Interface: When solicited by the strategy module, the human interface enables a human operator to change the content of the visual memory.It is important that the page analyzer and the human interface are able to produce *similar elements* in the visual memory of each image, as they have to collaborate in such a way that the human operator may sometimes *substitute himself to parts of the automated analysis to produce intermediary results*.

IV. Uniform Implementation for Directed and Spontaneous Interaction Models

We saw the page analyzer is a critical component to enable interaction. In previous papers, we separately demonstrated the interest of a directed interaction model (see [4], [6], and Sec. V-A) and the interest of a spontaneous interaction model (see [5] and Sec. V-B). Both models permit, when used appropriately, to reduce the amount of manual corrections required, for a given quality level, when compared to manual correction during post-processing.

This section shows how we integrated those two interaction models in a same page analyzer, thanks to a versatile implementation of the the directed interaction model which addresses not only specific cases where automatic error detection is possible, but also supports cases where no error detection is required. We briefly describe the DMOS-P system for which the prototype implementation was designed, then we focus on each interaction model and show they can be implemented using the same foundations.

A. DMOS-P Framework and Language

DMOS-P [7] is a concept-driven grammatical document analysis method. It uses a bi-dimensional extension of Definite Clause Grammars, called *Enhanced Position Formalism* (EPF), to describe pages. The following example illustrates its syntax: to recognize `A`, we try, at the `top` of the image, to recognize `B`, and if it fails, we try, at the `bottom`, to recognize `C`. Derivation is noted "`::=`", concatenation is noted "`&&`" and alternatives are expressed with multiples derivations for a same left-hand part.

```
A ::= AT(top) && B. % clause 1, tried 1st
A ::= AT(bottom) && C. % tried if 1 fails
```

The "`AT`" operator is used for the positioning the analyzer's "reading head": it defines a zone in the image plan, and only elements which are present in this zone will be considered in the rule which follows this operator.

Like attribute grammars, rules have input and output parameters indicated by "+" and "−" signs, as illustrated by:

```
recognizeNumber(+NumPos, -Value) ::=
   callClassifier(+NumPos, -Value).
```

B. DMOS-P Extensions for Directed Interaction

The directed interaction model we propose was introduced in [4] and its efficiency was demonstrated in [6]. This model relies on the introduction of 3 new operators in the page description language. The semantics associated to those operators permit to express properties about the page model. Their implementation allow to change the behavior of the page analyzer to automatically handle the interaction with a human operator or other components. It is based on an asynchronous information exchange made of questions and answers and using the visual memory, throughout the various page analyzer invocations at each analysis iteration.

The visual memory carries (and locates in the image referential) *questions* and *answers* using the containers:

$Q(Text, DataType)$
$A(Data)$

$Text$ gives information about the problem and $DataType$ indicates the expected type for the content of the answer, which is stored in its $Data$ attribute.

From a technical point of view, the operators were implemented using a continuation mechanism in the logical programming language used in DMOS-P. Continuations are, in this case, constructs very similar to the standard exception mechanism commonly used in imperative languages. Therefore, our approach should be easily adapted to some existing analysis system.

The implementation we propose enables an asynchronous error detection and correction, and also allows to recover from errors during a page analysis. In order to make a clear analogy between our new operators and exception handling, we chose names reflecting the behavior they produce during the analysis. Those operators we will now focus on are designed to: i) detect errors and ask questions; ii) ensure that answers are used by the analyzer to make progress instead of asking the same questions forever; and iii) make as much progress as possible in independent parts of the analysis if a problem arises.

1) Asking Questions (Error Detection): This operator is used to indicate that some part of the page cannot be understood properly. It can be to indicate that no valid interpretation could be proposed (all alternatives failed); or that recognized element are inconsistent with each other (non incrementing page numbers for example); or simply that the current case should better be processed by a human (not implemented, not reliable, etc.). The syntax and the algorithm describing the behavior of the page analyzer are the following:

```
raiseQuestion(+Text, +Zone, +DataType)
```

When called, it 1) adds a question in the visual memory M_I, with shape and position defined by `+Zone` to locate the issue; and 2) continues the analysis just after the latest invocation of `catchQuestion`. The following example illustrates how to ask a question if, in some mail we try to recognize, we cannot find the recipient.

```
mail ::= AT(top_right) && recipient(-R)
   && % use recipient information
mail ::= raiseQuestion("What is the
      recipient?", all_page, recipient).
```

Answering the questions is done outside of the page analyzer, in the human-machine interface, after the automatic step of the current iteration is complete. Answers are stored in the visual memory and will be usable in the next iteration.

2) Using Answer (Error Correction): This operator is used to indicate the part of the document model which need to be ignored when an appropriate answer exists in the visual memory. This part of the page model is supposed to produce a result, unless an error occurs and then the automated analysis will be replaced by a human operator, or a other automated process. This operator is the one which enables *both interaction models*, as we will see in Sec. IV-C. Syntax and implementation algorithm are:

```
getAnswerOrTry(+DataType, -Result, +Rule)
```

where `+DataType` indicates the type of the content of acceptable answers, and `+Rule` is a rule which has a unique output parameter whose type is also `+DataType`. The type of the element contained in the answer is the same as the type of the result produced by the automatic rule.

When called, it looks at the current search position for an answer $A(Data)$ in M_I where the type of $Data$ is `+DataType`. If it exists, the value of `-Result` is $Data$. Otherwise, it invokes the rule `+Rule` and the value of the output parameter of `+Rule` is used as value for `-Result`.

In order to avoid asking questions indefinitely, we need to ensure that if some appropriate answer was provided, the original question will not be asked again. To do so, we control that usage of `raiseQuestion` is only made inside rules annotated with a `getAnswerOrTry` with the same `DataType`. The following example illustrates this usage.

```
recognizeNumber(+NumPos, -Value) ::=
   getAnswerOrTry(number, -Value,
      callClassifier(+NumPos, -Value))
callClassifier(+NumPos, -Value) ::=
   % inner machinery
   % fail if confidence is too low
callClassifier(+NumPos, -Value) ::=
   raiseQuestion("What is this number?",
      +NumPos, number).
```

3) Continuing the Analysis (Error Recovery): This operator is used to indicate the scope of some error in the page model. Outside this scope, the analysis is not impacted by such error and can be continued if it occurs. This operator is noted:

```
catchQuestion(+Rule)
```

where `+Rule` is a rule which may have any parameter. When called, it invokes the rule `+Rule` and catches any interaction request raised with `raiseQuestion` in that rule. If no question is raised, the output parameters of `+Rule` are well defined, otherwise they are left uninstantiated. If, for some table recognition task we can process rows independently, then we could write:

```
rowlist ::= catchQuestion(one_row) &&
   % detect other rows independently
```

C. Introducing Spontaneous Interaction

Spontaneous interaction relies on a simpler protocol than directed interaction. There is no need to detect errors and ask questions, and no need to recover from error. However, the ability to replace some parts of the automated analysis by a manual edition made by a human operator is necessary. Therefore, the `getAnswerOrTry` operator contains all the logic we need to enable spontaneous interaction: if we allow (thanks to some modification in the protocol defined in the strategy module) the human operator to provide information with the right type, located at the right place in the image referential, then the analyzer will automatically use it and will skip specific parts of the automated analysis process. As a result, we should not call "answers" every piece of information provided by a human operator, but we should simply say "information" about the image content.

This approach allows to annotate specific parts of the document model which may be manually corrected by human operators during interaction steps. This allows, for example, to correct erroneous element or add missing results in an intermediary structure, like in the following example:

```
line_of_numbers ::= AT(line_start) &&
   number_sequence.
number_sequence ::=
   getAnswerOrTry(number, -Num,
      find1Number(-Num)) &&
   AT(right_of(+Num)) &&
   % loop until no more numbers
```

If the system misses a number, a human operator can add it in the visual memory and during the next analysis iteration, it will be used just like if it was detected in the image.

D. Implementation Advantages

The implementation we propose not only permits to introduce the two interaction model we propose in the same page analyzer, it also has two main advantages. The first advantage is to propose a clear separation between the page description and the implementation of the automated management of the interaction with some human operator (or other software components). It permits to keep the page description simple, and also to improve the interaction management without changing the page model. The second advantage is to integrate external information (human-provided, system-provided, or both), exactly as if they were extracted from the image. As a result, we can assign a confidence score to external information, validate it against the document model, and use it to produce final results.

V. Applications and Results

A. Directed Interaction for the Transcription of Documents from the 18^{th} Century

The directed interaction model was used to efficiently transcript handwritten words from sales registers of the 18^{th} century [6]. In the tests we conduced, 70 documents (1206 words) were processed, and extracted words were grouped by visual similarity in clusters (making use of redundancies of several words among pages), which were manually annotated if necessary. The iterative analysis permitted to reintegrate and validate external information (extracted using collection context) within a unique page model. This global approach diminished the human labeling of words from 21% (manual labeling of all suspicious elements) to 15% (-28%) for an overall recognition rate of 80%.

B. Spontaneous Interaction to Correct Number Segmentation Issues in Documents from the 18^{th} Century

The spontaneous interaction model was used to correct segmentation issues during the localization of numbers in an excerpt (50 pages with 1637 numbers) of a document dating from the 18^{th} century [5]. Those numbers were separated by some special characters which where not always properly detected, and caused under-segmentation issues. In the tests we conduced, we allowed the human operator to review to localization results, and to indicate the position of separators which were not detected. Using this approach, we were able to recover 40.6% of the missing numbers and to eliminate 30.0% of the erroneous zones. Correcting errors during the analysis stage allowed us to reduce by 29.8% the number of manual actions required to reach this quality level, when compared to a manual post-processing.

VI. Conclusion

Interacting with a human operator during the analysis stage is necessary to process old and degraded documents. While a directed interaction model may be the most efficient approach, as asking missing information enables to optimize human interaction, the error detection this model relies on is not always possible. Spontaneous interaction is a valuable alternative in those cases, as it also permits to reduce human workload when compared to post-processing. This paper show how we introduced both directed and spontaneous interaction models in the same page analyzer to be able to benefit from both approaches. The implementation we used enables an asynchronous error detection, correction and recovery throughout the iterations of the analysis. It is based on standard programming constructs very close to exceptions and should be easily adapted to other existing systems. Doing so, making use of external information during the analysis is very easy, and we were able to benefit from human-provided information, but also information extracted from collection context: redundancies of words among pages helped us to transcript documents from the 18^{th} century.

Acknowledgment

This work has been done in cooperation with the *Archives départementales des Yvelines* in France, with the support of the *Conseil Général des Yvelines*.

References

[1] N. Roussel, O. Hitz, and R. Ingold, "Web-based cooperative document understanding," in *Proc. of ICDAR*, 2001, pp. 368–373.

[2] B. Klein, A. Dengel, and A. Fordan, "smartFIX: An adaptive system for document analysis and understanding," in *Reading and Learning*, Dengel, Junker, and Weisbecker, Eds. Springer, 2004, vol. 2956 of LNCS.

[3] G. Nagy and S. Veeramachaneni, "Adaptive and interactive approaches to document analysis," in *Machine Learning in Document Analysis and Recognition*, ser. Studies in Computational Intelligence, S. Marinai and H. Fujisawa, Eds. Springer, 2008, vol. 90, pp. 221–257.

[4] J. Chazalon, B. Coüasnon, and A. Lemaitre, "Iterative Analysis of Pages in Document Collections for Efficient User Interaction," in *Proc. of ICDAR*, 2011.

[5] J. Chazalon and B. Coüasnon, "Iterative analysis of document collections enables efficient human-initiated interaction." in *Document Recognition and Retrieval XIX, Proc. SPIE*, 2012, accepted paper available at http://goo.gl/JqqRv for reviewers.

[6] L. Guichard, J. Chazalon, and B. Coüasnon, "Exploiting Collection Level for Improving Assisted Handwritten Words Transcription of Historical Documents," in *Proc. of ICDAR*, 2011.

[7] B. Coüasnon, "Dealing with noise in DMOS, a generic method for structured document recognition: An example on a complete grammar," in *Graphics Recognition*, Lladós and Kwon, Eds. Springer, 2004, vol. 3088 of LNCS.