(A)KDD for Structuring Destrured Documents

Jacques Pere-Laperne

(A)KDD for Structuring Destructured Documents

Jacques Péré-Laperne
1A3I , ESTIA, University of Bordeaux, LaBRI-CNRS, F-64210 Bidart, France , j.perelaperne@gmail.com

Abstract— *The worldwide volume of digital data doubles every 9 months. Over 75% of these data are unstructured. This paper concerns structuring the graphic information contained in vector files, including PDF (Portable Document Format) files, which represent a very significant share of all these vector files. We say these data are destructured because they are today produced by software. When there are stored or exchanged, (for example in PDF Files) only graphic information is kept and the data structure use to create the document is lost.*

To structure these data, we use Knowledge Discovery in Databases (KDD). The following two issues arise:

• Can the KDD method be adapted to "Structuring Destructured Documents"?

• If so, what points need adapting or highlighting in the method to solve the issue of structuring essentially graphic documents?

The answer to the questions is "YES" with the human in the loop. This is why we talk about the Anthropocentric Knowledge Discovery in Database method, abbreviated to (A)KDD.

Keywords: Data Minning, CHI, PDF, KDD, Graphic reconstruction, Pattern recognition

1. Introduction

1.1 Context

It is worth remembering just 3 figures. The first is that the volume of digital data doubles every 9 months. The second is the fact that this volume of data will represent 7 to 8 Terabytes per inhabitant on the planet in 2020. Finally, over 75% of these data will be destructured.

1.2 Real industrial need

For over 30 years software has produced destructured documents. There were two aims of producing destructured documents. The first was to enable these data to cross time and different environments. PDF files are a concrete example of this aim. The second, more or less clearly stated, was to provide information that could not be exploited, modified or extracted.

The data present in destructured files are visible: They are human-readable but not machine readable. The data can be printed, sent and archived, but they cannot be used or modified and information cannot be extracted from them, and, more generally, they are inoperable. Any mechanism enabling these data to be structured and used represents an extraordinary benefit for the owners of these data. These destructured documents do not necessarily need to be fully structured. Often partial reconstruction is sufficient. It is the user who decides on the level of reconstruction and the items to be structured.

In 2003 Tombre and Lamiroy [1] made the following observation : "Grassroots users, as well as large companies, have a huge amount of information at their disposal, but this information is available in very "poor" formats: paper documents, or low-level, poorly structured digital formats such as Postscript, PDF or DXF. The challenge is therefore to convert this poorly structured information into enriched information which can be used within an information system". This is the aim of this article.

1.3 Case study, example of application

There are very many cases of application. For example, the possibility of modifying an existing diagram stored in a destructured form. This concerns electrical, hydraulic, pneumatic, electronic and PID (Process and Instrumentation Diagram) diagrams, as well as all types of plans, including buildings and heating/air-conditioning systems. The second case of application is to carry out Reverse Engineering on these types of existing documents in businesses or from other environments. The third is the possibility to almost automatically introduce all the graphic information from old software into new software, for different reasons, the main one being that the old software is no longer used or has become obsolete

Today, for all these operations, the only solution is to fully and manually re-input information in software. This solution is very time-consuming and costly. Furthermore, it is also the source of numerous errors. A copy-paste is not a solution, by this way, only the graphical information is copy and not the structure, without structure the information cannot be used by a CAD software.

1.4 Our vision

When we began work on structuring destructured documents, we applied the KDD method defined by Usama Fayyad and others. We broke down each task to be carried out, in all around twenty, into 5 main sub–tasks of the KDD method. These 5 main sub-tasks are: 1 – selecting data, 2 – preparing and cleaning data, 3 – Coding data, 4 – Data-mining, 5 – Presenting results to users. The authors underlined, the KDD process is both complex and comprehensive.

After structuring, the information is produced in XML format and DXF/DWG format, a graphic format usable by most Computer-Aided Drafting, Computer-Aided Design and Desktop Publishing tools.

The first question to be asked is: why are convolution neural networks not used since they are becoming increasingly efficient? These convolution neural networks (Lecun, Bengio 2006 [2]) cannot be used to structure destructured documents for two main reasons: Firstly, the small amount of information available in companies (we shall explain why the volume of input data is sometimes very small) and secondly, the difficulty in gathering learning data. When a company wishes to structure a piece of information, it will have between 1 and 100 files, each with 20 to 100 pages. This is clearly a long way from the number of examples required for convolution neural networks. Our approach differs from this work on neural networks: as we said above, it is very close to the KDD method and places the user at the heart of the extraction process.

The user is present throughout all the structuring phases. Why? Because it is much cheaper and more efficient for the user to intervene throughout the process rather than at the end to validate and/or correct any errors. The fact that the user has to take decisions (which takes just a few seconds according to the Human-Machine Interface used) considerably reduces, and in some cases even cancels, the rate of errors for each task. The aim of structuring is to obtain a 100% exact result. The results produced by structuring must be 100% exact. Users of information produced by structuring will never accept a rate of only 70% or even 80%.

In the structuring process, there are around twenty tasks to be carried out. If each task and their sequencing is full automated, with each task having an error rate of 2 to 3% , we will at the end point, after carrying out 20 tasks, observe that validation by the user will require decisions on information with between 40% and 60% errors. Such an error rate leads the user to reject the system. In such cases, during the validation phase, the user will have to modify, reclassify or cancel over 40% of all results. If a destructured datum contains 10 000 basic pieces of information, the user will have to modify 4 000 pieces of information. These corrective tasks will take a considerable amount of time, which is inacceptable in the world of industry. The aim of many works is to automate tasks as much as possible, therefore reducing the user's intervention. In our case, it is exactly the opposite. The user will have to intervene in all tasks and their sub-tasks very quickly, so the exact result is 100%. Our aim is to structure data with on the end-user.

This paper is organized as follows:

• In paragraph 2, we look at the state of the art.

• In paragraph 3, we look at the constraints imposed by small volumes of KDD input data.

• In Paragraph 4, we introduce the incremental nature of structuring using the KDD method.

• In paragraph 5, we introduce ontologies in the KDD method.

• In paragraph 6, we define the anthropocentric nature of the KDD method.

• We then conclude and highlight upcoming works.

2. State of the art

We carried out a comprehensive start of the art on structuring documents in the paper presented at Smart 2017. Research has been carried out: (1) to structure press articles, (2) to structure tables and forms, (3) to extract scientific papers and PDF files from graphs and curves. The only work close to ours is that of the Mnémosyne team at INRIA Bordeaux [3].

In conclusion, very little research work has aimed to structure graphic information present in unstructured documents.There are three main reasons why structuring unstructured documents has not been the subject of much research work.

The first is that, as highlighted by Usama Fayyad and his co-authors, the KDD process to be implemented is a comprehensive and complex process. A single task is not enough to solve it. All the tasks need to be undertaken and solved.

The second is the volume of data. A great deal of work on KDD deals with the issues of very large amounts of data and their scaling up for a solution.The amount of input data may be very small. We talk about MEDIUM-DATA (for large groups) and SMALL-DATA (for ETI's and SME's), compared to BIG-DATA present in many works on KDD. The third reason is the need to place the user at the heart of the issue and that goes against a lot of research whose aim is to automate tasks as much as possible in order to take humans out of the equation..

3. The KDD method on medium-data&small-data

Indeed, data structuring can be applied to a very small amount of data, for example, to structure an electric folder available as a single PDF file.

This folder contains, for example, 40 pages or sheets. Each page consists of 1 000 to 4 000 graphic objects. After structuring these graphic objects, there are around a hundred symbols per page. Out of these 100 symbols, 20 are standard electric symbols (contacts, breakers, motors, etc.), 40 are equipotentials and equipotential numbering symbols, 30 are crossed-reference information and the remaining 10 are various different types of symbols. Thus, in the 40 page folder, there are 1 600 equipotentials with numbering, almost the same volume of crossed references and around 800 electric symbols which belong to a set of 80 different symbols. This means that the same symbol is present on average 10 times in the file. The underlying idea is that a structuring symbol appear just once in all the input data and that, on average, it will be present more than 10 times in the data, and a thousand times according to the volume of input data (around fifty folders).

The data-mining we are developing to structure graphic data has to find repetitive patterns and cluster them, as well as non-repetitions of patterns, which can represent a symbol because if they are neither equipotentials nor already identified symbols, they are certainly "free" symbols or drawings which only exist once in the input data.

4. Our (A)KDD method's incremental aspect

4.1 The KDD method

The way structuring works is quite simple. We search for (Data-mining part) "Graphemes" (as understood by Jacques Bertin, 1967 and 2005 [4]), which are repeated in the same types of document. We saw above the repetitive nature of a pattern can vary from 1 (in this case it is unique) to N, (according to the volume of data to be processed).

It is worth highlighting the fact that with regard to KDD methods applied to data-mining in databases, the user knows EXACTLY what should be obtained after restructuring, whereas with "classic" KDD methods the user is not too sure of the final result. The (A)KDD method we have developed is simply there to accelerate structuring. Indeed, we can take the example of a folder with 40 sheets. If the user has to re-enter all the information, he/she will mentally structure it to introduce it into new software. This will take around 8 hours per sheet (according to the amount of information to be input), ie. a total of 320 hours. Our aim is for structuring the entire folder to take between 1 and 2 hours, ie. 300 to 150 times faster, even if the user only has one folder.

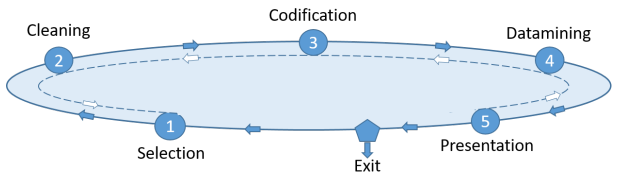
After searching and grouping into families of graphemes by the machine, the user validates, or not, all or part of these graphemes. After validation or non-validation, the graphemes are introduced into the ''Knowledge Base". In reality, "The" knowledge base consists of a series of knowledge bases, each of which corresponds to a type of document or sub-type of document, connected to one or several ontologies. Graphemes are entries of ontology objects. The entries and ontologies are in "The" knowledge base, and the graphemes are also stored in a graphic base in DXF/DWG format.

After validation, a new search and new "Clusterisation" are performed. This repetition is the Iterative aspect, with the user representing the Interaction aspect. Introduction into "The" knowledge base is the Integration aspect. Here we can see the KDD's 3 "I"s described by Usama Fayyad and his co-authors in 1996 [5]: Iterative, Interaction & Integration.

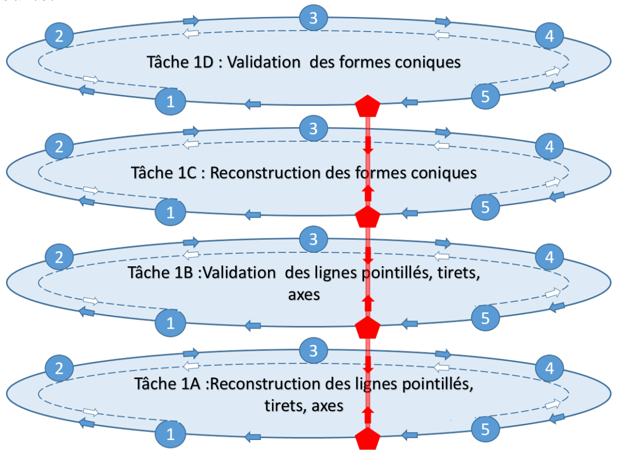
4.2 The incremental aspect

In our case, the principle of reconstruction is Incremental ((A)KDD's 4th "I"). When basic graphemes are recognised, we search for "Super Graphemes" which are the association of graphemes and basic graphic objects. In turn, they are proposed to the user and integrated in the knowledge bases and the graphic base. This process is recursive, until all the knowledge being searched for in the destructured documents is obtained. This is an example of the incremental aspect, which we shall describe in greater detail below.

It is worth remembering that each of the method's tasks is sub-divided into sub-tasks where we can find, 1 – Selecting data, 2 – Preparing and cleaning data, 3 – Coding data, 4 – Data-mining, 5 – Presenting results to users. We represent this set of tasks in the form of a double ring: the first rings shows the direction in which tasks are carried out and the second ring, the dotted line, shows the possibility of returning to any previous task. The ring's shape represents the method's Iterative aspect, with, of course, after the validation sub-task, the possibility of stopping the method's Iterative aspect.



The method's incremental aspect, which we propose, consists of stacking these rings on top of each other and enabling the user to be able to (1) Iterate the task's performance, or (2) Move on to the next task. This is the Incremental aspect, or (3) Return to the performance of one of the previous tasks because the results obtained do not match the expected results.


Tâche 1D : Validation des formes coniques
Tâche 1C : Reconstruction des formes coniques
Tâche 1B :Validation des lignes pointillés, tirets, axes
Tâche 1A :Reconstruction des lignes pointillés, tirets, axes
1
2
3
4
5

4.3 Stages of (A) KDD method

In the previous paper we identified the 7 following steps:

- Step 1: Structuring lines
- Step 2: Structuring letters

- Step 3: Structuring words
- Step 4: Structuring basic symbols
- Step 5: Structuring complex symbols
- Step 6: Structuring complex symbols
- Step 7: Structuring the folder

To explain the incremental aspect, we shall look at the first 4 steps in greater detail.

Step 1: Structure lines: Define the type of lines and identify them based on their composition (lengths of lines and spaces).

The aim of this step is to reduce the number of graphic objects by trying to group them together more holistically. For example, if we take the dotted line or with dashes, graphically it consists of a range of lines, each of which corresponds to the visible part of the dash or dotted line. Grouping them together reduces the number of graphic objects, for instance, from 50 graphic objects to one single graphic object.

Another example: All the conic shapes, like circles, ellipses, circle arcs or ellipsis arcs are made up of graphic objects which are segments. For a circle, according to its size, it can be broken down into 360 segments. In this case, each segment corresponds to a 1 degree corde d'un angle. NB. Circles or ellipses may be in dotted lines, which further increases the number of graphic objects making up the conic shape.

All these notions about the type of lines and type of conic shapes are defined in the ontology and the appearance of these notions in input data are instances of these notions.

Likewise, if you have a shape filled in, it is often shown by very narrow segments which give the impression of being filled in. There can be several thousand segments to be filled in in an area (like, for example, a logo).

Therefore step 1 can be broken down into 6 tasks:

- 1A: Reconstructing dotted lines, dashes, axes,...
- 1B: Validating dotted lines, dashes, axes,...
- 1C: Reconstructing conic shapes.
- 1D: Validating conic shapes.
- 1E: Reconstructing fillings.
- 1F: Validating fillings.

Why structure this information? There are three reasons:

1) After structuring it will be easier to identify "Symbols" which contain these graphic objects.
2) This is to reduce calculation times. Indeed, most of the algorithms used for data-mining are NP, N2 or N3 algorithms.. This means that if we wish to reduce the number of graphic objects represented by the letter N by 10 or 100, we can immediately see the impact which that can have on algorithm execution times. If we divide by 10 the number of graphic objects and the algorithm is N3, the time will be divided by 1000. We go from 10 minutes calculation time to 0.6 seconds. It should not be forgotten that the user is present throughout the entire process and it is ESSENTIAL that processing times are compatible with a user's waiting time which is close to a second.
3) By breaking down structuring into basic tasks focussed on a single graphic structure, we simplify and reduce the user's intervention time to validate or correct the results obtained by data-mining algorithms.

Step 2: Structuring letters: certain software packages draw letters directly in the form of filled in poly lines (cf. task 1E above) in order to, for example, trace on a tracer. Structuring letters involves defining the font, size, style, direction and letters which associate one or several graphics, for example "e", "é", "ë", a type of font (Arial, Courrier, ISO, etc).

We break down this step into two tasks: The first is the "Massive" recognition of graphics and letters. In the previous paper, we described information coding techniques enabling us to identify graphics, whatever the size and direction. Graphics are identified and "Clusterised" taking into account a distance function.

After "clusterisation", the grapheme is automatically recognised so as to identify it. This identification begins by searching in the knowledge base whether the same grapheme or one which is very close has already been instantiated (during reconstruction of a previous folder) in the knowledge base. If this is the case, the letter representing the grapheme is found. If this is not the case, the same search is carried out with regard to the fonts defined in the ontology.

The same letter can be present in the same folder with several different sizes (in general 3 sizes) and several directions (generally 3: horizontal, vertical, 45°). Given the large number of letters, the "Massive" recognition of letters is very weak (almost the same algorithms as used for symbols). On the other hand, it poses certain problems like, for example, capital "O", small "o" and according to the font, the number "0", or even a capital "I", small "l" and the number "1". This is why we add an additional task consisting of presenting the user all these problematic cases in order to attract their attention to particular cases. In any case, it may holistically validate the results provided by "Massive" recognition and make any necessary corrections in the second control task for particular cases.

Therefore, step 2 can be divided into 2 tasks:

- 2A: "Massive" recognition of letters
- 2B: Control particular cases

Step 3: Structuring words: This involves grouping together letters identified to create words, then text (series of words). For folders containing more text, like assembly instructions, we have to identify phrases and paragraphs.

This step can be divided into several tasks. The first is to identify words. It is worth remembering that writing can be in different directions: horizontal, vertical and inclined. A word is made up of 1 to N letters, N generally being between 3 and 5 since most words have less than 5 letters. In data-mining, algorithms used for associating letters in words take into account the space between letters and the direction in which a word is written. Once the words have been identified, they are presented to the user for validation. Next, the control task for particular cases of letters is relaunched, but this time with words. For example, identifying a capital "O" and small "o" or the number "0" is easier and faster within a word. The list of words is presented and the user simply validates, or not, the word and therefore the letter. During the control phase, words are displayed either alphabetically or with a word recognition trust index (cf. below).

When words are identified, dictionaries of words and word structures are defined in the field's ontology. For example, the word "MOTOR" can be pre-recorded in the ontology as an illustration of an existing word for the field. If the word "MOTOR" is not an entry and it is present in the folder and validated by the user, it becomes an entry in the dictionary and its frequency will help when checking because the word "MOTOR" will have a more or less high trust index according to the number of times it appears.

This is also the case for word structures. For instance, in electrical diagrams, the name "power contactors" begins with 2 letters 2 "KM" and are followed by 1 to 3 figures, for example, "KM21". We can see that by using these structures we can raise any doubt, if necessary, about the figure "1" and the letters "I" o' "l". These structures are created and recorded in the ontology and entries are recorded in the knowledge base. Thus, we can use the same trust index for structures.

Next, words with the same font (size and direction) are grouped together to form phrases. The phrases detected are controlled by the user and then, in turn, grouped together into paragraphs to be controlled. Therefore step 3 can be divided into 6 tasks:

- 3A: Word identification
- 3B: Word controls
- 3C: Phrase identification
- 3D: Phrase controls
- 3E: Paragraph identification
- 3F: Paragraph controls

Step 4: Structuring basic symbols: Defining these symbols and associating basic graphic items and text. In the ontology, families and categories of symbols are defined. For each category, we also specify the attributes which make up these categories. In general, these attributes correspond to areas of text close to the symbol and the connection points of these symbols with equipotentials (in the event of electrical diagrams). Data-mining searches for graphemes are repeated inside input data. These graphemes take into account structuring already carried out, like, for example, dotted lines, hatches or words. Then, the distance is calculated like with letters in order to "Clusterise" them.

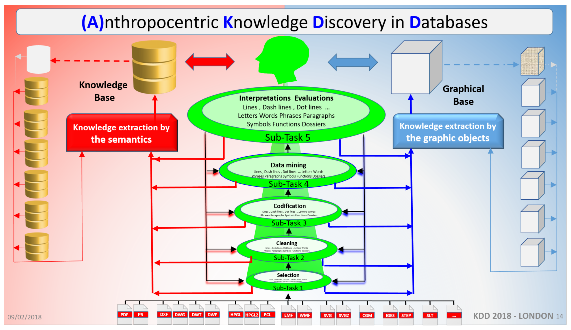
This step is highly iterative. First, the graphemes which are most present in the input data are proposed. The user validates, or not, the detection of these graphemes as symbols. If the user validates them, they become ontology entries in the knowledge base. As with letters, but based on a slightly different principle, after "Clusterisation", data-mining attempts to find in the knowledge base whether an identical or similar grapheme has already been identified. If so, this grapheme becomes a symbol entry in the knowledge base.

Therefore, step 4 is divided into 2 tasks:

- 4A: "Massive" recognition of letters
- 4B: Controls for particular cases

With the description of all these tasks we can observe that reconstructing destructured documents is an incremental task. We start by identifying types of lines, then conic shapes, then fillings, then letters, words, paragraphs and symbols.

By breaking down structuring into basic, but incremental tasks, the user's work is made lighter since it is no longer necessary to focus on a specific task and this task only concerns a single feature of objects. However, we can also see that the user's intervention and ontology are essential to our approach. This is the subject of the next two chapters.


(A)nthropocentric Knowledge Discovery in Databases
Knowledge Base
Graphical Base
Knowledge extraction by the semantics
Knowledge extraction by the graphic objects
Interpretations Evaluations
Lines , Dash lines , Dot lines ...
Letters Words Phrases Paragraphs
Symbols Functions Dossiers
Sub-Task 5
Data mining
Sub-Task 4
Codification
Sub-Task 3
Cleaning
Sub-Task 2
Selection
Sub-Task 1
PDF
PS
DXF
DWG
DWT
DWF
HPGL
HPGL2
PCL
EMF
WMF
SVG
SVGZ
CGM
IGES
STEP
SLT
....
09/02/2018
KDD 2018 - LONDON 14

5. Ontology & the (A)KDD method

Since the beginning ontologies have been considered as enabling the KDD method to be more precise and efficient. Usama Fayyad and the co-authors state in the paragraph "Concluding Remarks: The Potential Role of AI in KDD" that "Knowledge representation includes ontologies, new concepts for representing, storing, and accessing knowledge. Also included are schemes for representing knowledge and allowing the use of prior human knowledge about the underlying process by the KDD system".

The "Semantic Data Mining: A Survey of Ontology-based Approaches" paper provides a comprehensive overview of the role of ontologies in data-mining.

Ontology is the explicit specification of conceptualisation and a formal way to define the semantics of knowledge and data. The formal structure of the ontology makes it a natural way to code the field's knowledge to use data exploration.

Structuring documents is highly prioritised. It is described by an ontology. The major lines of the ontology are given to the user. However, in the current state of affairs, it is impossible to give the final user an ontology which corresponds exactly to the problem. Therefore, the user must be able to select the parts of the ontology to be activated, and, also, remove or complete certain parts of selected ontologies.

We have seen that an ontology to reconstruct electrical diagrams is different from an ontology for structuring the plans of a building. Activating both in the same knowledge base would certainly lead to more analysis and reconstruction errors.

The ontology is developed throughout the (A)KDD process. The ontology must adapt to what the user wants to structure, making structuring more precise and efficient.

Moreover, by making knowledge bases and the graphic base as close as possible to the sub-nature of documents, we obtain better structuring rates. The knowledge base contains the ontologies and instances of graphemes for ontology objects. The graphic base contains the graphemes of entries. Reconstruction information provided to the user is more reliable. Although the basic symbols to be structured are almost always the same for AutoCad, Elec'view and See, (which are 3 design software packages for electrical diagrams), this is also the case for cartridges, entry and exit badges, switches, specific maps, etc. We can consider that all graphic data differ from one software package to the next. This is why it is important to be able to manage knowledge bases in terms of the sub-features of documents.

6. The user's essential role in the structuring process

The authors of the KDD method insist on the fact that knowledge should be extracted with the user's help.In 2003, during a KDD conference, it was observed that many works on KDD did not include the user. The user only intervenes in the final step (Fayyad, 2003 [6]).

In "Recherche anthropocentrée de règles d'association pour l'aide à la décision" (Chevrin, 2007 [7]) the authors place the user at the heart of the method. They insist on the user's role and it is one of the first times the term "Anthropocentric" has been given such importance in the KDD method. This is our observation today in 2018. In most works dealing with structuring destructured documents, the user is almost absent. These works are not "Anthropocentric" with regard to the user, but should be. The aim of many works about knowledge extraction is to automate this extraction as far as possible. This is certainly possible where there is a large amount of data to be processed, but in our case it is impossible.

Users wishing to structure data do not have enough data to fully and reliably automate the structuring process. However, the KDD method can be applied, even when the volume of data is small, provided that, as we have seen, an incremental and guided process is implemented by an ontology.

It is the user who pilots structuring with the help of the computer for reasons of cost, efficiency and to ensure the system does not reject it. But the question is how can the user pilot structuring? This is the subject of the following paragraph.

In the reconstruction process, there are some twenty tasks made up of 5 sub-tasks. This means that the user will interact with the system in 100 different sub-tasks. In this case, the human-machine interface is one of the major points in terms of acceptance or rejection of the system by the user. "One of the key words in HMI is user-centred. User-centred design (Norman & Draper 1986 [8]) is a paradigm which defines 12 key principles which place the user at the heart of the development process". This is the paradigm described by Dupuy-Chessa (2011, [9]), which we use to define our user and system tasks, and which we described in the previous paper.

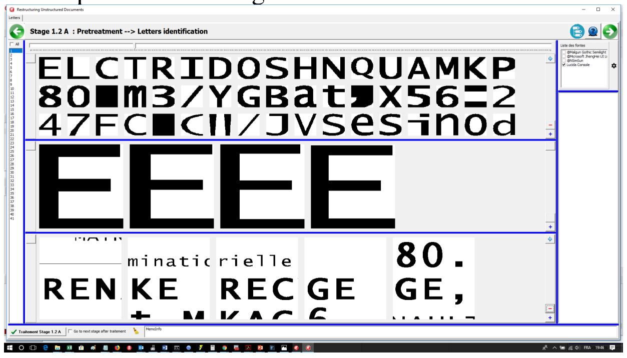
Structuring, like when implementing the KDD method, is long and complex. Data structuring must be 100% exact. It must even, wherever possible, correct any errors which are present in the folder. Indeed, we can observe that many electrical diagrams do not match terminal board diagrams , for example. This is a general problem when extracting knowledge from databases. Input data are sometimes inexact and the results obtained must not be biased by the inaccuracies.When reconstructing diagrams, this is also the case.

We have only described in detail the first 4 steps of structuring. In the final steps where complex symbols, functions, the folder and coherence controls to be carried out should enable this type of coherence to be detected, and in the event of divergence, it is the user who will determine which information is accurate or not.

One of the issues in Human-Machine Interfaces is the volume of data to be visualised. The main questions are as follows:

- How to visualise them?
- How to browse these data?
- How to validate information or not?

The problem's statement is simple, but the solution is more complex. In the previous paper we stated that the solution is a combination of the two methods: "Overview+Detail" and "Zooming+Panning". We shall take the example of visualising letters.



This is the interface of task 2A for the user to validate letter recognition. We shall only present the visualisation and browser interface inside the information displayed.

The screen is divided into 4 parts: 3 horizontal areas and one vertical area. The 3 horizontal areas relate to input data. The vertical area contains ontology information with regard to fonts and letters. The first horizontal area is to visualise Clusters identified by data-mining. The cursor present above this area enables the grapheme distance calculation inside a single cluster to be adjusted: the further right the cursor and the smaller the distance, the greater the number of clusters (NB. the same letter may be present several times). The further right the cursor and the greater the distance between graphemes, the smaller the number of clusters. The second horizontal area from the top displays all the different graphemes which have been grouped together in the cluster and the third area at the bottom displays all the occurrences of graphemes inside the input data. The "+" and "-" buttons to the left of each of the horizontal areas enable more or less information to be seen by displaying it on one or several lines. The "ScrollBar" enables informaiton to be scrolled through. "Splitters" enable an area to be enlarged compared to another area.

The 3 horizontal areas have the "Overview+Detail" functions starting from clusters as far as grapheme entries. The "Zooming+Panning" functions are provided by the "+" and "-" buttons and the Scroll Bars.

All the interfaces presenting data must be worked on with equal care. And every time the user intervenes, it must be ergonomic, fast and very precise.

7. Conclusion

In this paper we have described the incremental aspect of the (A)KDD method in detail. We have seen the importance of ontologies for obtaining a more efficient and precise result. And, above all, we have repeated the FUNDAMENTAL role of the user for small volumes of input data.

Developing the KDD method is both complex and long. It requires the implementation of many techniques, data mining algorithms, coding techniques, stringology algorithms applied to coding, clusterisation algorithms based on grapheme distance calculations. And, above all, a highly user-friendly, high-performance and simple interface, and very fast algorithms because their response time must be less than a second so that any time is acceptable to the user.

This is the direction of our work.

Next year, we shall, if possible, present at KDD 2019 in "Call for Applied Data Science Papers" the full implementation of "STRUCTURING DESTRUCTURED DOCUMENTS". This presentation will conclude our PhD work on this subject and will demonstrate that the (A)KKD user-based KDD method enables knowledge to be extracted for data sets, even when for small input volumes.

References

[1] K. Tombre and B. Lamiroy, "Graphics Recognition – from Re-engineering to Retrieval," in *7th International Conference on Document Analysis and Recognition*. Edinburgh, Scotland, UK: IEEE Computer Society Press, Aug. 2003, pp. 148–155, invited talk. Colloque avec actes et comité de lecture. internationale.

[2] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards ai," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA: MIT Press.

[3] I. Chraibi Kaadoud, N. P. Rougier, and F. Alexandre, "Implicit knowledge extraction and structuration from electrical diagrams," in *The 30th International Conference on Industrial, Engineering, Other Applications of Applied Intelligent Systems*, Arras, France, June 2017. [Online]. Available: https://hal.inria.fr/hal-01525028

[4] J. Bertin and M. Barbut, *Sémiologie graphique: les diagrammes, les réseaux, les cartes*. Gauthier Villars, 1973. [Online]. Available: https://books.google.fr/books?id=F4weAAAAMAAJ

[5] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Advances in knowledge discovery and data mining," U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, ch. From Data Mining to Knowledge Discovery: An Overview, pp. 1–34. [Online]. Available: http://dl.acm.org/citation.cfm?id=257938.257942

[6] U. M. Fayyad, G. Piatetsky-Shapiro, and R. Uthurusamy, "Summary from the kdd-03 panel: data mining: the next 10 years." *SIGKDD Explorations*, vol. 5, no. 2, pp. 191–196, 2003.

[7] V. CHEVRIN, O. COUTURIER, and E. MEPHU, "Recherche anthropocentrée de règles d'association pour l'aide à la décision," vol. 8, p. 30, 2007.

[8] D. A. Norman and S. W. Draper, *User Centered System Design; New Perspectives on Human-Computer Interaction*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1986.

[9] S. Dupuy-Chessa, "Modélisation en interaction homme-machine et en système d'information : à la croisée des chemins," Thèses et habilitations, 2011, habilitation à diriger des recherches, Université de Grenoble.