

Specifying a Generic Working Environment on Historical Data, Based on MetaindeX, Kibana and Gephi

Laurent Millet-Lacombe

¹MetaindeX, France

Abstract

We will try to define in this article basic user needs for a generic working environment on historical data, discuss then some key technologies and architecture orientations for online open-source application MetaindeX [1], which intends to fulfill those user requirements. At last, we will illustrate its usage with a real corpus of few thousands French archives from "Archives Nationales", from 16th and 17th century.

Keywords

statistics, charts, maps, links, graphs, user interface, NoSQL, metaindex, gephi, kibana

1. Introduction

This article takes place in the frame of "Effective interfaces for searching, browsing or visualizing historical data collections" topic proposed for HistoInformatics2021 workshop. It defines an overview of what user requirements could be for a working environment offering a ready-to-use, coding-free, quick and efficient access to advanced exploration, analytics and graphs capabilities over an historical data corpus for the researcher. It supposes that input data has already been collected, cleaned and reconciled, focusing on the exploration and analytics parts of historian workflow. In a second part, it then develops those needs through key technologies and architecture choices made for design and development of online open-source application MetaindeX [1]. At last, a sample use-case illustrates user experience and shows some analytics results from the use of MetaindeX and its ecosystem, based on a set of French archives from 16th and 17th century.

2. User Requirements Overview

Prior to defining user's expectations, we will start by defining his technical profile: he is quite at ease with standard tools such as Excel or even OpenRefine [2]; he is not used to or do not have time for developing his own analytic scripts in Python or so; his corpus is already available

HistoInformatics 2021 – 6th International Workshop on Computational History, September 30, 2021, online


✉ laurentmlcontact-metaindex@yahoo.fr (L. Millet-Lacombe)

🌐 <http://metaindex.fr/> (L. Millet-Lacombe)

🆔 0000-0002-2623-9648 (L. Millet-Lacombe)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

and consistent, but might be too big for convenient processing with Excel; he might work in international projects implying several other researchers from various countries.

In this context, following sections are a specification proposal from the author, intending to define what could be the basic needs of such a user to work on his corpus.

What is called a 'document' there is technically a database "entry", but what it represents depends on each use case: it could be actual documents (a book, a lawyer deed), but also more abstract entities studied by the researcher, such as an historical character, a city or an institution for example.

Basic requirements numbering [Reqxxx] is used to help the reader understanding which technical feature described in section 3 relates to which user requirement from this section.

2.1. Robustness to Volume and Contents

The user wants to potentially work with large amount of documents (up to 100000 or even more) even if he only owns a simple notebook [Req110] (if handling few thousands documents is usually sufficient for a physical corpus, this can quickly reach a significantly bigger size for a natively digital corpus). Documents are made of attributes (which we will call 'fields') whose list may vary for each document [Req120]. The user also wants to represent links between documents and associated weights [Req130]. The user wants to explore his corpus, using advanced queries like setting constraints on specific fields or using fuzzy (i.e approximative) matching rules [Req140]. If documents have associated ad-hoc contents (such as a picture typically), the user wants to store and reference them too [Req150].

2.2. User Interface Look and Feel

The user wants only minimal configuration steps prior to import and work with his data, typically limited to defining fields datatype [Req245]. With little more setup, the user wants too the interface to match his project specificities, in terms of spoken language(s) [Req210], wording of main concepts [Req220], and contents rendering layout [Req225]. He also wants to customize document's summary text displayed when browsing a search result [Req230], and optionally which picture shall be used as a thumbnail for each document [Req240]. Finally, he wants as fewer interface panels as possible [Req250], in order to enforce interface clarity.

2.3. Access Control and Collaborative Work

The user wants to control who can access his documents [Req310]: this is important to preserve exclusivity for an upcoming publication, and becomes critical when access to studied documents is restricted by law. In the meantime, he wants to let trusted persons access contents either as read-only observers or full partners able to change contents.

2.4. Ensure Interoperability

The user wants to work with an ecosystem of applications continuously evolving rather than a single multi-function application, this way his working environment is much more resilient, scalable and adaptive to future tools still to come. Using at least CSV format for data import

and export is a must as a standard pivot format [Req410], while importing contents directly from Excel file would be an appreciable feature [Req420]. GEXF format is preferred for graphs definition, as a well-known, formal and flexible way of defining networks [Req430]. In order to promote use of independent and interoperable tools, a toolbox area shall be proposed containing sample tools and code (typically in Python) for data cleaning and reconciliation. [Req440].

2.5. Perform Analytics

The user wants to produce two types of analytics over his corpus without having to code: advanced statistics charts based on documents' fields, and graph representation of links between documents. Statistic charts would include at least basic pie charts, line or area plots, and histogram diagrams. Colored maps with documents geolocation (based on fields containing political zones names or longitude/latitude coordinates) shall also be available. Those charts and maps shall be updated automatically as the contents of the database changes [Req510]. Graph analytics shall allow to customize nodes size and color based on documents fields values [Req520]. An "aggregated-graph" mode shall also allow to study connections between groups of documents, each group containing documents sharing a common value for a given field [Req530].

3. Focus on Key Implementation Details for MetaIndex

3.1. A Lightweight Client

In order to provide quick access to a running application and keep good performance independently from user computer characteristics [Req110], a 'light-weight client' architecture has been chosen, based on a web HTML5 client (Firefox) connected to an online dedicated web server. Such a client-server architecture allows also easier management of collaborative work and data access control [Req310] since operations and data storage are handled on a centralized location.

3.2. Key User Experience Features

3.2.1. Several operations from same page

Client-server communication uses extensively websockets technology for a smoother user experience, allowing then to stay on same page while performing several operations, rather than reloading HTML page every time an action is performed [Req250].

3.2.2. Rendering documents and customizing interface

For efficient search results overview, a grid of cards is proposed, each card corresponding to a document (or a line of loaded CSV file) and displaying a contextual title and optional thumbnail picture. For document contents, since each document might have various list of fields, an adaptive grid layout was also chosen [Req120].

Since product is generic, some customization shall be allowed to get interface consistent with each project specificities. Following customizations are applicable at catalog level:

- list, name and type of possible fields in documents. Type is important here (string, integer, float, date, geo-point) for proper value interpretation when computing statistical analysis [Req510].
- cards text and picture: exploring catalog's contents is done via displaying cards (one card per document), each one having a summary text and optionally a picture as a background. Default summary text is obviously document's ID, which is rarely useful nor explicit. User defines then a list of fields to concatenate in order to build a summary text specific to each document [Req230]. For example 'firstname, familyname, birthdate' could generate 'John Dow 1765' and 'Marty McFly 1985' for two different documents. A field containing a permalink to a picture could also be used as a background thumbnail picture adapted to each document [Req240].
- fields perspectives: user creates its own predefined layouts (called perspectives) where fields are organized by tabs and sections, each section containing a chosen subset of fields to be displayed [Req225].
- catalog's lexic: user set personalized names [Req220], in various languages [Req210] to be used in user interface for fields titles and main catalog's concepts. For example, in some projects a "document" might be called an archive, while for others it might be called a book, a city or a character.

3.3. A Powerful Database

NoSQL database "Elasticsearch" [3] has been chosen since it is natively documents oriented [Req120], it requires little data model configuration [Req245], it is scalable and fast, offers a very powerful query engine (Lucene [4]) allowing for example fuzzy search [Req140], and, at last it allows smooth integration with Kibana [5] application, giving user ability to define its own analytic charts, maps with documents geolocation and dashboards in a 'self-service' approach [Req510].

3.4. Linking Documents

Elasticsearch links model is quite limited for our use case, and a complementary specialized tool or database such as Neo4j [6] or Nodegoat [7] would probably add serious overhead to preserve consistency of data model and contents. In order to keep data model simple, chosen solution is to define links as basic 'string' fields to be interpreted at application level, meaning that links do not use any database built-in logic, data structure nor metadata, excepted the name of the link field itself. Links then simply contain identifiers of target documents to point at, optionally with associated 'weight' information [Req130]. For example, value "doc001 : 4 , doc003" for such a 'links' field of document "doc002" would mean a link to "doc001" with a weight of 4 and a link to "doc003" with implicit weight of 1. Main drawback of this approach is though that "in-depth" queries (i.e. defining constraints through links) are not directly possible (for example, considering a family tree, "retrieving all persons who have at least one child born before 1700"), but can usually be balanced with extra information added during data preparation or sequencing several queries.

About links weight, its actual meaning is left to user interpretation. As an example, a first link between two "cities" could represent some amount of commercial exchange and a second link the amount of people travelling. If link is between two "artists", it could then represent the number of peaces of art they own from each other.

Design by the researcher of data model to apply will depend on analytics objectives and corpus constraints. Continuing on our last example, the researcher could have for example considered a peace of art itself as a document to be explicitly described, with for example title, creation date, type, a link towards its creator and another link towards its successive owners. This alternative would lead to easier analytics focusing on peaces of art themselves, but more complex data structure if objective is to study relations between artists. If both objectives are targeted, maybe both approaches can be applied, at the cost of some redundancy in database contents.

3.5. Data import/export

As we said, interoperability is a critical feature to let user choosing the best application for its specific needs. Standard import formats .xls, .odt, and .csv are supported [Req410,Req420], while only CSV is available for export, since most of applications will accept this format. For a first 'from-scratch' import, only thing the user has to configure is fields datatype (string, integer,etc.). This operation is then not needed anymore for next imports [Req245].

Graphs export module generates GEXF graph file out of documents' "link" fields [Req430] for edges, while other fields are used as nodes contents. [Req520]. Moreover, an additional aggregation mode for graphs is available [Req530], where nodes and links are gathered together for documents sharing the same value for a given field (it can be seen as a simplified version of similar approach developed for Neo4j database [8]). This "group-by" feature will allow user to get more abstract and readable graphs, focused on studied topic.

3.6. Store and Reference Associated Files

User corpus can be associated with some files such as scans or illustrations typically. A standard SFTP service is then used to store those files on server [Req150]. Permalinks are generated to allow references to those files from documents fields (for example add a portrait picture to a document representing a character, or a scan for a lawyer deed). Proper access rights control is performed when accessing those contents both via SFTP and through permalink, ensuring that only logged-in and allowed users can actually access those files [Req310].

4. An Example with French Archives from Ancient Régime

We will show here how to work with a corpus from French archives, issued from a PhD work whose author kindly accepted to lend its data to run this demonstration.

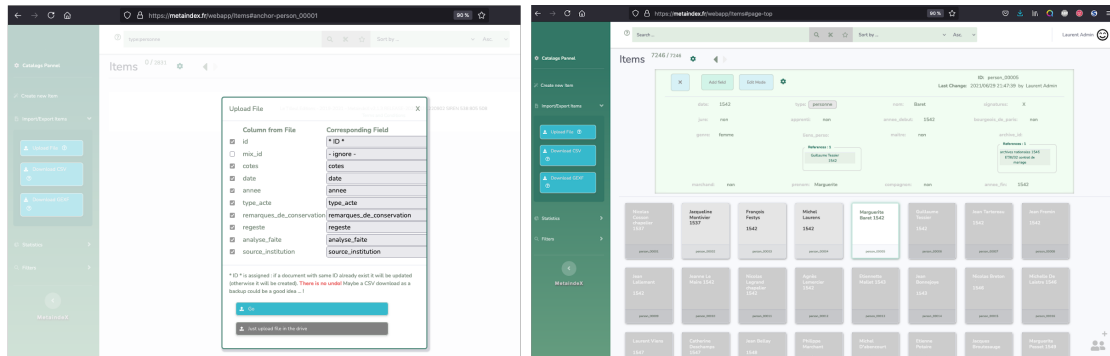


Figure 1: Screenshots from Metaindex. Left: CSV import module; Right: cards and document contents

4.1. Corpus Overview and Upload

4.1.1. Delivered corpus

The corpus is about French notarial archives (estate lawyers) from 16th and 17th century in Paris, and has been delivered as a set of Excel files of about 20000 lines, each line representing a specific person (name, gender, professional situation, textual description of personal and professional connections, signature), seen within a specific archive (reference, institution, date, type).

We will focus our study on two axes: building few statistical charts on the one hand, and generating some graphs of social relationships on the other hand.

Data preparation (done with Openrefine and Python scripts from Metaindex' toolbox) consisted in extracting a list of unique persons and a list of unique archives as two separate CSV files (about 4400 individuals and 2800 archives), and performing reconciliation of textual links descriptions into unambiguous lists of documents identifiers.

4.1.2. Loading and Exploring corpus

Dragging those CSV files over Metaindex user interface opens CSV-import module, which let the user set mapping of CSV columns to catalog's fields as shown on Figure 1 (left side).

Once loaded, documents are represented as cards, each card being a single line from our input CSV files (in our case either an archive or a person). A card can be expanded to see or edit document's fields as illustrated on Figure 1 (right side).

We can notice that links are displayed with a summary of linked documents (as for the cards summary text), rather than simply IDs list. Also jumping through links is possible by clicking on it, allowing user to conveniently navigate through those connections.

Lucene query syntax [4] is available as a search engine, we can for example look for "all persons whose first name approximately equals to 'Antoine' and was born before year 1700", which would translate to following query:

```
type:person AND firstname:Antoine~ AND datestart:<1700
```

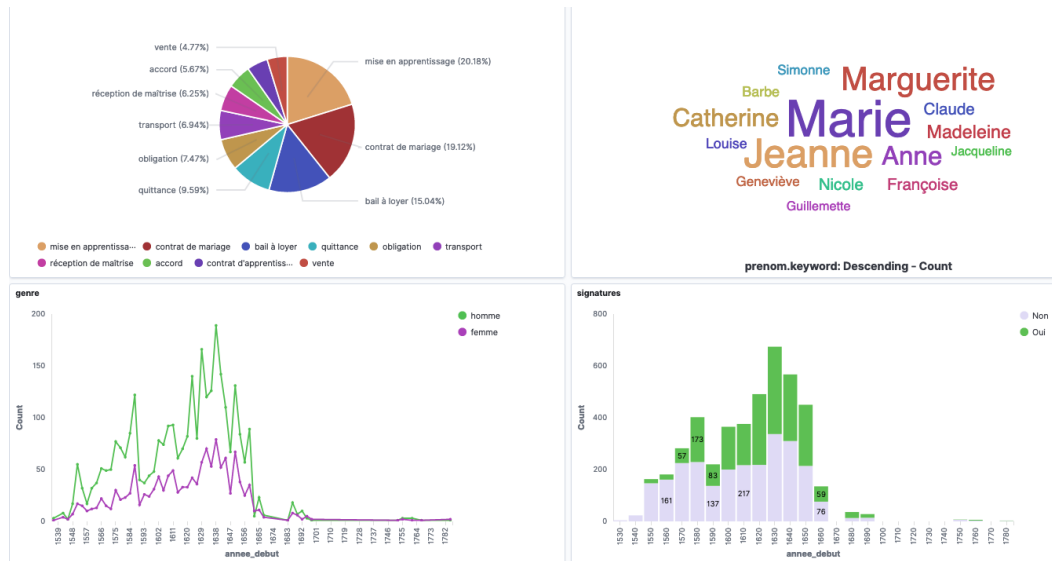


Figure 2: Dashboard analytics from Kibana: archive types (top left), popular female firstnames (top right), gender representation by decades (down left), ratio of people who knows how to sign per decade (down right)

4.2. Analytics Charts

Kibana is a statistic analysis environment running over Elasticsearch. In our case, integrated Kibana and Metaindex applications use the same Elasticsearch database instance, and so user will find same contents on both applications. Detailed usage of Kibana is out of scope of this document, but some example charts could easily be created in few minutes with our sample corpus and grouped into a dashboard presented Figure 2.

4.3. Studying Links and Graphs

4.3.1. Basic Graph Generation

Metaindex can generate a graph description file (GEXF format [9]) where each document would be a node and each link would be an edge. Such file can then be loaded with Gephi [10] tool. Generation module let the user select which fields to be injected as nodes' metadata, allowing then fine graph rendering customization based on those contents, for example by assigning a color to nodes depending on value of a given field.

Detailed usage of Gephi is out of the scope of this article, but once GEXF file generated from our data and loaded in Gephi, with only few settings we can already identify some clusters within professional and personal relationships among persons, as shown on Figure 3 (left side) (each grey dot represents a person, names have been hidden for better readability of the networks).

Such information could help the researcher to get a better vision of social and professional relationships over his corpus, and maybe interpret with better accuracy historical facts he could

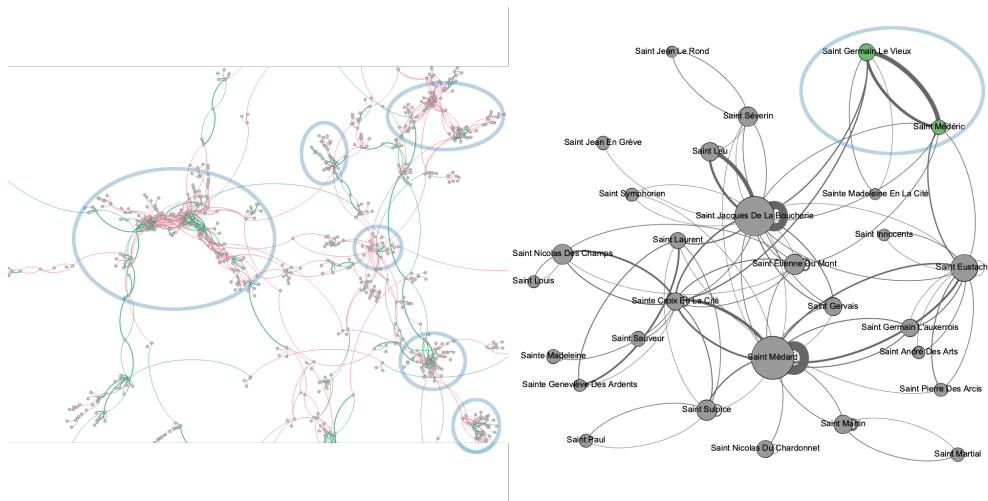


Figure 3: Left: professional (green) and personal (pink) relationships among individuals (grey dots); Right: professional relationships grouped by parish.

find on archives contents.

4.3.2. Aggregated Graph Generation

Since bigger graphs interpretation is a difficult task, Metaindex offers also a specific algorithm to generate simplified aggregated graphs. This way, nodes and links are gathered up when documents share the same value for a given field. This "group-by" feature allows to get much more readable graphs directly focused on topic the researcher is interested in. Following our example, we can group persons by parish they live in, which means that all persons having a "parish" field with a similar value will be grouped within a single node, and their respective links will also be aggregated to this node.

Figure 3 (right side) shows as a result all parishes found in the corpus, their size depending on amount of individuals registered as living there, while links thickness is based on amount of links that all individuals from given parish have with individuals from another parish. On that graph, we can see that parishes Saint-Germain-Le-Vieux and Saint-Médéric (top right) seems to have quite numerous professional relationships (thicker link) despite their smaller amount of persons recorded to live in (smaller nodes size) from our corpus. That could maybe lead the researcher to a new approach or hints to understand social relationships of this community.

5. Conclusion

We started with a proposal of user requirements for a generic historical data management tool, and detailed some technical choices made for online open-source tool Metaindex, which intends to fulfill those user requirements. At last, with a real-life example of thousands elements extracted from French Archives Nationales, demonstration has been done that Metaindex and

its ecosystem offers an efficient working environment on historical data collections for the researcher. Upcoming evolutions of Metaindex are still to be refined, but we can list following features as possibly of interest: advanced data representation approaches; import data from RDF [11] format or from a DTS API [12]; advanced text statistics, for example with integration of TXM [13] tools suite for text analytics; export corpus for public diffusion, with tools such as Omeka [14] or similar.

References

- [1] L. Millet-Lacombe, 2020, Metaindex cataloger, accessed September 1 2021. URL: <https://metaindex.fr>, source code at <https://github.com/laurentmldev/metaindex>.
- [2] S. van Hooland, R. Verborgh, M. D. Wilde, 2013, Cleaning data with openrefine, accessed September 1 2021. URL: <https://programminghistorian.org/en/lessons/cleaning-data-with-openrefine>.
- [3] E. B.V., 2021, Elasticsearch guide, accessed September 1 2021. URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html>.
- [4] T. A. S. Foundation, 2013, Apache lucene - query parser syntax, accessed September 1 2021. URL: https://lucene.apache.org/core/2_9_4/queryparsersyntax.html.
- [5] E. B.V., 2021, Kibana guide, accessed September 1 2021. URL: <https://www.elastic.co/guide/en/kibana/current/introduction.html>.
- [6] I. Neo4j, 2010, Github neo4j, accessed September 1 2021. URL: <https://neo4j-contrib.github.io>.
- [7] LAB110, 2021, Nodegoat website, accessed 25 September 2021. URL: <https://nodegoat.net>.
- [8] Neo4j, M. Junghanns, 2010, Graph grouping with neo4j, accessed September 1 2021. URL: <https://neo4j-contrib.github.io/neo4j-apoc-procedures/3.5/virtual/graph-grouping/>.
- [9] G. W. Group, 2009, Gexf file format, accessed September 1 2021. URL: <https://gephi.org/gexf/format/>.
- [10] M. Bastian, S. Heymann, M. Jacomy, Gephi: An open source software for exploring and manipulating networks, Proceedings of the International AAAI Conference on Web and Social Media 3 (2009) 361–362. URL: <https://ojs.aaai.org/index.php/ICWSM/article/view/13937>.
- [11] W3C, 2014, Resource description framework, accessed September 1 2021. URL: <https://www.w3.org/RDF/>.
- [12] D. W. Group, 2020, Distributed text service specification, accessed 20 September 2021. URL: <https://w3id.org/dts>.
- [13] S. Heiden, J.-P. Magué, B. Pincemin, 2010, Txm : Une plateforme logicielle open-source pour la textométrie - conception et développement, accessed September 1 2021. URL: <https://halshs.archives-ouvertes.fr/halshs-00549779/>.
- [14] R. R. C. for History, N. Media, 2021, Omeka website, accessed September 1 2021. URL: <https://omeka.org/>.