

THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

Improvement of User Interaction on visualization

Application to EVOQ a text visualization tool for Structural Analysis

Agossou, Bidossessi Emmanuel

Award date:
2019

Awarding institution:
University of Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



**UNIVERSITÉ
DE NAMUR**

FACULTÉ
D'INFORMATIQUE

**Improvement of User Interaction on visualization:
Application to EVOQ a text visualization tool for
Structural Analysis**

Bidossezi Emmanuel AGOSSOU

UNIVERSITÉ DE NAMUR

Faculty of Computer Science

Academic Year 2018-2019

**Improvement of User Interaction on visualization:
Application to EVOQ a text visualization tool for
Structural Analysis**

Bidossezi Emmanuel AGOSSOU



Supervisor: _____ (Signed for Release Approval-Study Rules art. 40)

Pr. Bruno DUMAS

Co-Supervisor: Pr. Isabelle LINDEN

A thesis submitted in the partial fulfilment of the requirements
For the degree of Master of Computer Science at the University of Namur

Abstract

Visual text analysis techniques have been one of the most prominent topics over the past decades. Many researchers have recently been interested in using text visualizations techniques and tools to conduct the structural analysis of a text. Text visualization indeed uses graphical representations to help text analysts to draw more knowledge from a raw text and to easily uncover the relations between the terms of a text whether explicitly or not. However the analyst's interactivity on the visualizations is determinant for a deeper understanding of the text. This work aims at improving the user interaction on the visualizations provided by EVOQ, an existing tool for text structural analysis. We have implemented a new algorithm for EVOQ which not only generates visualizations based on a pre-encoded terms relation dictionary but also helps analysts to construct the visualizations at ease. After evaluating our proposed solution with some users and the existing visualization interaction techniques, we have suggested some guidelines for future improvements.

Keywords: text visualization, structural analysis, relations dictionary, visualization techniques, interaction.

Résumé

Les techniques d'analyse visuelle de texte ont été l'un des sujets les plus en vue au cours des dernières décennies. De nombreux chercheurs se sont récemment intéressés à l'utilisation des techniques et outils de visualisation pour effectuer l'analyse structurelle d'un texte. En effet, la visualisation de texte utilise des représentations graphiques pour aider les analystes à tirer davantage de connaissances d'un texte brut et à découvrir facilement les relations entre les termes d'un texte de façon explicite ou non. Cependant, l'interactivité de l'analyste sur les visualisations est déterminante pour une compréhension plus approfondie du texte. Ce travail vise à améliorer l'interaction de l'utilisateur sur les visualisations fournies par EVOQ, un outil existant pour l'analyse structurelle de texte. Nous avons implémenté un nouvel algorithme dans EVOQ qui permet non seulement de générer des visualisations à partir d'un dictionnaire relationnel de mots pré-encodés, mais aussi aide les analystes à construire les visualisations à leur aise. Après avoir évalué notre solution avec des utilisateurs et les techniques d'interaction de visualisation existantes, nous avons proposé des directives pour les améliorations futures.

Mots-clés: visualisation de texte, analyse structurelle, dictionnaire des relations, techniques de visualisation, interaction

Acknowledgements

I would firstly like to say a big thank to the almighty GOD for enabling me with health and strength to finish this master program.

Special thanks to my thesis supervisor Pr. Bruno Dumas and my co-supervisor Pr. Isabelle Linden from the University of Namur. They have always opened the door of their office to me anytime I faced difficulties and needed to discuss ideas. Thank you for always been available to challenge me and provide me with directions when needed.

I would also like to thank the whole management of the University of Namur for granting me with UNamur Fundraising scholarship which enables me to do this program. My thanks also go to all the staff of the International Relations Service and more specifically Mr. Henrich Brunke whose friendship and kindness have been a great support for me.

Also, I thank my fellow students from Students For Christ (SFC) specifically Christian Nazili and François Kouptchinsky who have both been more than friends to me. Finally, I will always be grateful to my parents Mr. Medard Agossou and Ms. Jeanne Wamasse for their everlasting support and to my lovely wife Sarrah Agbogba and my dear children for their unfailing support and encouragements. The accomplishment of this thesis would not have been possible without them.

Contents

Chapter 1: Introduction	6
Chapter 2: State of the Art.....	8
2.1. Information visualization pipelines	8
2.1.1. Fry's visualization pipeline	9
2.1.2. Malyanov et al. Visualization pipeline stages	9
2.1.3. Jansen and Dragicevic Visualization pipeline stages.....	11
2.1.4. Liu et al. pipeline Visualization stages	12
2.2. Text visualization techniques	14
2.3. Brief description of Structural analysis.....	25
Chapter 3: EVOQ	27
3.1. Context and Goals of development.....	27
3.2. Development Methodology and technologies.....	28
3.2.1. Development Methodology.....	28
3.2.2. Technologies used in EVOQ.....	30
3.3. Software architecture and functionalities.....	31
3.3.1. Global architecture.....	31
3.3.2. Main Functionalities	34
3.3.3. EVOQ Main page Interface	35
3.3.4. Two ways of building the nodes-links graph.....	36
3.4. User interactions in EVOQ	42
3.4.1. Linking and Brushing.....	42
3.4.2. Construction-Based Graph vs. Dictionary-Based Graph.....	43
3.5. Existing Interaction techniques and Analyses.....	44
3.5.1. Existing interaction techniques in information visualization	45
3.5.2. Interaction Analysis	46
Chapter 4: Evaluation of EVOQ.....	48
4.1. Evaluation with Users.....	48
4.1.1. Participants	48
4.1.2. Evaluation Methodology.....	48
4.2. Evaluation Results.....	51
4.3. Suggestions of future improvements on EVOQ.....	53
Chapter 5: Conclusion	55
Bibliography.....	57
Appendices	60

Chapter 1: Introduction

Large amounts of information such as word processing documents, web pages, blogs, online news, publications and articles are more available today in form of text. This increasing availability of textual data has been made possible due to the rapid digitalization of paper documents. However, one of the problems with raw textual documents is that they are so voluminous. Therefore, finding relevant knowledge from them has become a tedious task. In order to extract knowledge, assimilate and navigate more easily through these textual documents, text mining and visualization techniques and tools have emerged as relevant solutions. Since the 1960s, many text visualization techniques have been extensively studied and developed for different purposes. The recent works provide the overview of the existing research in text mining, taxonomies and visualization techniques ((Yang et al.,2007), (Gupta and Lehal, 2009), (Kucher and Kerren,2015), (Cao and Cui, 2016), (Liu et al., 2018)). Visual text analytics has recently been one of the most prominent topics in both academic research and commercial world (Liu et al., 2018).

The text analysis technique of interest in this thesis is called *structural analysis*, a qualitative text analysis technique. Structural analysis is a text analysis technique used by researchers in the field of humanities which consists in matching related terms (single words or groups of words) manually in order to help uncover the relations between the terms of a text whether explicitly stated in the text or not. The difference between the structural analysis and text mining is that text mining tools provide the result of the analysis whereas in structural analysis it is the analyst who provides the result by making use of the tool. Introduced in 1997 by Jean-Pierre Hiernaux (HIERNAUX, 1977) and amplified later by (Piret et al.,1996), the objective of structural analysis is to understand the implied and connotative meanings that hide in a text.

Wallemacq et al. presented *EVOQ* (Wallemacq et al.,2004), a tool specially designed for structural analysis. This tool allows analysts to specify relations by themselves and visualize the relations networks on 2D node-link or a 3D landscape interactive representation.

The general question of research of this thesis is the following:

RQ: How can we improve the text analyst interactions on a text visualization tool based on structural analysis?

In order to efficiently answer this question our work has explored the following:

(1) Providing an up-to-date literature review;

Using (Clarinal et al., 2018) work as a starting point, we have made a deep research on related works in information visualization and more precisely in text visualizations and user interactivity in order to provide an up-to-date literature review.

(2) Improving EVOQ interface to enable text analysts to visualize both the text and visualization diagrams at the same time;

By using open source web technologies and agile method of software development which enables both increments and iterations on the designed interfaces, we have implemented a new work interface for EVOQ in order to help analysts operate both the visualizations and text in the same windows.

(3) Implementing a visualization algorithm which enables text analysts with more dynamic interactions on the visualizations.

In the previous EVOQ the analyst generates the visualizations based on a pre-encoded terms relations dictionary and cannot easily interact with the visualizations. Therefore, we researched on how we could increase the analyst interaction on the visualizations and implement a new algorithm which would help the analyst to build (construct) visualizations rather than generating them. We finally discovered **Viz.js** (sometimes denoted *Vis.js*), a JavaScript open source library, which we used for the implementation of our solution.

The remaining part of this thesis is organized as follows. Chapter 2 presents the state of the art, the relevant literature review of the researches already done in the field of information visualization and more precisely in text visualization. Chapter 3 describes EVOQ, the software solution we have proposed. It gives details on the context of the development, the methodology and technologies used, the new software architecture and functionalities. It also presents an analysis of the user interactions we have implemented. Chapter 4 presents the evaluation of our new solution with some users and provides guidelines for future improvements of EVOQ software. Finally, chapter 5 concludes the thesis by recalling the contributions and the limitation of the work and the possibilities of future work as well.

Chapter 2: State of the Art

There has been a growing interest for visualization process and text visualization techniques in the past decade. In fact (Nan and Cui, 2016) have reviewed over 200 papers on text visualization techniques summarized in the Text visualization Browser (TVB) an online platform developed by the ISOVIS Research Group from Linnaeus University in Sweden in 2014.

In this chapter our purpose is to present an up to date overview of the existing visualization techniques that are relevant to the present thesis. We have organized the chapter in three sections.

The first section presents the information visualization process or pipelines. In this first section we have explained the different transformation stages a raw text must go through in order to present visual representations which enable text analysts to cognitively formulate knowledge based on the visualizations.

In the second section the most up-to-date relevant text visualization techniques have been explored. Moreover this section puts an emphasis on the relevant text visualization techniques which enable the user (text analyst) with dynamic interactions.

The third section provides brief explanations on how a structural analysis is conducted.

2.1. Information visualization pipelines

Also called “InfoVis pipeline”, the visualization process is a transformation of information content (*raw text, data, meta data, etc.*) into knowledge via a visual representation. It involves both *visualization systems* that transform content into pictures termed visualizations and *humans* who transform visualizations into knowledge (Malyanov, et al., 2013). In the visualization process humans can have roles of users and thinkers. Human can alternate visualizations by interacting with the systems and as a thinker cognitively formulating knowledge (discoveries) based on the visualizations. Some researchers have proposed pipelines to describe the visualization process step by step. The four following pipelines are the most found in the literature.

2.1.1. Fry's visualization pipeline

Fry (2008) presented a pipeline explaining seven (7) stages process for manipulating and making sense of data from its acquisition through its visualization and ultimately to its interpretation. The model of Fry is shown in figure 2.1. According to Radburn et al., Fry was not rigid but draws attention to the inter-dependencies between the various elements of the visual data analysis process (Radburn et al., 2010).

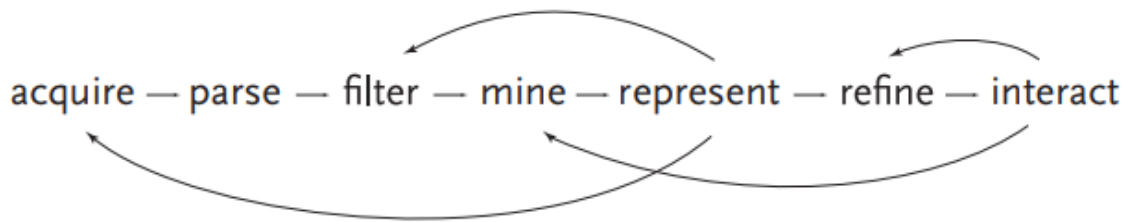


Figure 2.1: The seven Stages of Visualization (Fry, 2004 & 2008)

Fry visualization pipeline was used by Radburn in 2010 in *vizLib project* which objective was to develop capacity for exploratory data analysis in local government through the Visualization of Library Customer Behavior.

Chen and Jänicke in 2010 introduced a visualization pipeline containing five (5) consecutive stages such as *filtering, visual mapping, rendering, displaying and optical transmission*. In their paper, the stages of visualization are described as stages of signal transmission affected by errors (Chen and Jänicke, 2010). Their pipeline focuses more on the theoretical aspects than the visualization mechanics.

2.1.2. Malyanov et al. Visualization pipeline stages

In 2013, Malyanov et al., refined the work of Chen and Jänicke and presented five (5) stages of visualization process that transform raw data into an interactive rendering (Malyanov et al., 2013). The pipeline described in their paper is similar to the one in Chen and Jänicke paper (2010) but focuses less on the theoretical aspects and more on the visualization mechanics. Malyanov et al's, pipeline focused on the transformation of abstract data to visual information in order to facilitate human knowledge. According to Malyanov et al's; visualization experience is obtained during an interaction with the

Information System and reflects human sensations arising during the visualization process and represents the degree of cohesiveness, knowledge formulation and satisfaction in the visualization environment. Figure 2.2 bellow presents the five stages as far as Malyanov et al. pipeline is concerned.

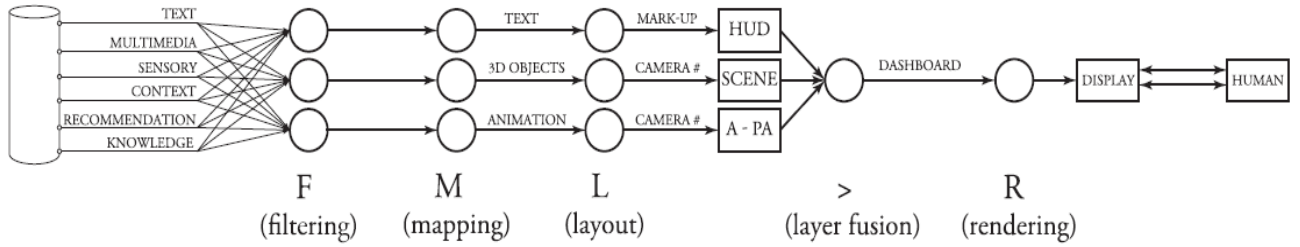


Figure 2.2: Information Visualization pipeline (Taken from (Malyanov et al. 2013))

The visualization process in Malyanov et al. pipeline has five stages: *filtering* (F), *Visual mapping* (M), *Layout* (L), *Layer fusion* (>), and *Rendering* (R). Each stage is a family of functions that process parallel pipelines. The input data flows through the pipeline being transformed in various ways until it generates the output images.

Filtering stage: At this stage, we have to decide which are the data's important aspects, or features, we are interested in. In most cases the imported data is not one-to-one with the aspects we want to get insight into. We must distill our raw data sets into more appropriate representations, also called enriched datasets, which encode our features of interest. This process is called *data filtering* or *data enriching*. On the one hand, data is filtered to extract relevant information. On the other hand, data is enriched with higher-level information that supports a given task. For example, medical specialists are usually interested in seeing only specific anatomical structures related to a certain condition, which are a subset of the entire dataset they obtain from scanning devices.

Mapping stage: The mapping function transforms abstract streams to visual streams. The filtering operation produces an enriched dataset that should directly represent the features of interest for a specific exploration task. Once we have this representation, we must map it to the visual domain. We do this by associating elements of the visual domain with the data elements presented in the enriched dataset. The visual domain is a multidimensional space whose axes, or dimensions are those elements we perceive as quasi-independent visual attributes, such as *shape*, *position*, *size*, *color*, *texture*, *illumination*, and *motion*.

Typically, a visual feature is a colored, shaded, textured, or animated 2D or 3D shape. Data mapping is probably the operation in the visualization pipeline that is most characteristic for the visualization process as it influences the resulting image more than any other step.

Layout stage: The layout function selects and sets appropriate parameters from abstract streams based on semantics or domain knowledge. For 2D graphics these parameters include (X,Y) position of a visual element. For 3D graphics these include (X,Y,Z,view) where *view* is a virtual camera (Malyanov et al. 2013).

Layer fusion Stage: In this stage, a function is used to fuse multiple layouts into an output view based on abstract.

Rendering stage: The rendering operation is the final step of the visualization process. Rendering takes the 2D/3D scene created by the mapping operation, together with several user-specified viewing parameters such as the viewpoint and lighting, and renders it to produce the desired images. In typical visualization applications, viewing parameters are considered part of the rendering operation. This allows users to interactively navigate and examine the rendered result of a given visualization.

2.1.3. Jansen and Dragicevic Visualization pipeline stages

Jansen and Dragicevic proposed an information visualization pipeline that shares many similarities with the above-mentioned previous models, but has been extended to better capture non-conventional operations. In fact, they focused more on the user interactions with the graphical representation and explained that a user could get multiple perspectives from a representation and combined them to form a mental visual model of what they saw in order to get insight of the raw data (Jansen and Dragicevic, 2013). Figure 2.3 shows the infovis pipeline as extended by Jansen and Dragicevic.

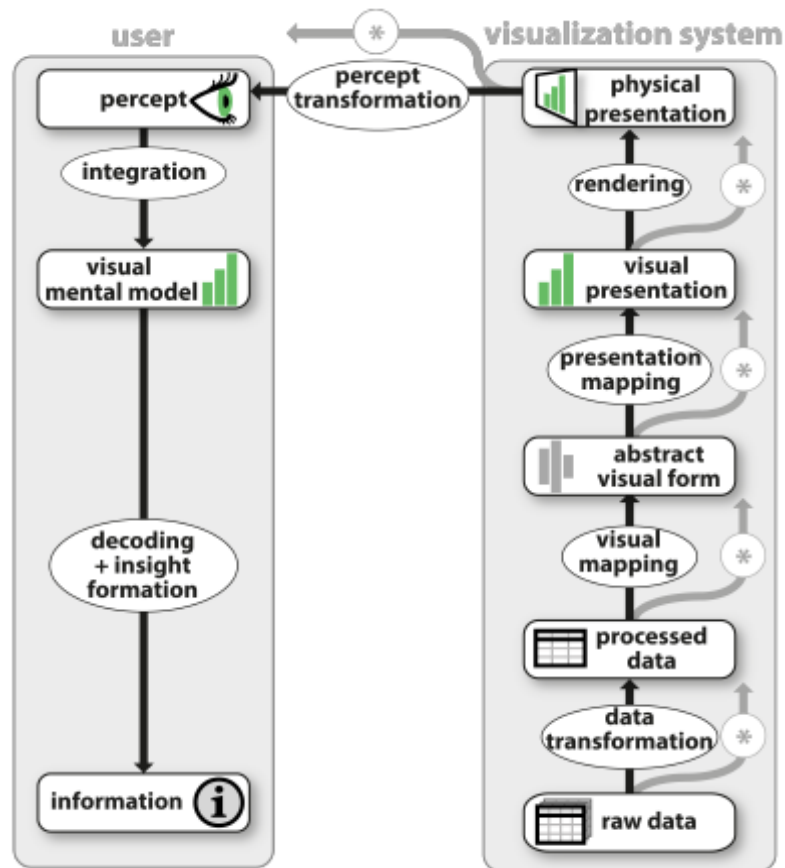


Figure 2.3: Information Visualization pipeline (Taken from (Jansen and Dragicevic, 2013))

2.1.4. Liu et al. pipeline Visualization stages

In January 2014, Liu et al., proposed an information visualization pipeline made of five (5) main modules or stages: *data transformation and analysis, filtering, mapping, rendering, and UI controls* (Liu et al., 2014). The input is a collection of data that can be structured or unstructured.

Stage 1: data transformation and analysis

The data transformation and analysis module is tasked with extracting structured data from the input data. With the structured data, this module then removes noise by applying a smoothing filter, interpolating missing values, or correcting erroneous measurements.

Stage 2: *filtering*

The output of stage 1 is then sent to the filtering module, which automatically or semi-automatically selects data portions to be visualized (focus data).

Stage 3: *mapping*

Given the results produced by the filtering module, the mapping module maps the focus data to geometric primitives (e.g.: points, lines) and their attributes (e.g.: color, position, size).

Stage 4: *rendering*

With the rendering module, geometric data are transformed into image data. Users can then interact with the generated image data.

Stage 5: *UI controls or User Interface Control*

Users can interact with the image rendered through various UI controls to explore and understand the data from multiple perspectives.

Figure 2.4 shows the pipeline presented by Liu et al.

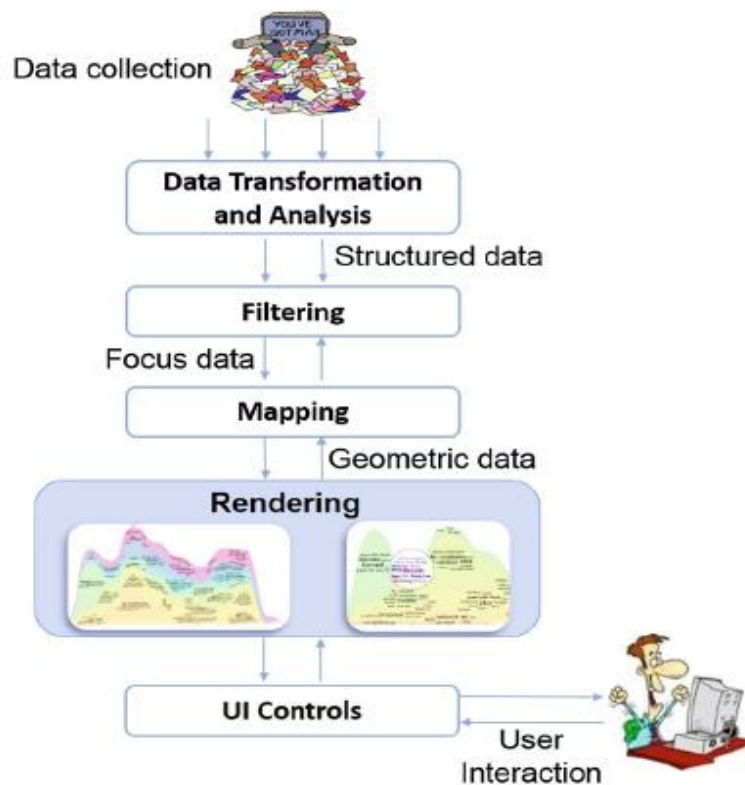


Figure 2.4: InfoVis pipeline (Taken from (Liu et al., 2014))

All the four pipelines mentioned above have integrated at different scales the user interactivity on the visualization. In addition, while Fry draws attention on the inter-dependencies between the various elements of the visual data analysis process and the recursion between various steps, (Jansen and Dragicevic, 2013) and (Liu et al., 2014) emphasize on the possibilities for the users to interact with the visualizations. (Jansen and Dragicevic, 2013) pipeline provides more details on how the user can form a mental visual model from visualizations whereas (Liu et al., 2014) pipeline provides the users with the best way to interact with the visualization through various user interface controls.

Therefore, when combined together, (Jansen and Dragicevic, 2013) and (Liu et al., 2014) pipelines integrate a very good approach of user interactivity which enables the user not only with the possibility to get multiple perspectives from visualizations in order to form visual mental model but also to efficiently interact with the visualizations. Both pipelines better align with the context of this thesis which aims at improving the user interactions on the visualizations based on structural analysis of texts.

2.2. Text visualization techniques

In 2002, Paley proposed *TextArc* to visualize the distribution of terms in a text with a circular representation. All the terms are shown at once in order to enable the analysis of co-occurrence relationships between them (Paley, 2002).

Wattenberg and Viégas in 2008, introduced *WordTree* a text visualization technique to summarize text data via a syntax tree in which sentences are aggregated by their sharing words and split into branches at a place where the corresponding words in the sentences are divergent (Wattenberg and Viégas, 2008).

In 2009, *Phrase Net* a node-link interactive visualization that extracts relationships between terms was introduced by (Van Ham et al, 2009). *PhraseNet* employs a node-link diagram, in which graph nodes are keywords and links represent relationships among them and are determined by a regular expression indicated by users. For the example shown in Figure 2.5 (a) and (b), a user can select a predefined regular expression from a list to extract a relationship such as “X and / is / of Y” or by imputing the regular expression by their own.

In this same year 2009, (Viégas et al., 2009) proposed *Wordle* which displays terms of text according to their frequency of occurrence. If the term is more frequent in the text, it appears bigger in size in the *Wordle* visualization.

Two years later in 2011, a group-in-a-box (GIB) layout was introduced by (Rodrigues et al., 2011). GIB is a network graph visualization used for representing clusters (group of words) with emphasis on the edges within and between clusters. The GIB layout visualizes sub graphs within a graph using a Treemap space filling technique and layout algorithms for optimizing the layout of sub-graphs within the space, such that related sub-graphs are placed together spatially. Figure 2.6 shows a sample group-in-a-box visualization.

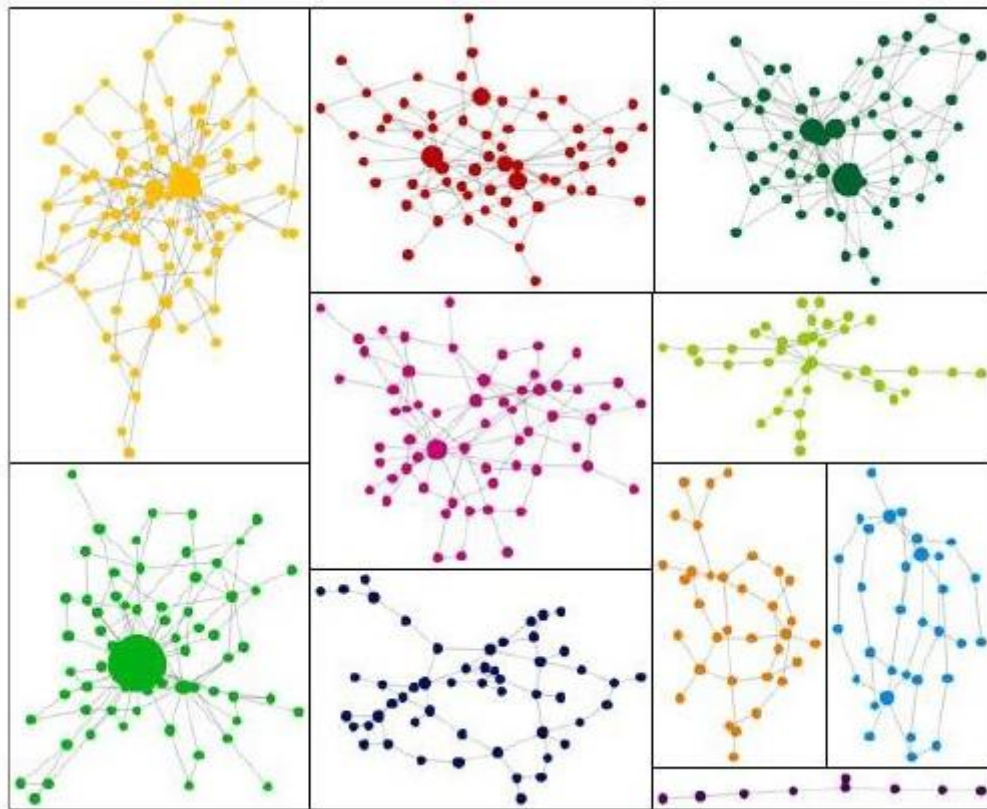


Figure 2.6: A sample GIB layout from (Rodrigues et al., 2011). The layout visualizes clusters distributed in a treemap structure where the partitions are based on the size of the clusters.

Curtotti et al., in 2013 developed a prototype of software tools demonstrating novel applications of visualizations for representation and analysis of definition networks within contracts. This software tool enables a user to input text via a web interface and presents the user with a number of alternative visualizations of definitions in a contract such as: *single layer pop-up hyper-linking of defined*

terms, representation of frequency and other information. It also provides application of 'word cloud' techniques to enable the rapid and global visualization of a defined term as well as some metrics reflecting the semantic content of the term; multi-layer hierarchical navigation tools enabling in-situ navigation of 'definition networks' from the rule where a definition is used; visual presentations of definitions as a link and node graph; and matrix representation of definition usage within a contract.

Figure 2.7 (a) and (b) respectively show node-link graph diagram and matrix representation.

Figure 2.7 (a) shows the relationships between a definition and the defined terms it uses. The visualization provides an immediate sense of the relationship between defined terms. It intuitively represents the complexity of definition, providing an opportunity to a drafter to consider revision to reduce complexity, or to a reader to explore concepts utilized by a rule. A reader is similarly alerted to semantic relationships.

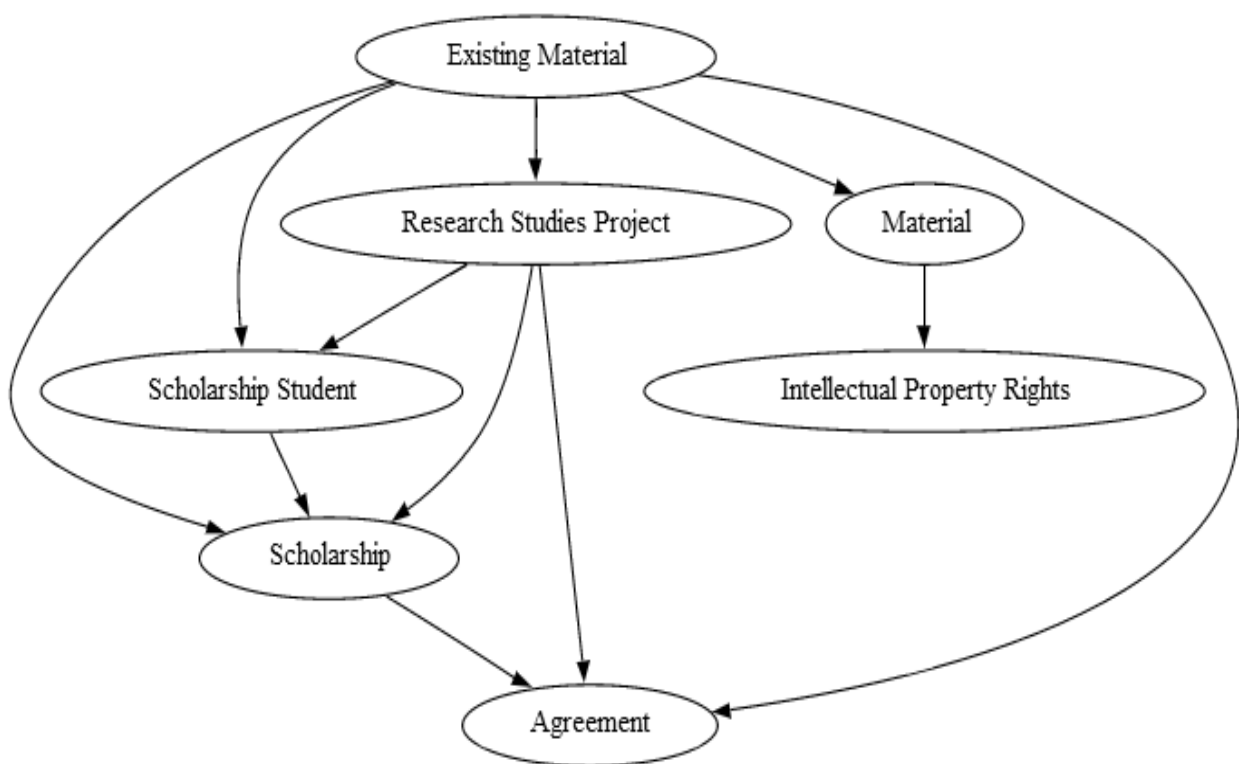


Figure 2.7(a): A node-link graph diagram showing the relationships between defined terms which have been extracted from a natural language contract.

Figure 2.7 (b) provides a representation of the relationship between clauses and definitions as a weighted bimodal adjacency matrix. Each square in the matrix represents a definition-clause relationship and the darkness of the square indicates the relative frequency with which a defined term is used in a particular clause. A column provides a visual summary of the importance of definitions used within a particular clause, while a row summarizes the use of a particular definition across the agreement.

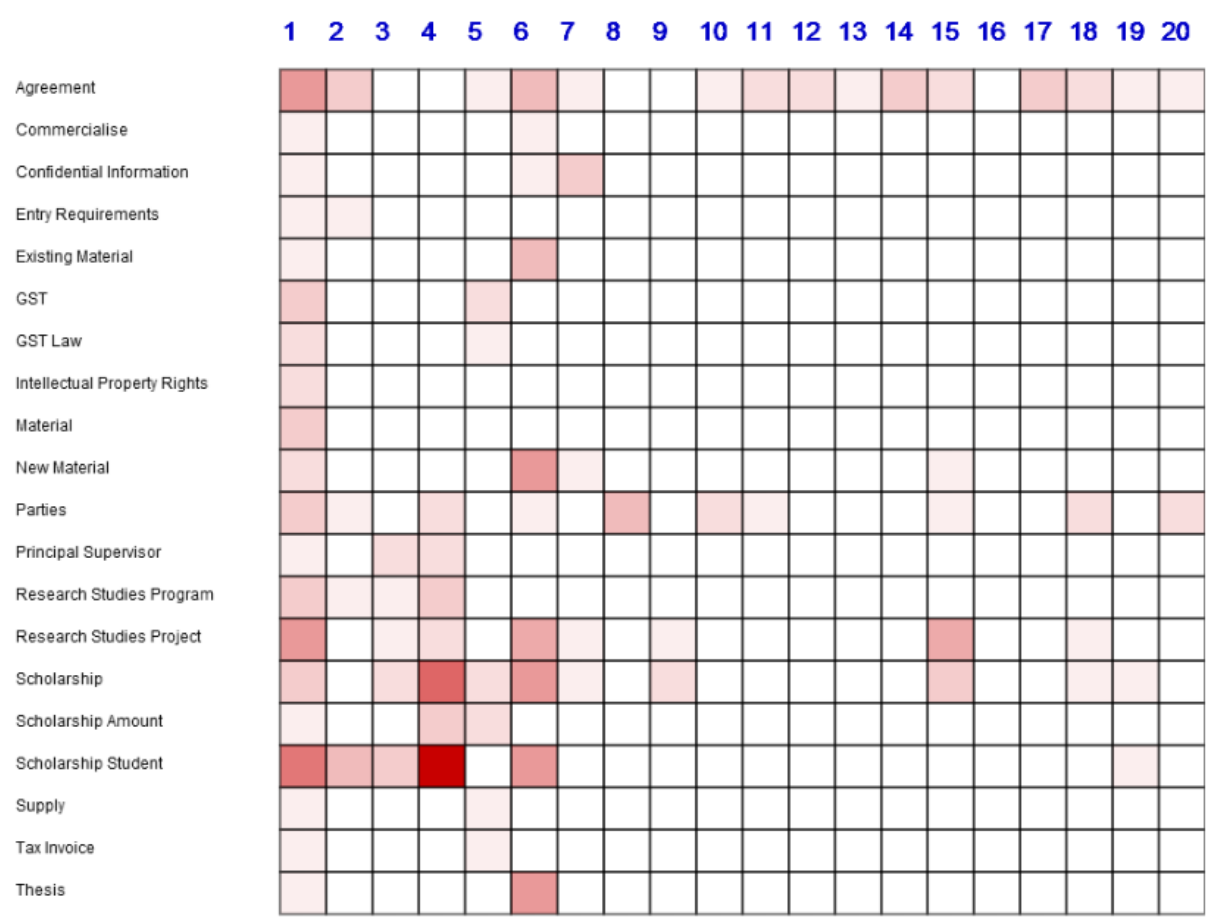


Figure 2.7 (b):A matrix representation of the relationships between definitions and clauses.

Figure 2.7: Visualization of Definition Networks in Legal Contracts through Node-link graph diagram and matrix representation

In 2014, Smith et al., presented a relationship-enriched visualization tools to help users explore topic models through word and topic correlations. They based their work on GIB layout (Rodrigues et al., 2011) and Latent Dirichlet allocation (LDA) algorithm (Blei et al., 2003). In their visualization, individual topics are represented as network graphs where nodes represent terms and

edges represent frequent term co-occurrence. A user can hover over a topic to see the related topics. To support user interactions, they have implemented an edit mode which allows among others adding words to a topic, removing words from a topic. In fact, Latent Dirichlet allocation (LDA) algorithm (Blei et al., 2003) is a common topic modelling technique which is used to classify text in a document to a particular topic. It takes a group of documents (anything that is made up of text), and returns a number of topics in order to link nodes based on the topic similarity of the documents.

In 20015, Dörk and Knight introduced a navigational approach to text visualization. They designed *WordWanderer* system a visualization technique that extends tag clouds into a navigational interface for text (Dörk and Knight, 2015). The tools supports the gradual movement between word 'context-view', which represents the words that co-occur in the neighborhoods of the selected word, and word 'comparison views', which arrange words based on their association strengths between two selected words. Figure 2.8 (a) and (b) show respectively 'context-view' and 'comparison-view' with WordWanderer.

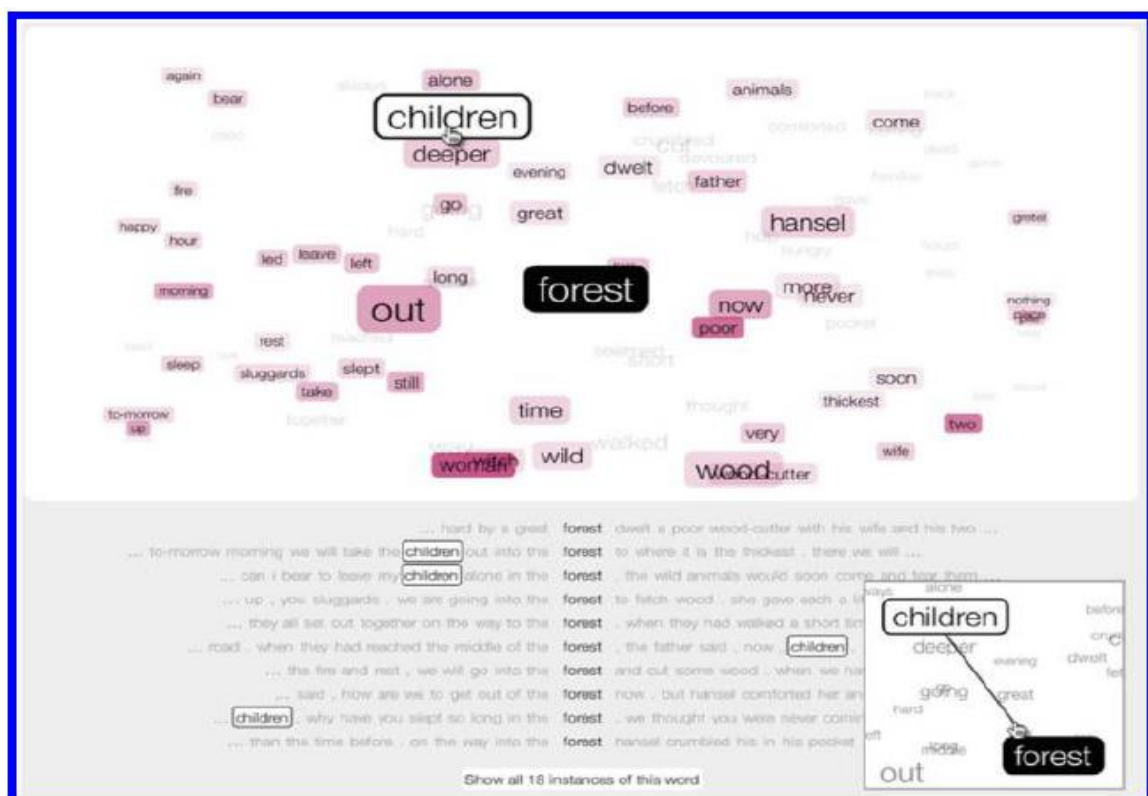


Figure 2.8(a): After choosing *forest*, the context view arranges the collocates according to their relative proximity in the text. Dragging a line between *children* and *forest* activates the comparison view

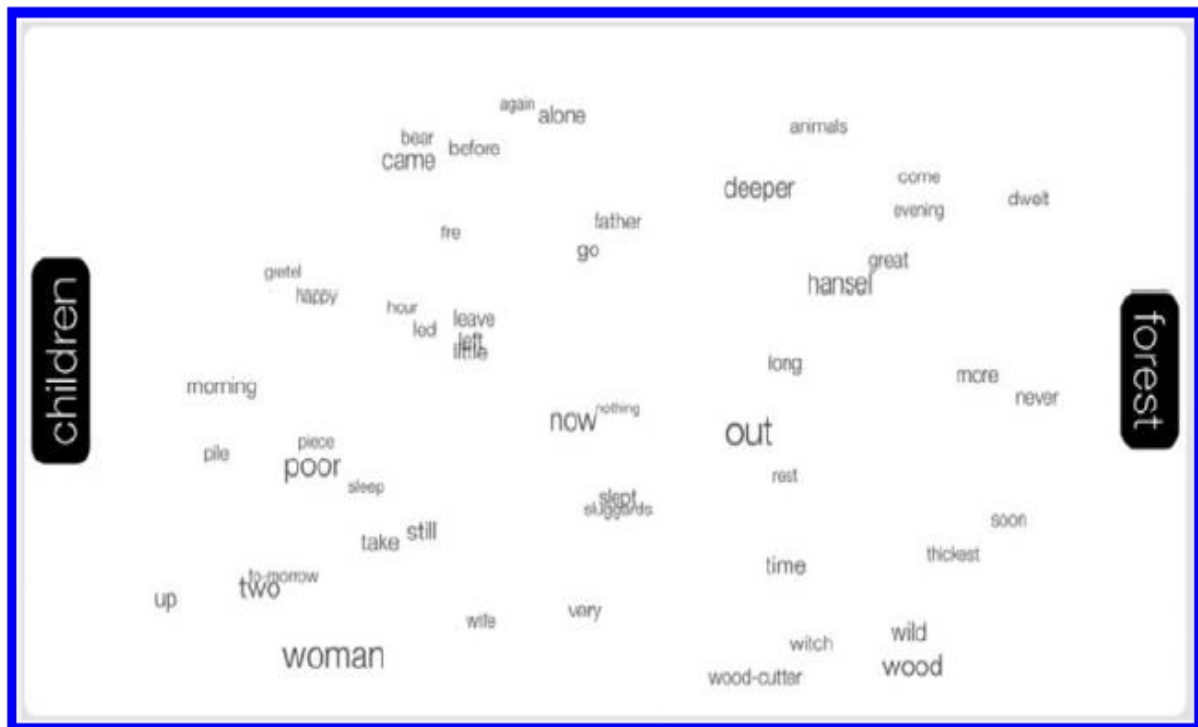


Figure 2.8(b): The comparison view arranges the collocates according to their relative strength of association to each of the two selected word

Figure 2.8: 'context-view' and 'comparison-view' with WordWanderer
(Extracted from (Dörk and Knight,2015))

Kucher and Kerren presented in 2015 an interactive visual survey of text visualization techniques that can be used for getting an overview of the field, for teaching purposes, and finding related work based on various categories defined in a survey taxonomy (Kucher et Kerren, 2015). Their work is available online as an interactive browser (Text Visualization Browser, abridged TVB¹). It currently gathers 440 techniques (last updated on June, 2019) and is considered as the most complete taxonomy. TVB offers the user a search bar and various filters like: *time filter*, *analytic task filter*, *visualization tasks filter*, *data source filter*, *data properties filter*, *domain filter*, *visualization dimensionality filter*, *visualization representation filter* and *visualization alignment filter*.

Figure 2.9 (a) shows the TVB web based user interface and Figure 2.9 (b) shows the taxonomy of text visualization techniques used in visual survey.

¹ <http://textvis.lnu.se/>

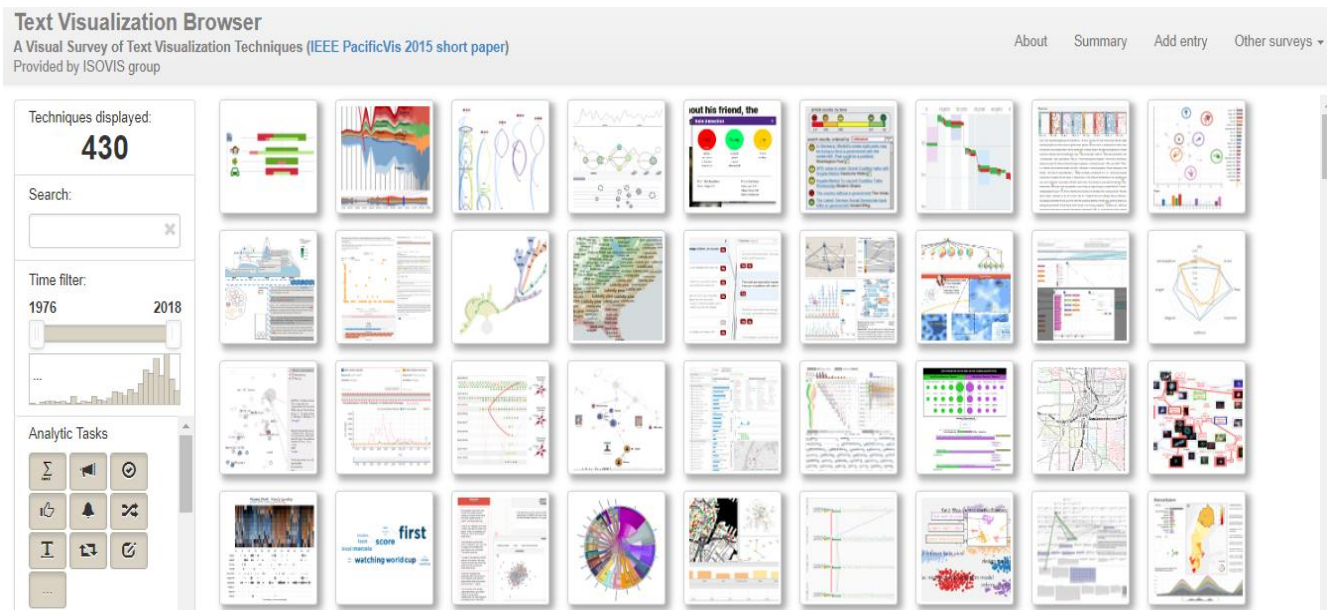


Figure 2.9 (a) Text Visualization Browser web based user interface

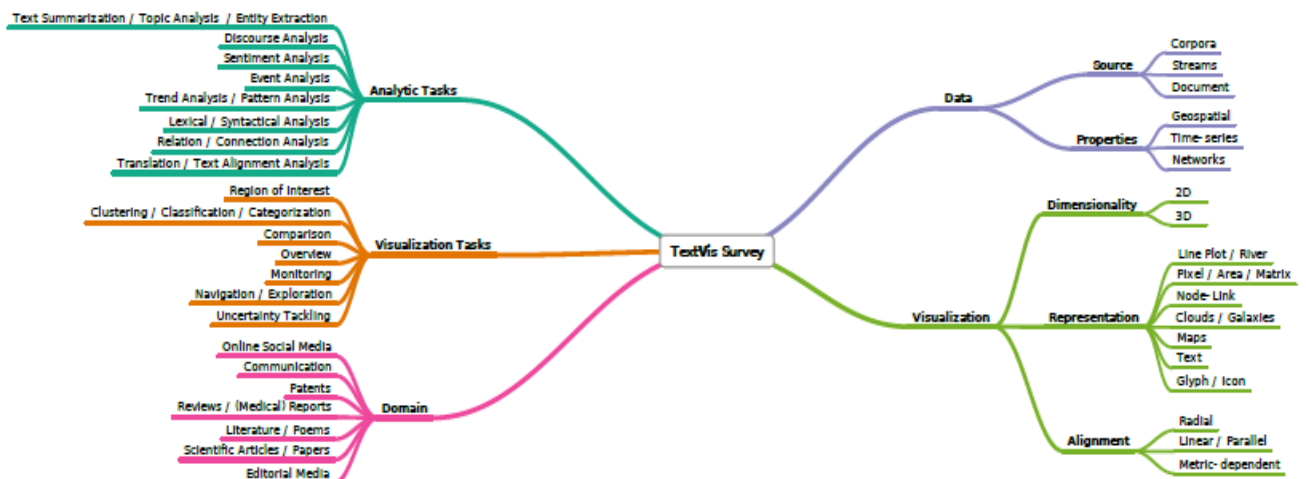


Figure 2.9 (b) Taxonomy of text visualization techniques used in visual survey
(Extracted from ((Kucher et Kerren, 2015))

According to Kucher and Kerren, text visualizations can be broadly divided into three (3) classes based on how they combine the text and the visualization. We can characterize them as *direct*, *indirect* or *hybrid visualizations*.

Direct text visualizations rely on the visual properties of the text such as size, color, location, orientation, style, value, and shape. A popular example in this class is tag cloud.

Indirect text visualizations quantify some aspect of text and visualize the numbers with a suitable technique. A popular approach is to compute term vectors or weights of each word in a document and visualize the vectors.

Hybrid text visualizations combine the text and the visualization in a meaningful way. This requires easy movement between the text, the corresponding spot in the visualization, and vice versa.

In this same year 2015, Teng et al., introduced *LinkScope* an interactive Graph analysis of Unstructured Text. LinkScope² is an open-source and web-based toolkit for performing visual interactive analysis of structured or unstructured text data, primarily through node-link data representations. The system supports three main tool types:

(1) graph building tools and filters based on underlying tabular data, (2) visual components such as automated layout algorithms and parallel statistical representations, and (3) novel interactive tools for graph manipulation and exploration.

Their research focused heavily on the interaction components, combining several variants of a novel interpolation layout methods based on mouse gestures. These methods are useful for interactive analysis of node-link graphs, particularly when they can be coupled with dynamic attribute extraction from tabular data, and work best when attribute values are represented on the graph as connected nodes. The goal of the framework is to test the utility of two novel techniques for the interactive manipulation of node-link graphs within a variety of such workflows. The first technique is based on interpolation of mouse gestures over the graph, and a second technique applies EvoCut³ conductance-based clustering (Andersen et al., 2008) and other methods prior to gesture interpolation). Both techniques are designed to work in tandem with existing graph layout algorithms, to help analysts make sense of result graphs, particularly when they become too complicated for a standalone layout algorithm.

Analysts can manipulate raw data in tabular form to construct graphs by selecting various column headers to link data entities derived from free text. This enables the analyst to construct a broad variety of node-link graphs tailored to a given analysis task. Filters can be also applied based on attribute types in the data to further refine the graph. Once a graph has been built, automated force-based algorithms can be applied to produce graph layouts. A set of

² <https://github.com/johnodonovan/LinkScope-Graph-Toolkit>

³ 'EvoCut' local clustering algorithm to find a local cluster around a target node in a smarter way,

statistical analyses tools and visualizations can be used in parallel to help the analyst understand more complicated graphs, for example by examining link distributions and clusters.

Figures 2.10(a) and (b) give details on an example workflow in *LinkScope*, showing an interpolated layout of filtered NSF⁴ data.

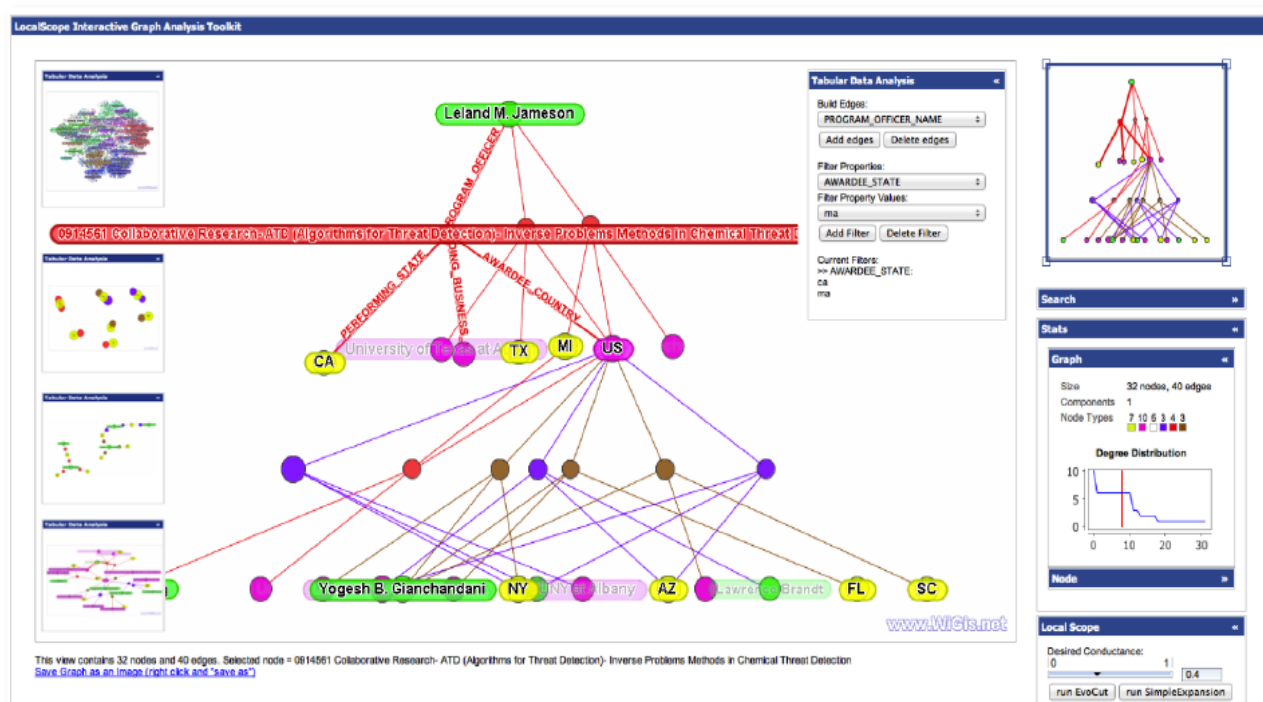


Figure 2.10 (a): Example workflow in LinkScope, showing an interpolated layout of NSF data

⁴ **National Science Foundation (NSF)**- an independent agency of the United States Government to financially support basic scientific research

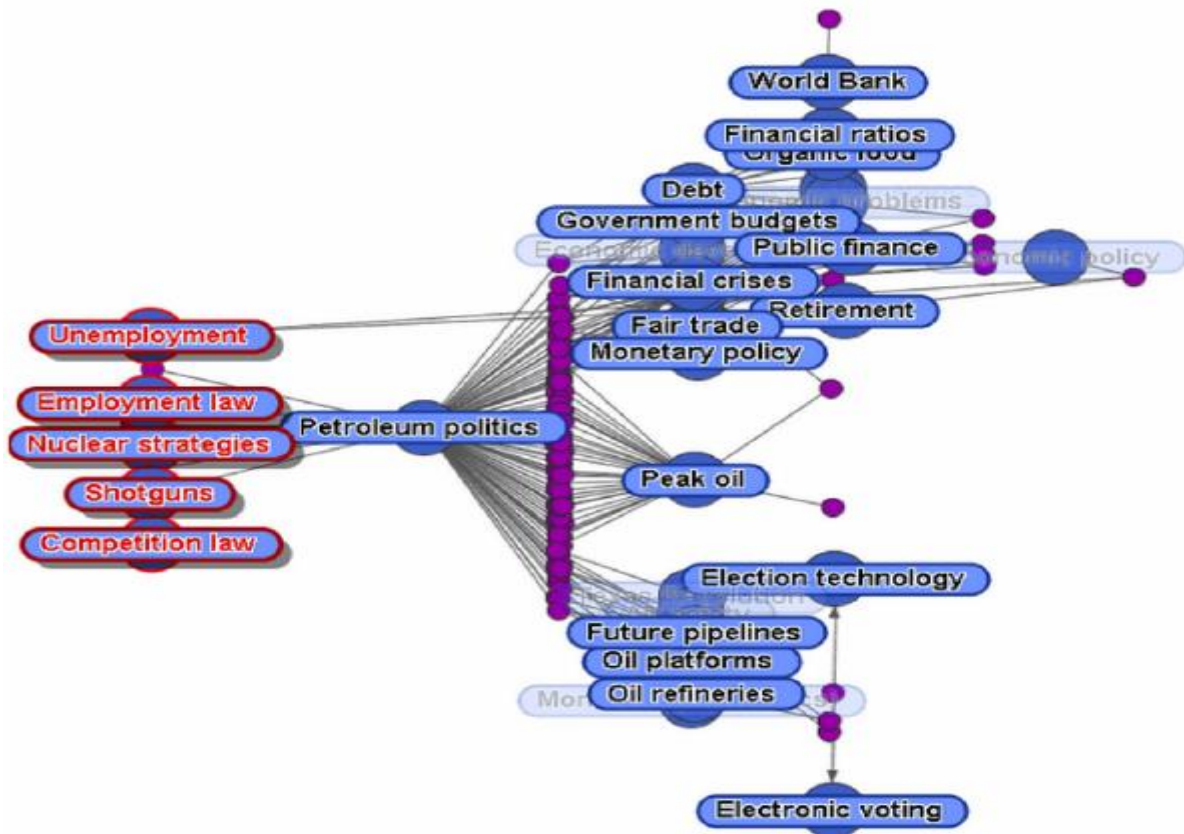


Figure 2.10 (b): Example workflow in LinkScope, showing an interpolated layout of filtered NSF data.

Figure 2.10: An interpolated layout of filtered NSF data using LinkScope.
(Extracted from (Teng et al., 2015))

In addressing the problem of interactive text visualization in the field of Humanities especially in sociolinguistic language study, Siirtola et al., introduced in 2016 a software tool called *Text Variation Explorer (TVE)* for sociolinguistic language study (Siirtola et al., 2016). TVE is based on interactive visualization with a direct manipulation user interface, and aimed for exploratory corpus linguistics. TVE has proven to be useful in supporting the study of language variation and change in its social context or sociolinguistics. It is a tool for quick exploration of text structure, complexity and variation.

In their work on *Text visualization Techniques*, Nan and Cui stated that existing techniques to show documents content at the word level are developed to address three general problems: (1) how to represent the words esthetically in

a visual form to clearly depict the content of the text; (2) how to summarize and represent the semantic relationships such as “A is B” and “A of B” between words in the text, and (3) how to reveal word-level patterns such as repetitions and co-occurrences.

Among these techniques are *TagCloud* and *Wordle* which are the state-of-the-art techniques that produce aesthetic word packing results by precisely calculating the word boundary and randomly inserting the word into empty spaces guiding by a spiral line (Nan and Cui, 2016).

Although widely used, *TagClouds* fail to reveal the relationships between words. Therefore, many tree or graph based visualization techniques were introduced such as WordTree (Wattenberg and Viégas, 2008) and PhraseNet (Figure. 2.5) which is explained in one of the paragraphs above.

2.3. Brief description of Structural analysis

The structural analysis method is one of the methods of content analysis developed by Jean-Pierre Hiernaux in 1977 (HIERNAUX, 1977). It is a qualitative method to describe and analyse content which takes into account the semantic nature of the data. The advantage of this approach lies in the fact that the researcher (the analyst) gives him (her)-self more opportunity to catch new elements in the text in relation to his prior knowledge of the subject. *It is all about identifying a structure in a discourse, meaning a structured set of elements that have no sense in themselves but derive their meaning from their relationship with other elements.*

A structure is a combination of elements. One element alone has no meaning. At the most basic level, the structure assumes that at least two elements have a relationship with each other (REMY et al., 1990). The Structural analysis is therefore a technique based on a **binarity postulate** according to which any speech can be reconstructed by means of a "toolbox" which minimal instrument is a relation between **two terms**. This fundamental relationship is called the **disjunction relation** ((REMY et al.,1990) & (Piret et al., 1996)). For example, the notion of "**good**" is only relevant in the structural analysis if the notion of "**evil**" is also present in the text. Both derive their meaning from the "**good and bad opposition**" denoted **good / bad**.

A **term** is thus constituted by the relation of disjunction with another term which is its inverse provided that the two terms belong to a common totality. If the two terms can be compared and then distinguished it is because their opposition is on the same axis. For example the opposition **man / woman** postulates a common point, the human race. The common denominator of both terms is called the "**semantic axis**" (REMY et al., 1990) & (Piret et al., 1996). Let consider the following example:

Example 1: "Human relations are good, but sometimes they are bad"

In this example, we will say in the terminology of the structural analysis that the two terms "good" and "bad" are united by a relation of disjunction and we note conventionally: **good / bad**. The semantic axis in the example is "**Human relations**". The whole description is represented by figure 2.11(a) as follows:

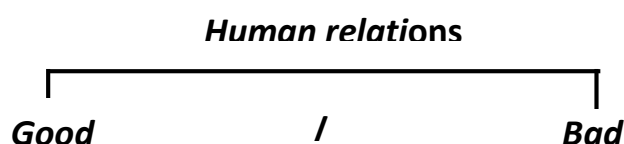


Figure 2.11 (a): Representation of disjunction (with mentioned semantic axe)

Sometimes it happens that the semantic axis to which the disjunction refers is not mentioned in the text. In this case we said the axis is "**not-manifested**" and the decoder (the analyst) then formulates a hypothesis as to the probable axis and places it between parentheses. Let us consider example 2.

Example 2: « Cryptography is easy to understand, Programming is so difficult »

In the text the disjunction relation is **Cryptography / Programming** but the semantic axe is not mentioned. We can therefore give a probable semantic axis and enclosed it with parentheses, like **(Computer Science course)**. Figure 2.11(b) gives us the graphical representation.

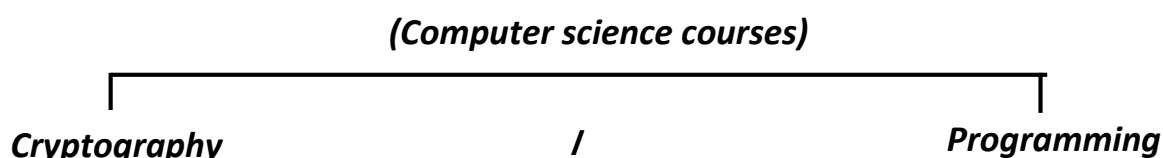


Figure 2.11 (b): Representation of disjunction (with "**not-manifested**" semantic axe)

Chapter 3: EVOQ

This chapter is organized in five (5) sections and presents the new version of the software EVOQ we have designed. Section 1 presents the context of the development of EVOQ and its goals. Section 2 describes the methodology we have used for the development as well as the implemented technologies. In Section 3, the new software architecture of EVOQ as well as a brief description of its main functionalities are presented. Section 4, presents and compares the two different (2) ways the analyst can build the nodes links graph with EVOQ. And finally section 5 analyses the different user interactions in EVOQ and compares them to a comprehensive list of visualizations interaction techniques found in the current state of art.

3.1. Context and Goals of development

EVOQ is a software that attempts to implement the structural analysis technique to help text analysts understand the semantic meaning of a text.

The first version of EVOQ is a cognitive mapping software and was developed in 2004 by (Wallemacq et al., 2004) in the context of EVOQ project. The programming language used for the development was Java 1.4. The objective of this tool is to use structural analysis technique to analyze a text by deriving the semantic fields surrounding its author. The semantic fields indeed represent the author's perceptions and form the network of the evocations (*i.e. the different terms evoked by a term T*) revolving around the terms as a term is well defined by its relationships with other terms.

The second version of EVOQ was developed in 2018 by (Clarinval et al., 2018) in the context of the EFFaTA-MeM research project at both the Faculty of Economics, Social and Management Sciences and the faculty of Computer Science of the University of Namur (Belgium). The goal of this current version of EVOQ is to improve the previously version by exploring additional visualization techniques. In fact, EFFaTA-MeM research project focuses on techniques that exploit the whole inner richness of a text and make it fully computer-process able. With this tool in hand, the EFFaTA-MeM research hopes to gather new insight about what makes a text analysis tool genuinely helpful for structural analysis.

The goal of the present work is to improve the current version of EVOQ (Clarinal et al., 2018) by enabling the text analyst to have more interactions with the visualizations. More specifically, our work aims at implementing a new attractive interface which will enable text analysts to monitor both the text and visualizations in the same window; implementing an algorithm to enhance analysts' interactions on the visualizations such as *linking and brushing, adding and deleting terms or relations (disjunction and association), editing terms, changing terms color etc...*all directly and interactively from the matrix or the nodes-links diagram; and finally helping analysts construct the relational graph.

3.2. Development Methodology and technologies

3.2.1. Development Methodology

The development methodology we have used for this work is “*Agile model*” of software engineering. Indeed *agile* method is a change-driven development approach based on a mix of iterative and incremental processes and matches better with our context of development. In fact, in the context of this work we were given a paper prototype of the desired new interface as well as a non-exhaustive list of the new desirable functional requirements as starting point. We were required to analysis them in line with the current state of the art and to propose a better solution which would improve the user interactions on the visualizations in EVOQ. Therefore, we were not given any rigorous specifications or a well-organized planning to follow step by step.

Our methodology can be summarized in the four following steps.

Step 1: Familiarization with EVOQ

Before any design and implementation, we took a couple of weeks to familiarize with the existing software EVOQ and the key concepts of structural analysis technique. The familiarization with the structural analysis technique helped us to have a good understanding of the roles of a text analyst and how structural analysis technique could be implemented through computer programs.

The familiarization with EVOQ provided us with a complete understanding of the existing functionalities as well as the technologies used to implement them.

Step 2: Analysis of the new prototype and requirements

We held many work sessions with our supervisors in order to elucidate the new desirable prototype and the functional requirements and to prioritize the requirements.

Step 3: Proposal of the new solution

Exploring the existing state of the art in the fields of text visualization and interactivity has helped us understand current visualization techniques and different features included in user interactions. This enabled us to propose a new architecture for EVOQ as well as an interactive construction algorithm which could improve user interactions on the visualizations in EVOQ.

Step 4: Design and implementation

EVOQ has a modular design working with four (4) distinct interoperating modules. The new version of EVOQ we have developed uses on the same modules found in the current EVOQ (Clarinval et al., 2018) which are presented as follow.

Text Module: The text module contains the text under analysis which can be either uploaded from an existing EVOQ project (*.evoq*), or from a text file (*.txt*) or typed.

Dictionary module: The dictionary module contains the semantic dictionary which is the set of terms and relations between these terms. A semantic dictionary is bound to a given context.

Matrix Diagram: it is a two dimensional (2D) matrix representation representing terms and the relations between them either *disjunction* or *association*.

Node-link diagram: also called node-links or relations graph, it is a dynamic 2D node-link diagram where the nodes are the terms and the edges or links are the relations between these terms. The visualization in this module is interactive. The user can drag, fix and remove terms or edges. The module also offers a feature allowing users to show the related terms of a chosen term. Two terms linked by a disjunction relation will tend to repel each other whereas two terms linked with conjunction will attract each other.

As far as the implementation is concerned we have used web technologies which are explained in paragraph 3.2.2. More information is given on the design and the implementation in section 3 which deals with the architecture of the software and its functionalities.

3.2.2. Technologies used in EVOQ

The recent version of EVOQ was developed with web technologies mainly HTML (Hypertext Markup Language), JavaScript, CSS (Cascading Style Sheet) and D3.js⁵ library. D3.js (Data-Driven Documents) is one of the most popular and extensive JavaScript data visualization library. It is built for manipulating documents based on data and bring data to life using HTML, SVG, and CSS. However D3.js is a bit more of a lower-level library compared to other strictly charting solutions, so it requires more boilerplate code to get similar results (Sonalake, 2016).

In the present work, we have used the same web technologies (HTML, JavaScript, CSS and PHP) for the implementation of codes. However, we have decided to use Viz.js library for the implementation of the visualization algorithm which enables the analyst to graphically construct the nodes-links visualization.

During our research we found out more than eleven (11) popular JavaScript Libraries for data visualization. Examples include *D3Js*, *ChartJs*, *TreeJs*, *C3Js*, *VizJs* just to name few. This thesis does not intend to present the literature review of JavaScript data visualization libraries, therefore we won't give details about them.

We have used VizJs in our development for some important reasons including *its simplicity, efficiency, rapidity and security*. In fact, Vis.js library is a JavaScript data visualization Open source Libraries developed by [Almende B. V](#)⁶ with a dual licenses under both Apache 2.0 and MIT(Massachusetts Institute of Technology) between 2010-2017. Vis.js is designed to be easy to use, to handle large amounts of dynamic data, and to enable manipulation and interaction with data. The Library consists of the components DataSet, Timeline, Networks Graphs 2D & 3D. VisJs runs fine on almost all the updated browsers: *Chrome, Firefox, Opera*,

⁵ <https://d3js.org/>

⁶ <http://visjs.org>

Safari, IE9+, and most mobile browsers. Its web site contains documentation, downloads and live examples. The source code of vis.js is available on Github⁷.

Finally, VizJs helps to perform some operations on graph networks more easily like nodes styling, edges styling, labelling, layouts, events management, dynamic data and interfaces configuration.

3.3. Software architecture and functionalities

3.3.1. Global architecture

The global architecture of EVOQ is based on various files which have specific role. They can be organized into four (4) categories:

PHP files: We have five (5) main PHP files: *index.php*, *Evoq.php*, *Matrix.php*, *ForceGraph.php* and *GraphConstruct.php*. They are made of php and html codes. The two visualizations diagrams (*matrix and node-links*) and the interactive Graph construction diagram are in three (3) separate files. Separating files offers many advantages. Indeed when a visualization needs to be modified, there is only one file to edit. Also modifying the code of a visualization will not have bad effect on the main page code neither on the other visualizations codes.

CSS files: They are responsible for the aesthetics of the software interfaces and their components. We have used both external CSS files (*CSS codes stored in separate .css files*) and embedded styles (*CSS codes inside php files*). We used seven (7) main external CSS files all contained in a specific Style folder.

JavaScript Files: The JavaScript files comprise JavaScript functions that are called by the visualization pages and by the main page. We used two categories of JavaScript files. The first category are those written as a constructor function. Each of them represents a concept of structural analysis (*terms, relations, terms dictionary, relations dictionary*) and provides functions to interact with them such as *getters, setters, and display functions, delete functions*. The second category are JavaScript files written as a set of functions that address a cross-cutting concern such as:

- **EVOQparsing.js** to handle the parsing of JSON (*JavaScript Object Notation*) into concepts of structural analysis;
- **EVOQlang.js** to handle the language-specific concerns.

⁷ <https://github.com/almende/vis> (accessed on February 7th 2019)

- **EVOQtextProcessing.js** to handle textual data processing.
- **Lemmatizer.js** to return the base or dictionary form of a word, which is known as the lemma in order to help find words synonyms or antonyms.

These functions are called from *php* files and from the *JavaScript Object* files. They are stored in a separated Library folder for reusability and maintainability purposes.

External Library: they include D3.js, JQuery (for events handling) and viz.js for the interactive construction of the node-link graph.

The global architecture of our new version of EVOQ software is represented by Figure 3.1 below.

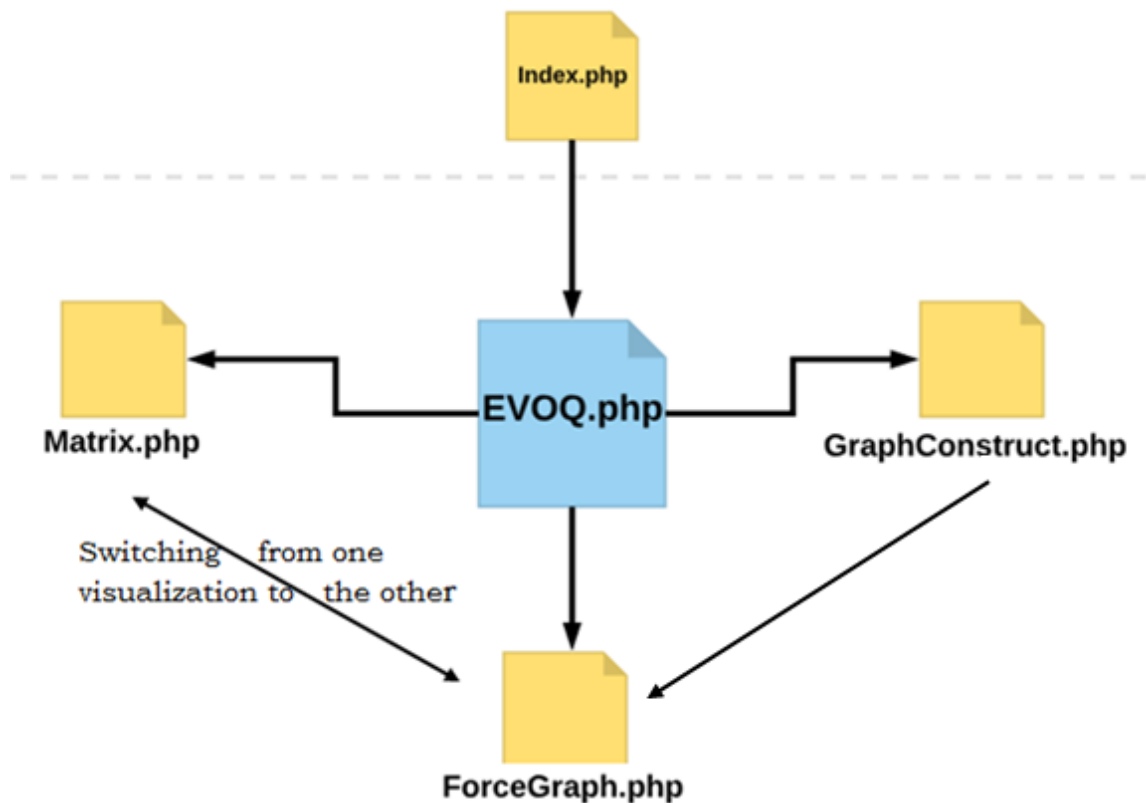


Figure 3.1: Global file architecture in EVOQ

Index.php page is the login page. After a secured login, the user directly has access to **EVOQ.php** page.

EVOQ.php page is the main page of the software which helps to have access to the other three pages. It also helps to manage the text, the terms dictionary and the relations dictionary.

Matrix.php page is called from EVOQ page to display and help interact with the matrix visualization of the structural analysis.

ForceGraph.php page is called from EVOQ page to display and help interact with the node-links visualization (relations graph) of the structural analysis.

GraphConstruct.php page is called from EVOQ page and assists text analysts to graphically build the nodes-links visualization at their ease.

Compare to (Clarinval et al., 2018), our new architecture offers additional features such as *the increase of linking and brushing* and *the construction of the relations graph visualization*. Indeed, with the new architecture, analysts can easily update a visualization (whether Matrix or Relational graph) and switch from one visualization to the other without losing any updated information. Doing so, it increases linking and brushing interaction. Analysts can also fully construct (build) the relation graph visualization rather than generating it from a pre-encoded dictionary.

Figure 3.2 below shows a simplified class diagram of the different components used in EVOQ.

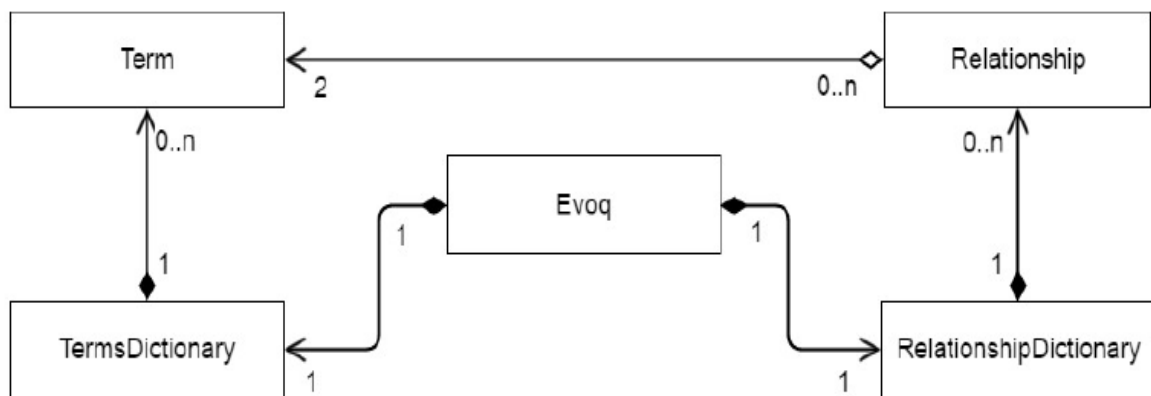


Figure 3.2: Simplified Class diagram (Taken from Clarinval et al., 2018)

From the above class diagram, we can clearly understand that:

- Each *evoq* project is composed of only one *Terms dictionary* and only one *Relationships dictionary*,
- A *Terms dictionary* is made of many *Terms*, and a *Relationship dictionary* is also made of many *Relationships*,
- A *relationship* is made of two *Terms*.

3.3.2. Main Functionalities

From the main page of EVOQ, the analyst can perform any of the following tasks:

1. **Manage project:** *import* an existing project, *save* a current project with *.evoq extension*, *change* project parameters and *close* project.
2. **Manage text:** *import a text* or *save (export)* the current text as *.txt file*
3. **Manage terms dictionary:** *display* the terms dictionary (*the list of the encoded terms*); or *import* a term dictionary (a text file with *evoqtd extension*) or *save (export)* the current term dictionary.
4. **Manage terms relationships dictionary:** *display* the relations dictionary (*the list of the encoded terms relations*); or *import* a relations dictionary (a text file with *evoqrd extension*) or *save (export)* the current relations dictionary. In the previous EVOQ designed by (Clarival et al., 2018), analysts can encode relations by using a form. With our solution, in addition, relations can be encoded in three other ways: *either via the constructed graph, or via the matrix or via the text*.
5. **Display and manage matrix visualization:** *display* the matrix diagram corresponding to the encoded relations and manage it by adding new relations, deleting relations or terms, or changing terms color.
6. **Display and manage dictionary-based graph visualization:** *display* the nodes-links graph (*also called relational graph*) corresponding to the encoded relations and manage it by adding new relations, deleting relations or nodes, or changing nodes color.
7. **Construct the relations graph:** our interactive algorithm helps analysts construct the nodes-links graph by themselves without having to encode the terms and relations first. Paragraph 3.3.4 gives details on how analysts can construct the relations graph.

Details on the functionalities are given in the appendices.

3.3.3. EVOQ Main page Interface

The present main page of EVOQ is composed of two workspaces with the following elements:

- (1) **A menu:** to manage project, text, terms and terms relations;
- (2) **A text input area:** where the text under analysis is uploaded or typed;
- (3) **Research bar :** which is used to research terms in the text;
- (4) **Project title input:** where we can rename the project;
- (5) **Matrix visualization tab:** to display and interact with the matrix diagram corresponding to the encoded relations dictionary;
- (6) **Graph visualization tab:** to display and interact with the relations graph corresponding to the encoded relations dictionary;
- (7) **Graph construction tab:** to help analysts to graphically construct the relations graph without having to encode relations dictionary first. This indeed increases the analyst interaction on the visualization as he/she has more control over the visualization.

Figures 3.3 (a) gives an overview of the main page interface.

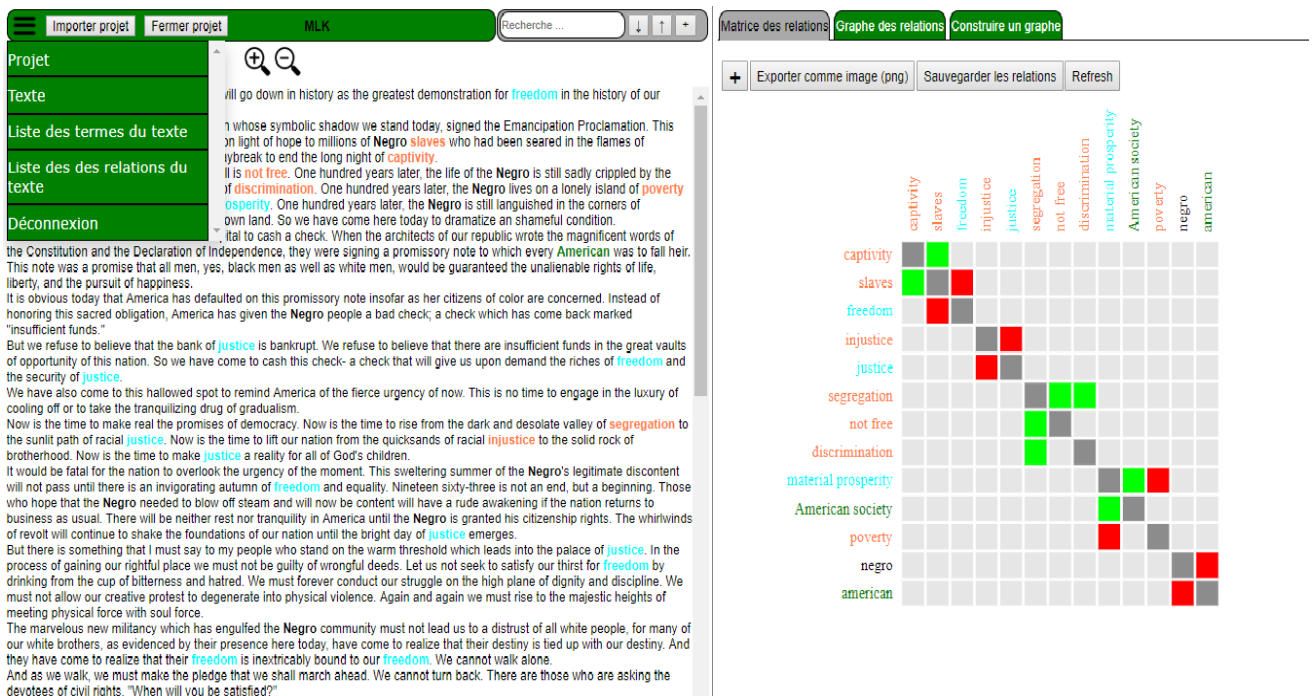


Figure 3.3 (a): Main page interface of EVOQ showing the different menu

User's interactions on the matrix include: *edit a term, change a term color, delete a term and add new relations via the matrix.*

The user can also interact with a word (term) from the text by directly double-clicking on the word and *add it to the dictionary, delete it from the dictionary, change the term color, create a relation with the term.*

Figure 3.3 (b) shows user interactions with the matrix term.

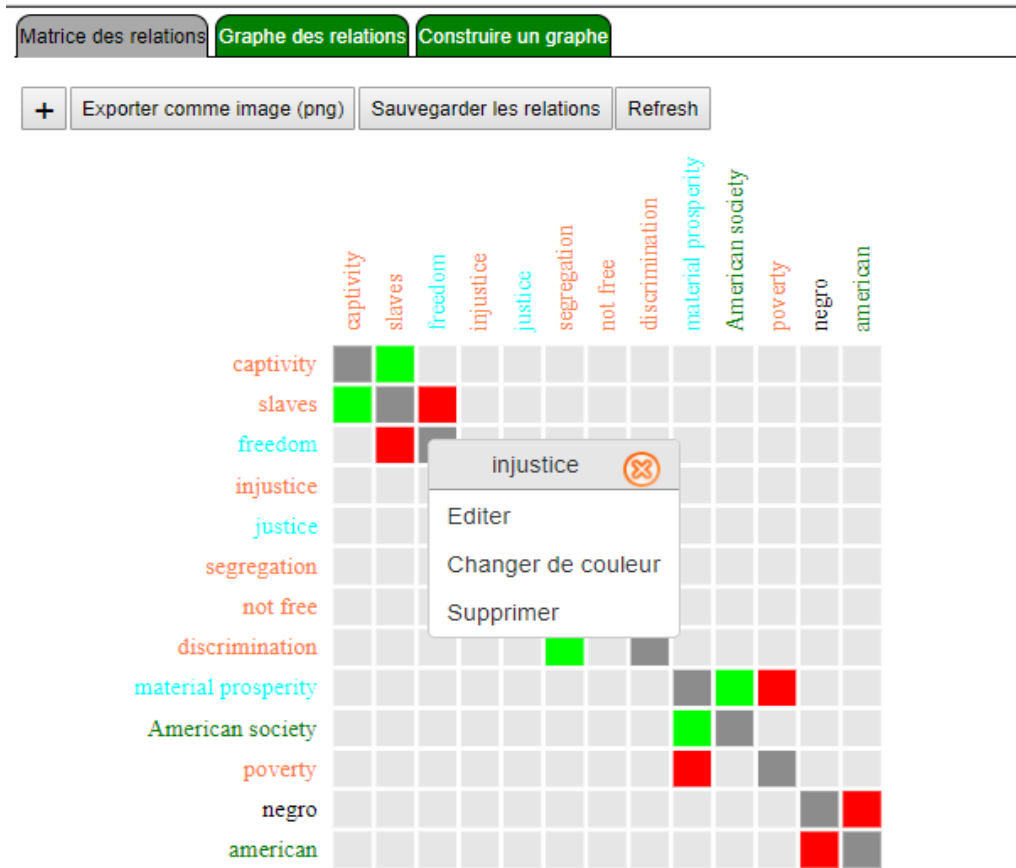


Figure 3.3 (b): Interactions with matrix terms

3.3.4. Two ways of building the nodes-links graph

Also called relations graph, the node-link diagram is a bi-dimensional graphical representation of the relations between terms. The *terms* are the *nodes* and *relations between terms* are the *links* connecting *nodes*. We used a colored disk labelled with the corresponding term to represent a node. A link (*also called edge*) is represented by a colored line drawn from a node to another. The color of the line depends on the type of the relationship between the terms: a *green color is used for an association relationship* whereas a *red color for a disjunction relationship*. These colors were chosen because they remind respectively *association* and *opposition*.

With our new solution, analysts can build the nodes-links graph in two different ways which we respectively named: (1) *Dictionary-Based Nodes-links graph* and (2) *Construction-Based nodes-links graph*.

(1) Dictionary-Based Nodes-links Graph

Here, the graph visualization is built based on the pre-encoded relation dictionary. It is also referred to as *dictionary based graph*.

In fact, after importing or typing the text, the analyst must encode a set of terms and relations before generating the visualization. Encoding a set of relations creates both terms and relations dictionaries. After encoding relations, the analyst can generate the relation graph corresponding to the relations he/she has encoded.

As the analyst has a critical role for the interpretation in structural analysis, the node-links visualization provides him/her with different ways to extract knowledge through several interactions such as: *drag and drop nodes, delete nodes or links, change nodes colors, add new relations, displays terms synonyms and antonyms*.

Figures 3.4 (a) and 3.4 (b) respectively show an encoded relations dictionary and the corresponding node-link graph.

✓	Terme 1	Force	Terme 2	Supprimer
✓	negro	<>	american	✗
	american	<>	negro	
✓	poverty	<>	material prosperity	✗
	material prosperity	<>	poverty	
✓	material prosperity	=	American society	✗
	American society	=	material prosperity	
✓	discrimination	=	segregation	✗
	segregation	=	discrimination	
✓	segregation	=	not free	✗
	not free	=	segregation	
✓	injustice	<>	(justice)	✗
	(justice)	<>	injustice	
✓	freedom	<>	slaves	✗
	slaves	<>	freedom	
✓	captivity	=	slaves	✗
	slaves	=	captivity	

Figure 3.4 (a): Encoded Relation Dictionary

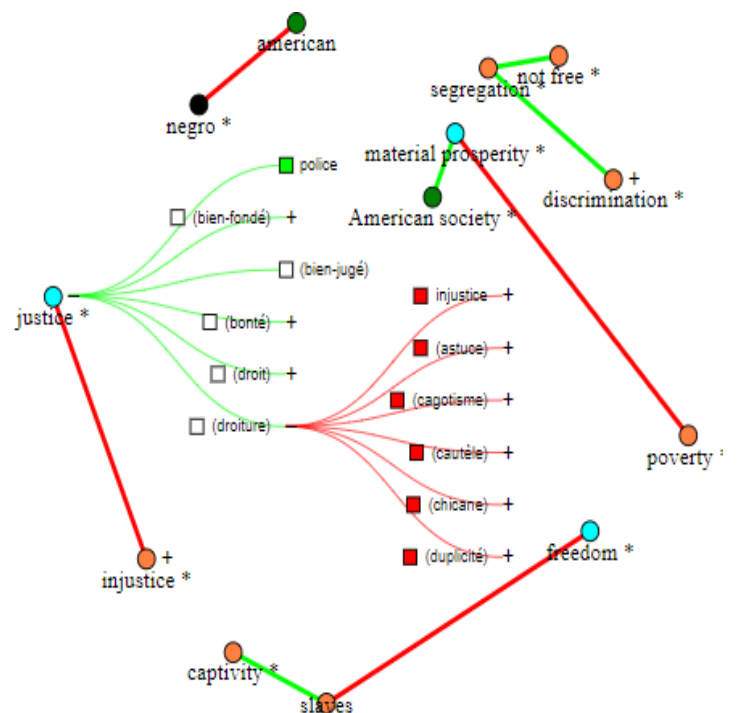


Figure 3.4 (b): Dictionary based graph with synonyms and antonyms for "Justice"

After importing (or typing) the text, analysts can directly and graphically construct the nodes-links graph at their ease. There is no need to encode terms and relations dictionary first.

The construction of graph starts from a single *disjunction relationship* between two empty nodes referred as *undefined nodes* and labelled with a question mark '?' as shown in figure 3.5.

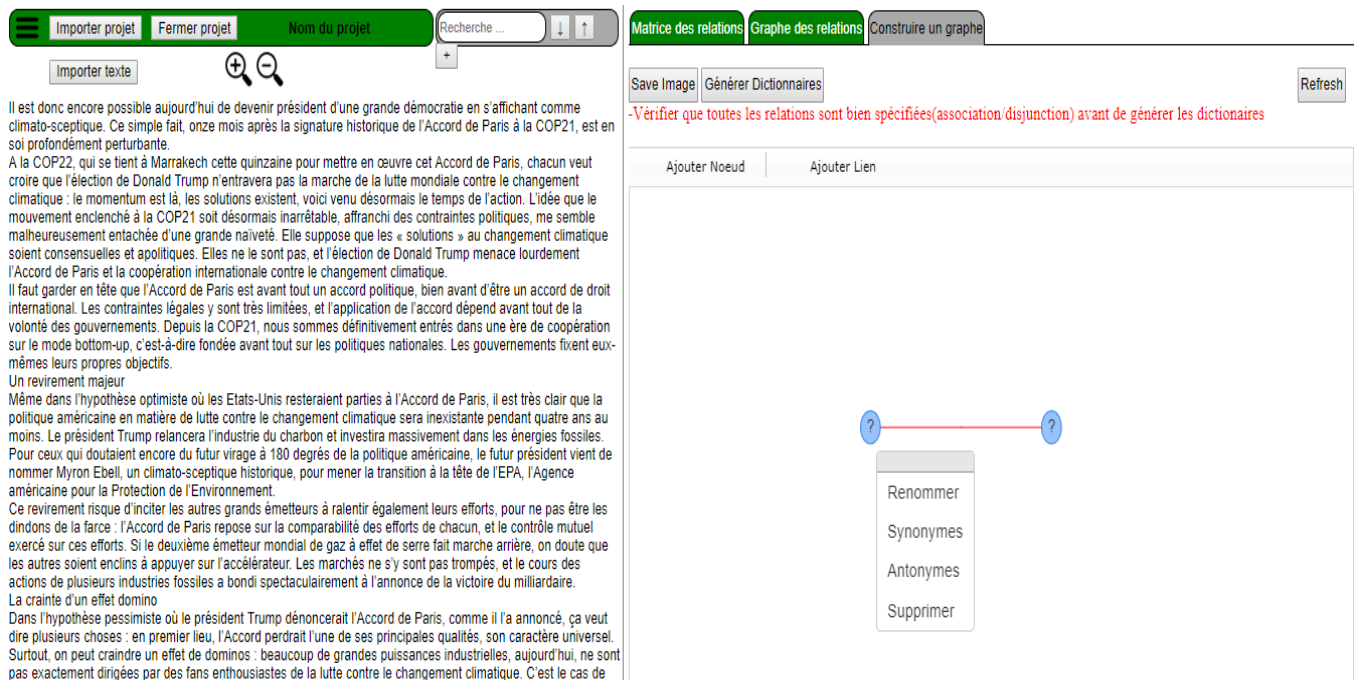


Figure 3.5: Starting interface for Construction-Based Node-link Graph

A node represents a term and an edge represents a link or a relationship between two terms. The edge is in red color (*respectively green color*) when it is a disjunction (*respectively an association*) relationship.

Analysts can interact with both nodes and edges at their ease. These interactions includes: *add, rename, delete, drag and drop nodes or edges, display nodes' synonyms or antonyms, set an edge as association or disjunction relation*. Interactions menus are accessed by double clicking on a node or an edge.

- **Generating relations from a node**

The user can generate a suggested list of *associations* or *disjunction relations* which are related to a given term (node). This suggested list of associations is made of the closest synonyms of the node term whereas the suggested list of disjunction is made of the closest antonyms of the node term.

Suggested associations (*respectively disjunctions*) are obtained by double-clicking on the concerned node.

Figure 3.6 shows the result of the graph when the user generates association relations for 'gouvernement' and disjunction relations for 'démocratie'.

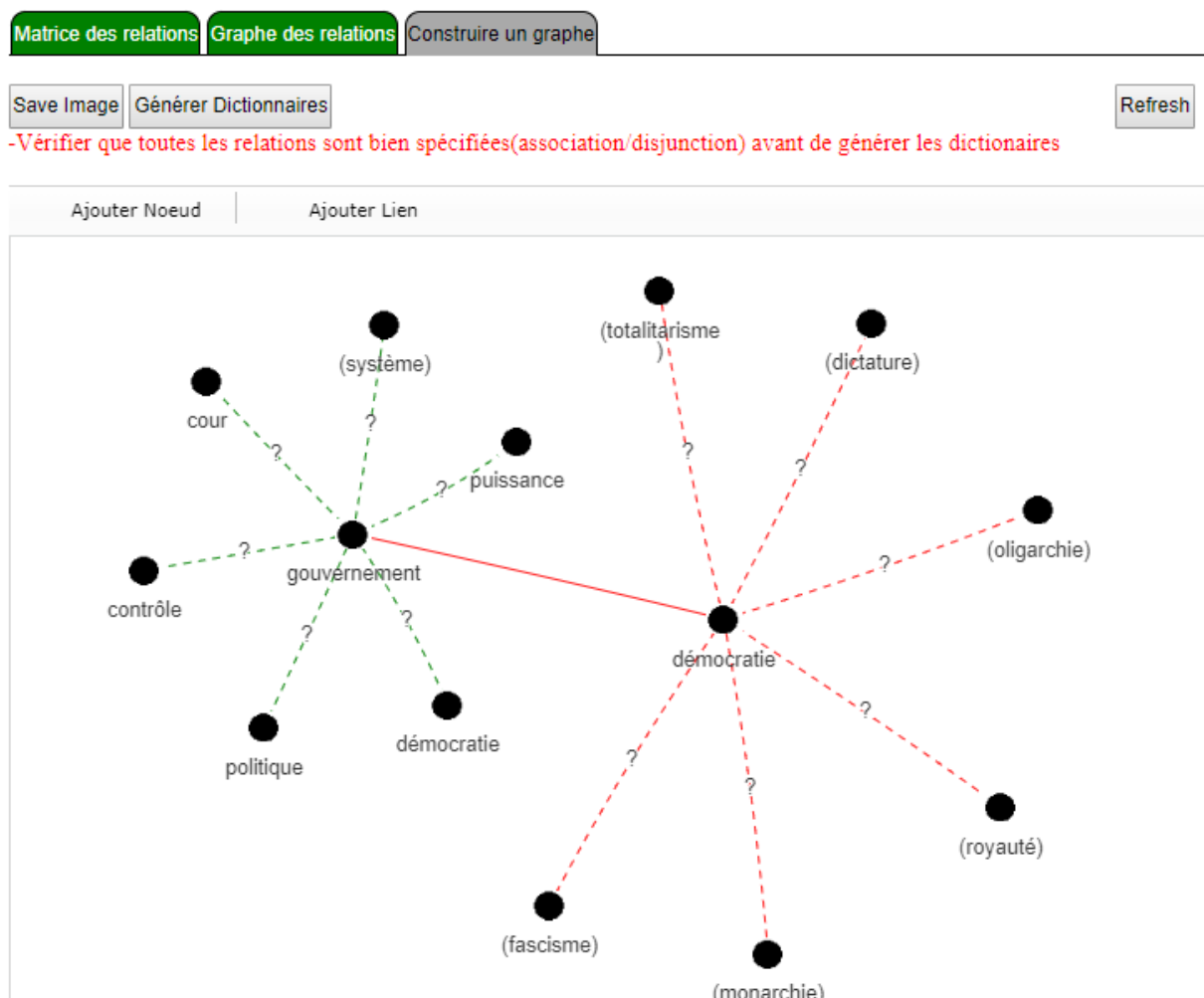


Figure 3.6: Generating Association relations for 'gouvernement ' and Disjunction relations for 'démocratie'

Terms which are not found in the text under analysis are put in brackets. Also the generated links are all labelled with a question mark '?' which indicates that

they are suggested relations which the analyst can validate: *either accept or reject them*.

- **Validation of suggested relations**

The analyst can validate a suggested relation and set it *as association or disjunction*. He/she can also simply choose to delete the suggested relation.

Figure 3.7 shows the final graph visualization after the analyst has validated the suggested relations and the interaction menu for an edge (link).

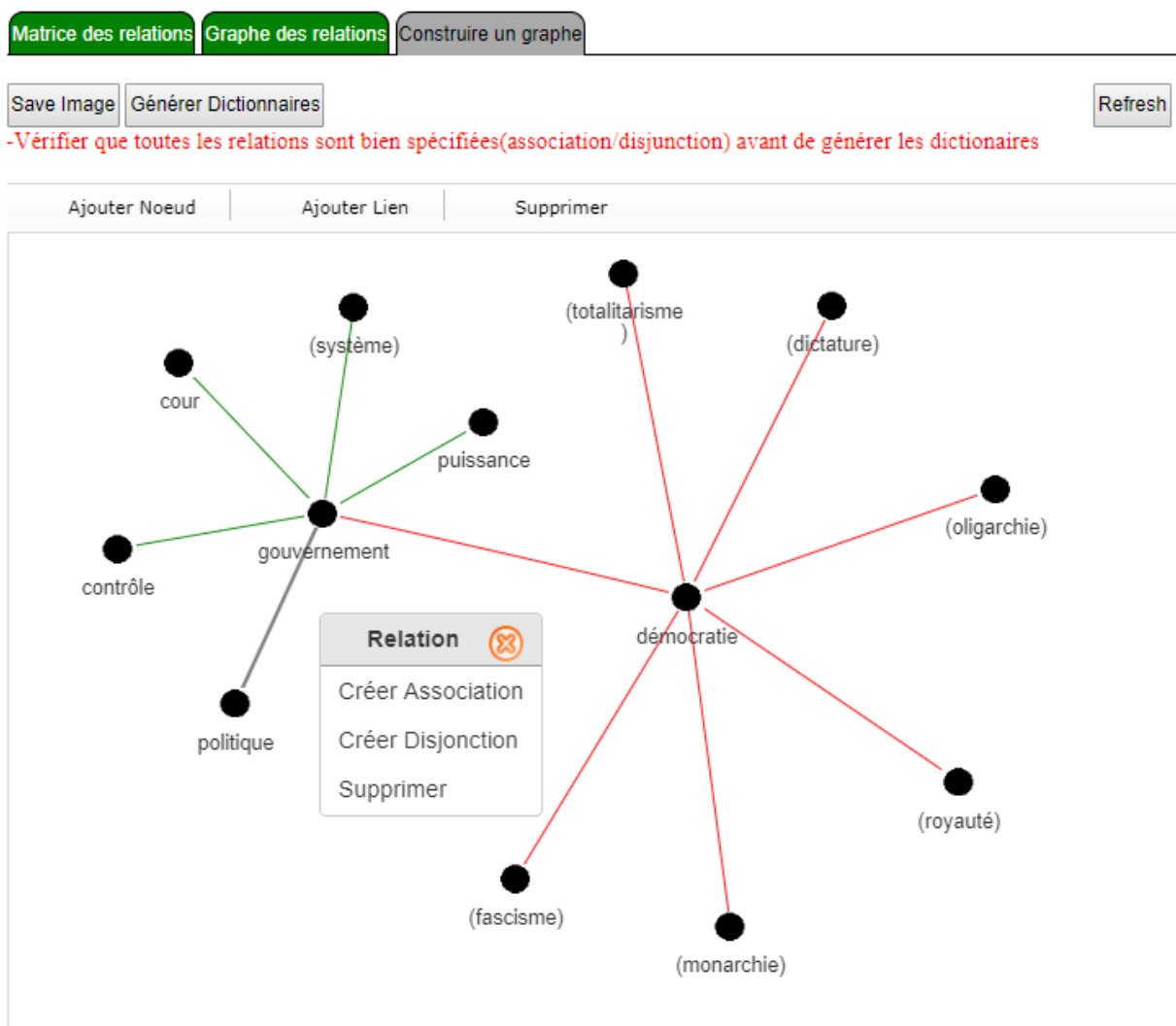


Figure 3.7: Final graph after user validation and edge-interaction menu

- **Adding new nodes and edges**

Analysts can add new nodes and edges from the menu bar buttons, drag and drop nodes and edges at anywhere on the network panel. A new added node is labelled with “new” which the analyst has to rename.

Each new edge created between two nodes has to be set either as *association* or *disjunction* relation.

- **Generating Dictionaries**

After constructing the nodes-links graph the analyst can generate the corresponding terms and relations dictionaries just by clicking on ‘*Générer Dictionnaires*’ button. Once the dictionaries are generated successfully a confirmation message is delivered.

- **Switching to Matrix and Dictionary-Based graph**

After generating the dictionaries from the constructed graph, the text analyst can easily switch either to the matrix visualization or to the *dictionary based* nodes-links graph for future update without losing any information. Figure 3.8 shows the corresponding matrix diagram and dictionary-based graph.

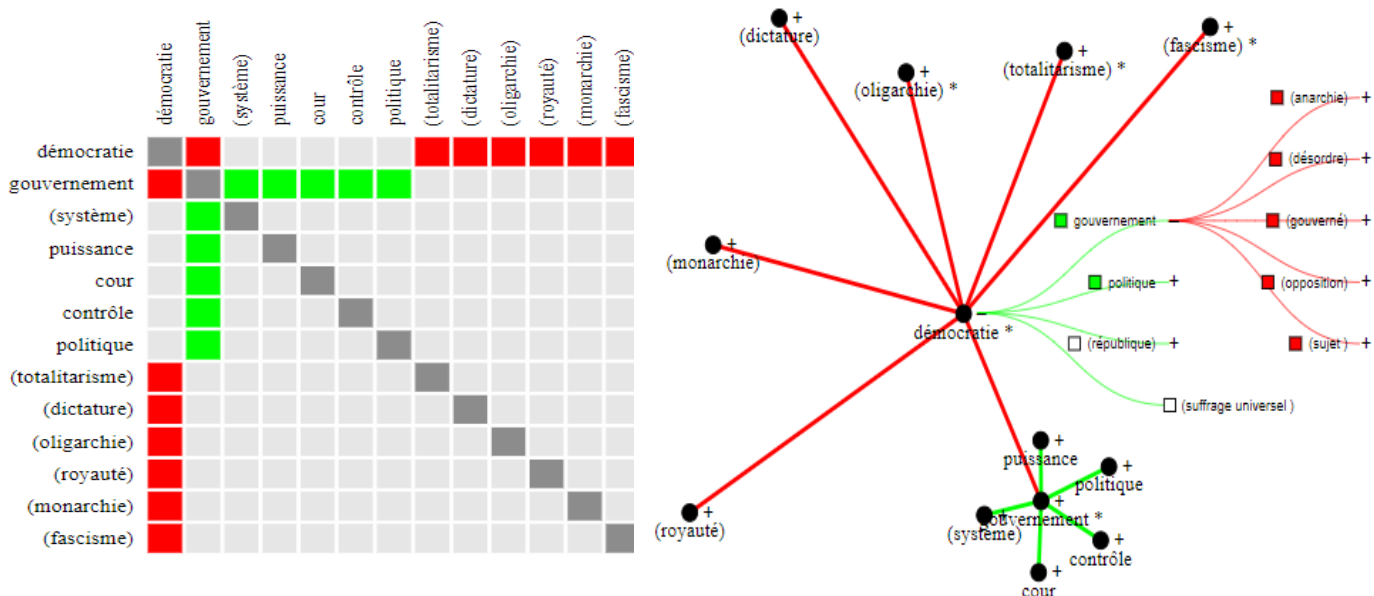


Figure 3.8: From Construction-Based visualization To Matrix and Dictionary-Based visualizations.

3.4. User interactions in EVOQ

The solution we have implemented presents a lot of advantages as far as user's interactions on visualizations are concerned.

Firstly it keeps the text close to the visualizations by displaying both the text and visualizations on the same window. This enables the analyst to have full control on the text and visualizations at the same time. Therefore, analysts are not disconnected from the text while working on the visualizations and vice versa.

Secondly, it offers more features regarding user interactions with the visualizations such as linking and brushing, selection (interactions on a particular node or edge), details on demand etc... The paragraphs in this section and the next section give more details on the interaction features.

3.4.1. Linking and Brushing

Linking and brushing is defined by Keim as *"the idea to combine different visualization methods to overcome the shortcomings of single technique"* (Keim, 2002). Wills writes that visualizations are linked if *"when the user interacts with one view [...], the other views will update and show the result of such an interaction"* (Wills, 2008). This implies that when the user makes a change on one visualization the others should also be consequently updated. The user can therefore easily switch from one visualization to the other without losing any information. Visualizations bound with linking and brushing are more informative than the same visualizations considered independently (Keim, 2002).

The interactions features with the matrix diagram (*respectively node-links diagram*) have linking and brushing with the node-links diagram (*respectively matrix diagram*).

Table 3.1 summarizes some features offered by EVOQ as regards to linking and brushing.

Table 3.1: Linking and Brushing in EVOQ

Interactions	Source entities	Impacted entities
Deleting a term, node or relation	Matrix diagram, Relations graph, terms and relations dictionaries	Matrix diagram, Relations graph, terms and relations dictionaries
Adding a relation	<i>Add button</i> , Text, Matrix, Relations graph	Matrix diagram, Relations graph, terms and relations dictionary
Change a term colour	Matrix diagram, Relations graph, Text	Matrix diagram, Relation graph, and the Text
Edit a term	Matrix diagram, Relations graph	Matrix diagram, Relations Graph, terms and relations dictionaries
Switching	From Matrix diagram to Relations graph and vice versa	

3.4.2. Construction-Based Graph vs. Dictionary-Based Graph

In this present work, we offer the analyst two different ways to build visualizations mainly the relations graph visualization. Both ways of building the graph have the same objective which is to help the analyst draw much understanding from a text using visualization. However each of them implements the visualization differently.

The *construction-based graph* also referred as *construction-based solution* helps the analyst to construct the nodes-links (*terms relations*) graph graphically and generate the dictionaries at the end of the construction whereas the *dictionary-based graph* requires relations dictionary to be encoded first before generating the visualization. Encoding terms and relations dictionaries could be time consuming when the number of terms or relations increase. In such case *construction-based graph* which offers a way to encode relations via the graph could help the analyst save time much more than the *dictionary-based graph*.

Also, the level of interactions with the visualization is higher when constructing the graph because the user has more control over the graph and can update or adjust the construction at ease.

Therefore the construction-based graph could increase the user interactivity more than the dictionary-based graph.

3.5. Existing Interaction techniques and Analyses

In traditional or rigid visualization systems the user has no interactions with the visualizations. As stated by (Brodbeck et al., 2009) *“traditional systems work in batch mode: first the search query is completely specified, then the query is sent to the system. The system executes the query and finally shows the results.”*

Highly interactive systems on the contrary work incrementally and reversibly. Every change is immediately sent to the system and a dynamic feedback is generated. Changes can be easily reverted. These systems work after the principle of *“direct manipulation”*: relevant objects and processes are visually represented and can be manipulated directly with the mouse. The manipulations take the place of a complex syntax and results are immediately visible. Interactive visualization systems combine visualization with interactivity (Brodbeck et al., 2009).

Interactivity allows users to effectively uncover the information visualization insights and is one of the most important parts of visualization: *“being able to not just see the data, but quickly change the view, add different data, etc., makes analyzing it much faster and more effective”* (Kosara et Mackinlay, 2013).

Successful interactive visualization combines expressive graphical representations and effective user interaction (Tominski, 2015).

3.5.1. Existing interaction techniques in information visualization

Currently many researchers proposed taxonomies for interaction techniques in visualization. Table 3.2 bellow gives an overview on the proposed interactivity techniques by (Shneiderman, 1996), (Keim, 2002) and (Yi et al., 2007).

Table 3.2: Interaction techniques taxonomies by (Shneiderman, 1996), (Keim, 2002) and (Yi et al., 2007) extracted from (Ana Figueiras, 2015)

Authors	Interactivity techniques
Shneiderman	Overview, zooming, filter, details on-demand, relate, history, and extract
Keim	Dynamic projections, interactive filtering, interactive zooming, interactive distortion, and interactive linking and brushing
Yi et al	Select, explore, reconfigure, encode, abstract/elaborate, filter, and connect

In order to more systematically explore the purpose of interactivity in information visualization, (Ana Figueiras, 2015) proposed a framework to help discuss and evaluate interaction techniques based on eleven (11) categories of interaction techniques. Basing her work on the existing techniques proposed by (Shneiderman, 1996), (Keim, 2002) and (Yi et al., 2007), she conducted an extensive review of popular visualization techniques and their interactive capacities. Then, she evaluated 232 visualizations⁸ that were popular on the web and studied the type of interaction they use.

The eleven (11) visualization interaction techniques provided by *Figueiras* framework and a brief description of each technique are presented in Table 3.3.

⁸ The visualizations are available at www.rethinkingvis.com/

Table 3.3 Framework for interaction techniques from (Ana Figueiras, 2015)

Interaction techniques	Description
Filtering	Only show me the data in which I am interested
Selecting	Mark or track items in which I am interested
Abstract/Elaborate	Adjust the level of abstraction of the data
Overview and Explore	Overview first, zoom and filter, then details-on-demand
Connect/Relate	Show me how this data is related
Reconfigure	Give me a different arrangement of the data
Encode	Give me a different representation of the data
History	Allow me to retrace the steps I take in the exploration of the data
Extraction of features	Allow me to extract data in which I am interested
Participation/Collaboration	Allow me to contribute to the data
Gamification	Show me the data in a more playful way

3.5.2. Interaction Analysis

This paragraph compares the user interactions we have implemented in EVOQ with some of the interaction techniques provided by *Figueiras* framework such as *selecting*, *overview and explore*, *connect/relate*, *reconfigure*, and *encode*.

Selecting: EVOQ offers the possibility to the user **to focus on a particular node** of the graph. This reduces the opacity of any node or link which is not connected

to a focused node. The user can also have access to a context menu of a given node or term *to change its colour, edit it or delete*.

Overview and Explore: With EVOQ the user can overview nodes, select a particular node and get more details on that node (term). The details include *the selected term's synonyms (in green edges) and antonyms (in red edges)* and are represented in form of a collapsible tree. Figure 4.1 shows an example.

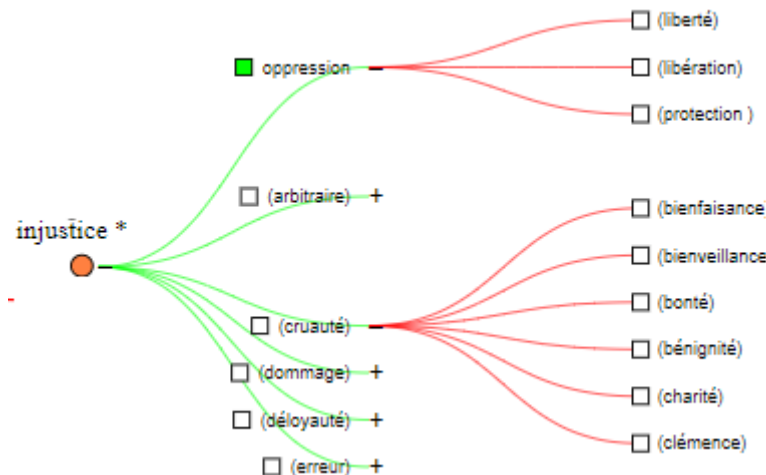


Figure 3.9: D3 collapsible tree for "injustice"

Connect/Relate EVOQ node-links or relations graph reveals the kind of connection or relation between terms (nodes). *Associated terms* are linked with a *short green edge* to show their closeness whereas *disassociated terms* are linked with a *long red edge*.

Reconfigure: On the relations graph provided by EVOQ, the user can rearrange at his/her ease the position of each node just by dragging and dropping the node. Rearranging the visualization helps the analyst to improve both its esthetic and his/her understanding of the text.

Encode: EVOQ offers the analyst the possibility to view different representations of the data. He/she can view data in form of a *matrix diagram* or a *nodes-links graph* and can combine these two different representations to get better insight into the analysis. Also, encoding data on one visualization directly update the other visualization in such a way that the analyst can easily switch from one visualization to the other without losing any information.

Chapter 4: Evaluation of EVOQ

User experience evaluation (UXE) or user experience assessment (UXA) refers to a collection of methods, skills and tools utilized to uncover how a person perceives a system (product, service, or a combination of them) before, during and after interacting with it. It is non-trivial method to assess user experience since user experience is subjective, context-dependent and dynamic over time (Law et al., 2009). Measures used as the basis for defining the user experience include among other *utility, usability, and aesthetics*.

This chapter aims at evaluating EVOQ with some users and proposing guidelines for future improvements.

The evaluation's objectives are to access the usability of the application and compare the three different ways users can encode relations with EVOQ in order to check the improvement of the users' interactions on the visualizations.

The first section of this chapter describes the evaluation methodology. The second section presents the evaluation results. And the third section proposes future improvements that could be implemented in the present EVOQ to increase the user's interactions and thoroughly match with the existing interactions techniques.

4.1. Evaluation with Users

An evaluation session with users was held on June 27, 2019 in the faculty of computer science of the University of Namur. The overall evaluation process lasted two hours.

4.1.1. Participants

Five management masters students of the faculty of Economy of the University of Namur participated in the evaluation. They were aged between 23 and 30 years old. Two of them were studying in management and finances and the other three were studying in international economy and development.

They were given the same text and were asked to encode nine relations.

4.1.2. Evaluation Methodology

The methodology used for the evaluation belongs to *within-subjects* category. Indeed there are two primary ways in which experimenters (participants) can

experiment a product: *within-subject design and between-subject design* (Charness et al., 2012).

In a “within-subject” designed experiment, each individual is exposed to more than one of the treatments being tested, whether it be playing a game with two different parameter values, being treated and untreated, answering multiple questions, or performing tasks under more than one external stimulus. With such designs, causal estimates can be obtained by examining how individual behaviour changed when the circumstances of the experiment changed (Charness et al., 2012).

In a “between-subject” designed experiment, each individual is exposed to only one treatment. With these types of designs, causal estimates are obtained by comparing the behaviour of those in one experimental condition with the behaviour of those in another (Charness et al., 2012).

Both designs have their merits, and their choice depends on the context of the question being studied. According to (Charness et al., 2012) “within-designs” have three main advantages with respect to between-designs. First, their internal validity does not depend on random assignment. Second, in many frameworks they offer a substantial boost in statistical power. Finally, they are more naturally aligned with most theoretical mind sets. However, in environments where an individual is likely to only face a single decision, a between design might have more external validity.

The different steps of the evaluation are as follow:

Step 1: Introduction

We began the session with a brief introduction on the fundamental concepts of text structural analysis and visualization. Then we explained the participants the purpose of EVOQ and the importance of the evaluation session.

Finally participants were granted access to the test version of the application online and were asked to use the application and proceed with structural analysis without being given any explanation on how to use the software. While participants were exploring the application we were carefully observing every action they were doing as well as their reactions on each interface. This first exercise helped us to check how easy it could be for users to find out EVOQ

interfaces and functionalities without being given any training, and how they feel about their use of EVOQ.

The participants at this steps were able to import a text and to scroll among the menu items. Although they could guess the function of some menu items and buttons, they were not able to encode relations and interact with the visualizations.

Step 2: Demonstration and Testing

The second step started with a brief demonstration on how to use EVOQ and ended with the testing of the tool with a concrete example of text structural analysis. Focus were put on the different ways user can encode relations, the generation and construction of visualisations both matrix and relations graph and on the different interactions with the visualisations.

The participants were given the same text and nine relations. Each participant was asked to encode the relations in the three following order:

- (1) the three first relations should be encoded via the graph using the construction-based solution;
- (2) the three next relations should be encoded via the matrix;
- (3) and the three last relations should be encoded via the text.

When the participants faced difficulties, they were encouraged to try to solve the problem for at least two minutes before we helped them. The advantage of this technique is to evaluate how easy the user would actually use EVOQ and deal with encountered difficulties. The users can then efficiently express themselves on the unclear elements of the design.

The results of the participants' evaluation are presented in paragraph 4.2.

Step 3: Filling evaluation questions

After the participants had processed all the tasks, they were given an evaluation form to fill. The evaluation form we used was taken from the User Evaluation Questions available online at <https://www.ueq-online.org/>. In addition, we have added three other questions to check how easy or difficult the participants found each of the three above mentioned methods for encoding relation.

4.2. Evaluation Results

The evaluation of EVOQ with the participants lasted two hours and reveals the following limitations:

- Absence of a *help menu* to provide information on how to use the application;
- It is difficult for the participants to intuitively encode a relation unless they are shown how to do it;
- The application is less fast when displaying the relations graph than the matrix and updating visualization takes some seconds;
- Sometimes there are spontaneous movements of the nodes during the construction of the graph;
- The button '+' is not explicit enough to guess its function.

The average results of the participants' evaluations are summarized on the User Evaluation Questions presented by figure 4.1. Figure 4.2 presents the result for the evaluation of the three different ways to encode relations.

UEQ- User Evaluation Questions

Example:

attractive ○ ⊗ ○ ○ ○ ○ ○ unattractive

This response would mean that you rate the application as more attractive than unattractive.

Please assess the product now by ticking one circle per line.

	1	2	3	4	5	6	7											
annoying	○	○	○	○	○	●	○	enjoyable	1	unlikable	○	○	○	○	○	●	pleasing	14
not understandable	○	○	○	○	○	●	○	understandable	2	usual	○	○	○	○	○	○	leading edge	15
creative	○	○	○	○	○	○	○	dull	3	unpleasant	○	○	○	○	○	○	pleasant	16
easy to learn	○	○	○	○	○	○	○	difficult to learn	4	secure	○	○	○	○	○	○	not secure	17
valuable	○	○	○	○	○	○	○	inferior	5	motivating	○	○	○	○	○	○	demotivating	18
boring	○	○	○	○	○	○	○	exciting	6	meets expectations	○	○	○	○	○	○	does not meet expectations	19
not interesting	○	○	○	○	○	○	○	interesting	7	inefficient	○	○	○	○	○	○	efficient	20
unpredictable	○	○	○	○	○	○	○	predictable	8	clear	○	○	○	○	○	○	confusing	21
fast	○	○	○	○	○	○	○	slow	9	impractical	○	○	○	○	○	○	practical	22
inventive	○	○	○	○	○	○	○	conventional	10	organized	○	○	○	○	○	○	cluttered	23
obstructive	○	○	○	○	○	○	○	supportive	11	attractive	○	○	○	○	○	○	unattractive	24
good	○	○	○	○	○	○	○	bad	12	friendly	○	○	○	○	○	○	unfriendly	25
complicated	○	○	○	○	○	○	○	easy	13	conservative	○	○	○	○	○	○	innovative	26

Figure 4.1: Average Results of the UEQ

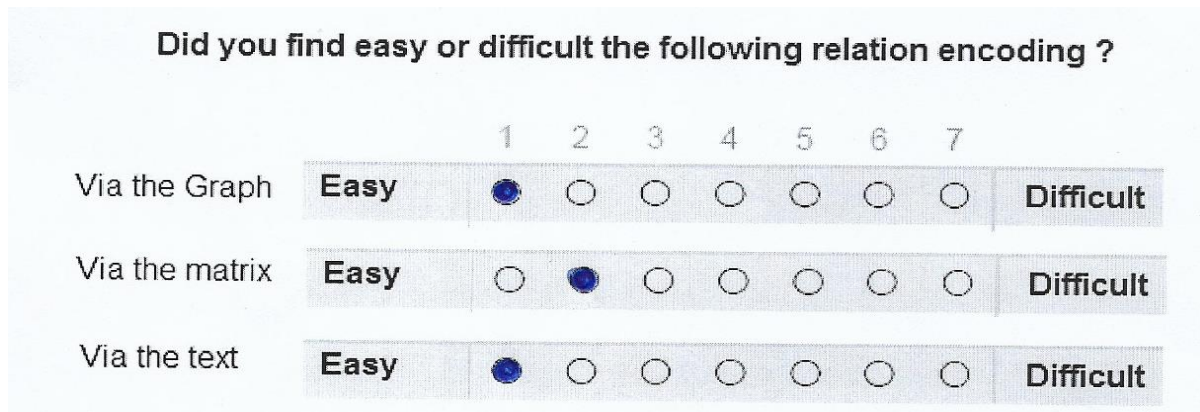


Figure 4.2: Average Results for relation encoding

Although the number of participants is statistically small, the analysis of the results of this preliminary evaluation reveals the following indications:

- The application interfaces are very attractive; the menu, buttons and labels are simple, clear and easy to understand;
- The application is pleasant and easy to use after little training;
- The three ways to encode relations are not complicated. But encoding relations *via the graph* and *via the text* are easier and faster;
- The application efficiently generates the expected results;

The participants also found the different interactions with the visualisations easy and interactive although some actions may take few seconds to be completed. They mainly appreciated the interactive updates on the visualizations and the possibility to switch from one visualization to the other.

As regards to the research question, based on this preliminary evaluation we could say that some improvements have been made concerning the user's interactions on the visualizations in the present EVOQ. These improvements concern the linking and brushing, the interactive updates on the visualizations, and relations encoding via the visualizations (matrix or graph). A further assessment with a greater number of participants could be held for accurate validation.

The functionalities which were not covered in the user evaluation were explained to the participants. Despite the aforementioned limitations, the

participants were very enthusiastic about the solutions offered by EVOQ regarding text analysis and about the whole process of the evaluation.

4.3. Suggestions of future improvements on EVOQ

This paragraph proposes three guidelines for the improvement of EVOQ that future works could take into consideration in order to increase user interaction on the visualizations.

Guideline n°1: Improving performance

Some functionalities of EVOQ such as *displaying relations graph*, *updating terms* and *generating dictionaries* are less faster and could take much more time when the number of relations increases. Therefore, it could be convenient to reduce the complexity of the algorithms for those functionalities in order to make their execution faster.

Guideline n°2: Improving filtering

One way to reduce complexity is by filtering the data. The most successful way to filter data is through the use of dynamic filters which will allow the user to quickly see on the visualizations only the items he/she is interested in. A dynamic filter could be implemented in EVOQ for both matrix and relation graph visualizations. Examples of filter could be '*Terms filters*' and '*Relationships filters*'.

Terms filters would help arranging term on the matrix diagram in alphabetic ascendant or descendant order, or selecting only a list of terms to display on the visualizations.

Relations filters would help to filter and display only association or disjunction relations on the visualizations.

Guideline n°3: Improving Zooming

Zooming allows the user to see an overview of the visualization or to see a smaller more detailed view without fundamentally altering the representations. *Zooming-in* enlarges smaller data elements in which the user is interested, removing from view or reducing the size of large uninteresting data. The *zooming-out* produces the opposite result, it reduces larger data elements.

Zooming could be implemented on the visualizations offered by EVOQ to allow the user to get more insight on interested data. User could zoom on the whole or a precise part of the matrix or relation graph.

Guideline n°4: Improving Reconfigure

The reconfigure interactive technique provides the users with different perspectives about the data set by changing the spatial arrangement of the representation (Yi et al., 2007). The arrangement of the data allows the user to have different perspectives that he/she probably would not have with a rigid representation.

In EVOQ, the user can rearrange the nodes-links graph by dragging nodes and edges. In addition, we could implement a function to save the final configuration of the graph by saving the position (x,y) of each node while saving the project so that the analyst would not lose the positions of the nodes when he/she reopen the project. This would be very helpful to the analyst in case of complex graphs with a great number of nodes and edges.

Chapter 5: Conclusion

The interest of this thesis lies in the importance of user interactivity in visualization and more specifically in text visualization. The general question of research of this thesis was how to improve the text analyst interactions on a text visualization based on structural analysis.

In order to efficiently answer this question our work explored the three following approaches:

- (1) Providing an up-to-date literature review on text visualization and user interactivity;
- (2) Improving the existing text visualization tool EVOQ to enable text analysts to visualize both the text and visualizations at the same time;
- (3) Implementing a visualization algorithm which enables text analysts with more dynamic interactions on the visualizations.

To achieve the first approach we used (ClarINVAL et al., 2018) work as a starting point and conducted an up-to-date state of the art on visualization techniques and tools that represent relationships between text elements, as well as on user interactivity.

As far as the second approach is concerned, we used an agile method of software development and open source web technologies to implement a new work interface for the existing visualization tool EVOQ.

The previous EVOQ enables the text analyst to generate a relational graph visualization from an encoded terms (words) relations dictionary. We referred this way of generating the visualization as *dictionary-based graph*.

To achieve the third approach, in addition to *dictionary-based graph* we proposed in the new version of EVOQ an algorithm referred to as *construction-based graph algorithm* which helps the analyst to fully construct the relational graph rather than generating it. Also, given a node or a term, the analyst can easily generate its synonyms or antonyms (associations or disjunctions relations) making therefore the construction easier and faster. At the end of the construction, the analyst can generate the relations dictionary and switch to the matrix visualization. This new approach of constructing the visualization seems to

increase the analyst interactions on the visualization. We used a JavaScript open source library **Viz.js** to implement the *construction-based graph algorithm*.

We compared our solution with existing interaction techniques found in the current literature. Our analysis was based on the 11 visualization interactions techniques described in *Ana Figueiras* framework (Ana Figueiras, 2015).

We also conducted a preliminary evaluation with five participants to access the usability of EVOQ and evaluate the three different ways users could encode relations which are: *via the graph, via the matrix and via the text*. The overall evaluation results reveal that the present EVOQ offers some improvements as regard to user interactions on the visualizations; mainly: linking and brushing, interactive updates on the visualizations, and relations encoding via the visualizations (matrix or graph). A further evaluation with a significant number of participants is determinant for accurate results and conclusion.

Finally, we proposed four guidelines for future works to improve user interactivity in EVOQ in order to fully match EVOQ with the existing visualization interactions techniques.

Bibliography

- Jansen, Y. and Dragicevic, P. (2013). An interaction model for visualizations beyond the desktop. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), pp.2396 - 2405.
- Liu S., Cui W., Wu Y., and Liu M. (2014). A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12), pp.1373-1393.
- Wallemacq A., Jacques J.M., and Bruyninckx V. (2004). Dans le sillage des mots : EVOQ. Logiciel de cartographie cognitive. *Presses universitaires de Namur*.
- Kucher K. et Kerren A. (2015). Text visualization techniques: Taxonomy, visual survey, and community insights. In *Visualization Symposium (PacificVis '15)*, 2015 IEEE Pacific, pages 117-121. *IEEE*.
- Nan C. and Cui W. (2016). Overview of text visualization techniques. In *Introduction to Text Visualization, Atlantis Briefs in Artificial Intelligence*, pages 11-40. *Springer*.
- Yang Y.Y., Akers L., Klose T., and Yang C.B. (2007). Text mining and visualization tools-Impressions of emerging capacities. *PIUG 2007 Annual Conference, Costa Mesa, CA, USA*.
- Piret A., Nizet J., and Bourgeois E. (1996). L'analyse structurale: une méthode d'analyse de contenu pour les sciences humaines. *De Boeck Supérieur*.
- Radburn R., Dykes J., and Wood J. (2010). vizLib: Using The Seven Stages of Visualization to Explore Population Trends and Processes in Local Authority Research. *giCentre, School of Informatics, City University London, EC1V 0HB*.
- Malyanov I., Brian J.d'Auriol, and Lee S. (2013). Visualization experience and related process modeling. *Journal of visual Languages and Computing* 24 pages 223-233.
- Smith A., Chuang J., Hu Y., Boyd-Graber J., and Findlater L. (2014). Concurrent Visualization of Relationships between Words and Topics in Topic Models. *Workshop on Interactive Language Learning, Visualization, and Interfaces*, page 79-82, Baltimore, Maryland, USA.
- Fry B. (2004). Computational Information Design, *Massachusetts Institute of Technology*.
- Fry B. (2008). Visualizing Data. Sebastopol. CA: O'Reilly, 382 pp.
- Liu S., Wang X., Collins C., Dou W., Ouyang F., El-Assady M., Jiang L., and Keim A.D. (2018). Bridging Text Visualization and Mining: A Task-Driven Survey. 2018 *IEEE Transactions on Visualization and Computer Graphics*.

Gupta V., and Lehal S.G. (2009). A survey of Text mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, Vol.1.August 2009.

M. Chen, H. Jänicke. (2010). An information-theoretic framework for visualization, *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), pp. 1206–1215.

Clarinval A., Linden I., Wallemacq A., Dumas B. (2018). Evoq: A Visualization Tool to Support Structural Analysis of Text Documents. In *Proceedings of the 18th ACM Symposium on Document Engineering*. ACM Press, Halifax, Canada.

W Bradford Paley. 2002. TextArc: Showing word frequency and distribution in text. In *Poster presented at IEEE Symposium on Information Visualization*, Vol. 2002.

Viegas, F. B., Wattenberg, M. et Feinberg, J. (2009). Participatory visualization with wordle. *IEEE transactions on visualization and computer graphics*, 15(6)

Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Marc Smith, Ben Shneiderman, and Derek Hansen. 2011. Group-in-a-box layout for multifaceted analysis of communities. In *ICSM*, pages 354–361. IEEE.

Alison Smith, Jason Chuang, Yuening Hu, Jordan Boyd-Graber, Leah Findlater. (2014). Concurrent Visualization of Relationships between Words and Topics in Topic Models. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 79–82, Baltimore, Maryland, USA, June 27, 2014. c 2014 Association for Computational Linguistics

Michael Curtotti, Eric McCreath, Srinivas Sridharan. (2013). **Software Tools for the Visualization of Definition Networks in Legal Contracts**. *Proceedings of the 14th International Conference on Artificial Intelligence and the Law*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Machine Learning Journal*, 3:993–1022.

R. Andersen and Y. Peres. (2008). Finding sparse cuts locally using evolving sets. *Proceedings of the 41st annual ACM symposium on Theory of computing*, page 20, 2008.

Y Teng, T Höllerer, J O'Donovan. (2015). LinkScope: Interactive Graph Analysis of Unstructured Text. *University of California, Santa Barbara*.

F. Van Ham, M. Wattenberg, and F. B. Viegas. (2009). Mapping text with phrase nets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1169–1176, 2009.

Wattenberg M., Viégas F.B. 2008. The word tree, an interactive visual concordance. *IEEE Trans. Vis. Comput. Graph.* **14**(6), 1221–1228 (2008)

Graham Wills.2008. Linked data views. In *Handbook of data visualization*. Springer,217-241

R. Kosara and J. Mackinlay. 2013. Storytelling: The next step for visualization. *Computer* 46, 5 (2013),44–50.

Christian Tominski. 2015. Interaction for visualization. *Synthesis Lectures on Visualization* 3, 1 (2015), 1–107.

Dominique Brodbeck, Riccarde Mazza, Denis Lalanne. 2009. **Interaction visualization: A survey.** Denis Lalanne, Jürg Kohlas (Eds.), *Human Machine Interaction*, Springer-Verlag, Berlin (2009), pp. 27-46

B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations", *Visual Languages 1996. Proceedings. IEEE Symposium on*, pp. 336-343, 1996.

J. S. Yi, Y. ah Kang, J. Stasko, J. Jacko, "Toward a deeper understanding of the role of interaction in information visualization", *Visualization and Computer Graphics IEEE Transactions on*, vol. 13, no. 6, pp. 1224-1231, 2007.

Sonalake Limited, <https://sonalake.com/latest/popular-open-source-javascript-frameworks-for-data-visualisation/>, accessed on February 7th ,2019

Siirtola H., Isokoski P., Säily T., Nevalainen T. 2016. Interactive Text Visualization with Text Variation Explorer. in E Banissi (ed.), *Proceedings of the 20th International Conference on Information Visualization (IV 2016)*. Information Visualization, IEEE Computer Society, Los Alamitos, California, pp.330-335

HIERNAUX, Jean-Pierre .(1977). L'institution culturelle. II. Méthode de description structurale. *Louvain-la-neuve, Presses de l'Université de Louvain (U.C.L.)*.

D. Keim, "Information visualization and visual data mining", *Visualization and Computer Graphics IEEE Transactions on*, vol. 8, no. 1, pp. 1-8, 2002.

Figueiras, A. 2015. Towards the understanding of interaction in information visualization. In: *19th International Conference on Information Visualization (2015)*

Jean REMY, Danielle RUQUOY. (1990). Méthodes d'analyse de contenu et sociologie, *Bruxelles, Publications des Facultés Universitaires Saint-Louis*

Keim, D. A. (2002). Information visualization and visual data mining. *IEEE transactions on Visualization and Computer Graphics*, 8(1):1-8.

Law, E., Roto, V., Hassenzahl, M., Vermeeren, A., Kort, J. (2009). **Understanding, Scoping and Defining User Experience: A Survey Approach.** In *Proceedings of Human Factors in Computing Systems conference, CHI'09. 4–9 April 2009, Boston, MA, USA*.

Gary Charness, Uri Gneezy, Michael A. Kuhn. (2012). **Experimental methods: Between-subject and within-subject design.** *Journal of Economic Behavior & Organization*, Pages 1-8

Appendices

Description of the functionalities of EVOQ

1. Managing evoq project menu

The analyst can either import an existing project, save a new project or modify project parameters.

Import a project: clicking on this button allows the user to import an existing evoq project file (.evoq). When the project file is successfully imported, the text of the project is loaded in the text input bock and the related terms dictionary and terms relations dictionary are also loaded in their respective data structures. The analyst can zoom on the text.

Save a project: the analyst can save the current project. Saving a current project saves the project text as well as all the related terms dictionary and terms relations dictionary. The project file is saved in the format *filename.evoq* where filename is the name encoded by the user in the project *title input*.

Project parameter: the user can change project parameters such as *project name*, *number of synonyms and antonyms to display for terms in the node-links graphs etc.*

Close a Project: the user can close a project. When the user clicks on this button, a confirmation is asked from him before closing the current project. We user can also save the project during the closing process.

Figure 1 shows the project management menu with its items.

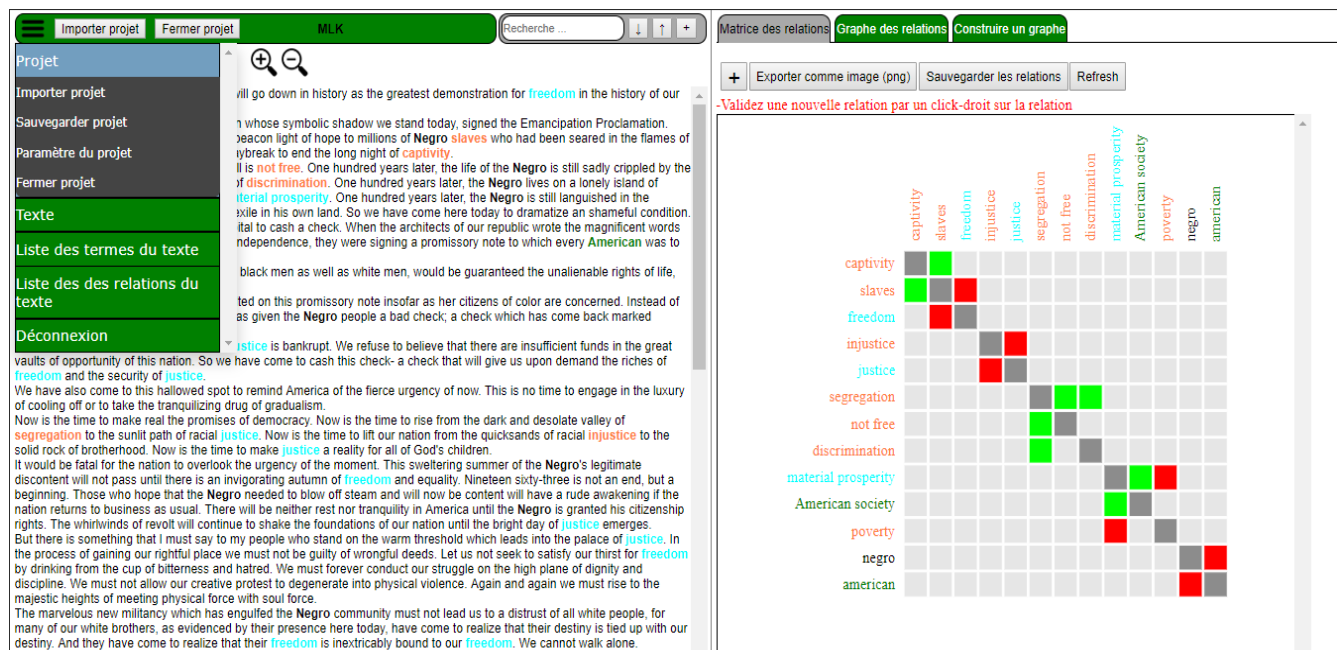


Figure 1: EVOQ the project management menu

2. Managing a text menu

The analyst can import a text or save a project text.

Import a text: clicking on this button allows the user to import an existing text. The accepted format for the text is *.txt*. Once imported, the text is displayed in the text input block.

Save a text: this button saves the project text alone in *.txt* format.

3. Managing terms dictionary menu

In this submenu, the analyst can either display the list of terms from the term dictionary, import an existing list of terms or save (export) the current list of terms in a file.

Display term list: this button displays all the terms related to the current *evog* project. It also enables the analyst to delete a term from the term dictionary. Figure 2 gives an overview of the terms dictionary.

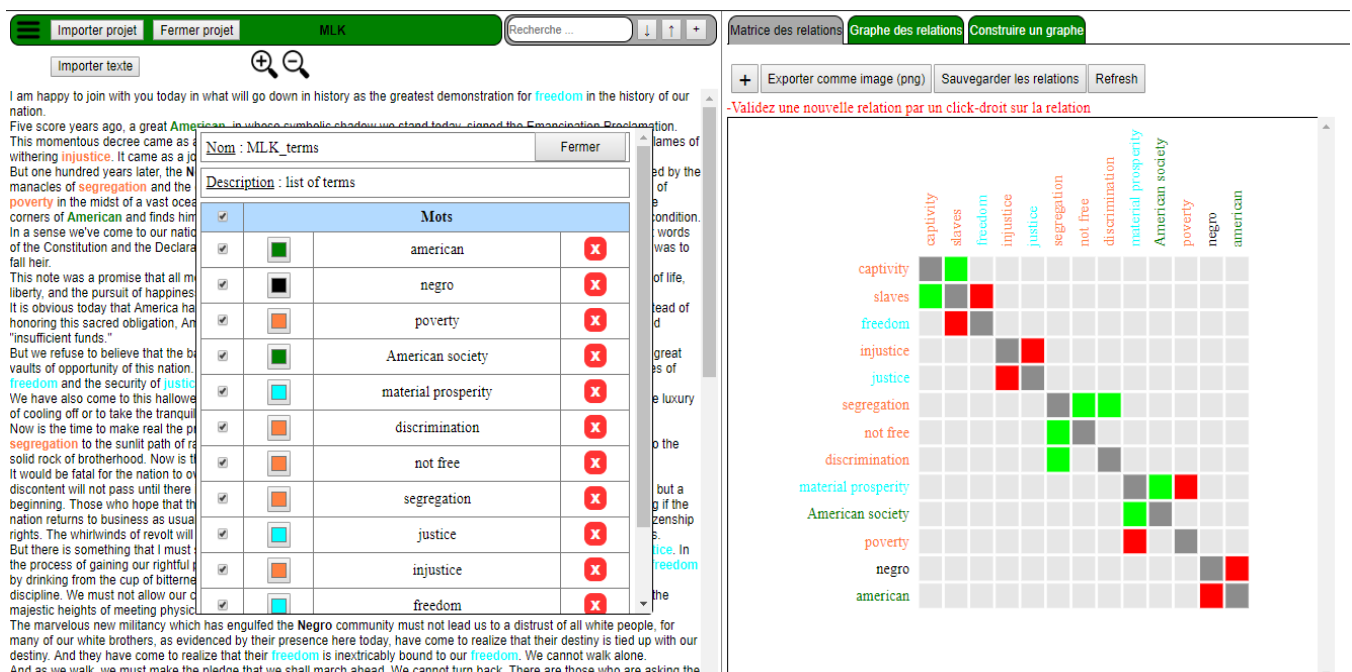


Figure 2: Terms dictionary of a given project

Import terms dictionary: clicking on this button helps the analyst to import an existing terms dictionary and load them into the current project term dictionary data structure. The imported file must be with the extension *.evogtd*

Save term dictionary: by clicking here the analyst can save or export the current project terms dictionary with the extension *.evogtd*

4. Managing terms relations dictionary menu

In this submenu, the analyst can either display the term relation dictionary, import an existing relation dictionary or save (export) the current terms relation dictionary in a file.

Display term relation dictionary: this button displays all the terms relations related to the current *evoq project*. It also enables the analyst to delete relations from the dictionary. When the user deletes a relation from the relation dictionary the visualizations are automatically updated (i.e. the deleted relation is removed from all the visualizations, both the matrix diagram and nodes-links graph). Figure 3 (a) and (b) give an overview of the terms relation dictionary.

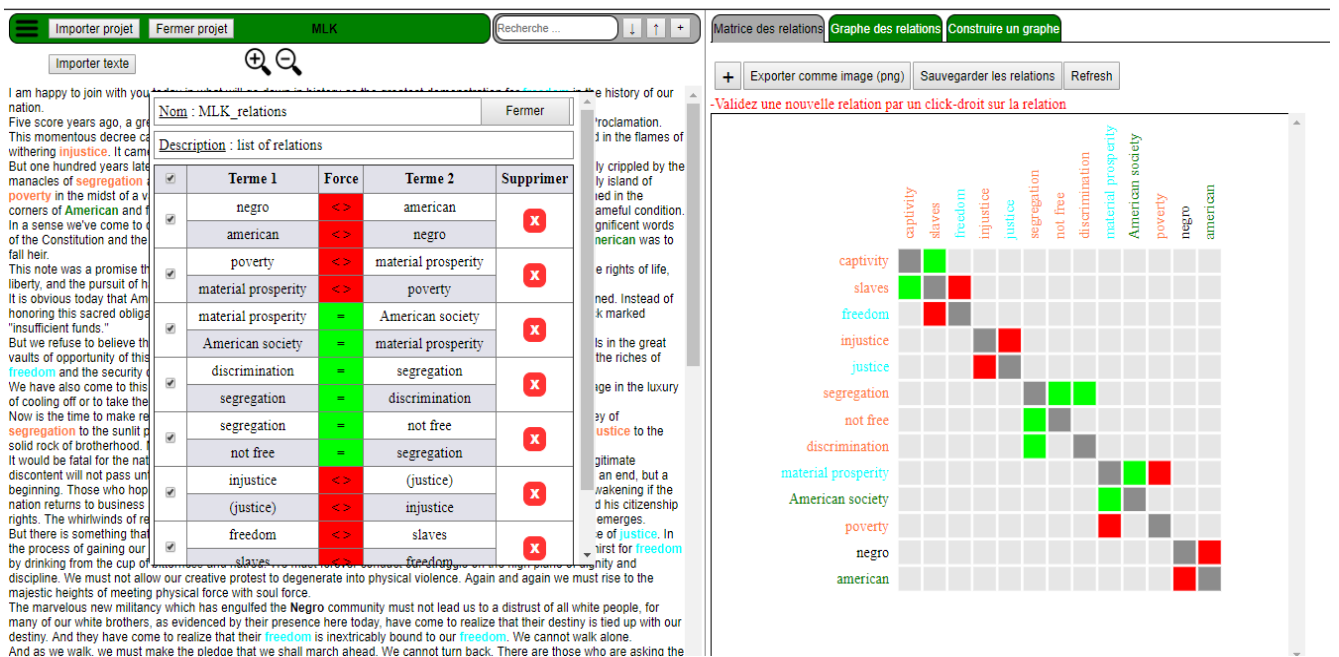


Figure 3 (a): Relations dictionary of a given project

<input checked="" type="checkbox"/>	Terme 1	Force	Terme 2	Supprimer
<input checked="" type="checkbox"/>	negro	<>	american	<input checked="" type="checkbox"/>
	american	<>	negro	
<input checked="" type="checkbox"/>	poverty	<>	material prosperity	<input checked="" type="checkbox"/>
	material prosperity	<>	poverty	
<input checked="" type="checkbox"/>	material prosperity	=	American society	<input checked="" type="checkbox"/>
	American society	=	material prosperity	
<input checked="" type="checkbox"/>	discrimination	=	segregation	<input checked="" type="checkbox"/>
	segregation	=	discrimination	
<input checked="" type="checkbox"/>	segregation	=	not free	<input checked="" type="checkbox"/>
	not free	=	segregation	
<input checked="" type="checkbox"/>	injustice	<>	justice	<input checked="" type="checkbox"/>
	justice	<>	injustice	
<input checked="" type="checkbox"/>	freedom	<>	slaves	<input checked="" type="checkbox"/>
	slaves	<>	freedom	
<input checked="" type="checkbox"/>	captivity	=	slaves	<input checked="" type="checkbox"/>
	slaves	=	captivity	

Figure 3(b): Relations dictionary of a given project

The force of a relation is either an association (=) or a disjunction (<>).

Import terms relation dictionary: clicking on this button helps the analyst to import an existing terms relation dictionary and load it into the current project term relations dictionary data structure. The imported file must be with the extension *.evoqrd*

Save term relation dictionary: by clicking here the analyst can save the current project terms relation dictionary with the extension *.evoqrd*.

5. Display matrix diagram

When the analyst clicks on “*Matrix des Relations*” button, the matrix diagram is displayed. The matrix visualization represents a set of relationships of the terms. The terms are displayed on the two axis of the diagram. The brackets are used to indicate that the term is not present in the text. The color of the square indicates the type of the relation. A green color of a square indicates that there is an association relation between its two terms whereas a red color indicates an opposition or disjunction relation between its two terms. A gray color of a square means there is no assigned relation between the two terms.


The user can add a new relation by clicking on the add button.

The **Export image button** helps to export the matrix diagram as an image whereas the **save relations button** is used to save (export) the terms relations encoded from the matrix visualization.

Figures 3 above shows the matrix visualization of a given project we have chosen.

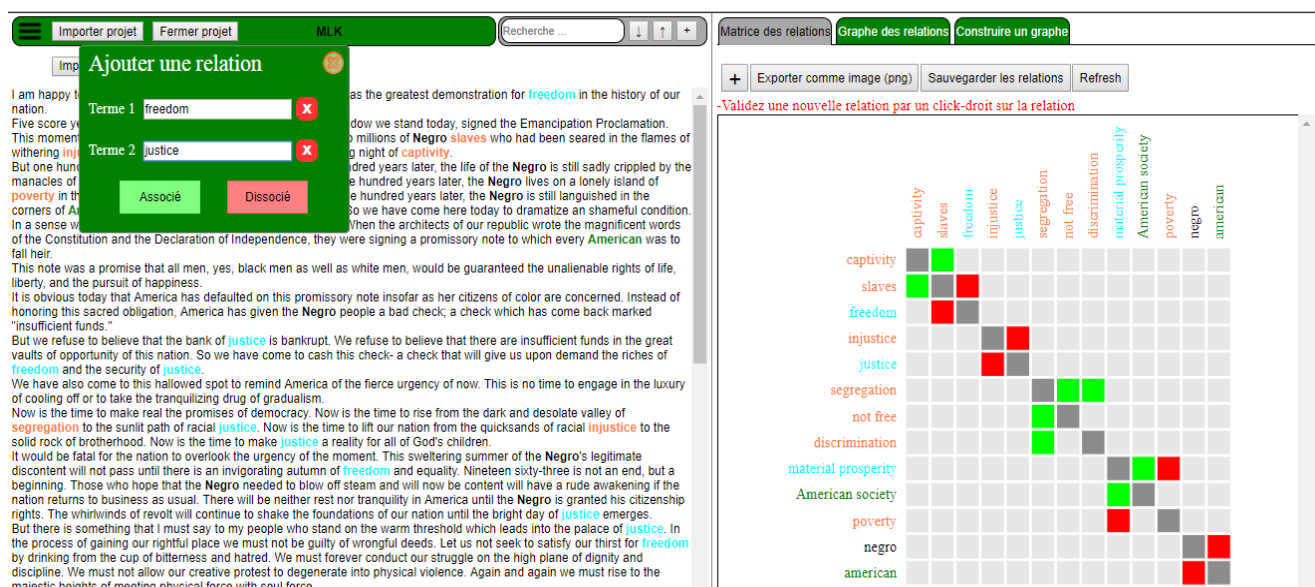
Interactions with the matrix

The matrix visualization allows some interactions from the user *such as adding new relation, deleting a term or a relation, editing a term, and changing term color*.

To add a new relation the user has to click on the  “add button”.

Users can also add a new relation or term by double-clicking on a term from the text and filling the relation form. Finally users can add relation via the matrix by right-clicking on the concerned relation from the matrix and validating. Figure 4 (a) and (b) show respectively examples for adding a relation via the text (or via button ‘+’) and via the matrix.

When the user add a new relation, the terms dictionary and the relations dictionary are updated automatically. The visualizations are also updated automatically.



The screenshot displays the application's interface. On the left, a text document is open, and a dialog box titled "Ajouter une relation" is overlaid. The dialog has two input fields: "Terme 1" with the value "freedom" and "Terme 2" with the value "justice". Below these fields are two buttons: "Associé" (green) and "Dissocié" (red). On the right, the "Matrice des relations" (Relations Matrix) is shown. It is a grid where rows and columns represent terms. The terms listed are: captivity, slaves, freedom, injustice, justice, segregation, not free, discrimination, material prosperity, American society, poverty, negro, and american. The matrix shows various colored cells (green, red, grey) representing different types of relations between these terms. For example, "freedom" is associated with "captivity" and "slaves", while "justice" is associated with "injustice".

Figure 4(a): adding new relation form (via text and button ‘+’)

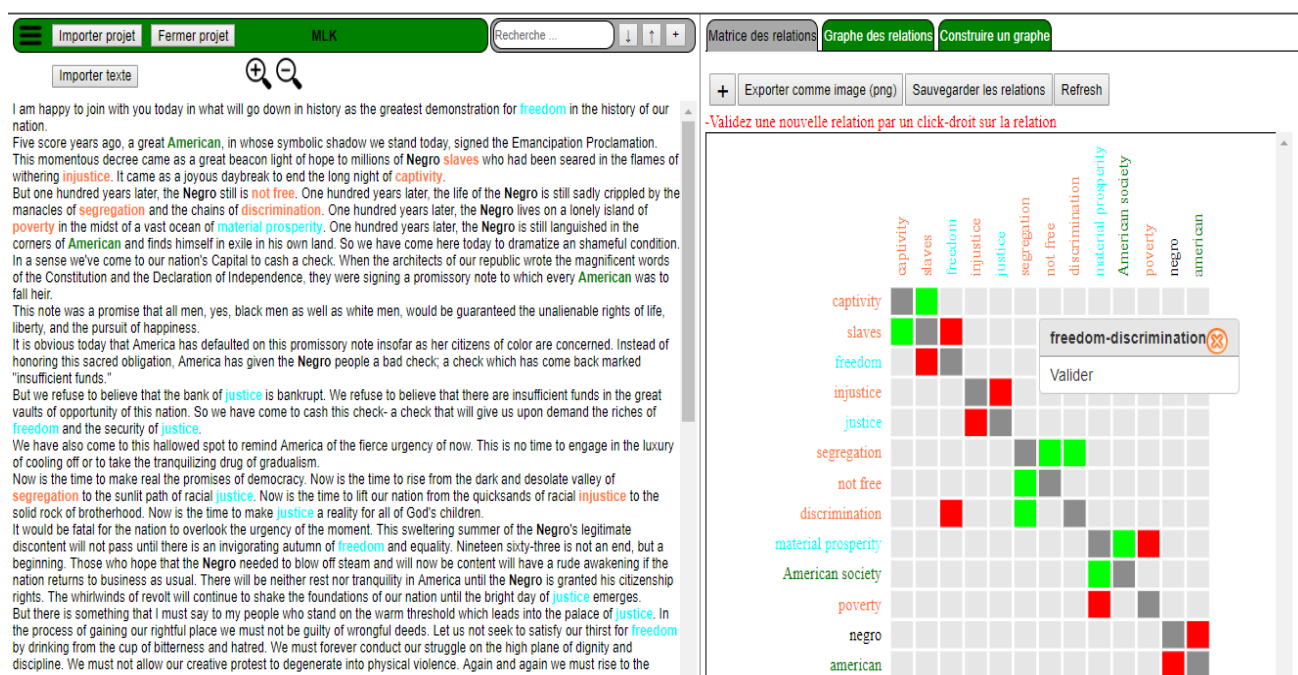


Figure 4(b): adding new relation via the matrix

To delete a term, the user right-clicks on the term, then clicks on ‘Supprimer’ in the context menu and confirms by clicking on “OK”. When a term is deleted the terms dictionary and the relations dictionary are updated automatically meaning that all the relations concerned by the *deleted term* are removed from the relation dictionary. The visualizations are also updated automatically.

To edit a term, the user right-clicks on the term, then clicks on ‘Editer’ in the context menu. Then he/she inputs the new name of the term and confirms. The term is therefore renamed in the terms and relation dictionaries as well as on the visualizations.

To change a color for a term, the user right-clicks on the term, then clicks on ‘Changer de couleur’ in the context menu. Then he/she chooses the new color from the color panel and confirms. Therefore the color of the term changes automatically in the term dictionary, in the text and on the visualizations.

The color of a term can also be changed from the text by double-clicking on the term and choosing ‘change term color’.

Figures 5 (a) and (b) show respectively the interfaces to change a *term color* and to *delete a term*.

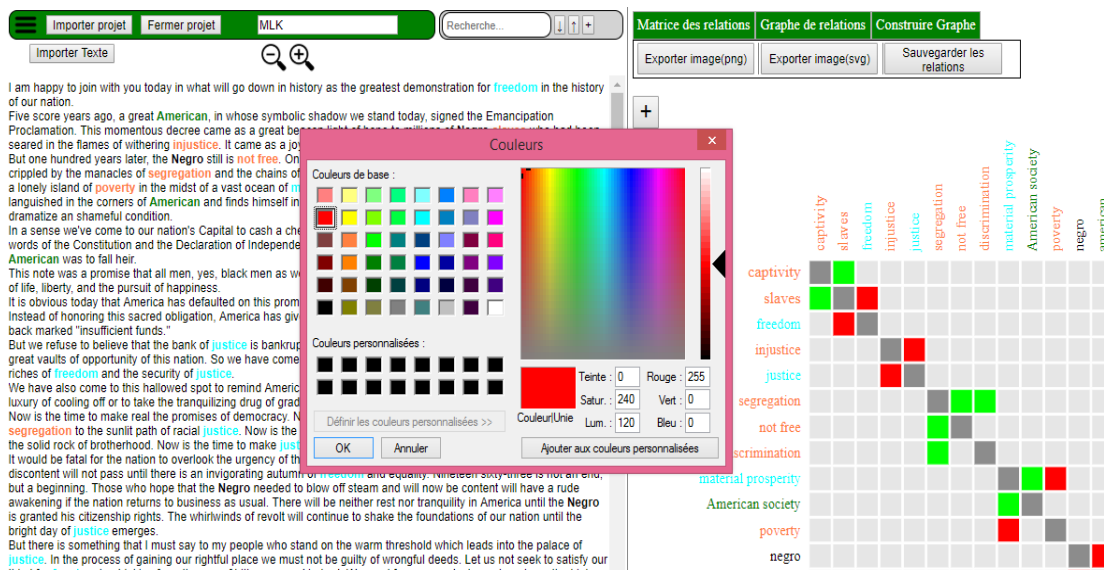


Figure 5: (a) Change the color of a term

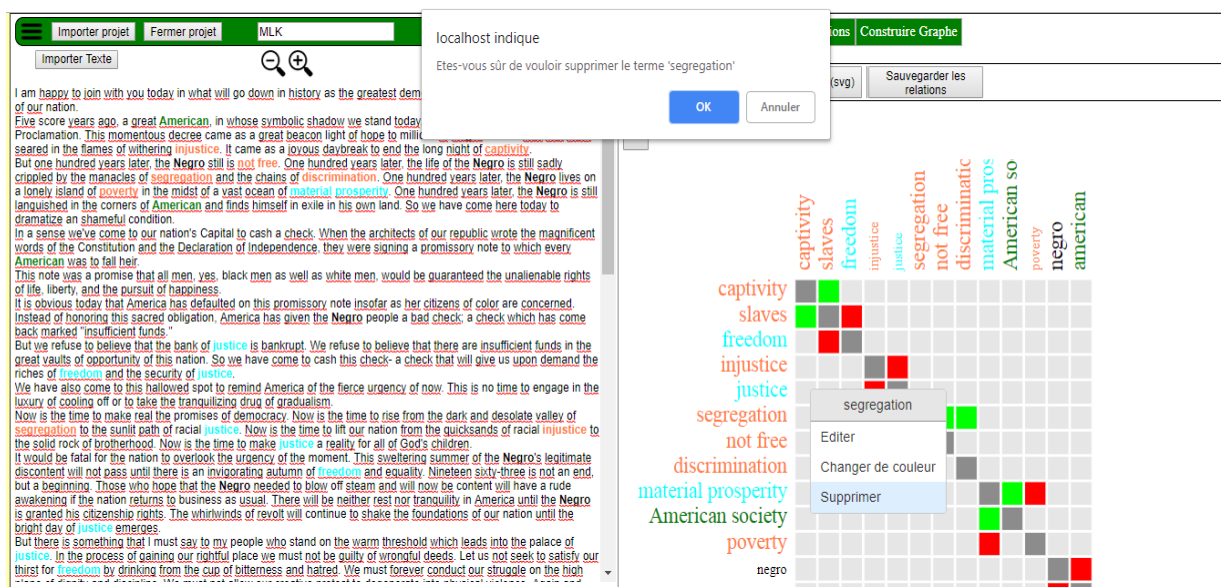


Figure 5(b): deletion of the term “segregation”

6. Display node-links diagram

The node-link diagram (also called relation graph) is the graphical representation of the terms relations as a bi-dimensional graph. The terms relate to the nodes and relations relate to the links between two given nodes. We used a colored disk containing the corresponding term to represent a node and we present a link as a colored line drawn from a node to another. The color of the line depends on the type of the relation it represents: we used *green color for a conjunction relation* and *red color for a disjunction relation*. These colors were chosen because they remind respectively

association and opposition. On the other hand, the user can choose any color for a node by right-clicking on the node.

Figures 6 (b) and (c) show an example of a node-link diagram generated by the following relations dictionary of Figure 6 (a):

Terme 1	Force	Terme 2
negro	< >	american
american	< >	negro
poverty	< >	material prosperity
material prosperity	< >	poverty
material prosperity	=	American society
American society	=	material prosperity
discrimination	=	segregation
segregation	=	discrimination
segregation	=	not free
not free	=	segregation
injustice	< >	justice
justice	< >	injustice
freedom	< >	slaves
slaves	< >	freedom
captivity	=	slaves
slaves	=	captivity

Figure 6 (a): a given relations dictionary

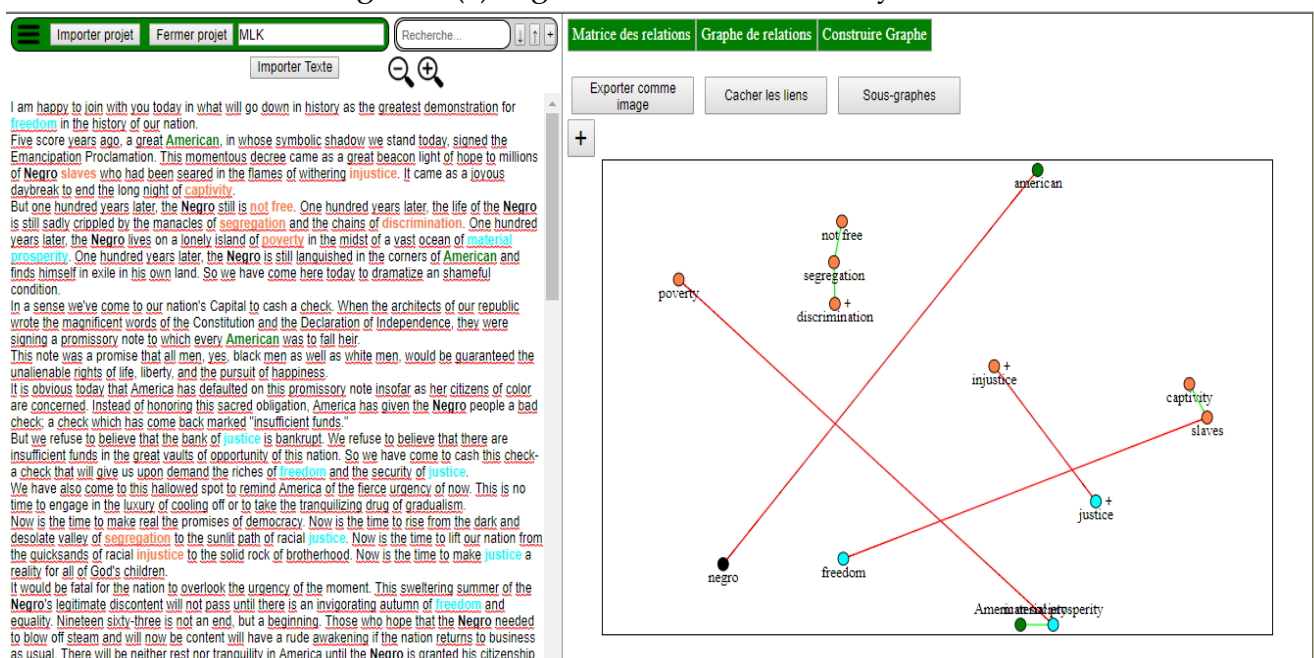


Figure 6 (b): nodes-links graph visualization

When we click on the + of the node we can have the trees of the synonyms of the terms.

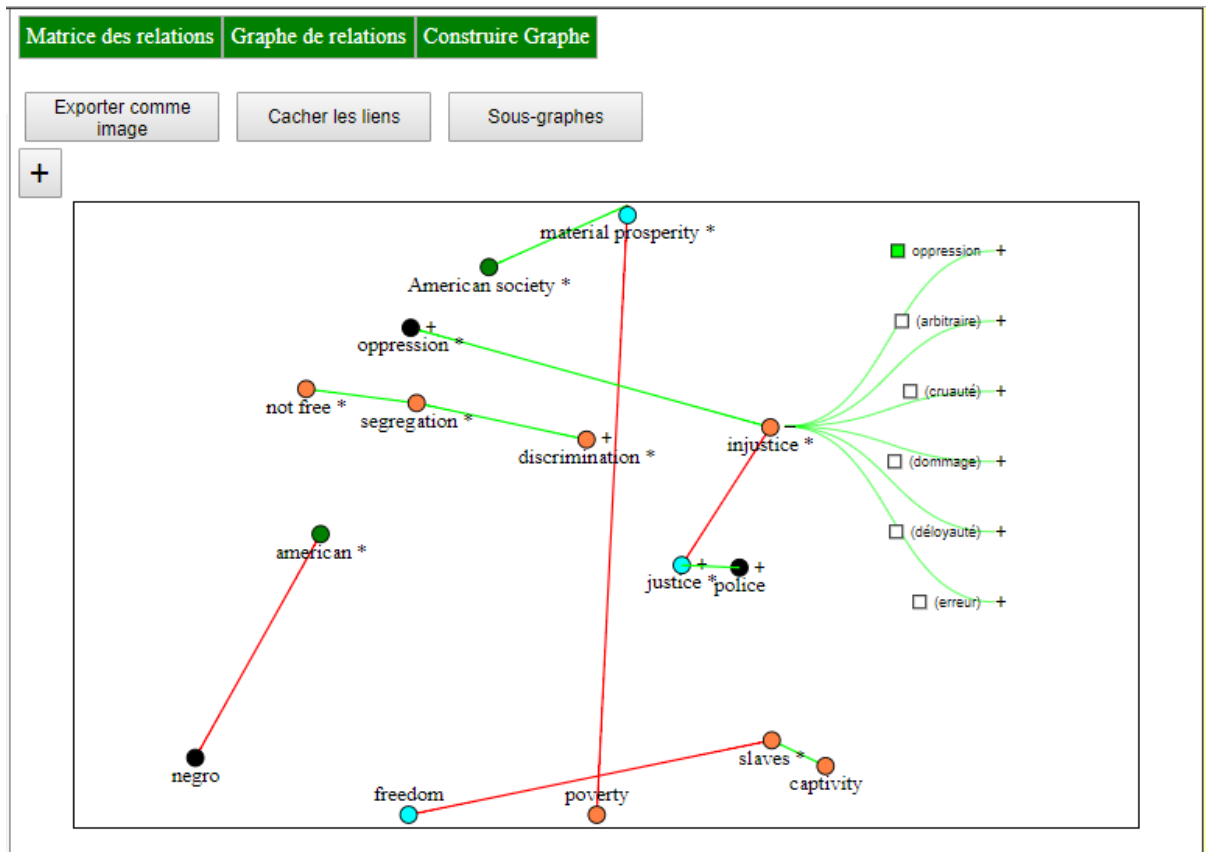


Figure 6 (c): nodes-links graph visualization showing some synonyms of “injustice”

As the analyst has a critical role for the interpretation in structural analysis, our node-link visualization provides him ways to extract knowledge by several interactions possibilities offered to the user such as:

- **Drag and drop a node**

The drag and drop of a node is the most important interaction feature. When a user drags a node, they can observe the effect it has on the semantic field represented by the diagram. This is precisely what an analyst who performs structural analysis is looking for because it is the type of knowledge they need and that is difficult to get without a visualization. *This is what makes the node-link diagram the visualization that brings the most added-value to the analyst.* The focus, links hiding, and the drag and drop interactions allow the user to reduce the number of visible edge crossings in the graph. In this regard, they improve the legibility of the diagram. The number of edge crossings is indeed the most penalizing issue for legibility in a node-link diagram.

- **Change the color of a node**

To change a node color, the user right-clicks on the node and selects ‘*Changer de couleur*’ in the displayed context menu. Then he/she chooses the color and validates.

Figure 7 bellow shows a change of color for the node “poverty”.

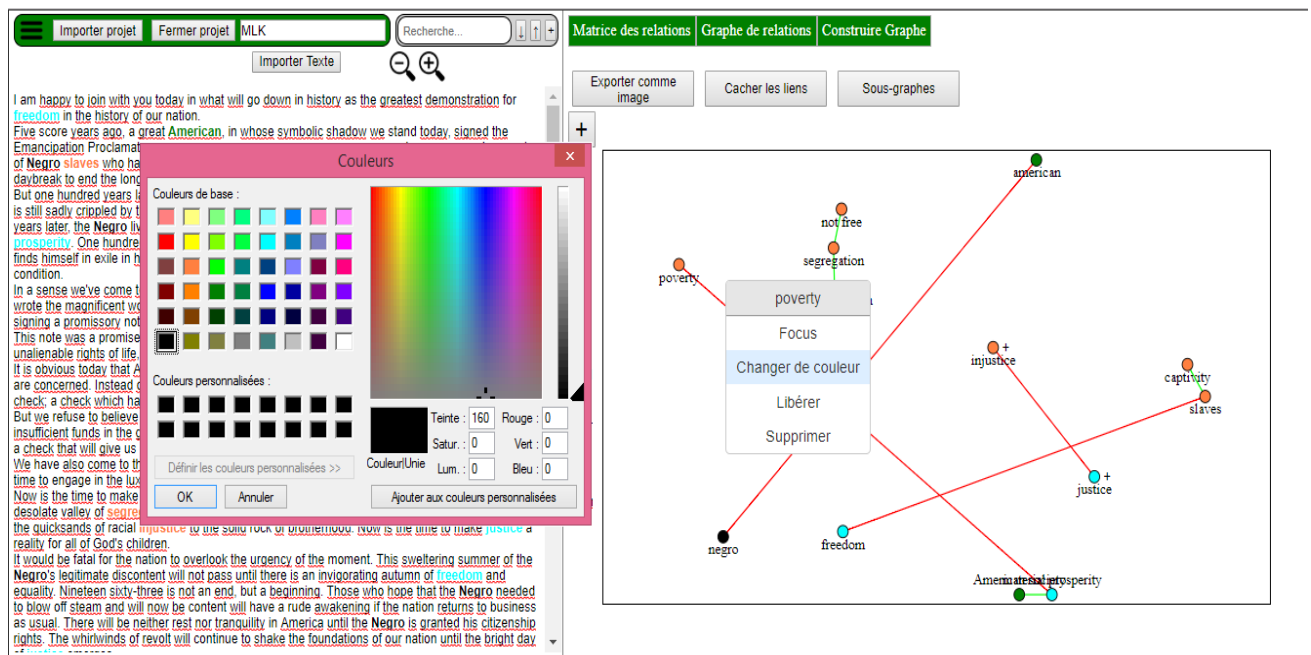


Figure 7: changing the color for node “Poverty”

When a node color is changed, the color of the corresponding term changes automatically in the term dictionary, in the text and on the visualizations.


- **Remove a node**

To remove or delete a node, the user right-clicks on the node and clicks on ‘delete’ in the displayed context menu and confirms. When a node is deleted, all the relations concerned by the corresponding terms are deleted. Therefore the terms dictionary and the relation dictionary as well as the visualizations are updated.

- **Remove a link**

To remove or delete a link, the user right-clicks on the link and clicks on ‘delete’ in the displayed context menu. This deletes the corresponding relation and updates dictionaries and visualizations.

- **Add new relations**

The user has to click on the  “add button”. Figures 8 (a) and (b) show an example of adding relation to a graph.

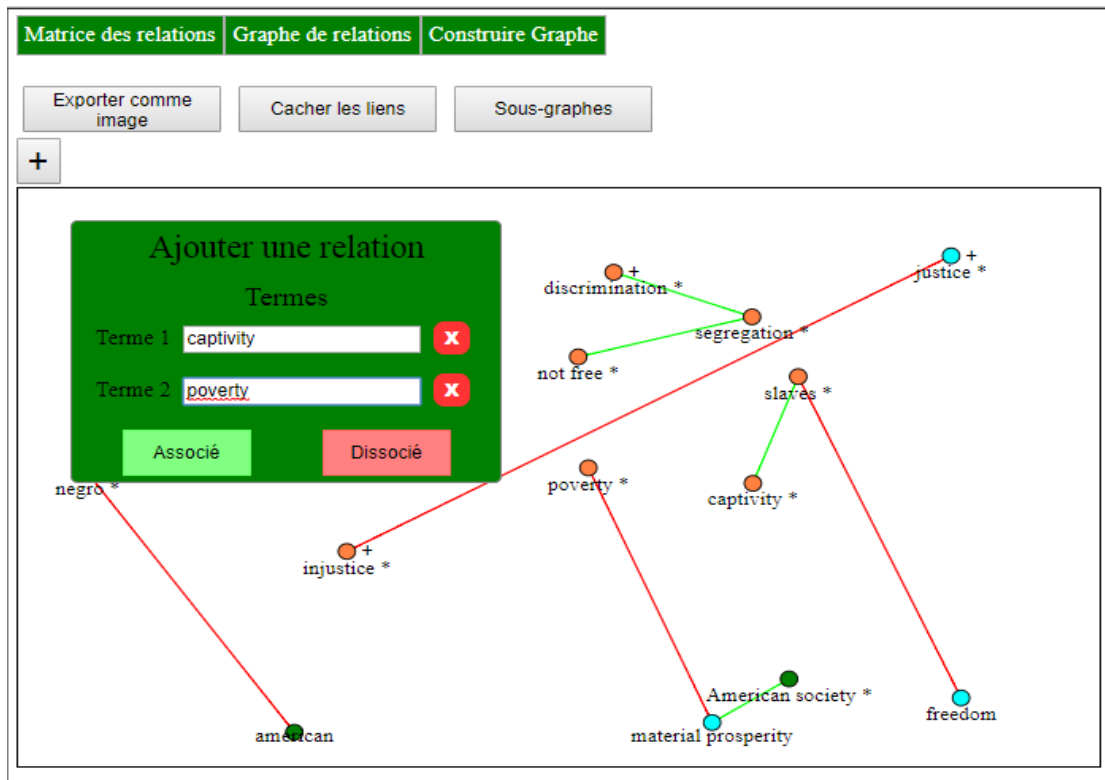


Figure 8(a): adding new relation to the node-graph

If the user clicks on “**Associé**” button an association relation will be created between *captivity* and *poverty* and the node-graph will display as shown in figure Figure 8(b)

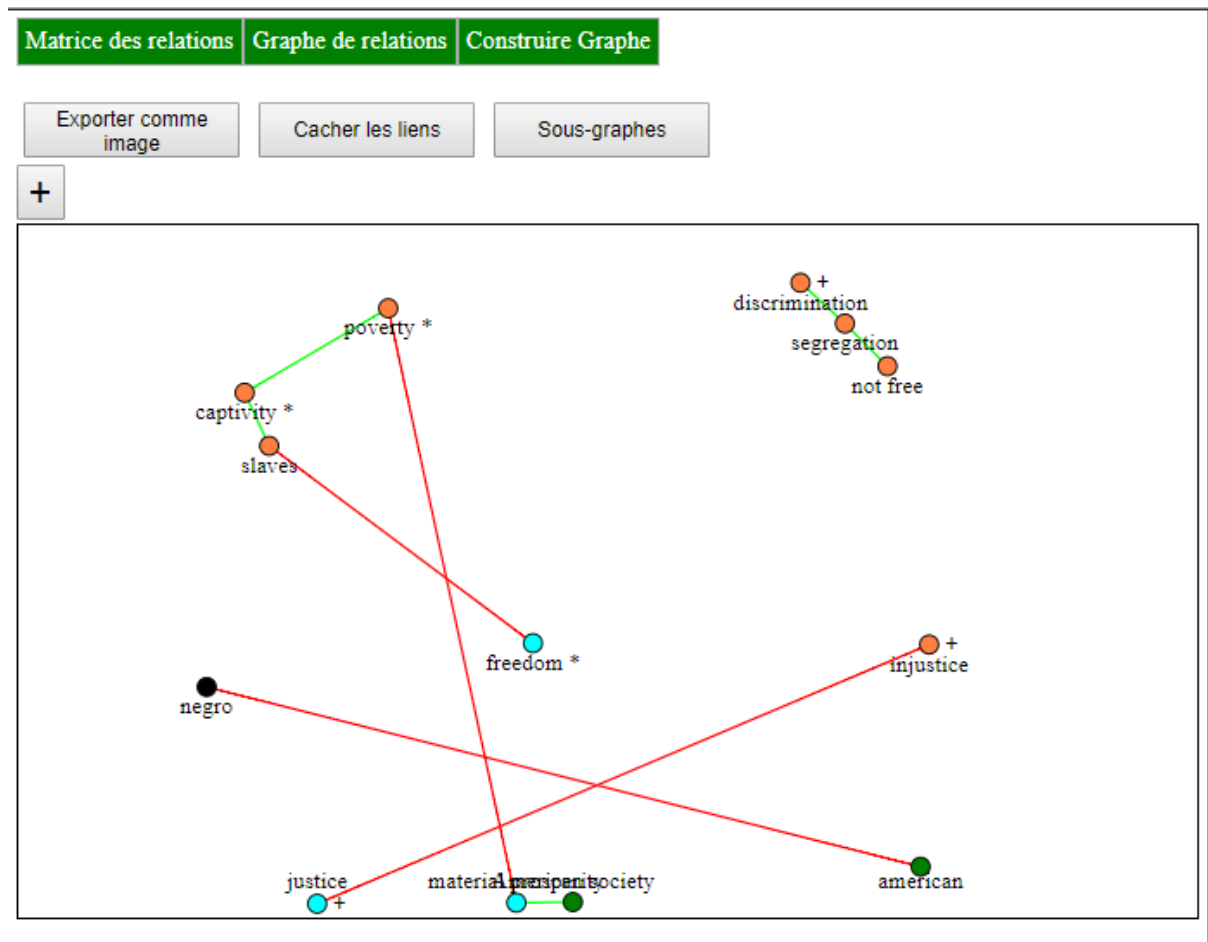


Figure 8(b): association link created between *poverty* and *captivity*

- The **Export image button** helps to export the node graph diagram as an image.
- The **Hide links button** hides the links of the graph when clicked on.