This document is a step by step guide for the indexation of all fictional characters and real persons found in La Comédie humaine. It explains how the ongoing indexing work should be pursued and/or potentially rearranged according to more specific needs. Knowledge in programming is not a prerequisite for these tasks, since programs and scripts have specifically been developed to help you in that purpose.

A French version of this report is available online on GitHub:

https://github.com/Armellei/IndexBalzac

Report Concerning the Indexation
of all Fictional Characters and Real Persons
Found in *La Comédie humaine*

Foreword

1.1 Presentation of the indexing guiding principles
1.2. Use of scripts for a semi-automatic working process
1.3 Required programs to start the indexation

Indexation

1.1 General workflow
    1.1.1 Aims of the workflow
    1.1.2 Default options regarding the indexation
    1.1.3 Customized indexation procedures and potential openings
1.2 Indexing rules
    1.2.1 Rules as defined in La Pléiade
    1.2.2 Naming using a unique user ID
    1.2.3 Leads and aliases
    1.2.4 Sections and subdivisions of the index
    1.2.5 Characters' descriptions and discriminating criteria
    1.2.6 Nomenclature and syntax of the databases
    1.2.7 Case sensitivity
        1.2.7.1 Repeated syntactic unit terms susceptible to start with a capital letter
        1.2.7.2 Adding exceptions to case-sensitivity
        1.2.7.3 Banned exceptions
    1.2.8 Orthographic and typographic rules
    1.2.9 Names to index and not to index
1.3 Use of the scripts
    1.3.1 Dutocq
    1.3.2 Rabourdin
    1.3.3 Colleville
    1.3.4 Bixiou
    1.3.6 txt2json
1.3 Tracking of the duplicates
    1.3.1 Tracking on the HTML indexing worksheet
    1.3.2 Use of the tracking scripts
1.4 Removal of the duplicates
    1.4.1 Main principles
    1.4.2 How-to guide
1.5 Automatic XML-TEI markup
    1.5.1 Default markup
    1.5.2 Customized markup
        1.5.2.1 Refining possibilities
        1.5.1.2 Step-by-step modification of the python files
    1.5.3 HTML display of the names tagged in the XML-TEI files

How-to guide to generate the index

1.1 Preparing the files before starting the indexation
1.2 Writing the databases
1.3 Using the scripts to generate the HTML worksheet index
1.4 Final checking

Illustrations

1.1 Editing the illustrations, icons and coats-of-arms
1.2 Photo-manipulation advice

Recommendations

1.1 General advice

Common mistakes while indexing and debug of the scripts

1.1 Reading the debug indications given by the scripts
    1. 1.1 Debug operation and common types of mistakes
    2.1. 2 Examples
1.2 Mistakes regarding the characters
    1.2.1 Removing the duplicates
    1.2.2 Removing nonidentical data about a single user ID
    1.2.3 Errors while filling in the fields
1.3 Mistakes regarding the syntax of databases
    1.3.1 Txt and JSON syntax errors
    1.3.2 UTF-8 encoding errors and special characters
1.4 Multiple mistakes

File upload

1.1 Database updates from /IndexBalzac on GitHub
1.2 Preparing the files for a web integration

Source code

Foreword

1.1 Presentation of the indexing guiding principles
1.2. Use of scripts for a semi-automatic working process
1.3 Required programs to start the indexation

1.1 Presentation of the indexing guiding principles

The indexing work consists in creating databases listing all fictional characters and real persons found in every single book from *La Comédie humaine* in its Furne corrigé version (1844-1848). Each book is matching with to tailor-made databases, specifically created for each and every text written by Balzac, listing the two types of people that can be found within: the real persons belong to the first of these databases, and all fictional characters to the other one. These bases are simple text files easy to edit and use that will be converted into JSON files, a more practical and web-friendly choice, at the end of the indexing process.

Each book from *La Comédie humaine* has been generated in a XML-TEI version that can be found on GitHub, and in an HTML version, also available online on ebalzac.com. The databases are linked to these two files with which they can exchange data.

The indexing work of all the characters found in the databases must be done “manually” by the operator who has to read the books one by one, with a little help from five scripts allowing to index way much faster. All checking steps (errors, special cases) are done semi-automatically. For each book, two databases are created. Then, we can check automatically whether there are any errors left in the files (e.g. duplicates) or not, and when the scripts have green-lighted the work, an HTML index allowing another important checking step can be generated. This index is a worksheet whose function is to help the operator indexing the characters: once a book has been processed, the index is automatically updated. Afterwards, the automatic markup of the XML-TEI file of the book we are working on can be launched to locate the characters that have been registered in the two databases. This marking will allow, amongst many other possibilities, to provide dynamic search and data exploitation tools to be used on the index and HTML versions of the books. Finally the text databases are converted into JSON files before they are uploaded on the website, and they will automatically update the existing index.

Since indexing proper nouns with particular programs developed in that purpose (TXM, etc.) have shown disappointing final results and a rather high rate of errors, the following indexing process should be automatized with custom-made scripts and these have been designed to fit as well as possible with Balzac's writing style and his never-ending, subtle yet duplicitous plays on words.

On average, one to two books can be indexed every day, depending on the operator's reading speed.

Use of scripts for a semi-automatic working process

Five scripts, whose description abounds in details in section 1.3, are helping you to automatize a huge part of the work: these are small programs coded in python and awk. Nonetheless, indexing without using the scripts or without even understanding how they work is absolutely possible: just click on their icons and they will do the job on their own.

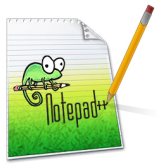
The scripts have to be launched one by one at each step of the indexing process: they have

intentionally not been combined together in an executable to allow the operator to do any modification if needed and to look at the file to track potential errors with more precision. However, generating a unique executable with the four python files and the awk program is still possible, but up to you.

1.3 Required programs to start the indexation

Two programs are required to start the indexation. Both are easy to install and use. Furthermore, two languages must be installed as well on your computer in order to start and read the scripts.

- a **text editor** such as Notepad++ ;
The editor must allow a display of the characters in UTF-8.



- a **terminal** ;
A terminal is a command line interpreter.

Mac: Terminal
Windows: Do not use an ugly and outdated piece of garbage such as cmd - rather download Git Bash > https://gitforwindows.org/
Linux: Terminal, gnome-terminal, konsole, xterm, etc.

- **Python** ;
Python is a language allowing you to read and write scripts.

Get Python > https://www.python.org/download/

- **Awk.**
Awk is a language for text processing that will be used here to automatize the indexation work.
NB: If you use GitBash, Awk already comes with it, you do not need to install it.

Mac: Install Homebrew, open a terminal, enter the command:

```
brew install gawk
```

Windows: install GitBash
or install gawk for Windows from:
http://gnuwin32.sourceforge.net/packages/gawk.htm

Linux: install gawk with your software package tool (APT, etc.)

Indexation

1.1 General workflow
    1.1.1 Aims of the workflow
    1.1.2 Default options regarding the indexation
    1.1.3 Customized indexation procedures and potential openings
1.2 Indexing rules
    1.2.1 Rules as defined in La Pléiade
    1.2.2 Naming using a unique user ID
    1.2.3 Leads and aliases
    1.2.4 Sections and subdivisions of the index
    1.2.5 Characters' descriptions and discriminating criteria
    1.2.6 Nomenclature and syntax of the databases
    1.2.7 Case sensitivity
        1.2.7.1 Repeated syntactic unit terms susceptible to start with a capital letter
        1.2.7.2 Adding exceptions to case-sensitivity
        1.2.7.3 Banned exceptions
    1.2.8 Orthographic and typographic rules
    1.2.9 Names to index and not to index
1.3 Use of the scripts
    1.3.1 Dutocq
    1.3.2 Rabourdin
    1.3.3 Colleville
    1.3.4 Bixiou
    1.3.6 txt2json
1.3 Tracking of the duplicates
    1.3.1 Tracking on the HTML indexing worksheet
    1.3.2 Use of the tracking scripts
1.4 Removal of the duplicates
    1.4.1 Main principles
    1.4.2 How-to guide
1.5 Automatic XML-TEI markup
    1.5.1 Default markup
    1.5.2 Customized markup
        1.5.2.1 Refining possibilities
        1.5.1.2 Step-by-step modification of the python files
    1.5.3 HTML display of the names tagged in the XML-TEI files

1.1 General workflow

1.1.1 Aims of the workflow

The indexing workflow consists in the following:

- Manual writing of the two databases (simple text files)
- Potential manual un-marking of the XML-TEI file thanks to a debugging file
- Potential un-marking of the databases with a python script: Dutocq
- Automatic generation of a HTML index worksheet with a python script: Dutocq

- Checking of the duplicates with a python script: Bixiou
- Automatic markup of the XML-TEI version of the books with a python script: Rabourdin
- Databases conversion from txt to JSON files with an awk script: txt2json
- Upload of the JSON databases and freshly tagged XML-TEI files on eBalzac.com

1.1.2 Default options regarding the indexation

The indexation process consists in listing all the characters as well as a bunch of criteria describing their life, gender, place of origin, relations, etc. This information is compiled within the databases, which are linked to the XML-TEI files thanks to a deliberate and very simple markup: <persName> tags containing a single user ID allowing us to identify who is who, and it is sufficient. These identification numbers (ID) cross several files such as the HTML index, the XML-TEI versions of the books, the txt and the JSON databases to make the work easier.

This simple markup prevents us from:
- having heavy XML-TEI files, which are already quite important, since the index communicates directly with the databases when you do a research through this interface. The JSON databases can although contain a lot of useful information that could not be easily tagged in the XML files: the characters' relations, the places they have been to, etc. JSON files are dedicated to this.
- having markup errors if we had decided to tag any other information such as places, dates, etc., which can be annoying if they have to be detected in a heavy XML-file. Their correction is way much easier in a simple text database.

Finally, the XML-TEI markup is done automatically with a script that reads the databases, checks whether there are errors in the files or not, and add the desired tags on its own if it has the go-ahead.

1.1.3 Customized indexation procedures and potential openings

You can automatically markup:
- the existing <persName> tags with a higher grade of precision – the one you want, there is no limit
- any other syntagma or group of alphanumeric characters with the TEI tags of your choice.

In order to modify the <persName> tag, please modify the Rabourdin script as described down below:

- Open Rabourdin.py in a text editor such as Notepad++
- Modify line 7 with a new or more tag names
- Modify the lines 42, 43, 68, 69 (useful information is provided in the file)
- Save and close Rabourdin

If any other info is tagged (places for instance), new databases can be created to list it all before allowing an automatic markup. Their syntax and principles can be based on the characters' databases.

1.2 Indexing rules

1.2.1 Rules as defined in La Pléiade

Most of the time, all indexing rules follow those enacted in La Pléiade (*La Comédie humaine*, volume XII) about the indexation of all fictional characters and real persons. In case of doubt, get back to it. It will be helpful to check the spelling of a character if Balzac himself plays with several

written forms of the same name, if someone's personal data is unclear or when the author has made chronological mistakes in the books.

1.2.2 Naming using a single user ID

Each character has a unique ID allowing us to identify who is who. This ID is followed by the letter P for fictive characters, and by the letter R for real persons.

Example:

Modeste Mignon is identified as: 181P
Aspasie is identified as: 200R

1.2.3 Leads and aliases

Lead

- Each character is known by its main name – the lead – and by other forms of the same name, named aliases.

- All characters having several identities are indexed under the most common identity (lead), all other identities being listed as aliases.

Alias

Other denominations for a specific character which are not the lead are known as aliases and must be listed the exact way Balzac writes them in his texts. Aliases might be:

- First names ;
- Maiden names ;
- Family names for people married with several characters ;
- Nicknames ;
- Pet names – and names of the pets ;
- Pen names, stage names and pseudonyms ;
- Noms de guerre ;
- People designated by their honorific ;
- Any other written form of a name referring to a character.

→ If a character is never called by his first name, this first name may never appear in the index.

1.2.4 Sections and subdivisions of the index

The index is divided into two main sections that can be displayed together or separately on the website:

fictional characters / real people

The index is then subdivided by:

social class / occupation / birth place / gender / contact network (family / friends / lovers / colleagues / etc.) / reappearing nature / any other discriminating information that has been listed in the databases

1.2.5 Characters' descriptions and discriminating criteria

Each character can be described as much as necessary: the length of the description is not limited to a certain number or characters. Real persons are briefly described by default, but fictive characters can get more data, especially if this data is not listed in La Pléiade and adds useful content.

1.2.6 Nomenclature and syntax of the databases

Each book is linked to two text databases and to another file named “debugging file”. Their structuration, syntax and nomenclature are very simple yet unchangeable for the scripts to interpret them properly. The following rules must therefore be observed:

Nomenclature

- The fictive characters database is a text file whose name must be the one of the book followed by this extension: _P.txt ;

- The real persons database is a text file whose name must be the one of the book followed by this extension: _R.txt ;

- The debugging file is used freely by the operator: you can write whatever you want on this file. However, its name must be the one of the book followed by the extension: _DEBUG.txt .

Syntax

- Each fictive or real character is listed on a paragraph within the database it belongs to. Each group of information referring to the same character should be on the same line.

- Each fictive or real character is listed at least on a first line, which is a compulsory line. This line is a first group of information but several optional lines can be added if more information is added.

- Each line of text starts and ends with an alphanumeric character. No space must be left at the beginning or end of a line, or at the beginning or end of a database.

- The first line of required information is subdivided into six fields and consists in a first group of data. Four are compulsory and two are optional (the third and fourth ones). Five angle brackets separate the fields. The fields refer to the following in this specific order:

single ID >Lead>Free description of the character>Place>Genre(0/1/2/3)>Letter for indexing

- The second group of information is optional and must be used to describe the contact network of the character. Starting with an at symbol, it is followed by a colon (@:) and by each single user ID our character is in contact with. These contacts must be separated by commas.

- The third group of information is also optional and lists all existing aliases for a single character. Each alias is listed in a different line.

- The five brackets of the first group of information are compulsory.

- The main structure for each paragraph must be the following:

single ID >Lead>Free description of the character>Place>Genre(0/1/2/3)>Letter for indexing

@relations

Alias

→ Several relations and aliases can be listed. In this case, the structure must be the following:

single ID >Lead>Free description of the character>Place>Genre(0/1/2/3)>Letter for indexing
@relations: ID1
@relations: ID2, ID3
@relations: ID4
Alias 1
Alias 2
Alias 3

1.2.7 Case sensitivity

Case sensitivity is compulsory to get a proper markup in the XML-TEI files: it tells the difference between the proper nouns and the common ones, between the different written forms of someone's name, and helps us not to get confused with special cases such as *Le Tasse* and *une tasse*, *Racine* and *une racine*, etc.

Reapeted syntactic unit terms susceptible to start with a capital letter

However, writing names twice in the databases (as aliases) if they appear to be written both with and without a capital letter might be quite annoying and time consuming. You can for instance come across this case:

madame de Latournelle
Madame de Latournelle

Adding exceptions to case-sensitivity

The script Rabourdin.py is able to remember some exceptions for you not to list all the words being written both with and without a capital letter in Balzac's texts. Several terms, such as « madame », « monsieur », « mademoiselle », « lord », « lady », etc. are not case sensitive. They can for instance be found at the beginning of a sentence with a capital letter, and anywhere else with lowercases.

Other terms can be remembered by Rabourdin if you want to remove the case sensitivity for a specific word:

- Open Rabourdin in a text editor such as Notepad++
- Write the exceptions on line 13 and save. Close Rabourdin.

```
11  # Add a word here if you want rabourdin to check it with and without a capital letter!
12  # NOTE: ALWAYS add words in lowercase in this list! :]
13  CASE_INSENSIVE_WORDS = ["monsieur", "madame", "mademoiselle"]
```

Banned exceptions

Beware! Exceptions such as « le », « la », « les », « de », « du », are strictly forbidden! Only terms such as « madame », « monsieur », « mademoiselle » or honorifics might be listed in Rabourdin's source code.

1.2.8 Orthographic and typographic rules

Choice of a letter to index a character

Each characters' name is alphabetically listed and sorted in the index thanks to its first letter. However, finding what this first letter should be can be quite challenging: the letter is not chosen by the indexing scripts but by the operator because of complex cases you can encounter.

If the character has a family name with an aristocratic particle, the letter for indexing must follow these examples:

Name	Registration in the index	Letter chosen to index the name
Machin Chouette	CHOUETTE (Machin)	C
Machin de Chouette	CHOUETTE (Machin de)	C
Machin des Chouettes	CHOUETTES (Machin des)	C
Machin Du Truc	DU TRUC (Machin)	D
Machin de La Chouette	LA CHOUETTE (Machin de)	L
Machin Le Truc	LE TRUC (Machin)	L
Machin La Chouette	LA CHOUETTE (Machin)	L
Machin de L'Houette	L'HOUETTE (Machin de)	L
Machin d'Houette	HOUETTE (Machin d')	H
Machin Chouette de Saint-Truc	CHOUETTE DE SAINT-TRUC (Machin)	C

Example:

Jeanne de La Peyrade des Canquoëlles will be listed in the index as:
LA PEYRADE DES CANQUOËLLES (Jeanne de).

Use of small caps

Family names are written in small caps, followed by first names in lowercase and/or honorifics in lowercase. Some particles such as *de, d'* and *des* are not written in small caps but in lowercase.

Foreign particles follow these rules:

Particle	Name example	Composition
di	PIOMBO (Bartholoméo di)	lowercase
van	GOBSECK (Esther van)	lowercase
von	WITTGENSTEIN (Peter-Ludwig von)	lowercase

Family names and occupation

The letter used to sort and list the names should be the one of the family name. However, some characters are not described with a family name but with a job. It is often the case for domestic servants, poor people or little people.

Special cases: diplomats (ambassadors, consuls, etc.) are indexed by occupation and not by family name; valets and servants described with their job only as well, but a characteristic should be added into parentheses if possible, since there are lots of servants in Balzac's books. This characteristic

could be a physical description for instance. It should be used only if the job is a very common one (e.g. chambermaid, janitor, cook, etc.)

Examples:

Vieil ambassadeur de Sardaigne should be indexed as:
Ambassadeur de Sardaigne

Valet de chambre de XXX should be indexed as:
Valet de chambre

Vieille femme de chambre de XXX should be indexed as:
Femme de chambre (vieille)

Honorifics

Honorifics are included in the person's description within the databases, and they can also appear in the lead name. Honorifics could be the following:
comte, comtesse, vicomte, vicomtesse, baron, baronne, marquis, marquise duc, duchesse, prince, princesse.

Example:
baron MACHIN DE LA CHOUETTE
LA CHOUETTE (baron Machin de)

Madame, mademoiselle, monsieur
If someone is known by one of these terms followed by a family name only, they must be indexed as the following:

MACHIN (madame)
MACHIN (mademoiselle)
MACHIN (monsieur)

People well known as “le père Machin” or “la mère Machin” will be indexed the same way:

MACHIN (le père)
MACHIN (la mère)

Identical names within the same line of descent
People having the same family name than ancestors or descendants are discriminated as much as possible by their rank in the family (mother, daughter, father, son, etc.):

MACHIN (mère)
MACHIN (fille)

Inarguable pests:
- People getting married again and again, using their maiden name although they are married, giving different family names to their children (each child having a different family name than his siblings), changing their identity, having several civil status (a French one, a Spanish one...), using a nom de guerre or pseudonym, nickname, pet name, stage name, etc. will be indexed just like La Pléiade does.

1.2.9 Names to index and not to index

Must be indexed:

- All names except those which must not be index (see below) ;
- Religious people (apostles, prophets...) except saints and God: Jesus-Christ, the Virgin Mary, Adam and Eve, Noah, Muhammad, etc. are indexed;
- Unknown tribes helping us to understand the cultural or family background of a character (e.g. les Abencerrages).

Must not be indexed:

- Names found in dedications ;
- Characters being part of the title of a book quoted in the text (« vous lirez dans LOUIS LAMBERT que… ») ;
- Saints (Saint Michel, Sainte Geneviève, croix de Saint-Louis, pont Sainte-Anne, ville de Saint-Denis...) ;
- God ;
- Spouses of kings, daughters and sons of queens and kings (Madame la Dauphine…) whose name is not given ;
- Balzac's own name ;
- Tribes and ethnic groups;
- Rose and Colas, in *Une Double Famille*.

1.3 Use of the scripts

1.3.1 Dutocq

Dutocq will generate an index in HTML. This index is a worksheet that helps the operator, and it is generated after Dutocq reads all the databases previously created. Dutocq also knows how to debug: it can track duplicates, syntax errors within the text databases and wrong unique ID numbers given to the characters listed.

1.31.2 Rabourdin

Rabourdin's role consists in reading the text databases to extract the names of the characters (leads and aliases), before it marks up the XML-TEI file automatically with the information it has gathered about the characters of the book you are working on. It simply adds a <persName> tag to each fictive character or real person found on each line of the XML file, and the tag is followed by a @ref providing a single user ID number. Rabourdin is a case sensitive script that can deal with exceptions you can configure as you please.

1.3.3 Colleville

Colleville helps both to check whether the TEI markup has been done properly by Rabourdin or not without having to open the file manually to check it all, and it also checks if the names listed in the databases have been taken into account. Each name tagged in TEI will be highlighted in bright yellow for you to check the automatic markup by Rabourdin in the HTML version of the texts. Errors and oversights can then be found easily.

1.3.4 Bixiou

Bixiou sorts all names already indexed in alphabetical order, since the HTML worksheet is not able to do it. The script can then be launched to look for potential duplicates.

1.3.6 txt2json

txt2json is a simple converting tool. It converts the text databases, easy to manipulate, into pretty and web-friendly JSON files.

1.3 Tracking of the duplicates

1.3.1 Tracking on the HTML indexing worksheet

You can avoid creating duplicates if you look for a name in the index before you write it as a new one by default in the text databases: always check whether it already exists or not. However, duplicates will always happen, for instance if a character has changed its name and you won't know it until you read the lasts books of *La Comédie humaine*, or if the operator needs to get some sleep.

1.3.2 Use of the tracking scripts

The script bixiou.py has been coded to help you solve these problems: it helps you detect the duplicates while sorting all names in order.

Tracking the duplicates with Bixiou:

Create a file and name it tri.txt.
Copy and paste all names generated in the HTML worksheet index
Open a terminal and type:

```
python bixiou.py
```

The sorted list will be automatically displayed on your screen.

1.4 Removal of the duplicates

1.4.1 Main principles

Removing the duplicates is done by checking for each name present in several databases (thus, for each character present in different books) whether the information is strictly the same or not. It has to be the exact same, except for relations and aliases that can be completed in one database only (the will still appear on the index). If you find different descriptions for the same character (same ID), please correct this and get an identical paragraph.

1.4.2 How-to guide

1. Open all P or R files where the duplicates appear (you'll find this info on the index worksheet)
2. Choose the best description amongst the several ones you have got, copy it and paste it on the other databases to get the exact same info everywhere the character is listed. Then delete the wrong ID number and replace it with the correct one: you'll make a new ID number available.
3. Open the P or R files of the next book you will be working on and write the free ID number, followed by the 5 angle brackets that are compulsory. This ID will now be available for another brand new character.

1.5 Automatic XML-TEI markup

1.5.1 Default markup

The default markup is done by Rabourdin, with a <persName> tag and a single user ID (@ref)

Exemple: <persName ref="148P">Marie Willemsens</persName>

1.5.2 Customized markup

cf. 1.1.3 Customized indexation procedures and potential openings

1.5.2.1 Refining possibilities

cf. 1.1.3 Customized indexation procedures and potential openings

1.5.1.2 Step-by-step modification of the python files

All the python scripts have commentaries to help you understand and modify them as you wish.

1.5.3 HTML display of the names tagged in the XML-TEI files

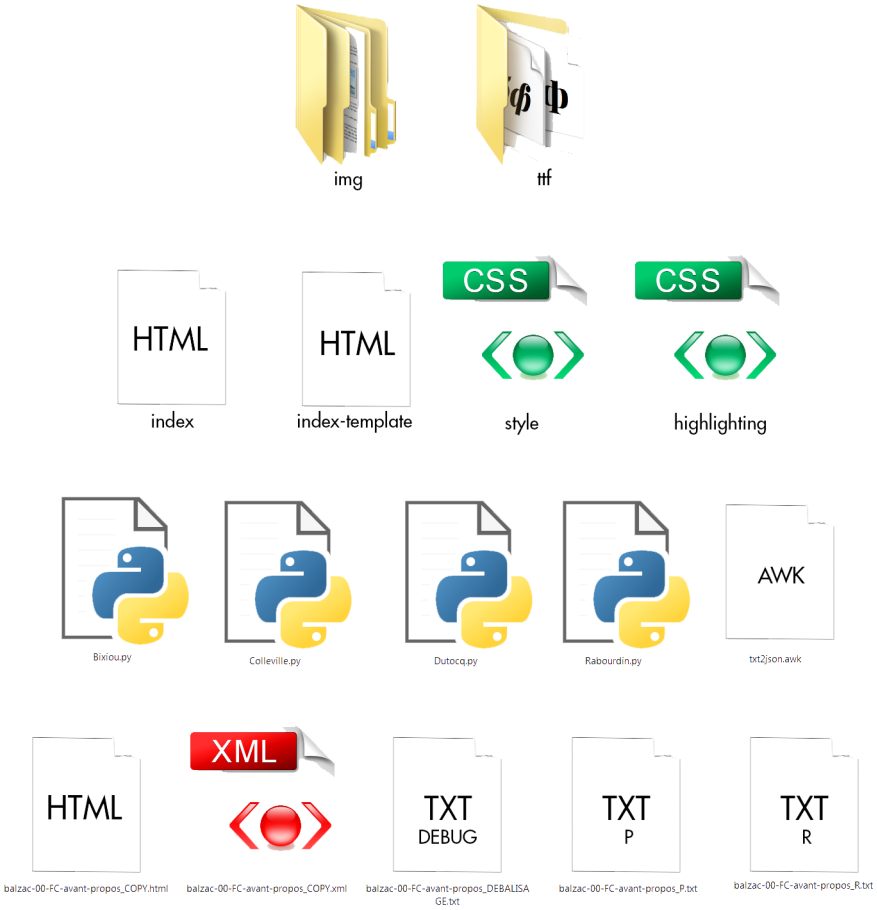
The display is done with the script Colleville.

How-to guide to generate the index

1.1 Preparing the files before starting the indexation

Before you start your indexation, you must create a folder – give it the name you want – on your computer. You are going to work on this folder. It is where the terminal will be launched to use the scripts. This folder will contain all scripts and files related to the indexing process that you can download from GitHub. These are the following:

- All P & R databases already created and completed and the one you are working on ;
- All debugging files already created and completed and the one you are working on ;
- All python and awk scripts: Dutocq, Rabourdin, Colleville, Bixiou, txt2json ;
- A copy of all the XML files of the books ;
- A copy of all the HTML files of the books ;
- A folder named img, containing the illustrations, icons and coats-of-arms ;
- A folder named ttf, gathering the fonts used for the HTML index worksheet ;
- A folder named index.html ;
- A folder named index-template.html ;
- A folder named style.css ;
- A folder named highlighting.css.



The first five elements of the above list allow you to build the index with the scripts and databases. The last six elements allow you to visualize it thanks to an HTML page generated and updated each and every time the indexing process keeps being done.

You can gather all these elements within separate folders, except for some files that have to be kept in the same folder: the four python scripts + the copies of the XML and HTML versions of the books + the databases. All other files can be sorted as you wish, but do not forget to link the two CSS files to the HTML ones (index.html and index-template.html must be linked to style.css and highlighting.css)

You can download all the files and scripts here:

https://github.com/Armellei/IndexBalzac

1.2 Writing the databases

1. Make a copy of the HTML and XML files of the book you want to work on. Put them in the working folder where you have your scripts and all the databases already created.

2. Create a new _P.txt, _R.txt and _DEBUG.txt file named with the name of the book you are working on. Open the three files on Notepad++.

3. Open the HTML version of the text you want to work on in your web browser.

4. Write in the P.txt file the last single user ID not assigned to anyone yet. You will find it if you open the last P.txt database completed and updated on GitHub. This ID will be free. Add some more free ID numbers (for instance, you've seen that 1024P was the last ID used, then you can write 1025P, 1026P, 1027P... 1050P as new and free user ID numbers in your new P.txt file).

5. Do the same for the R.txt file.

6. Read the text you want to markup line per line. For each name found, check whether it already exists in the html index: if yes, copy and paste its description from another database into the new one. If not, list it with a new user ID.

7. Repeat the procedure until the end of the text.

8. If a name must be tagged by hand (e.g. homographs), write it in the debugging file.

1.3 Using the scripts to generate the HTML worksheet index

9. At the end of the indexing, drag and drop the copy of the HTML book you are working on on Colleville's icon. Debug if necessary.

10. Double-click on Dutocq to generate the index.

11. Debug if necessary.

12. Open the index.html and check briefly if the indexing process worked properly (new names added must be visible).

13. Drop the XML file you want to automatically markup on the icon of Rabourdin. It will tag the XML in a few seconds.

14. Debug if necessary.

→ Once the scripts are launched they must shut down without your intervention. If it is not the case, there is an error in the terminal that you must take into account.

1.4 Final checking

15. Use Bixiou to check whether there are duplicates or not in the index. Create a file named « tri.txt », copy and paste in this file all the names found in the HTML index and launch Bixiou. Debug if necessary.

16. Open the HTML index worksheet and verify that your corrections have been taken into account if there were any.

17. Upload the databases and XML files on GitHub. Repeat the whole process from step 1 for next book.

Illustrations

1.1 Editing the illustrations, icons and coats-of-arms

Illustrations:

Size: 1350 x 2000px
Background: white
Sharpness: perfect at 100% (at least)
DPI: 150
Format: .jpg

Icons:

Size: 400 x 400px
Diameter of the circle: 400px
Background: transparent
DPI: 150
Format: .png

Coats-of-arms:

Size: 1670 x 2237px
Background: transparent
DPI: 150
Format: .png

1.2 Photo-manipulation advice (using Photoshop CS5, CS6 or CC)

- Color settings: greyscale mode
- Use the white, grey and black balance tools to clean the image
- Get a pure white (rgb 255,255,255) on the background
- Sharpness must be perfect at 100%
- Delete (overlapping) artifacts
- Strengthen the blacks on all engravers' and illustrators' names if necessary
- Remove all stamps
- Edit the files in .psd or .eps only before a final export in .jpg or .png

Recommendations

1.1 General advice

- Do not list names in the databases starting with a capital letter if they contain the following words: « madame », « mademoiselle », « monsieur ». List them in uppercases only.

- Always launch a research in each P or R database already created to check all weird, suspicious or shifty names. Intuition is key: everything that sounds like a play on words or like a nasty joke from Balzac should be carefully analyzed. Always verify whether these weird names are in the index of La Pléiade before doing anything.

Example: *Toulouse* is quoted twice by par Balzac as a stagecoach driver and as a city within the same sentence. In this case we should avoid tagging all the words *Toulouse* present in the book where *monsieur Toulouse* appears to be doing something since it is also used to talk about the city.

- Always write in the debugging file the terms that should be un-tagged if two characters have the same name, share a common first name, family name, honorifics, etc.

Example: *Bettina* in Modeste Mignon can be both the mother and the daughter. Make the choice to tag the one Balzac writes the most about and manually un-tag and re-tag the homograph in the XLM-TEI file with is proper unique ID number.

- Always keep the two databases for each book (never ever merge the databases, otherwise you will get boatloads of duplicates and our goose will be cooked). The indexation should always be done book by book, in the same order that the one displayed on ebalzac.com, Furne corrigé version.

Common mistakes while indexing and debug of the scripts

1.1 Reading the debug indications given by the scripts

1. Debug operation and common types of mistakes

Two scripts are here to help you debug: Dutocq and Bixiou.
Dutocq works with the databases: it displays our errors.
Bixou works with the HTML index: it sorts all names in alphabetical order to check whether there are duplicates or not.

Two types of errors can occur: errors related to the characters (duplicates, oversights, etc.) or errors related to the syntax in the databases, impeaching the scripts to work properly and the index to be generated. The scripts will display the errors as long as they remain present in the databases. The work won't be possible until you have not debugged your files. Nasty but necessary procedure.

Errors made by the operator could be the following while indexing a huge number of characters on a book:
- register of a character who already exists with a new ID ;
- addition of more data in a characters' description that would not be updated in all the databases where this character appears;
- syntax errors in the databases: missing spaces, @ symbols, angle brackets, empty compulsory field, etc.

Never forget to save and upload all the databases you have created, completed, checked and debugged on GitHub right after the work has been done.

2. Examples

The messages displayed in the terminal while launching the scripts will look like this if there are any errors:

1.2 Mistakes regarding the characters

1.1.1 Removing the duplicates

cf. 1.3 Tracking of the duplicates

1.1.2 Removing nonidentical data about a single user ID

cf. 1.3 Tracking of the duplicates

1.2.3 Errors while filling in the fields

cf. 1.2.6 Nomenclature and syntax of the databases

1.3 Mistakes regarding the syntax of databases

MESSAGE AFFICHÉ PAR DUTOCQ	TYPE D'ERREUR DANS LA BASE DE DONNÉES
	ERREUR DE SYNTAXE
`Error in the character file! Possibly a missing ">" or extra blank line for this or a reversal between a place and a number (0/1/2/3/4):`	le lieu et le genre ont été inversés OU un saut de ligne est mal placé ou en trop OU il y a moins de 5 chevrons
`There are more than 5 '>' for`	il y a plus de 5 chevrons
	ERREUR D'ÉCRITURE
`Names don't match for character 95 (Renée de) and Maucombey (Renée de)`	le nom n'est pas identique
`Occupation don't match for character 95 (Maucombe (Renée de)`	la description n'est pas identique
`Cities don't match for character 95 Maucombe (Renée de)`	le lieu n'est pas identique
`Gender don't match for character 95 Maucombe (Renée de)`	le genre n'est pas identique
`First letters don't match for character 95 Maucombe (Renée de)`	la lettre d'entrée dans l'index n'est pas identique
`Names don't match for character 267 (Pierrotin and Jean)`	un même ID est attribué à deux personnes différentes

1.2 .1 Txt and JSON syntax errors

cf. 1.2.6 Nomenclature and syntax of the databases

1.2.2 UTF-8 encoding errors and special characters

UTF-8 encoding

Character encoding can be tricky. All P and R databases must be encoded in UTF-8. If not, errors will occur while running the python script: convert the file into UTF-8 and relaunch the scripts.

Annoying glyphs

Nonbreaking spaces

Some names will not be displayed properly in the index: those cut by a nonbreaking space for instance. You have to copy and paste directly from the HTML version of the text found on ebalzac.com the name into the databases. The nonbreaking space should be included this way.

Names in small caps in the XML and HTML files

Names in small caps still remain an unresolved problem: good luck with it, Rabourdin will recognize them half the time. For now, you have to manually tag these names in the XML file since the problem has not been resolved. Enjoy.

Superscripts

Names composed with a superscript (<sup> tag in the XML file) are not tagged with the superscript: stop tagging before the <sup>. For instance: Charles I^{er} → just write « Charles I » in the text databases, that's all. Same joke for subscripts.

Special glyphs

Some glyphs might be listed in the text databases yet they won't be recognized by the XML-TEI file and Rabourdin won't tag them. It can be the case for the following glyphs that you must replace with their HTML code:

&#00C1; = Á
#U+00D0; = Đ
etc.

1.4 Multiple mistakes

Several errors might occur on the same book: the scripts will tell you more about that in the terminal that will get opened automatically immediately after launching the scripts. Press enter to see the next errors if there are several. Once all errors have been listed, the terminal will shut down. Just launch the script again to make it list all the errors again.

Employees on strike

Rabourdin refuses to launch:

A character might be causing you trouble thanks to a delicate UTF-8 encoding: character *Á* for instance will cause errors. Delete this glyph on your databases and replace it manually by taping the corresponding HTML code (should be like &#XXXX;).

Colleville does not tag anything:

If Colleville does not tag anything and shows an error before shutting down: there is an error in the P or R files. Launch Dutocq which will be tracking it, correct the mistake and launch Colleville again.

Rabourdin does not tag anything:

The databases are not within the same folder than Rabourdin.py. Remove them into the same folder and launch the script again.

→ All messages displayed by Rabourdin, Dutocq and Colleville looking like « X was not found ! » are normal: they are no errors but simple indications about who has been tagged or not in the XML-TEI file. The terminal will simply shut down after listing all names not found in the XML files.

File upload

1.1 Database updates from /IndexBalzac on GitHub

The existing databases, debugged and ready for web integration are available online:
https://github.com/Armellei/IndexBalzac/tree/master/Database

1.2 Preparing the files for a web integration

The files that need to be sent to the webmaster for web integration are the following:

- P and R databases converted into JSON by txt2json
- XML-TEI files that have been tagged by Rabourdin
- all edited images (illustrations, icons, coats-of-arms)

You do not need to provide the debugging files, the txt databases, the HTML version of the index and the scripts.

Source code

Source code is available here:
https://github.com/Armellei/IndexBalzac/tree/master/Indexing%20scripts