

Peritext

Publication multimodale
orientée contexte

Une introduction

MUSEI
WORMIANI
HISTORIA
LUGD-BATAVORUM
EX OFFICINA ELSEVIRIANA
Acad Typis & 1655.

Peritext - publication multimodale orientée contexte

Une introduction

Robin De Mourat

Sommaire

Introduction	7
Le format de données peritext	13
L'écosystème des modules peritext	18
Des systèmes documentaires riches et exploitables	19
Un design adaptable et modulaire	21
Contextualiseurs	23
Contextualiseur bib	24
Contextualiseur glossary	25
Contextualiseur webpage	26
Contextualiseur embed	29
Contextualiseur vidéo	33
Contextualiseur dicto	34
Contextualiseur codefiles	36
Contextualiseur table	38
Contextualiseur vegalite	39
Contextualiseur p5 (processing)	43
Contextualiseur data-presentation	45
Gabarits	50
Générateurs	51
Éditeur Ovide	53
Contribuer à peritext	54
À propos	57
Références	58
Glossaire	62
Index des auteurs	63

Introduction

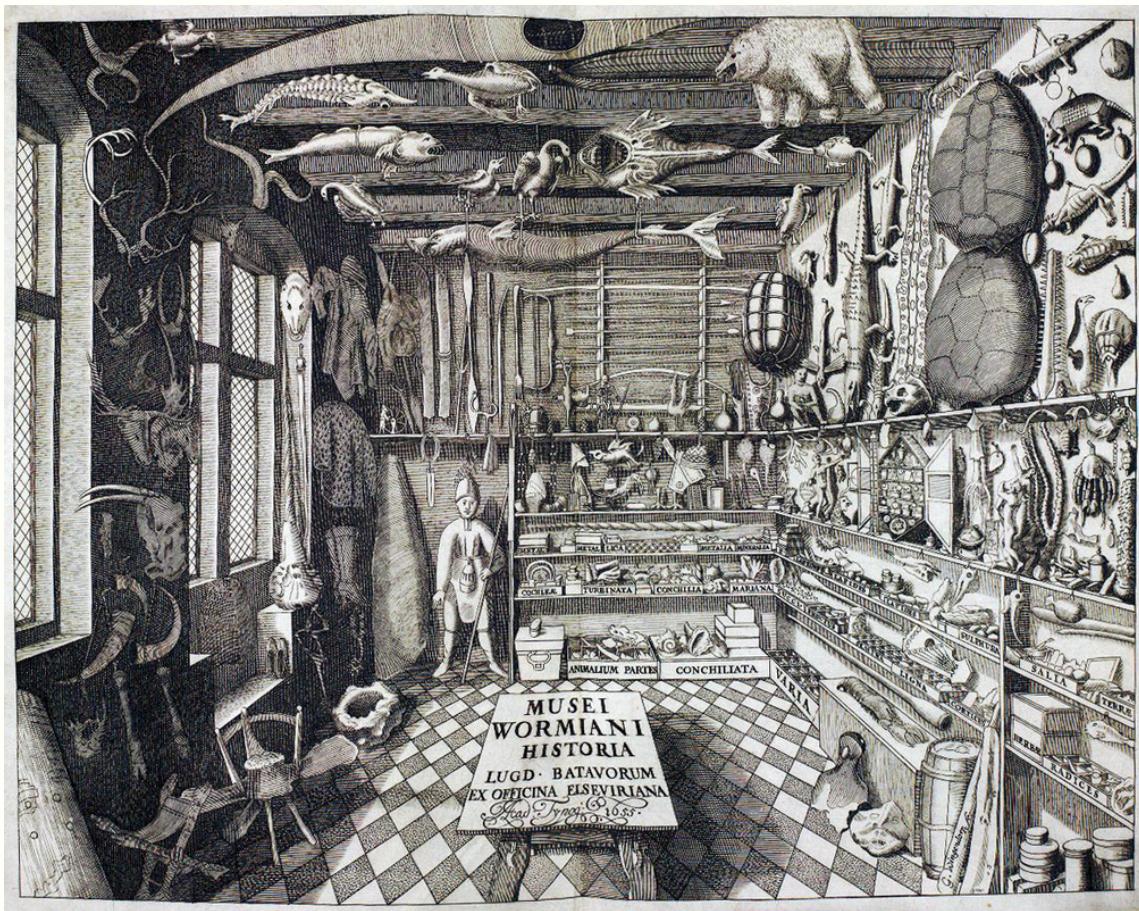
Ce document est dédié à la présentation (et la démonstration) de l'écosystème Peritext, un projet ouvert et libre développé pour supporter des pratiques de publication académique tirant parti des données du web et autres ressources numériques convoquées dans les recherches, et autorisant une publication à haute qualité graphique et éditoriale sur toute la palette des formats de communication utilisés par les chercheurs aujourd'hui.

En quoi consiste Peritext ? pour commencer, à produire le site que vous lisez en ce moment !

Pourquoi Peritext

En amont comme en aval de l'acte de publication, les textes produits et échangés par les chercheurs universitaires convoquent d'une manière de plus en plus intime les environnements numériques (au premier rang desquels le web) dans lesquels ils sont inscrits.

On assiste en effet, dans un large spectre de disciplines comme de sujets de publication, à la prolifération d'une variété d'objets hétéroclites qui participent de manière de plus en plus active aux processus de recherches. En témoignent des exemples aussi divers que l'utilisation croissante d'enregistrements audiovisuels en ligne produits ou analysés dans un contexte scientifique, l'utilisation des réseaux sociaux et de twitter comme élément de corpus et/ou de restitution des débats en cours, la mobilisation d'archives et de dépôts de sources numérisées rendant possible leur présentation sous forme diagraphmatique ou tabulaire, etc.



Le cabinet d'Ole Worm

Reproduction de la gravure conservée à Modène représentant un cabinet de curiosités

Source: *Bibliothèque Estense, Modène.*

Cela dit, dans les formats les plus répandus de l'édition scientifique, les traces du web et autres ressources numériques peinent encore à sortir de la marge des pages : on les remarquera subrepticement au détour d'une note de bas de page (sous la forme d'un discret hyperlien), ou timidement reproduites dans une annexe secondaire, à la fin d'une entrée bibliographique ou dans la légende d'une figure, etc. Ces connexions à des éléments externes désignent des ressources riches, dont le contenu contribue aux arguments développés, et est souvent disponible en intégralité en ligne par ailleurs. Elles restent pourtant souvent sur le seuil des échanges et des arguments, notamment du fait qu'elles s'effacent des documents une fois traduites et formatées selon les conventions de publication instituées (articles, monographies, actes, mémoires, thèses, ...).

D'un autre côté, l'essor de la publication en ligne, du mouvement de l'accès ouvert au résultats de la recherche, et de nouvelles pratiques de communication de la part des chercheurs eux-mêmes qui vont de l'usage des réseaux sociaux à l'expérimentation de diverses formes de publication en ligne, ont instauré un régime de production et de circulation des documents académiques extrêmement riche où ces derniers se voient discutés, critiqués, reformatés et éditorialisés par-delà une multitude de mondes scientifiques, sociaux et médiatiques.

Il est donc nécessaire d'inventer de nouvelles manières de connecter les modes de production des documents académiques avec les contextes du web. Cette recontextualisation doit opérer dans les deux sens : **les environnements numériques et toutes les ressources et formes qu'ils proposent doivent être plus significativement convocables dans les documents académiques, et les documents académiques doivent être plus significativement contextualisables dans les environnements numériques.** C'est là le projet à la base de Peritext.

Pour quels contextes Peritext est-il fait (et pas fait) ?

Peritext ne s'adresse pas à toutes les pratiques de publication académique, il n'entend en aucun cas proposer une solution totale prenant en charge l'intégralité d'un processus d'édition académique, ou couvrir la totalité des contextes de projet scientifiques ou éditoriaux.

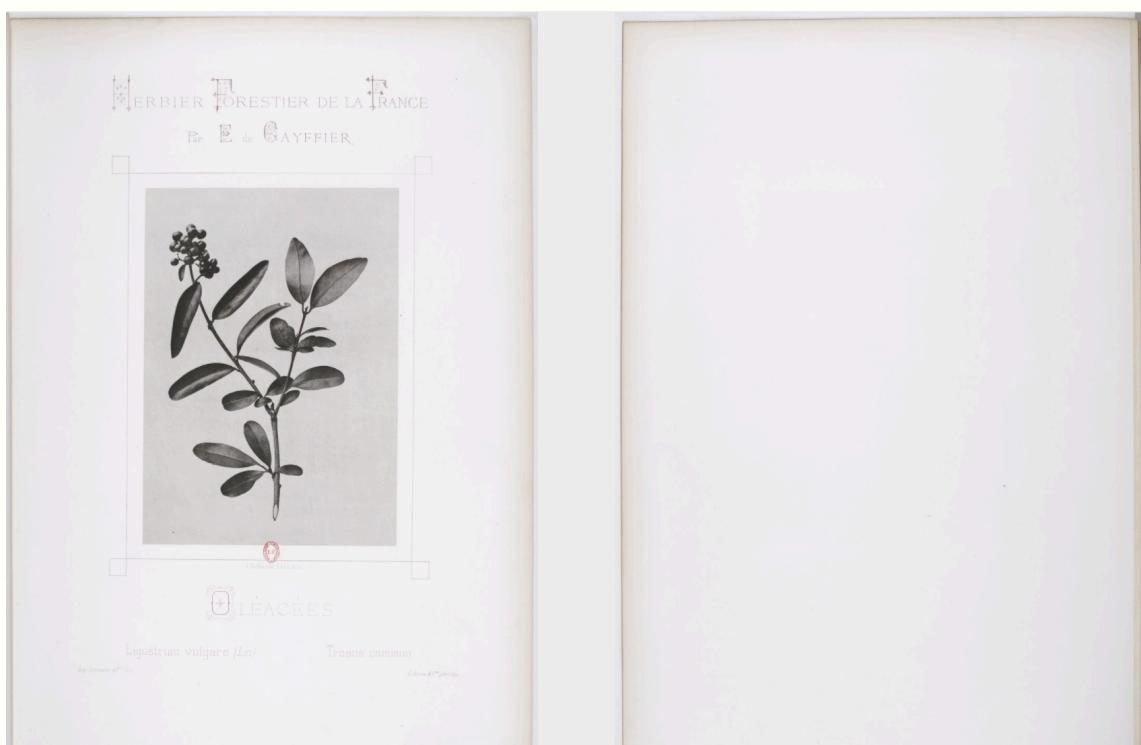
Il s'agit plutôt d'un écosystème ouvert (c'est-à-dire fait pour se connecter à un archipel de normes, outils, pratiques et projets existants), conçu pour des projets qui se placent à la marge des mondes académiques, et à la marge des formes rhétoriques propres aux formats établis, fortement (et souvent exclusivement) fondés sur une argumentation ou un compte-rendu sous forme textuelle.

Peritext pourrait donc être pertinent pour vous si vous remplissez au moins deux des conditions suivantes :

- **Vos recherches convoquent de manière très importante des documents extérieurs** (tableaux de données, médias, sites web, ...) et votre activité consiste notamment à les analyser et les commenter en profondeur
- **Vous désirez publier au-delà des formats traditionnels associés à la promotion et la reconnaissance officielle des chercheurs**, notamment en ayant une activité sur le web
- **Vous êtes attaché à la qualité technique & documentaire des documents que vous produisez**, notamment en termes de métadonnées et d'indexation
- **Vous êtes attaché à la qualité graphique et éditoriale des documents que vous produisez** en tant que chercheur, que ce soit vous ou un membre de votre équipe qui s'en occupe

Par ailleurs, Peritext en tant que tel ne prend pas en charge les dimensions suivantes du travail de communication académique, et ne sera donc pas pertinent pour vous si :

- votre projet éditorial mise de manière centrale sur la dimension collective & réticulaire du travail de communication scientifique (commentaires, réactions, collaboration, ...). Tout cela peut se passer autour d'un document peritext, mais pas en son sein. Ce point ne devrait pas évoluer dans le futur du projet, et pour l'instant l'utilisation d'outils tiers tels qu'Hypothes.is ou disq.us est favorisée pour cela.
- vous comptez expérimenter avec des structures rhétoriques qui se départissent de l'organisation traditionnelle des documents académiques, faits de sections (chapitres, parties, ...) se suivant de manière linéaire, et d'un hypertexte assez simple permettant de produire une table des matières, un glossaire, et une liste de référence. Ce point est soumis à une probable évolution ultérieure.
- vous comptez expérimenter de formes de publication académique qui ne prennent pas le texte pour médium principal et structurant pour l'argumentation (organisées autour d'une vidéo ou d'une image par exemple)



Herbier forestier de la France. Reproduction par la photographie... des principales plantes ligneuses qui croissent spontanément en forêt. Description botanique...

Source: Gallica

Qu'est-ce que Peritext ?

Peritext est d'abord une **spécification technique** pour la description de documents académiques visant à terme à devenir un format ouvert et interopérable⁽³⁰⁾. Autrement dit, il s'agit d'un mode d'emploi expliquant comment décrire le contenu d'un document académique d'une façon conventionnelle & prévisible. Ce dernier décrit comment les données d'un récit écrit par un chercheur avec un outil compatible doivent être structurées, et contribue à produire un *format de données* particulier, le *format Peritext*.

Autour du *format Peritext* s'articulent un ensemble de modules technologiques et des applications directement utilisables qui en actualisent les potentialités et lui donnent du sens⁽³¹⁾. Ces modules technologiques ont tous en commun le fait qu'ils visent à aider à automatiser certaines tâches relevant des normes techniques et éditoriales liées à la publication académique contemporaine⁽²⁾, tout en ménageant beaucoup de place et de souplesse pour l'affichage de contenus non-conventionnels et leur design en accord avec l'argument fait par le texte.

Une écriture et une édition multi-supports

Le premier atout du format peritext est que sa manière de décrire un document permet de produire une diversité de documents

1. Tous les modules sont écrits en Javascript et utilisent notamment sur la bibliothèque de rendu open source react, qui permet de produire indifféremment du code pour des applications client et des applications serveurs, et pour des environnements interactifs ou des environnements statiques (type générateur de pdf).
2. Des tâches comme la documentation des sources, et la description des métadonnées du document et des documents qu'il cite.

de type "codex" (pdf, epub) comme de type "site web" à partir d'une seule et même représentation des contenus.

Par exemple, ce document même est directement accessible comme un site web, un fichier pdf imprimable, et un fichier epub exécutable sur liseuse numérique, une API de manière à ce qu'il soit utilisable par une autre application, etc.

Peritext rentre donc dans la catégorie de technologies intitulées "COPE" ("Create Once, Publish Everywhere"³) qui consistent à produire une diversité de documents à partir d'un même texte source. Ainsi, avec Peritext les auteur.e.s peuvent écrire une seule fois leur article, chapitre ou livre, et disposer en retour et simultanément de diverses contextualisations de ce contenu pour les différents supports (site web, pdf, epub, ...).

Cela dit, Peritext se distingue une première fois de ses voisins dans la mesure où il permet également aux utilisateurs de se faire "éditeurs-designers" et disposer de divers moyens très détaillés de mettre en forme leur récit pour chacun des supports disponibles, en fonction de leur corpus, de leur argument et de leur projet intellectuel -- ce qui n'est pas le cas de la plupart de ses voisins.

Un modèle uniifié et souple pour gérer les arguments centrés sur des documents et autres éléments extérieurs

Le second atout du format Peritext réside dans sa vocation à soutenir la convocation de nombreux éléments extérieurs dans la structure argumentative d'un texte académique (références, données, documents audiovisuels, etc.). Pour ce faire, il repose sur un mode de représentation des contenus particulier, centré sur la séparation entre les ressources convoquées dans un récit et les différentes formes que ces dernières peuvent prendre dans le cadre de l'argumentation opérée par un texte.

Pour en savoir plus, rendez-vous à la section "Le format de données peritext".

Respect des normes techniques et éditoriales de la publication académique

Sur le plan documentaire, peritext prend en charge une série de nécessités spécifiques à la publication académique.

Pour en savoir plus, rendez-vous à la section "Des systèmes documentaires solides et faciles à mettre en place".

Adaptation et modularité

Peritext est pensé selon une logique de "boîte ouvrable", c'est-à-dire que ses différents modules fonctionnent avec des paramètres par défaut mais sont par ailleurs profondément paramétrables par les utilisateurs qui le souhaitent.

Outre les paramètres exposés par chacun des modules, peritext prend en charge une diversité de langages permettant de préciser et de modeler les contenus avec beaucoup de liberté, comme par exemple le langage markdown pour certains contextualiseurs et les langages de mise en forme que sont css et css print qui peuvent être utilisés de manière distincte pour chacun des types de supports produits à partir d'un contenu.

Enfin, il est construit selon une logique modulaire qui permet beaucoup de souplesse et de variations dans l'implémentation d'un écosystème éditorial spécifique (en termes de supports visés, de types de documents pris en charge, ...).

Pour en savoir plus, rendez-vous à la section "adaptation et modularité".

Économie de moyens technique

La représentation d'un document peritext est faite au **format json**, ce qui veut dire qu'elle ne nécessite pas de base de données et peut être stockée dans un simple fichier. De la même manière, les modules composant peritext ont vocation à être assemblés à l'intérieur d'une application javascript, pouvant être déployée sur quantité de services gratuits.

3. "Créer une seule fois, publier partout.". Voir par exemple la technologie Pandoc ([25](#)) dans ce registre.

Enfin, le générateur de site web actuel produit un répertoire de fichiers html simples, qui peuvent être exportés sur un serveur ftp, ou sur un service d'hébergement gratuit tels que github pages ou surge.

En conclusion, même s'il peut être intégré dans des **Très Grandes Infrastructures de Recherche** ou des plateformes de grande envergure, l'écosystème peritext est d'abord pensé pour des contextes d'implémentations légers, décentralisés et peu coûteux en équipement.

Pour en savoir plus, rendez-vous par exemple à la section "générateurs".

Le format de données peritext

Peritext est d'abord une **spécification technique** pour la description de documents académiques visant à terme à devenir un format ouvert et interopérable. Autrement dit, il s'agit d'un mode d'emploi expliquant comment décrire le contenu d'un document académique d'une façon conventionnelle & prévisible. Ce dernier décrit comment les données d'un récit écrit par un chercheur avec un outil compatible doivent être structurées, et contribue à produire un *format de données* particulier, le *format Peritext*.

Le schéma de données du format peritext est dédié à la description de "récits" académiques, qui peuvent correspondre à des formats traditionnels aussi variés que :

- un article de revue académique
- une monographie
- des actes de colloque
- un numéro de revue académique

... ou à des formes ne correspondant à aucun des formats de publication académique établis.

Techniquement, peritext se fonde sur une description au **format json**, qui permet de décrire un objet selon une structuration complexe faite d'éléments imbriqués, et de stocker cette description dans de simples chaînes de caractères et autres fichiers de texte.

La spécification exacte du format, actuellement encore en gestation⁴, est définie dans un premier module du projet intitulé *peritext-core* (30)⁵.

Les paragraphes qui suivent décrivent et justifient les grandes lignes du format peritext. Pour une idée plus précise et technique du schéma, voir le répertoire *peritext-core* ou sa documentation technique (31).

Caractéristiques générales du format peritext

Le format peritext est d'abord structuré autour des aspects suivants :

- un récit Peritext est décrit par une série de métadonnées qui décrivent : son titre et sous-titre, son ou ses auteurs, etc.
- un récit Peritext est structuré en *sections*. Une section peut être une partie, un chapitre, une introduction, des remerciements, ... les sections sont assemblées dans un ordre linéaire pour construire le *sommaire* du récit. Elles peuvent disposer d'un niveau hiérarchique qui permet de définir les sections les plus générales et les sections d'importance moindre.
- chaque section dispose de ses propres métadonnées (titre, auteurs, ...)
- chaque section dispose d'un contenu structuré autour d'un texte principal et d'une série de notes.
- les contenus d'une section ou d'une note en particulier sont aussi représentés sous la forme de blocs de textes annotés avec des styles et des entités spécifiques selon une norme établie⁶.
- outre les sections et leur contenu, au niveau d'un récit peritext sont également définis des paramètres (*settings*) qui définissent des informations relatives à l'affichage du récit telles que : le gabarit à utiliser, les règles de citation, des paramètres tels que la position des notes, et enfin le code css à injecter pour chacun des types de supports
- enfin, un récit peritext est décrit par une série d'objets relatifs à la contextualisation d'éléments extérieurs dans

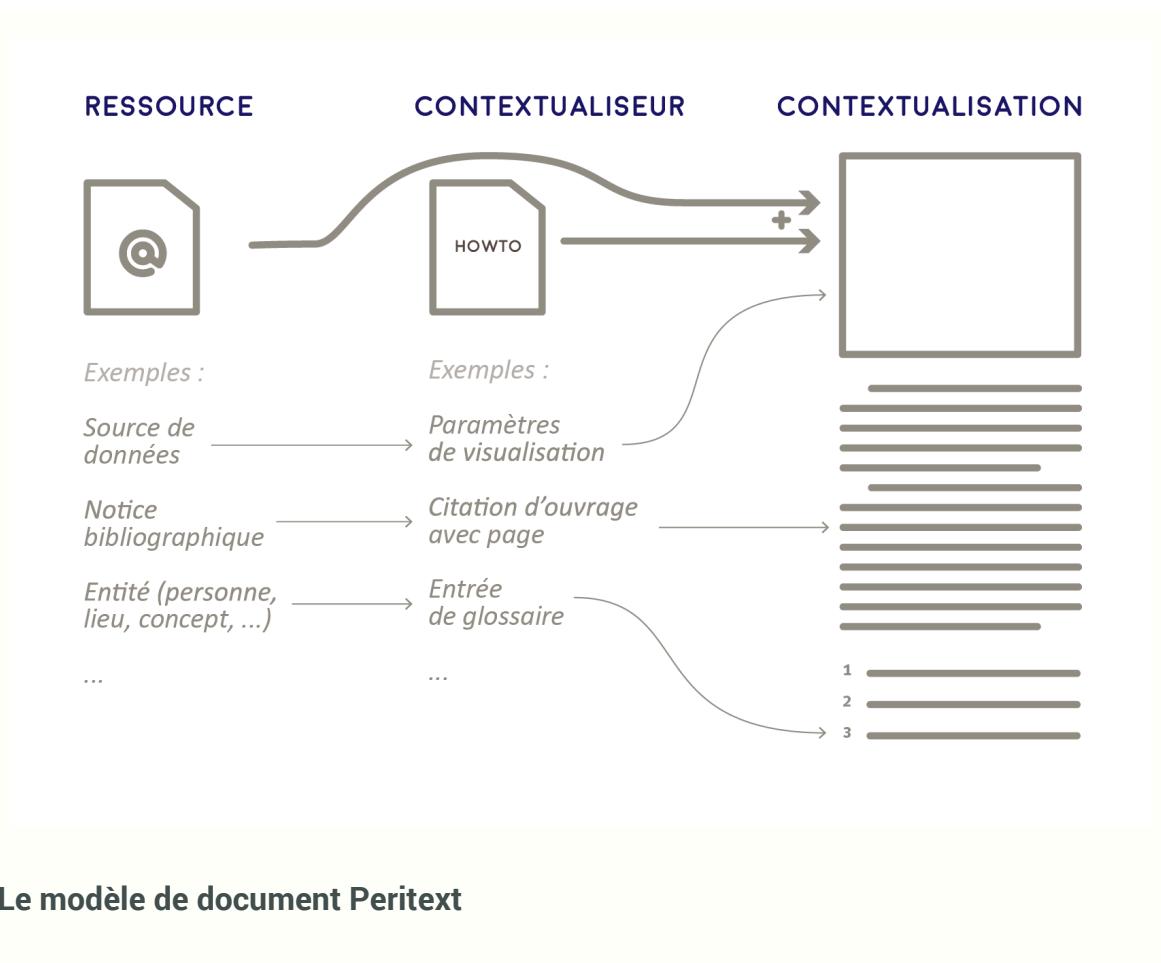
4. C'est pourquoi tous les modules du projets sont pour l'instant à l'état de version "pre-alpha", c'est-à-dire qu'ils sont instables & non utilisables pour des situations réelles car appelés à évoluer de manière non-compatible avec leur version actuelle.

5. Peritext-core est notamment dédié à décrire le format peritext au moyen de la spécification json-schema développée par l'Internet Engineering Task Force (32).

6. Ce type d'annotation s'adosse à la spécification open source draft-raw développé par l'équipe de développement de l'entreprise Facebook, qui sert à décrire au format json des contenus textuels "riches" pour des applications utilisant la technologie de rendu html intitulée react, qui est utilisée dans tous les modules de l'écosystème peritext. Voir le site du projet draft-js pour plus d'informations (28).

l'argument : les ressources, les contextualiseurs et les contextualisations

Caractéristiques spécifiques liées à l'argumentation fondée sur les documents : le modèle RCC



Le modèle de document Peritext

Peritext tire sa spécificité de sa vocation à soutenir la convocation de nombreux éléments extérieurs dans la structure argumentative d'un texte académique (références, données, documents audiovisuels, etc.).

Pour ce faire, dans un récit Peritext, la convocation d'éléments extérieurs dans l'argument du récit s'effectue via trois types d'objets :

- **les ressources** décrivent tous les éléments à convoquer dans le texte, selon un **registre documentaire** indifférent à la forme qu'ils prendront dans le récit. Les ressources peuvent désigner des éléments très variés dans leur nature (références, données, médias, mais aussi entités de glossaire telles que personnes, lieux, thèmes, ou encore morceaux de code, ...)
- **les contextualiseurs** décrivent un ensemble de paramètres permettant d'afficher une ressource d'une certaine manière. Ils agissent sur un **registre rhétorique** pour préciser comment la ressource doit être affichée.
- **les contextualisations** enfin sont la conjonction d'une ressource et d'un contextualiseur à un endroit précis de l'argument. Elles sont par ailleurs assorties d'un titre et d'une légende.

Le modèle de données de Peritext est donc intitulé RCC car il sépare la liste des Ressources utilisées dans le documents,

les Contextualisations de ces ressources à des endroits spécifiques et les Contextualiseurs qui les font s'afficher d'une manière spécifique.

Une démonstration du modèle RCC

George, chercheur en archéologie, explique le résultat de sa dernière expédition en Egypte via un tableau représentant la quantité de vases retrouvés sur une liste de sites qu'il a listés par lettres ("A", "B", "C", ...)

Au moyen d'un éditeur compatible Peritext comme Ovide (7) , George stocke d'abord son résultat (consigné avec un autre logiciel type excel dans un fichier de tableur) dans une ressource de type "table", c'est-à-dire un simple tableau de données.

Il est d'abord possible de la contextualiser au moyen d'un contextualiseur de type "table", qui est le contextualiseur par défaut pour ce type de ressource. Ce dernier consiste à représenter la ressource sous la forme d'un tableau. En l'utilisant pour contextualiser sa ressource dans le paragraphe qui suit, George obtient le tableau suivant :

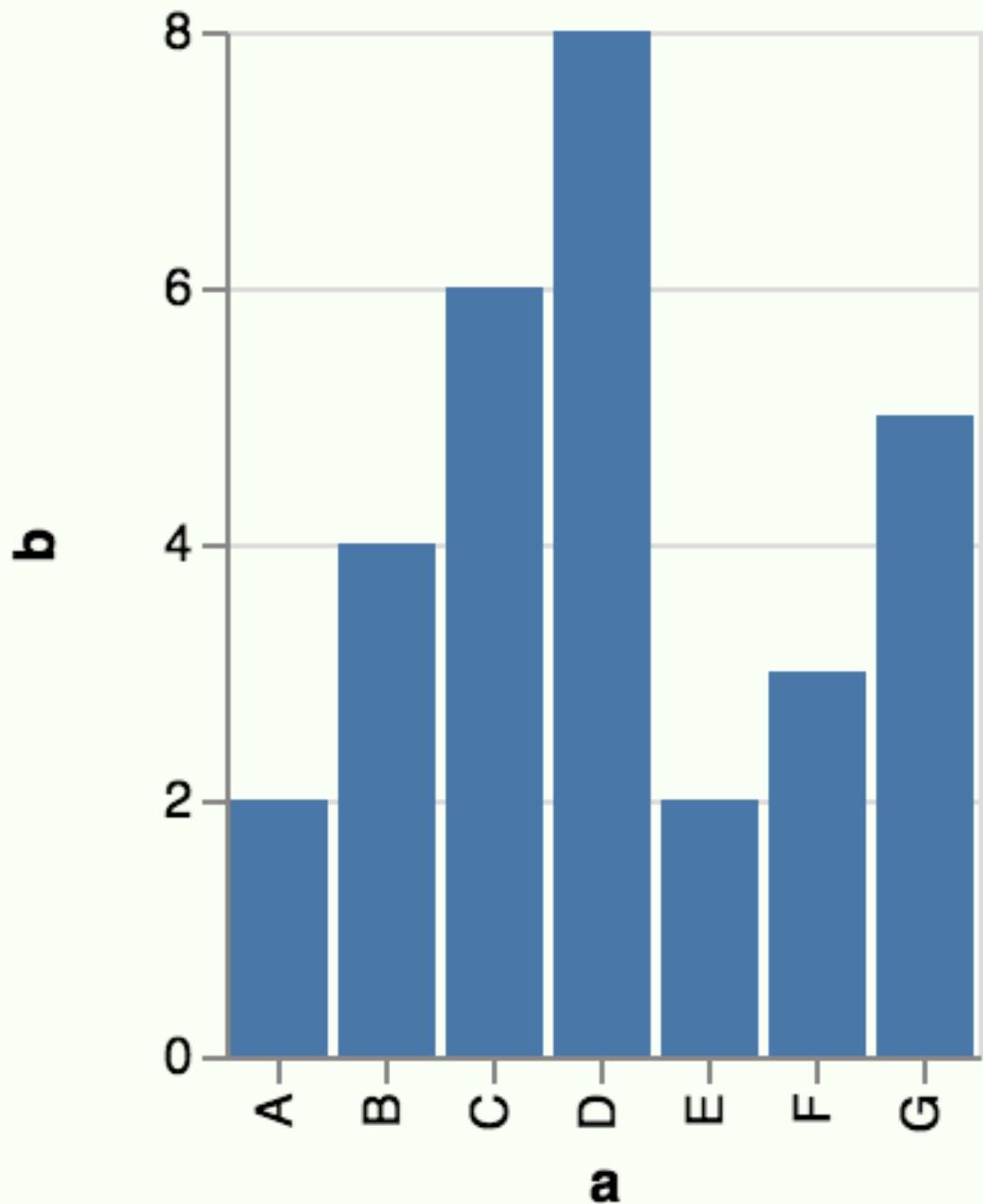
a	b
A	2
B	4
C	6
D	8
E	2
F	3
G	5

(exemple) une ressource contextualisée en table

Les vases de George représentés comme un tableau.

Maintenant, George souhaite à nouveau faire mention de cette ressource sans pour autant souhaiter l'afficher à nouveau sous la forme d'un encombrant tableau. Pour ce faire, il va utiliser le contextualiseur "bib" qui contextualise la ressource sous la forme d'une référence bibliographique. C'est ce qui va être fait à la fin de ce paragraphe(26).

Enfin, mettons que notre auteur décide maintenant de discuter de la comparaison des valeurs numériques décrites dans les données de cette ressource. Il utilisera alors un troisième contextualiseur de type "vegalite", qui permet de construire des visualisations de données. C'est ce qui va être fait dans le prochain paragraphe :



(exemple) La ressource contextualisée avec vegalite

Les vases de George représentés comme un histogramme.

C'est ainsi que le modèle RCC implémenté par le format Peritext permet :

- de représenter les éléments convoqués dans un argument de plusieurs manières différentes, selon les besoins de l'argumentation
- et de paramétrier finement la manière dont on veut voir affichée une ressource grâce aux paramètres des contextualiseurs et à l'injection de code css via l'éditeur
- tout en maintenant une rigueur documentaire qui pourra être exploitée ensuite pour l'indexation du document et pour la génération d'une interface utilisateur exploitant le système hypertexte ainsi créé de manière homogène malgré la diversité des contextualisations possibles. En effet, malgré la diversité des diverses formes de contextualisations pouvant être appliquées à une ressource donnée, le lien avec cette dernière restera toujours présent, permettant par exemple de produire une liste de mention ou une liant ressources et contextualisations. Ainsi, il est à noter que dans le présent gabarit de présentation en cliquant sur l'icône en dessous de chacune des contextualisations il est possible de prévisualiser toutes les autres contextualisations de la même ressource dans le récit et d'y accéder.

Remarque sur les principes du format peritext et sa relation à la forme

Dans une application d'édition suivant le modèle de données Peritext, l'auteur a donc non seulement la main sur les éléments extérieurs qu'il veut citer dans son texte, mais également sur la manière de les afficher et sur un ensemble de paramètres lui permettant de modeler précisément l'usage qu'il veut en faire dans son argument (par exemple : préciser un passage de texte, mettre en lumière une certaine dimension d'un jeu de données, spécifier un cadrage particulier sur une visualisation, etc.). **Il a ainsi ainsi la possibilité de développer un fil argumentatif pleinement articulé autour de la mobilisation des ressources qu'il convoque.**

Peritext tente de se départir de la traditionnelle distinction entre les modèles de description centrés sur les contenus (*What You See is What You Mean*⁷) et ceux centrés sur la forme propre à un type de support pour le document (*What You See is What You Get*⁸). Il s'agit plutôt de modéliser les *intentions argumentatives* attachées à un moment particulier du texte (*What you see is What You Argue*⁹), plutôt qu'une "forme finie" ou un "contenu sans forme". La notion de *représentation intermédiaire* joue ici un rôle crucial, comme nom de cet état intermédiaire des contenus décrits selon le format Peritext.

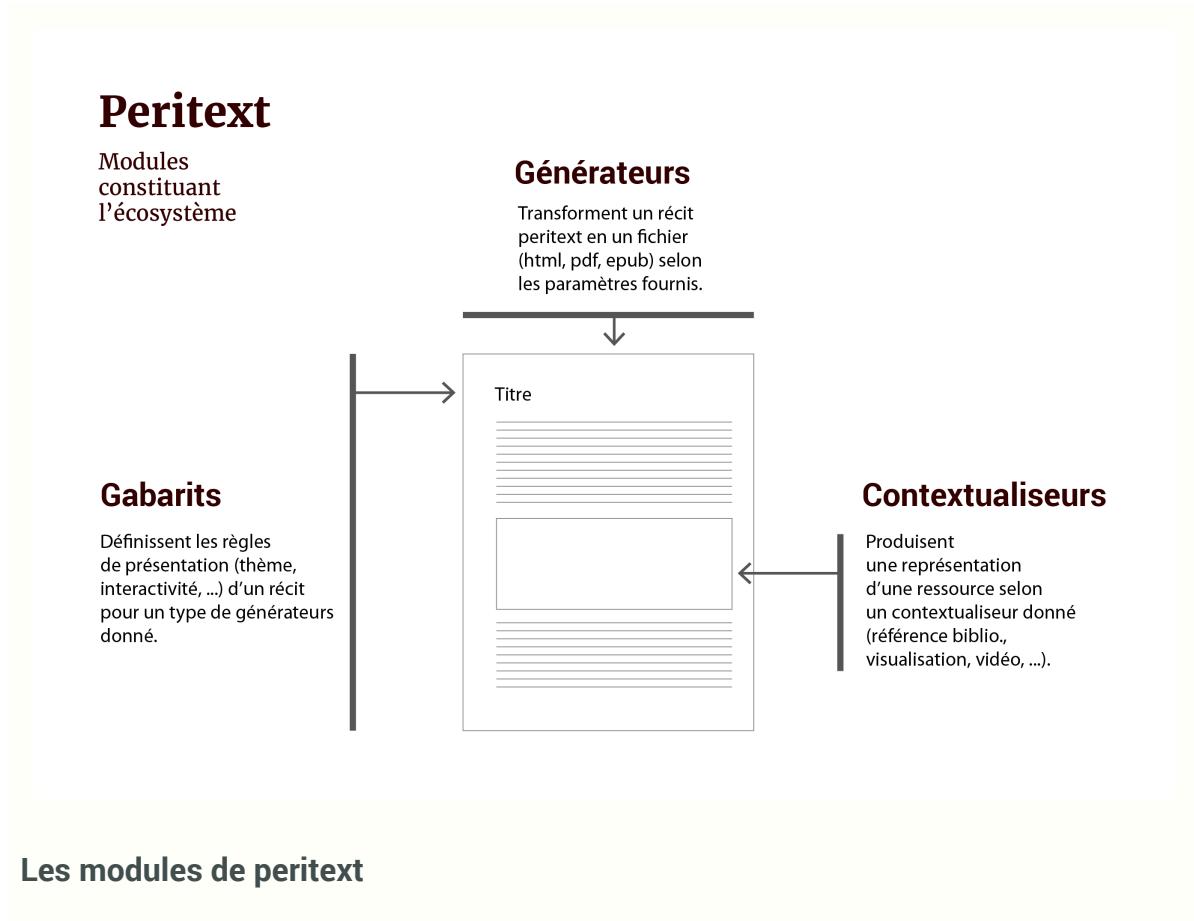
7. "Ce que vous voyez est ce que vous voulez dire". Voir des formats comme

8. "Ce que vous voyez est ce que vous obtenez" (à l'impression). Voir des formats comme word dans cette catégorie.

9. "Ce que vous voyez est ce que vous argumentez". Nous proposons cette expression pour décrire cette troisième voie qui par se référence à la rhétorique ne sépare pas la question de la forme de la question du sens.

L'écosystème des modules peritext

Autour du format Peritext s'articulent un ensemble de modules technologiques qui le comprennent et en exploitent les potentialités. Conceptuellement, ces modules correspondent chacun à l'une des dimensions entrant en jeu dans la mise en forme multimodale d'un récit peritext. Concrètement, ils correspondent à des bibliothèques javascript structurées selon un modèle normalisé qui permet la compatibilité entre les modules de l'écosystème.



Les modules de peritext

Les modules générateurs sont des modules qui "consomment" un récit pour le transformer en un document utilisable (un fichier pdf par exemple). Outre le récit à transformer, ils "consomment" également d'autres modules -- les gabarits et les contextualiseurs -- pour définir comment transformer les contenus et les contextualisations et organiser le document.

Les modules gabarits sont des modules qui définissent le thème visuel, l'organisation des contenus et les mécanismes d'interaction à utiliser par un générateur pour transformer un récit. En somme, ils définissent toute la mise en forme du récit, excepté les contextualisations.

Les modules contextualiseurs sont des modules qui définissent comment interpréter une série de paramètres de contextualisation contenues dans un objet contextualiseur (voir la section "le format de données peritext") selon le contexte d'implémentation (comme paragraphe, dans un paragraphe) et selon le générateur utilisé.

Des systèmes documentaires riches et exploitables

Le modèle RCC implanté dans Peritext (voir la section "le format de données Peritext") induit une haute qualité de description des ressources impliquées dans la production des documents peritext. Cette section décrit comment cette richesse est exploitée dans les modules constituant Peritext.

Gestion des références bibliographiques

Peritext s'adosse au standard international Citation Style Language ([21](#)) pour gérer les références. Il est ainsi capable de gérer la plupart des formats et langues de citation en utilisation. Les modules de l'écosystème permettent par ailleurs de prendre en charge n'importe quel type de ressource Peritext sous la forme d'une citation bibliographique.

L'éditeur démonstrateur Ovide (voir section afférente) propose par ailleurs une interface d'import de références au format BibTeX et une interface expérimentale d'ajout de références par interface graphique.

Métadonnées à l'échelle d'un récit et de ses sections

Dans les gabarits par défaut de l'écosystème de Peritext, les métadonnées renseignées au niveau des récits et des sections d'un document peritext sont mises à profit de la manière la plus complète possible.

Ainsi, les fichiers de sortie html de peritext présentent des métadonnées encodées au format DublinCore, Twitter, Open Graph & schema.org.

Métadonnées à l'échelle des contenus

À l'échelle des éléments affichés par les générateurs de Peritext (et notamment les contextualisations de ressource), Peritext entend fournir le meilleur degré de description machine possible.

Pour ce faire, chaque mention de ressource inclue l'imbrication automatique d'un composant COINS ([20](#)) permettant par exemple de l'enregistrer avec le plugin Zotero.

Est prévue dans la feuille de route des modules une implémentation ultérieure des normes d'encodage sémantique au niveau des composants html eux-mêmes (au moyen du format RDFa) afin de permettre la lecture optimale d'un document peritext par une machine.

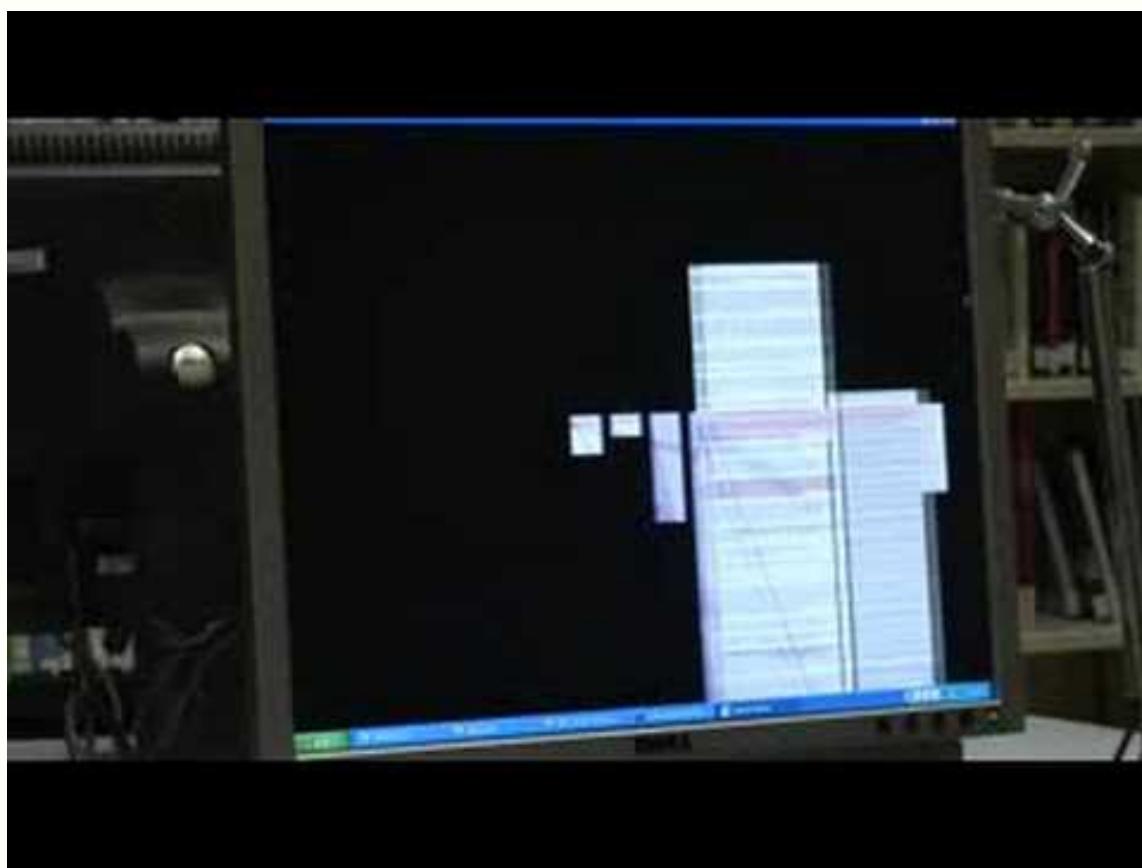
Systèmes hypertextuels à double-sens

Peritext est attaché à une utilisation optimale de la technologie hypertexte, initialement pensée pour les documents académiques ([29](#)). Sans explorer trop avant les possibilités rhétoriques et organisationnelles offertes par cette technologie dans le contexte académique¹⁰, Peritext tente d'optimiser les formes d'hypertexte existant dans les documents traditionnels (liste de références, glossaire, index d'auteurs) en les opérationnalisant techniquement comme des liens à double sens à tous les endroits où ils sont implémentés.

Ainsi, par exemple, une référence à une entrée de glossaire permettra toujours d'accéder à la liste de toutes les mentions de cette entrée, et cette liste permettra dans l'autre sens d'accéder à la localisation de chacune de ces mentions.

10. Voir le projet Scalar ([19](#)), qui expérimente des formes alternatives de structuration des arguments académiques en tirant parti des technologies de base de données relationnelles.

Peritext développe ici certaines idées éprouvées dans de précédentes expérimentations en Humanités Numériques¹¹ mais également dans des expériences plus anciennes telles que le projet Xanadu qui envisageait un système hypertexte permettant de parcourir les liens à double-sens¹²



Ted Nelson demonstrates Xanadu Space

Ted Nelson présente le logiciel Xanadu Space qui spatialise en trois dimensions un système hypertexte.

Source: photonhunter ([youtube: https://www.youtube.com/watch?v=En_2T7KH6RA](https://www.youtube.com/watch?v=En_2T7KH6RA))

Pour voir une concrétisation de ce principe, rendez-vous dans les sections "glossaire" et "références" de ce site web, qui sont non seulement pointées par des éléments du texte, mais également permettent d'accéder à la liste des mentions associées aux différentes ressources citées dans le texte depuis ces dernières.

Par ailleurs, pour information et comme démonstration d'une opérationnalisation possible de ce principe des liens à double sens, il est possible de cliquer sur les citations présentes dans cette page pour voir apparaître sur la colonne de droite une liste des mentions des ressources associées ailleurs dans le document.

11. Voir par exemple le projet AIME [\(10\)](#).

12. Voir notamment le project Xanadu [\(34\)](#) mis en place par Ted Nelson.

Un design adaptable et modulaire

Peritext est un projet ouvert, pensé selon une logique modulaire permettant une multitude de variations et d'adaptations autour d'un même modèle de données interopérable, celui du *format peritext*.

Le projet présente divers niveaux d'ouverture pour son adaptation aux besoins de chacun, qui correspondent aussi à différents niveaux d'expertise. Derrière ces choix de conception réside le projet de permettre un meilleur design des arguments académiques, dans toutes leurs spécificités et leurs complexités, et ce à travers l'ensemble des médias qui constituent aujourd'hui la place publique des échanges intellectuels.

Niveau d'adaptation 1 : mise en forme paramétrique et CSS

L'adaptation de la forme des récits peritext est d'abord permise par un ensemble d'options et de paramètres associés aux documents (exemple: position des notes) et des contextualiseurs utilisés dans un récit. Ceux-ci dépendent du **gabarit** utilisé pour représenter un récit.

Il est par ailleurs possible d'écrire du code css spécifique pour chaque récit, et de différencier les styles pour les supports de type "web" et pour les supports de type "codex". Concernant les supports de type "codex", les générateurs en place actuellement dans l'écosystème se plient à la norme css print (3).

Ce premier niveau est accessible à tout type d'utilisateurs, car il est notamment exposé par les éditeurs de contenus Peritext tels qu'Ovide (7).

Niveau d'adaptation 2a : gabarits

Un **gabarit** (ou template) décrit selon quelle forme interpréter les données incluses dans un récit au format peritext.

L'écosystème peritext propose pour l'instant un nombre réduit de gabarit. Il est possible de développer de nouveaux gabarits de mise en forme (sous la forme de composants React) ou de développer des alternatives aux existants, à l'échelle d'un projet en particulier ou comme contribution à l'écosystème général.

Voir la section gabarits pour en savoir plus.

Niveau d'adaptation 2b : contextualiseurs

Les générateurs et les éditeurs de l'écosystème peritext utilisent des modules contextualiseurs pour décider comment afficher une contextualisation donnée pour un type de support donné.

Chaque contextualiseur peut exposer jusqu'à 4 composants distincts suivant le support et la localisation de la contextualisation : comme paragraphe en mode web, dans un paragraphe en mode web, comme paragraphe en mode codex, dans un paragraphe en mode codex.

La logique modulaire des contextualiseurs permet d'implémenter des configurations spécifiques de Peritext en fonction des besoins de chacun (par exemple : plus orienté vers le partage de codes sources, plus orienté vers les médias vidéos et audio, plus orienté vers la visualisation des données, ...).

Il est possible d'adapter Peritext en développant de nouveaux contextualiseurs ou de développer des alternatives aux existants.

Rendez-vous à la section contextualiseurs pour en savoir plus.

Niveau d'adaptation 3 : générateurs

Peritext produit des documents (pdf, epub, site web) au moyen de modules intitulés générateurs qui consomment un récit peritext, des contextualiseurs et des gabarits pour produire un fichier utilisable et partageable. À ce niveau d'adaptation, il est possible de faire à peu près n'importe quoi à partir d'un document encodé au format peritext en écrivant un nouveau générateur.

Rendez-vous à la section générateurs pour en savoir plus.

Contextualiseurs

Selon la logique RCC, les générateurs de documents articulés autour de Peritext utilisent des modules *contextualiseurs* pour prendre en charge des ressources courantes dans la publication académique (référence bibliographique, image, élément de glossaire) mais aussi une variété d'autres éléments interactifs (vidéo, production dicto, production quinoa, visualisation vega, ...).

Les modules contextualiseur décrivent comment afficher une ressource dans un contexte particulier et selon les paramètres spécifiques fournis par un object contextualiseur dans un récit peritext. Ce sont en somme des sortes de "mini-programmes" qui *en entrée* lisent une ressource et une série de paramètres et *en sortie* produisent une mise en forme spécifique.

Il est important de comprendre que les contextualiseurs ne sont pas forcément attachés à un type de ressource, puisque c'est là l'une des spécificités principales de Peritext. Certains contextualiseurs sont associés à un seul type de ressource (exemple: dicto), d'autres peuvent être appliqués à plusieurs types de ressources différents (exemple: référence bibliographique).

Techniquement, ces bibliothèques exposent un ou plusieurs composants react ainsi que des feuilles de style.

Les sections suivantes décrivent par le menu les divers modules contextualiseurs de Peritext déjà mis en place et leurs options.

Liste d'idées pour des contributions portant de nouveaux contextualiseurs

- unity : un contextualiseur jouant un code source unity
- three.js : un contextualiseur jouant une géométrie 3D au moyen de la technologie WebGL
- pdf : un contextualiseur affichant un pdf à partir d'une url au moyen d'un module de lecture personnalisé
- facebook: rendre un post facebook selon une mise en forme de qualité ou personnalisée
- twitter : rendre un tweet ou une liste de tweets selon des paramètres personnalisés
- portofolio: un contextualiseur permettant de représenter des séquences de documents & esquisses commentées

Contextualiseur bib

Le contextualiseur "bib" (pour bibliographique) affiche une ressource sous la forme d'une référence bibliographique.

Celle-ci peut être une citation courte, c'est-à-dire dans la forme prise dans le corps des textes, comme à la fin de ce paragraphe [\(16\)](#).

Elle peut aussi être longue, c'est-à-dire dans la forme prise normalement dans les bibliographies, comme suit :

¹ RICCI, Donato. *An Account of Digital Humanities from the AIME Project* [en ligne]. 2016. Revue échappées. Disponible à l'adresse : <http://echappees.esapyrrenees.fr/content/2-numeros/3-numero3/5-an-account-of-digital-humanities-from-the-aime-project/echappees-an-account-of-digital-humanities-from-the-aime-project.pdf>

An Account of Digital Humanities from the AIME Project

Le formatage exact de la citation dépend du style et de la locale de citation définie pour le récit Peritext en entier (par exemple : ISO-61, APA, Chicago, ...). Le contextualiseur se met à jour en fonction des données de formatage qui lui sont fournies en amont, ce qui fait que toutes les citations d'un document sont toujours harmonisées et à jour.

Paramétrisation du contextualiseur bib

Il est possible de paramétriser le contextualiseur bib en indiquant pour une citation particulière un préfixe, un suffixe et/ou un localisateur comme un numéro de page. Voici un exemple [\(16, p. 2 version papier\)](#).

Extensibilité du contextualiseur bib

Outre la contextualisation de relevés bibliographiques, le contextualiseur bib peut être utilisé sur n'importe quel type de ressource grâce à un module qui convertit les métadonnées des ressources dans un format standard lisible par celui-ci¹³.

Voici par exemple la contextualisation bib d'une ressource de type vidéo [\(17\)](#) ou une autre d'une ressource de type site web [\(2\)](#).

13. Il s'agit du format établi par la spécification csl-json [\(21\)](#).

Contextualiseur glossary

Le contextualiseur glossary permet de signaler des entrées de glossaire dans un récit peritext.

Voici un exemple de mention de Michel-Ange.

Il est possible d'utiliser des alias par exemple en citant la même ressource que dans le paragraphe précédent mais en l'appelant Michel-Angelo.

Ce contextualiseur est appelé à évoluer pour s'appliquer de manière plus riche en fonction du type d'entités décrites (personnes, notions, lieux, évènements, ...).

Rendez-vous dans la section "glossaire" de ce document pour voir une liste des mentions de ressources interactives.

Contextualiseur webpage

Le contextualiseur webpage affiche la page web associée à une ressource.

Il peut s'agir d'un lien hypertexte, d'une vignette du site ou encore d'un iframe (site imbriqué) du site.

Page embarquée



Site web hypotheses

Extensibilité du contextualiseur webpage

Outre les ressources webpage, le contextualiseur webpage peut être utilisé sur toutes les ressources qui indiquent une url dans leur métadonnées, ou les ressources bibliographiques qui indiquent une url.

Exemple d'une ressource "dicto" affichée avec le contextualiseur webpage :



Revue vectors

Tour d'horizon des formats utilisés

Printemps 2016.

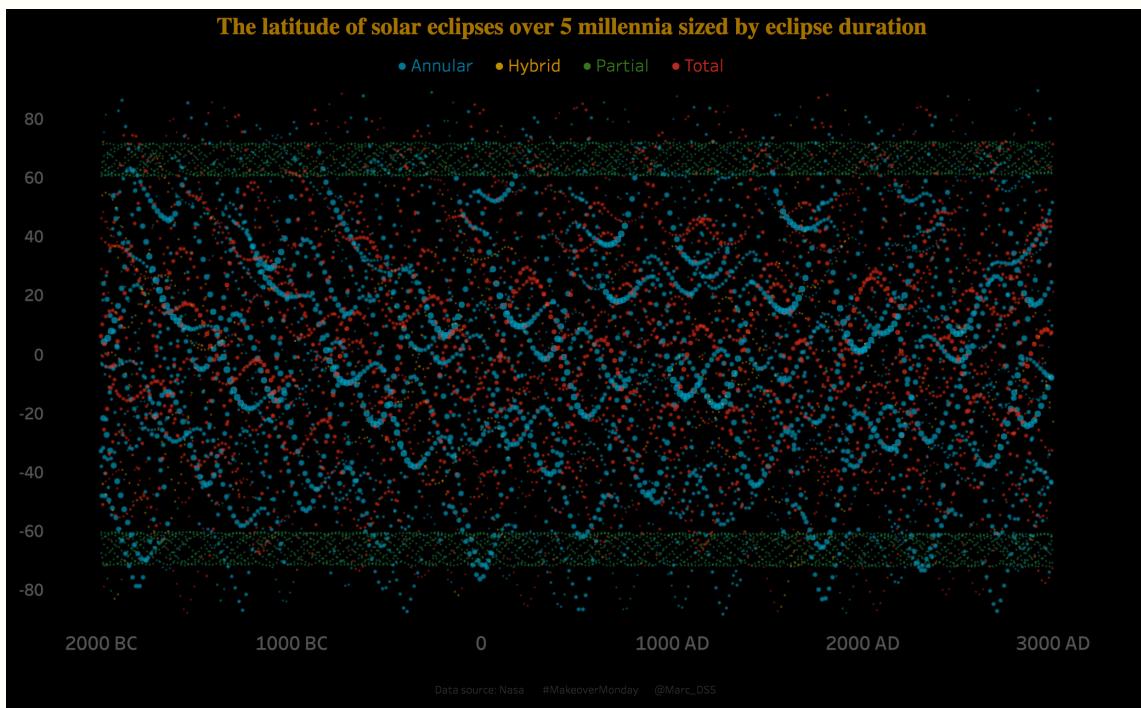
Vectors - une analyse

Analyse par étiquetage des différents articles du journal Vectors

Exemple d'une ressource référence bibliographique affichée avec le contextualiseur webpage (le lien pointe vers fichier pdf, qui sera donc affiché avec les moyens du navigateur - et sa présentation est désactivée pour les supports de type codex) :

An Account of Digital Humanities from the AIME Project

Exemple d'une ressource pointant vers une visualisation tableau :



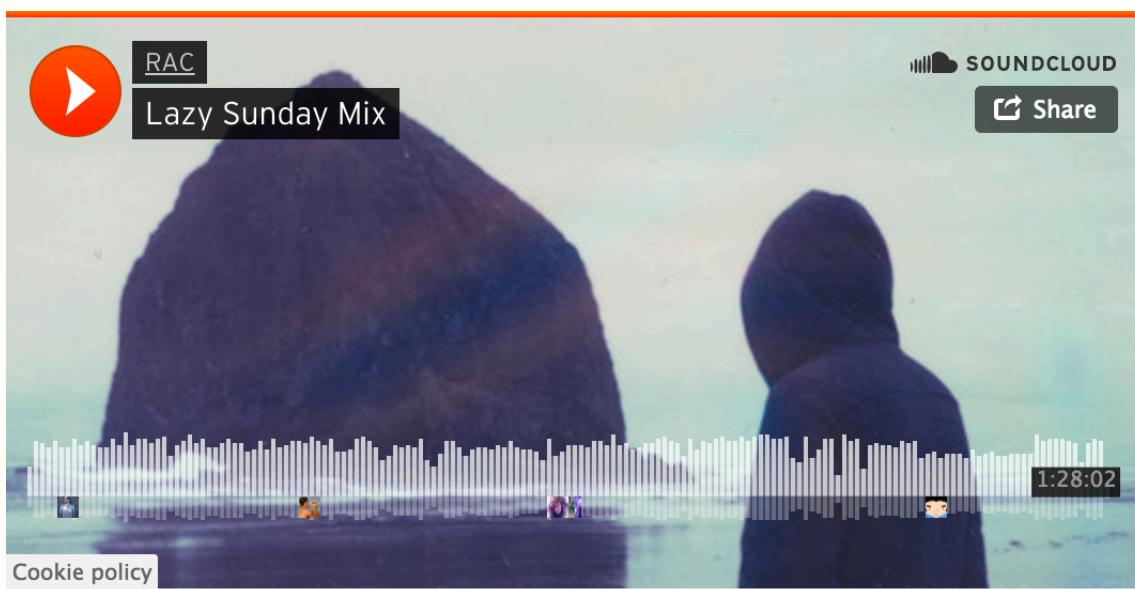
Une visualisation hébergée sur tableau

Accessible à https://public.tableau.com/views/MM34-5MillenniaofSolarEclipses/SolarEclipse?:embed=y&:toolbar=no&:loadOrderID=0&:display_count=yes

Contextualiseur embed

Le contextualiseur "embed" prend en charge les ressources de type "embed", qui ne sont rien d'autre que des portions de code html directement intégrées dans les contenus¹⁴.

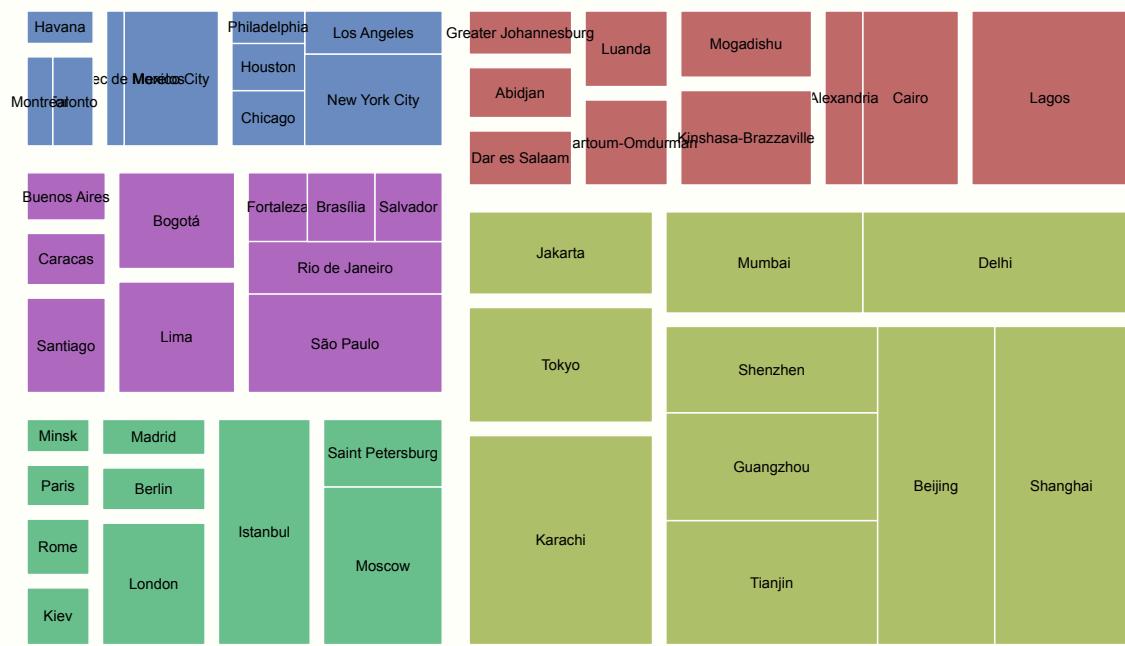
Exemple d'un embed soundcloud :



Un exemple d'embed soundcloud

Un exemple d'embed de code svg, construit avec l'outil raw graphs :

¹⁴. Il est à noter que le code javascript contenues dans d'éventuelles balises script injectées selon cette méthode ne sont pas prises en compte.



Exemple d'embed de code svg tiré de raw graphs



manovich

@manovich

Abonné

Stanford professor of Computer Science, Marc Levoy, offers a free course on digital photography. bit.ly/2vDL9mb

À l'origine en anglais



15:35 - 30 août 2017

15 Retweets 25 J'aime



Exemple d'embed basique d'un tweet

Exemple d'embed d'un post facebook :



Centre Pompidou

24 août, 10:53 ·

There's only way to celebrate [#DavidHockney](#)'s birthday: come and see his retrospective at the [Centre Pompidou Paris](#) !

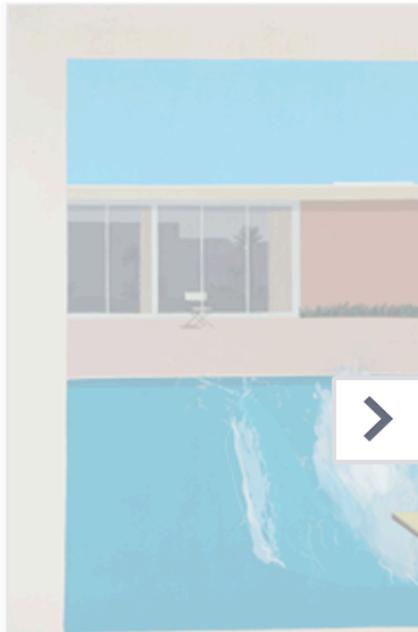
*Buy Your ticket online and enjoy all exhibitions, the Museum, the Kids Gallery and the View of Paris.



Portrait of an Artist

by David Hockney

Réserver



A Bigger Splash

by David Hockney

Exemple d'embed d'un post facebook

Contextualiseur vidéo

Le contextualiseur vidéo joue une ressource sous la forme d'un player vidéo.

Les sources prises en charge pour l'instant sont youtube, vimeo, et une url pointant vers une vidéo html5.

En mode codex, le contextualiseur affiche une vignette pointant vers l'url de la vidéo.

En voici un exemple :



Objectified

Source: Neojaponismo (youtube: <https://www.youtube.com/watch?v=TyofGn8fiUU>)

Contextualiseur dicto

À propos de Dicto

Dicto est un format de données, un éditeur et un composant de player développés par Robin de Mourat & Donato Ricci dans le cadre d'analyses de transcriptions¹⁵.

Dicto permet d'associer des sous-titres (éventuellement taggés) à une vidéo et de produire à partir de ces derniers une application interactive permettant de naviguer dans la vidéo via son texte (et vice versa).

Utiliser le contextualiseur dicto



Ressource dicto représentée avec le contextualiseur dicto

Analyse par étiquettage des différents articles du journal Vectors

Non-exclusivité du contextualiseur dicto

Une ressource de type dicto est naturellement prise en charge par le contextualiseur dicto. Cela dit, elle peut par ailleurs aussi être prise en charge par le contextualiseur "video" :

15. Voir une démonstration d'un format de sortie possible pour dicto ([i](#)) .



Revue vectors

Tour d'horizon des formats utilisés

Printemps 2016.

Ressource dicto représentée en vidéo simple

Analyse par étiquetage des différents articles du journal Vectors

Contextualiseur codefiles

Les ressources "fichiers de code" (codefiles) représentent du code source entré optionnellement divisé en une série de fichiers.

Le contextualiseur fichiers de code affiche une ressource de type "fichiers de code" (codefiles) sous la forme d'une interface interactive ou statique.

Voici un exemple de contextualisation codefiles :

second.css

```
.ovide-InlineCitation
{
    margin: 0;
    padding: $gutter-small 0 $gutter-small 0;

    cursor: pointer;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}
```

my_file.js

```
import React from 'react';

export default ({
    resource: {
        data: {
            code={}
        }
    },
    // contextualizer,
    // contextualization
}) => {
    return (
        <div className="peritext-contextualizer-static peritext-contextualizer-codefiles">
        {
            Object.keys(code)
            .map(fileName => {
                return <div key={fileName} className="file-display">
                    {Object.keys(code).length > 1 && <h2>{fileName}</h2>}
                    <pre>
                        <code>
                            {code[fileName]}
                        </code>
                    </pre>
                </div>
            })
        }
    </div>
)
}
```

Exemple de ressource codefiles

Contextualiseur table

Le Contextualiseur de type table affiche une ressource de type table (exemple: tirée d'un fichier csv) sous la forme d'un tableau traditionnel.

En version codex, il produit un tableau simple.

En version web, il produit un tableau interactif (pagination, recherche, ...)

En voici un exemple :

a	b
A	2
B	4
C	6
D	8
E	2
F	3
G	5

Dataset très simple

Contextualiseur vegalite

vega (et son petit frère allégé vega lite) est une grammaire de description de visualisations de données interactives et statiques. Il est développé par l'équipe Interactive Data Lab au sein de l'Université de Washington ([18](#)) .

Le contextualiseur vegalite permet de contextualiser une ressource de type table sous la forme d'une visualisation de données paramétrée selon la grammaire vegalite.

Utiliser le contextualiseur vega

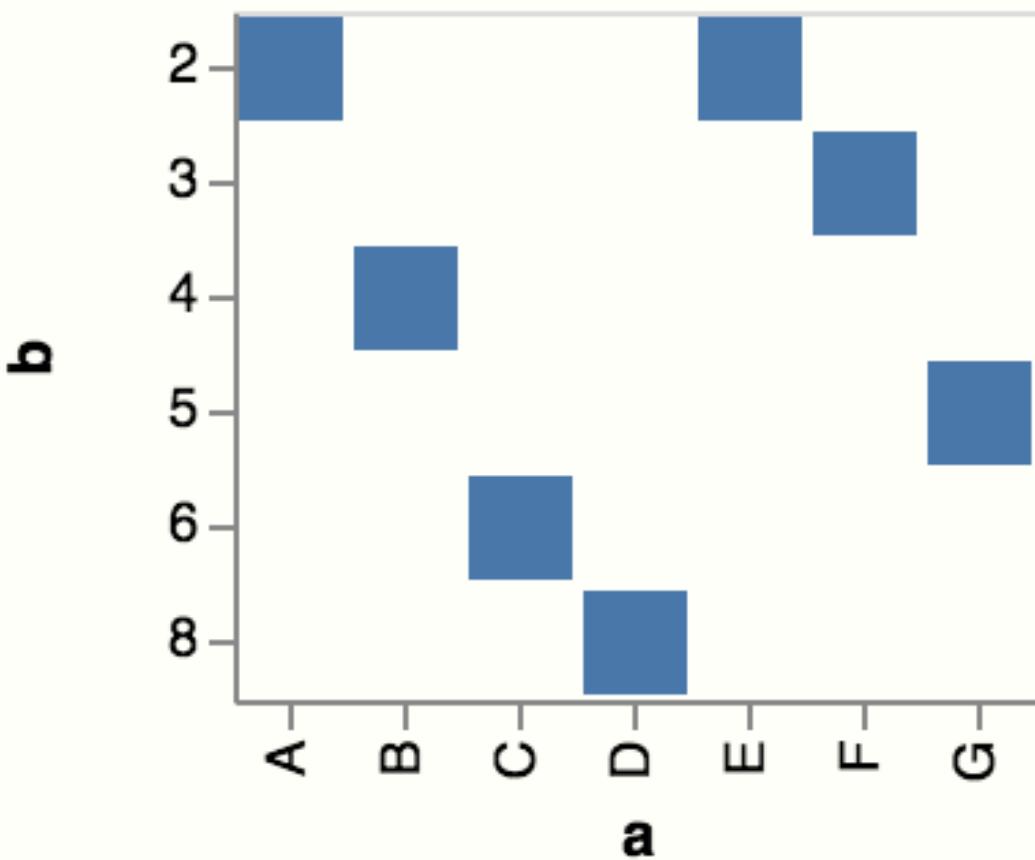
Soit la ressource suivante :

a	b
A	2
B	4
C	6
D	8
E	2
F	3
G	5

Dataset très simple

Elle est par défaut affichée via le contextualiseur table. La voici désormais affichée avec le contextualiseur vega¹⁶ :

¹⁶. Il est à noter que le module vega rend du svg. Pour une raison inconnue à ce jour cela dit, le module ne fonctionne pas en environnements serveurs ; pour palier à ce problème technique il est pour l'instant possible d'associer une vignette à chaque visualisation afin qu'elle soit utilisée comme image à la place du svg qui devrait normalement s'afficher.



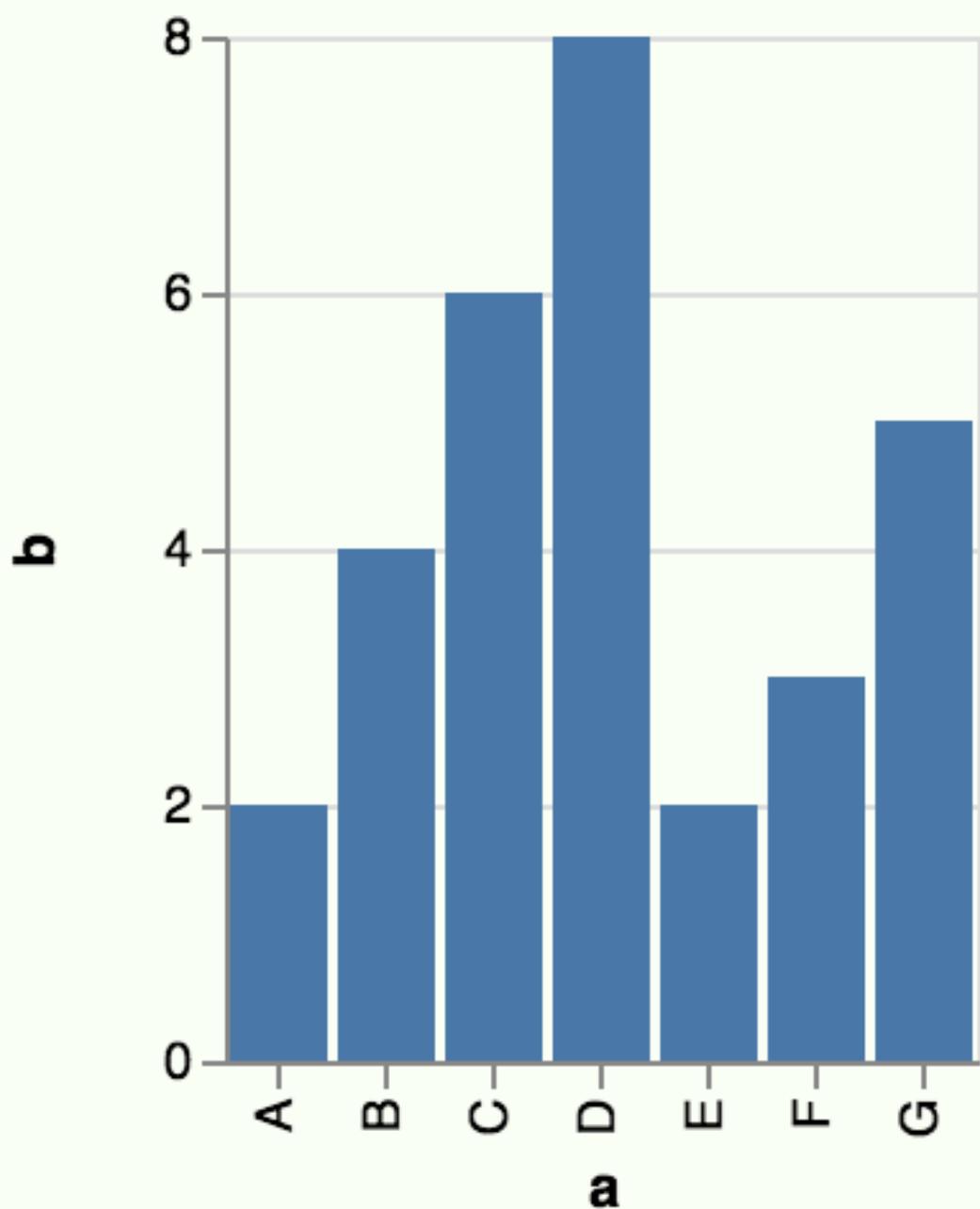
Visualisation sans spécification

Par défaut, le contextualiseur cherche un état minimal pour afficher la table sous la forme d'une visualisation, pas très intéressant donc.

Ajoutons-lui maintenant le paramètre de spécification suivant :

```
{
  "mark": "bar",
  "encoding": {
    "x": {"field": "a", "type": "ordinal"},
    "y": {"field": "b", "type": "quantitative"}
  }
}
```

On obtient :



Visualisation avec une spécification basique

Apprendre à utiliser la grammaire vegalite

Les possibilités offertes par vega sont très importantes et couvrent une grande partie des besoins courants en termes de visualisation de données. La documentation proposée sur le site de vega permet de s'en faire une première idée.

Pour prévisualiser et tester une spécification vegalite avant de l'intégrer dans un contextualiseur, rendez-vous sur l'outil de développement proposé par ses créateurs [\(36\)](#).

Contextualiseur p5 (processing)

Le projet p5.js

Processing est un langage de programmation, une bibliothèque Java, un éditeur et une communauté dédiées au développement d'outil de programmation pour les artistes visuels.

C'est aujourd'hui un langage utilisé et enseigné dans nombre de contextes artistiques et éducatifs.

p5.js est un portage de la syntaxe mise en place par Processing pour le langage javascript.

L'outil permet de créer diverses formes d'images mouvantes et interactives 2D et 3D, et optionnellement d'utiliser les options du navigateur (micro, caméra, ...) pour créer des expériences numériques merveilleuses.

Utiliser le contextualiseur p5

Le contextualiseur p5.js interprète une ressource de type "fichiers de code" contenant un *sketch* p5 pour en faire une application interactive, stockée dans une balise canvas.

Prenons la ressource de type "fichiers de code" suivante :

```
function sketch (p) {
  var rotation = 0;

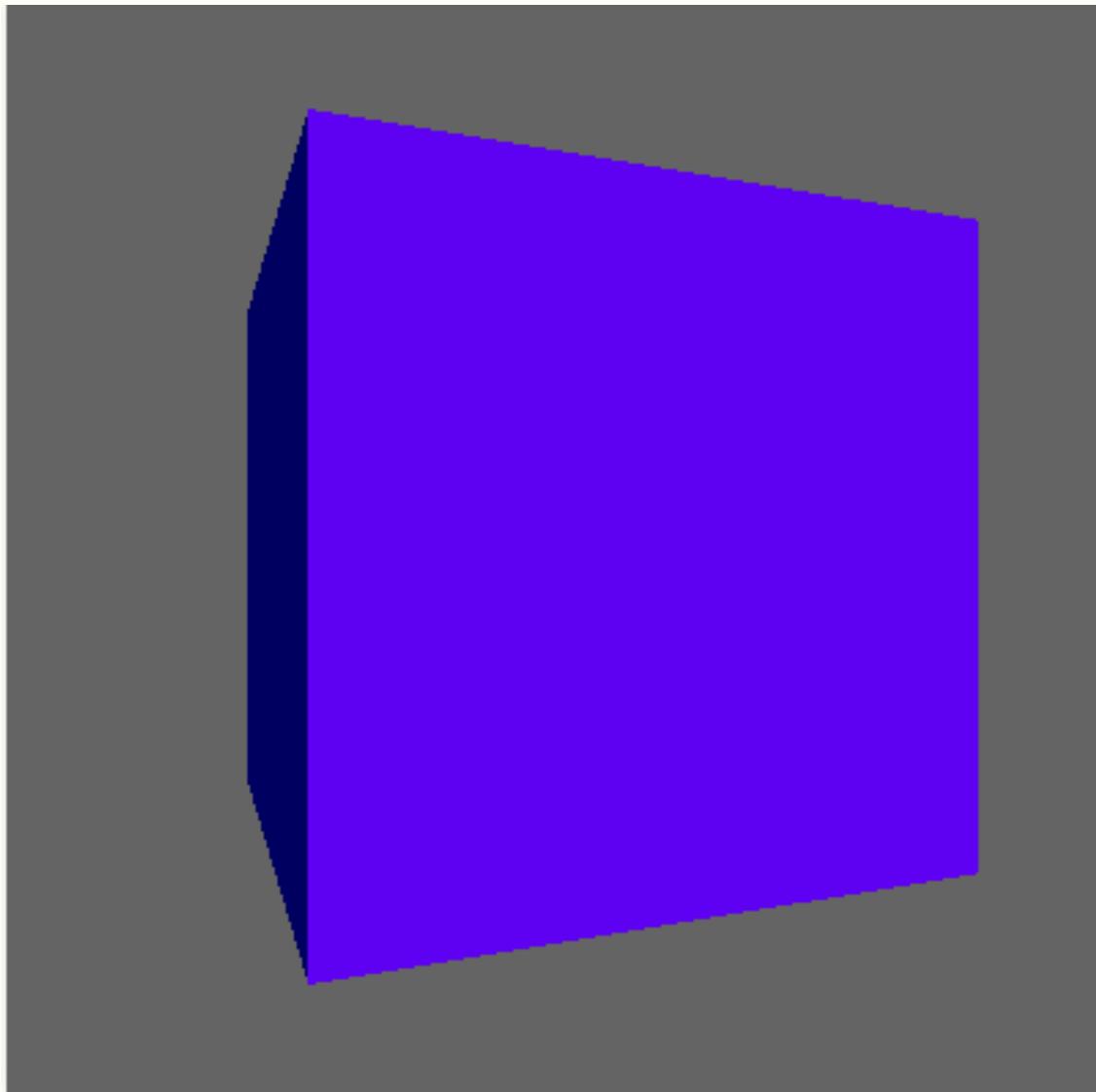
  p.setup = function () {
    p.createCanvas(200, 200, p.WEBGL);
  };

  p.draw = function () {
    rotation += 0.01;
    p.background(100);
    p.noStroke();
    p.push();

    p.rotateY(rotation);
    p.box(100);
    p.pop();
  };
}
```

Ressource processing js

Choisissons maintenant le contextualiseur p5 pour l'afficher. On obtient l'image suivante :



Ressource processing js

On obtient un "canvas" dans lequel est représenté un programme processing interprété à partir du code de la ressource¹⁷.

Les possibilités offertes par ce contextualiseur sont nombreuses.

Pour en savoir plus sur les capacités de p5.js, rendez-vous sur le site officiel du projet [\(27\)](#).

¹⁷. Pour l'instant, les contextualisations de ce type ne sont pas visibles sur les documents de type codex directement. Il est donc possible de spécifier une image de vignette au niveau du contextualiseur qui sera utilisée en remplacement.

Contextualiseur data-presentation

Les présentations de données

Le concept de présentation de données désigne des documents qui consistent à proposer aux lecteurs une visite guidée dans une visualisation. Le lecteur parcourt des slides, qui sont associées à une vue particulière dans la visualisation associée.

Le format des présentations de données est actuellement développé par le médialab de sciences po [\(6\)](#) dans le cadre du programme FORCCAST [\(8\)](#).

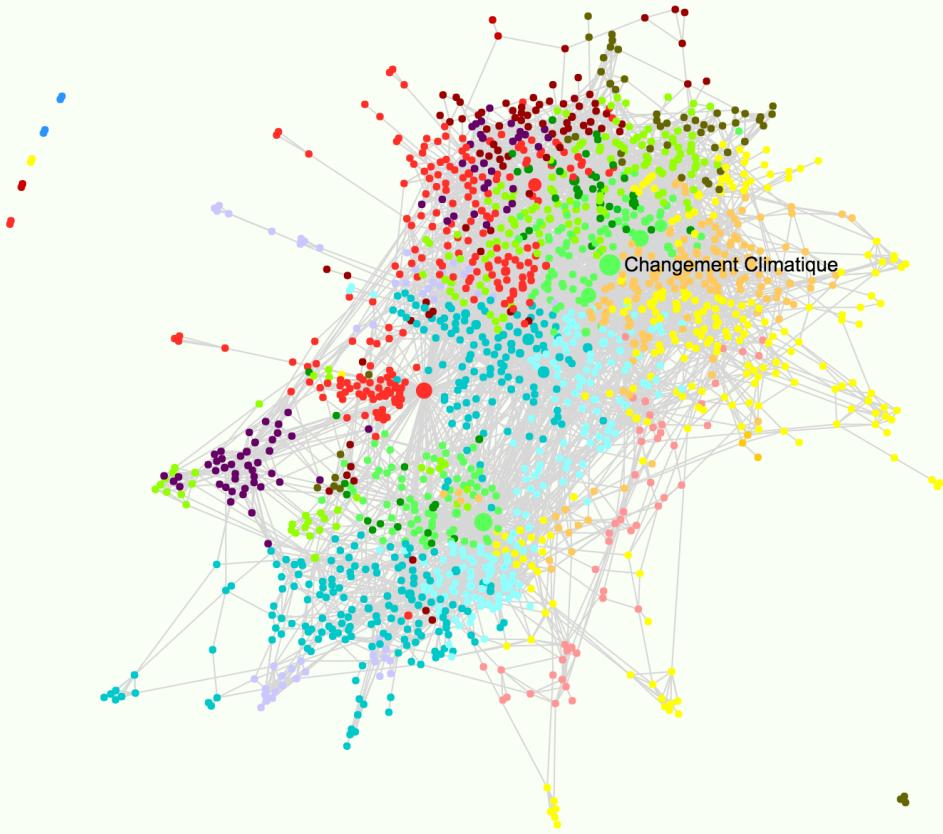
Un premier exemple de présentation de données fait pour les visualisations de réseaux peut être expérimenté via l'outil ManyLines [\(22\)](#).

Un outil plus récent permettant de construire des présentations pour des frises chronologiques, cartes, graphiques, ... est actuellement en développement.

L'outil permettant de construire les présentations qui suivent est appelé à être rendu public courant 2018.

Utiliser le contextualiseur data-presentation

Voici un exemple de présentation de réseau :

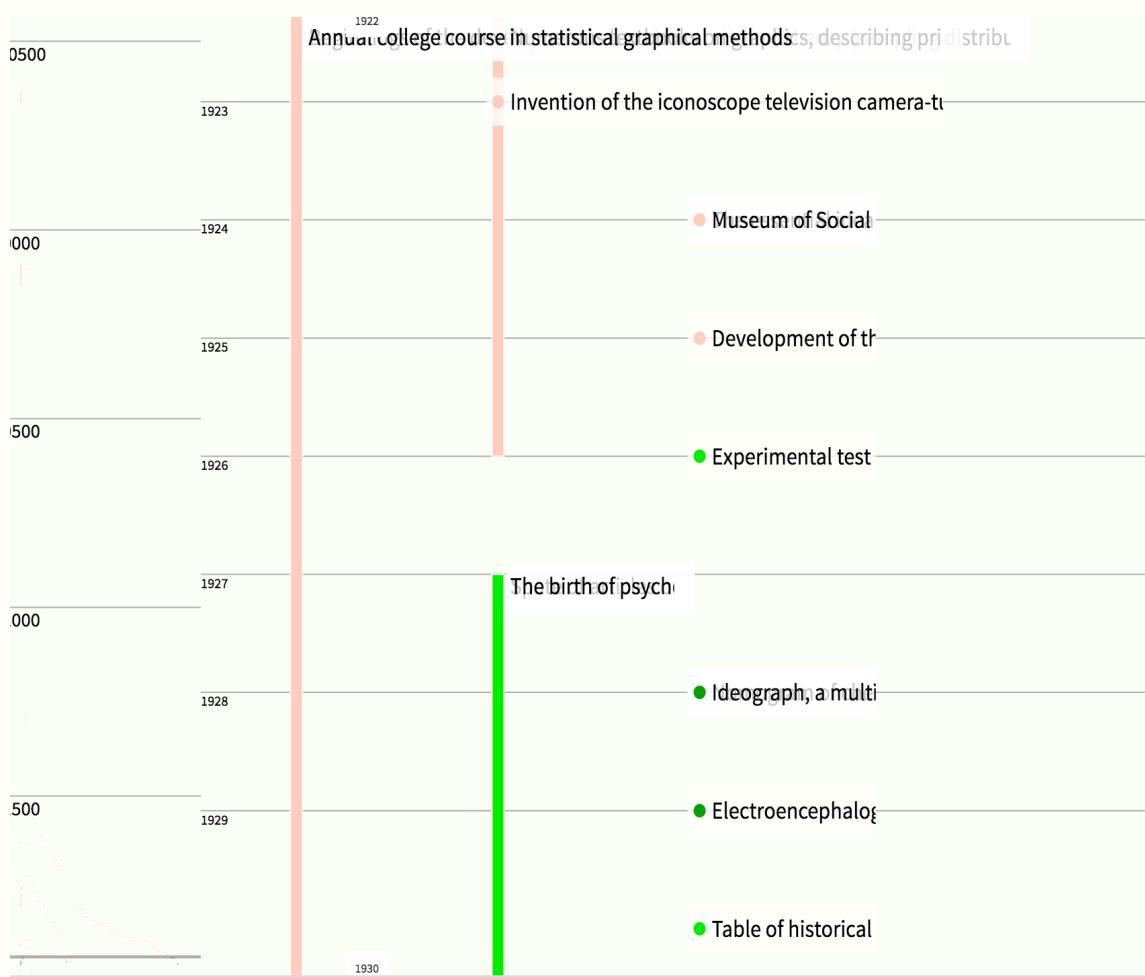


Mon beau réseau roi des forêts

Mon beau réseau roi des forêts

Les présentations de données fonctionnent, comme peritext, sur un modèle de gabarit. Pour l'instant deux gabarits sont possibles ("déroulement", comme dans la contextualisation précédente, et "étapes")

Voici exemple de présentation de frise chronologique :



Histoire de la visualisation

Robin.

Histoire de la visualisation

Cette présentation traite de l'histoire de la visualisation

Pour le mode codex, il est également possible de choisir si l'on veut afficher ou non les commentaires attachés à chacune des vues de la présentation.

Un exemple de présentation de carte avec les commentaires affichés :

Histoire du Fort et des combats de 1870-1871

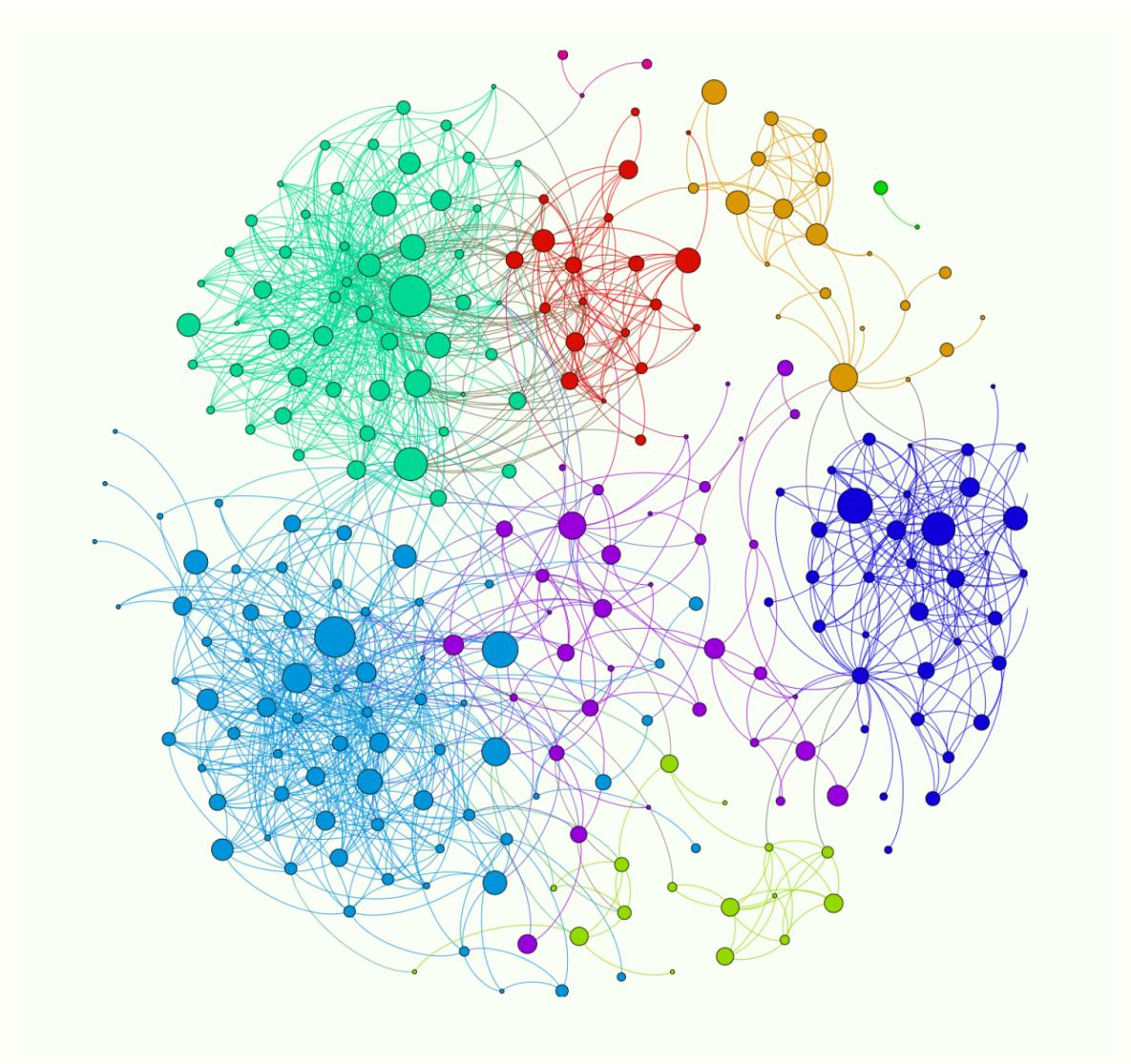
Robin.

Histoire du Fort et des combats de 1870-1871

Une histoire du fort d'Issy

Il est à noter que les types de visualisations pris en charge par les présentations de données incluent les fichiers svg, ce qui permet de proposer des visites guidées sur de simples images vectorielles.

Une exemple de présentation fondée sur un fichier svg :



Exemple de présentation SVG

Robin.

Exemple de présentation SVG

D'autres gabarits, types de présentation de données et options disponibles sont appelés à apparaître au fil du développement de ce projet voisin ...

Gabarits

Les **gabarits** définissent la manière selon laquelle doivent être affichés les contenus d'un récit. Ils concernent à la fois le code html/css de chaque portion de contenu, et l'organisation même des pages/sections pour les documents codex ou des routes pour les documents web.

Par exemple, le présent document est mis en forme au moyen des gabarits *peritext-template-web-garlic* et *peritext-template-codex-garlic*.

Les gabarits sont attachés à un type de générateurs particulier avec lequel ils sont compatibles (pour l'instant les deux types disponibles sont codex et web - voir la section [générateur](#)).

Gabarits existants

Gabarit Web Garlic

Il s'agit d'un **gabarit** pour générateurs de type web très simple, fortement calqué sur la structure rhétorique d'un livre académique traditionnel ou d'une thèse.

En ce qui concerne l'interface des contenus, le gabarit présente une colonne principale déroulant les contenus de manière linéaire, et une colonne optionnelle permettant de naviguer dans les différentes contextualisations d'une même ressource.

Il présente également un ensemble de "vues de coupe" dans le document calquées sur les formats imprimés traditionnels, comme une bibliographie/liste de ressources et un glossaire interactifs permettant de prévisualiser et de naviguer dans les contextualisations d'une même ressource.

Ce gabarit est pour l'instant testé avec le générateur suivant : *peritext-generator-next*.

Gabarit Codex Garlic

Il s'agit d'un **gabarit** pour générateurs de type codex très simple plutôt pensé pour les pdf interactifs et les epub, utilisant les polices Roboto et Merryweather et présentant une table des matières interactives, un glossaire, un index des auteurs, et des références interactives.

Ce gabarit est pour l'instant testé avec les générateurs suivants : *peritext-generator-pdf*, *peritext-generator-epub*.

Générateurs

Les générateurs produisent un fichier pour un support donné à partir d'un récit peritext et d'un ensemble d'options¹⁸.

Ils se divisent en deux grandes familles :

- les générateurs de type codex sont dédiés à tous les formats de sortie structurés autour de pages à tourner. Techniquement, ces formats de sortie n'intègrent pas de code javascript
- les générateurs de type web qui produisent d'une manière ou d'une autre un site web ou application assimilée (application mobile, application bureau, ...)

Générateur pdf (codex)

Ce générateur produit un fichier pdf interactif et imprimable, avec le moteur PrinceXML([2](#)) qui permet de gérer toutes les règles CSS print¹⁹.

Générateur epub (codex)

Ce générateur produit un fichier epub pour liseuses, prenant en charge les hyperliens internes et acceptant du code css personnalisé. Ce générateur est appelé à être grandement amélioré à l'avenir.

Générateur next (web)

Ce générateur produit un site web statique (série de fichiers html) au moyen du framework next.js ([4](#)). Le framework next.js a la particularité de générer des applications *isomorphiques* : l'intégralité du contenu de chaque page est écrit dans des fichiers html (pour l'usage des machines d'indexation, et des navigateurs interdisant javascript), mais le comportement et l'affichage est aussi pris à charge via une application javascript pour les clients qui le peuvent afin de profiter d'un maximum d'interactivité et de subtilité dans la présentation. Cette technologie permet donc d'allier indexabilité des contenus et interactivité des interfaces.

Ce générateur permet également d'utiliser le site statique qu'il produit comme API, puisqu'il expose les différentes parties du récit qu'il génère sous la forme de fichiers json séparés pouvant être requêtés par une application tierce (exemple d'une vue sur les ressources du présent document : ([35](#))).

Liste d'idées pour des contributions au niveau des générateurs

- Générateur de fichiers markdown (devra d'une manière ou d'une autre représenter les contextualisations)
- Générateur de fichiers LaTeX
- Générateur d'un fichier .doc
- Générateur d'un fichier xml/idml (compatible avec InDesign)
- Générateur (côté client) de code pouvant être embarqué dans un blog wordpress

18. Techniquement, ces bibliothèques exposent une fonction qui consomme un récit au format json et des modules de contextualisation et de gabarit, pour produire en sortie un fichier utilisable (e.g. ".pdf").

19. Note: prince xml est la seule technologie non libre utilisée dans un module peritext. Le choix de Prince XML est motivé par sa prise en charge hors pairs de la spécification w3c pour le css print([3](#)). Les autres technologies de conversion pdf en javascript, basées sur webkit ou phantom, ne prennent pas (ou très mal en charge) ces spécifications.

- Générateur d'extraits sous des formats consommables par les APIs de twitter, facebook, instagram, ... afin d'être implémentés dans des bots diffusant progressivement le contenu d'un récit dans ces formats par exemple
- Générateur de site *monopage* adapté aux articles scientifiques et devoirs d'étudiants
- Générateur de livre audio via une API de text-to-speech
- Générateur d'application de "spatialisation" de récits avec Unity ou mobilizing.js
- ...

Éditeur Ovide

Ovide est le premier éditeur développé pour l'écosystème Peritext. Il est dérivé de manière importante de l'éditeur Quinoa, en développement au sein du médialab sciences po⁽⁶⁾ et lui aussi développé en partie autour du format peritext.

Il a vocation à faire office de "preuve de concept" plutôt que d'application en soi, étant donné que sa configuration technique est davantage tournée vers l'expérimentation que vers une utilisation à grande échelle comme un outil "sur étagère"⁽²⁰⁾.

Les données de l'application sont stockées sur le navigateur de l'utilisateur, et il peut les télécharger à tout moment.

L'éditeur propose une interface d'édition graphique, mais prend aussi en charge la syntaxe markdown⁽⁵⁾ comme raccourci.

L'éditeur permet d'écrire un récit peritext et de l'exporter aux formats epub, pdf, et html, via des générateurs installés sur un serveur avec lequel il doit être connecté.

Rendez-vous sur la version alpha de l'application [\(7\)](#) pour la découvrir :

The screenshot shows the Ovide application interface. At the top left is the Ovide logo (a stylized butterfly icon). Next to it is the title "Ovide" and the subtitle "l'éditeur de documents multimodaux académiques". Below this is a brief description: "Ovide est un outil qui sert à éditer des documents académiques riches, faits de visualisations, de vidéos et autres éléments interactifs, et les publier à la fois aux formats pdf, epub et web." A language switch "fr/en" is shown. A large blue button in the center says "commencer un nouveau récit". Below the button is a link "... ou continuer un récit stocké dans le navigateur". The bottom section shows a dark green sidebar with the Peritext logo and text: "PERITEXT", "Publication Multimodale Orientée", and "Une introduction". To the right of the sidebar is a preview of a document titled "Peritext - publication multimodale orientée contexte". The preview text discusses the need to connect academic document production modes with web contexts. On the far right of the preview are three buttons: "éditer", "paramètres", and "supprimer".

Éditeur Ovide

20. Ainsi, Ovide utilise par exemple l'intégralité des contextualiseurs à disposition dans l'écosystème, ce qui ne devrait pas être le cas pour des applications plus spatialisées.

Contribuer à peritext

L'intégralité du code de peritext est placée sous licences libres LGPL-3 & CECCIL-B. Il peut donc être modifié et utilisé hors exploitations commerciales. L'ensemble du projet est déposé sur github, au sein de l'organisation peritext ([33](#)).

Pour l'instant, les modèles de données associés aux ressources, contextualiseurs, gabarits de peritext ne sont pas stabilisés, car une utilisation prolongée est nécessaire pour les gérer et les fixer de manière intelligente²¹. Il en est de même pour l'API des modules, qui évolue en fonction de l'expérience tirée des premières implémentations du projet.

Ainsi, les possibilités de contribution à peritext sont amenées à s'élargir petit à petit, une fois le projet stabilisé et apte à fonctionner comme une plateforme pour de nombreuses initiatives.

Créer/hacker un contextualiseur

Un contextualiseur doit exposer l'ensemble des éléments suivants :

- une série de composants exposés au propriétés BlockStatic, BlockDynamic, InlineStatic ou InlineDynamic pour chacun des cas de figure pris en charge par le contextualiseur
- une propriété metadata présentant les propriétés du contextualiseur, les modules disponibles, et les options qu'il propose à l'utilisateur
- une propriété defaultCss qui expose le code css par défaut à injecter quand le contextualiseur est implémenter

Exemple de l'API du contextualiseur vegalite :

```
{  
  BlockDynamic: Component1,  
  BlockStatic: Component2,  
  metadata: {  
    type: 'peritext-contextualizer',  
    id: 'vegalite',  
    name: 'Vega lite',  
    coverage: {  
      inlineStatic: false,  
      blockStatic: true,  
      inlineDynamic: false,  
      blockDynamic: true,  
    },  
    model: {  
      acceptedResourceTypes: [  
        {type: 'table'}  
      ],  
      block: {  
        expandable: false,  
        options: [  
          {  
            type: 'jsonInput',  
          }  
        ]  
      }  
    }  
  }  
}
```

21. Avant de stabiliser le schéma de données de peritext, il faudra notamment :
intégrer dans le modèle des données la possibilité d'écrire des liens internes au récit (vers une section, version une contextualisation)
écrire un schéma de description des métadonnées des contextualiseurs
décider où stocker les schémas des types de ressources et leur modèle (dans les contextualiseurs par défaut ? dans des modules séparés ?)
faire évoluer le modèle des contextualisations (et contextualiseurs) vers un modèle multi-ressources
trouver la meilleure manière de gérer la définition des "plans de sites" et de la structure des récits en ce qui concerne les couples gabarit web / générateur de sites web

```

        id: 'spec',
        title: {
          fr: 'Spécification vega lite',
          en: 'Vega lite specification'
        },
        placeholder: {
          fr: 'Écrire ou coller une spécification vega-lite (tester ici : https://vega.github.io/editor/#/custom/ve',
          en: 'Write or paste a vega specification (test here : https://vega.github.io/editor/#/custom/ve'
        }
      }
    ]
  }
}
}
}

```

API du contextualiseur vegalite

... plus d'indications bientôt.

Créer/hacker un gabarit/template

L'API des gabarits varie selon qu'ils se destinent à des générateurs de type "codex" ou "web".

Pour les deux, un gabarit doit exposer :

- une propriété metadata
- un ensemble de composants correspondant aux différentes vues/sections d'un récit (à normaliser)
- un composant centralisateur permettant d'afficher tout le récit (par exemple pour une utilisation du module comme aperçu)

Exemple de l'API du contextualiseur web-garlic :

```
{
  metadata: {
    name: 'web garlic',
    type: 'peritext-web-template',
    id: 'web-garlic',
    options: [
      {
        type: 'boolean',
        id: 'disqus',
        title: {
          fr: 'Intégrer le service de discussion disqus sur les page',
          en: 'Integrate discussion service disqus on pages'
        },
      },
      {
        type: 'select',
        id: 'dynamicNotesPosition',
        title: {
          fr: 'Position des notes',
          en: 'Notes position'
        },
      },
    ],
  }
}
```

```

options: [
  {
    id: 'foot',
    labels: {
      fr: 'Fin de page',
      en: 'Page end'
    }
  },
  {
    id: 'aside',
    labels: {
      fr: 'Marge de page',
      en: 'Page margin'
    }
  }
]
},
PreviewContainer: Component,
Bibliotheca: Component,
Dimensio: Component,
Home: Component,
Lexicon: Component,
Section: Component,
typefaceNames: Array<String> // liste des noms de fontes à intégrer
}
}

```

API du gabarit "web-garlic"

... indications plus précises bientôt

Créer/hacker un générateur

Un générateur est un module qui prend en argument un récit et des modules à consommer (un modulegabarit, des modules contextualiseurs, une liste de traductions de termes) et produit un fichier.

Voici par exemple l'API du générateur epub :

```

function generateEpub{
  story : Object, // le récit à consommer
  template : Object, // le gabarit à utiliser
  contextualizers : Object, // hashmap de modules contextualiseurs
  locale : Object, // hashmap de traduction de termes (dépend du gabarit)
  outputDirPath = './output' : String, // chemin de sortie
  tempDirPath = './temp' : String, // chemin de stockage des fichiers temporaires
  additionalStylesheets = [] : Array<String>, // feuilles de style
}

```

API du générateur epub

... indications plus précises bientôt

À propos

Peritext est un projet libre développé dans un contexte de recherche en design et esthétique. Il a également beaucoup à voir avec les sciences de l'information et de la communication, les sciences de la documentation et l'informatique.

Le projet est né dans le contexte de la construction des humanités numériques françaises d'une part, et d'un intérêt croissant des institutions et des étudiants de design français vis-à-vis de la recherche académique d'autre part. Ces deux milieux (humanités numériques et recherche en design) partagent la nécessité de mettre en scène et discuter de manière élaborée une diversité d'éléments hétéroclites impliqués dans les recherches concernées.

Il s'agit également du volant pratique d'une recherche doctorale menée par Robin de Mourat sur les formats de publication académique²²

Histoire du projet / remerciements

- 2013-2016 : Robin de Mourat bénéficie d'un contrat doctoral de 3 ans de la part du Ministère de l'Enseignement Supérieur et de la Recherche via l'ENS Cachan pour conduire des activités de recherche et d'enseignement auprès de Nicolas Thély à l'université Rennes 2, dans le cadre du groupe de recherche MONADE (23) . La naissance de Peritext doit beaucoup à ce contrat et à cet environnement.
- Janvier 2014 - Mars 2015: la recherche doctorale conduit à une phase d'observation participante sur le projet Enquête sur les Modes d'Existence (9) , projet philosophique soutenu par un écosystème d'instances imprimées et de sites web - qui sera analysé via la conduite d'entretiens commentés et d'une enquête numérique(10). Une partie importante des principes de conception de Peritext découlent de cette rencontre et des nombreux échanges qu'elle a occasionné.
- 2015 : rédaction par Donato Ricci & Robin de Mourat d'un texte de recherche en design portant sur le projet EME, dont émerge la centralité du concept de *contexte* dans les enjeux actuels de la publication académique (15)
- Juin 2015 : workshop "Open AIME", dédié à la confrontation de l'écosystème EME (ses composantes techniques et processus de design) aux cas d'étude d'autres chercheurs (13) - qui permet de cerner les perspectives de recherche qui seront développées par peritext
- Janvier 2016 : Publication collective de l'article "Clues, Anomalies, Understanding"(14) qui sur la forme propose une première itération de peritext, via la réalisation d' un site web généré à partir de contenus écrits selon une syntaxe markdown+json (11)
- Été 2016 : Première spécification & implémentation de Peritext sous la forme d'une bibliothèque javascript unique utilisant des données encodées en markdown & BibTeX et stockées dans un dossier de fichiers textes (12)
- Septembre 2016 : présentation du projet Peritext à la Maison des Sciences de l'Homme de Lille
- 2016-2017 : développement de Peritext dans le cadre du projet Quinoa pris en charge par le médialab Sciences Po (6) pour le programme FORCCAST (8) - les spécifications du format évoluent à cette occasion
- Mai-Septembre 2017 : exposition du projet Peritext au festival international de design graphique de Chaumont lors de l'exposition "une cartographie de la recherche en design graphique" (24)
- Juin-Août 2017 : refactorisation du projet comme un écosystème ouvert de modules, écriture de la spécification opérationnelle du format peritext en json-schema, développement de l'éditeur Ovide

22. Thèse actuellement en cours de rédaction ...

Références

1. DE MOURAT, Robin. Vectors – une analyse [en ligne]. Disponible à l'adresse : <http://dicto-formats.herokuapp.com/share/transcription/vectors-une-analyse?display=fullscreen&showtags=1&showrailway=1&leftcolumn=50>
Analyse par étiquettage des différents articles du journal Vectors
Mentions: , p. 34, p. 35, p. 27
2. Site web hypotheses. [en ligne]. Disponible à l'adresse : <http://hypotheses.org/>
Mentions: , p. 26
3. DE MOURAT, Robin et RICCI, Donato. Dicto – site web. [en ligne]. 2017. Disponible à l'adresse : <http://dicto-playground.herokuapp.com/>
Site web de démonstration de l'outil Dicto
Mentions: , p. 34
4. HUSTWIT, Gary. Objectified. [en ligne]. Disponible à l'adresse : <https://www.youtube.com/watch?v=TyofGn8fiUU>
Mentions: , p. 33, p. 24
5. *Dataset très simple*.
Mentions: , p. 38, p. 39, p. 40, p. 41, p. 15, p. 15, p. 16
6. Prince XML. [en ligne]. Disponible à l'adresse : <https://www.princexml.com/>
Mentions: , p. 51, p. 24
7. Spécification W3C pour le css print. [en ligne]. Disponible à l'adresse : <https://www.w3.org/TR/css-print/>
Mentions: , p. 51, p. 21
8. Framework javascript next.js. [en ligne]. Disponible à l'adresse : <https://zeit.co/blog/next3>
Mentions: , p. 51
9. Markdown – wikipedia. [en ligne]. Disponible à l'adresse : <https://fr.wikipedia.org/wiki/Markdown>
Mentions: , p. 53
10. Médialab Sciences Po. [en ligne]. Disponible à l'adresse : <http://www.medialab.sciences-po.fr/fr/>
Mentions: , p. 53, p. 57, p. 45
11. DE MOURAT, Robin. Éditeur Ovide. [en ligne]. Disponible à l'adresse : <http://ovide.surge.sh/>
Mentions: , p. 53, p. 53, p. 15, p. 21
12. Programme FORCCAST. [en ligne]. Disponible à l'adresse : <http://controverses.org/>
Mentions: , p. 57, p. 45
13. An Inquiry Into Modes Of Existence. [en ligne]. Disponible à l'adresse : <http://modesofexistence.org/>
Mentions: , p. 57
14. AIME, expérimentation de quelques instruments d'écoute. [en ligne]. Disponible à l'adresse : <http://modesofexistence.org/-recherche-en-design-re-penser-les-normes-de-la-production-scientifique/>
Mentions: , p. 57, p. 20
15. *Un exemple d'embed soundcloud*.
Mentions: , p. 29
16. *Exemple d'embed de code svg tiré de raw graphs*.

Mentions: , p. 30

17.Exemple d'embed basique d'un tweet.

Mentions: , p. 31

18.Exemple d'embed d'un post facebook.

Mentions: , p. 32

19.

Clues, Anomalies, Understanding – digital companion. [en ligne]. 2015. Disponible à l'adresse : <http://modesofexistence.org/anomalies/>

Mentions: , p. 57

20.Première implémentation de Peritext. [en ligne]. Disponible à l'adresse : <https://github.com/peritext/peritext>

Mentions: , p. 57

21.

Compte-rendu du workshop < open aime >. [en ligne]. 2015. Disponible à l'adresse : <http://modesofexistence.org/open-aime-workshop-11-12-june-paris/>

Mentions: , p. 57

22.

RICCI , Donato, DE MOURAT , Robin, LECLERQ, Christophe et LATOUR, Bruno. Clues, Anomalies, Understanding. Detecting underlying assumptions and expected practices in the Digital Humanities through the AIME project. [en ligne]. 2015. Visible Language. Disponible à l'adresse : <http://visiblelanguagejournal.com/issue/172/article/1172>

Mentions: , p. 57

23.

RICCI , Donato, DE MOURAT, Robin et BOULANGER , Pierre-Laurent. AIME: opening the context of a Humanities inquiry. [en ligne]. 2015. Disponible à l'adresse : <http://echappees.esapyrrenees.fr/content/2-numeros/3-numero3/5-an-account-of-digital-humanities-from-the-aime-project/echappees-an-account-of-digital-humanities-from-the-aime-project.pdf>

Mentions: , p. 57

24.

RICCI, Donato. An Account of Digital Humanities from the AIME Project [en ligne]. 2016. Revue échappées. Disponible à l'adresse : <http://echappees.esapyrrenees.fr/content/2-numeros/3-numero3/5-an-account-of-digital-humanities-from-the-aime-project/echappees-an-account-of-digital-humanities-from-the-aime-project.pdf>

Mentions: , p. 27, p. 24, p. 24, p. 24

25.

Outil et de prévisualisation d'édition de la grammaire vega. [en ligne]. Disponible à l'adresse : <https://vega.github.io/editor/#/edited>

Mentions: , p. 0, p. 42

26.Vega lite. [en ligne]. Disponible à l'adresse : <https://vega.github.io/vega-lite/docs/>

Mentions: , p. 39, p. 0

27.Exemple de ressource codefiles.

Mentions: , p. 36

28.Mon beau réseau roi des forêts.

Mentions: , p. 46

29.Scalar. [en ligne]. Disponible à l'adresse : <http://scalar.usc.edu/>

Mentions: , p. 19

30.

Ted Nelson demonstrates Xanadu Space. [en ligne]. Disponible à l'adresse : https://www.youtube.com/watch?v=En_2T7KH6RA
Ted Nelson présente le logiciel Xanadu Space qui spatialise en trois dimensions un système hypertexte.

Mentions: , p. 20, p. 20

31. La norme OpenURL et la technologie Context Objects in Span. [en ligne]. Disponible à l'adresse : <http://blog.stephanepouyllau.org/116>
Mentions: , p. 19
32. CitationStyles.org. [en ligne]. Disponible à l'adresse : <http://citationstyles.org/>
Mentions: , p. 19, p. 24
33. Ressource processing.js.
Mentions: , p. 43, p. 44
34. ManyLines. [en ligne]. Disponible à l'adresse : <http://tools.medialab.sciences-po.fr/manylines>
Mentions: , p. 45
35. DE MOURAT, Robin. *Exemple de présentation SVG*.
Mentions: , p. 48
36. DE MOURAT, Robin. *Histoire du Fort et des combats de 1870-1871. Une histoire du fort d'Issy*
Mentions: , p. 47
37. DE MOURAT, Robin. *Histoire de la visualisation*. Cette présentation traite de l'histoire de la visualisation
Mentions: , p. 47
38. Le modèle de document Peritext.
Mentions: , p. 14
39. Projet MONADE. [en ligne]. Disponible à l'adresse : https://www.mshb.fr/projets_mshb/monade/2289/
Mentions: , p. 57
40. LE SIGNE. Une cartographie de la recherche en design graphique - Biennale de design graphique 2017. [en ligne]. Disponible à l'adresse : <http://www.centreinternationaldugraphisme.fr/fr/cig/page/biennale-de-design-graphique-2017/expositions/une-cartographie-de-la-recherche-en-design-graphique>
Mentions: , p. 57
41. Pandoc. [en ligne]. Disponible à l'adresse : <https://pandoc.org/>
Mentions: , p. 11
42. Une visualisation hébergée sur tableau. [en ligne]. Disponible à l'adresse : https://public.tableau.com/views/MM34_5MillenniaofSolarEclipses/SolarEclipse?:embed=y&:toolbar=no&:loadOrderID=0&:display_count=yes
Mentions: , p. 28
43. COMMUNAUTÉ P5.JS. Site officiel de p5.js. [en ligne]. Disponible à l'adresse : <https://p5js.org/>
Mentions: , p. 44
44. Conventions de conversion de données pour le projet draft-js. [en ligne]. Disponible à l'adresse : <https://draftjs.org/docs/api-reference-data-conversion.html>
Mentions: , p. 13
45. BUSH, Vannevar et OTHERS. As we may think. *The atlantic monthly*. 1945. Vol. 176, n° 1, pp. 101-108.
Mentions: , p. 19
46. Le cabinet d'Ole Worm. 1655. Reproduction de la gravure conservée à Modène représentant un cabinet de curiosités
Mentions: , p. 8
- 47.

DE MOURAT, Robin. Répertoire peritext-core sur github. [en ligne]. Disponible à l'adresse : <https://github.com/peritext/peritext-core>

Mentions: , p. 13, p. 10

48. Peritext core – documentation technique. [en ligne]. Disponible à l'adresse : <https://peritext.github.io/peritext-core/>

Mentions: , p. 13

49. INTERNET ENGINEERING TASK FORCE. Spécification JSON Schema. [en ligne]. Disponible à l'adresse : <http://json-schema.org/>

Mentions: , p. 13

50. L'organisation peritext sur github. [en ligne]. Disponible à l'adresse : <https://github.com/peritext/>

Mentions: , p. 54

51. API du générateur epub.

Mentions: , p. 56

52. API du contextualiseur vegalite.

Mentions: , p. 54

53. API du gabarit << web-garlic >>.

Mentions: , p. 55

54. Exemple d'utilisation du générateur next comme API. [en ligne]. Disponible à l'adresse : <http://peritext.robindemourat.com/static/generated/resources.json>

Mentions: , p. 51

55. EUGÈNE DE, Gayffier. Herbier forestier de la France. Reproduction par la photographie... des principales plantes ligneuses qui croissent spontanément en forêt. Description botanique... [en ligne]. 1868. Disponible à l'adresse : <http://gallica.bnf.fr/ark:/12148/btv1b86260659/f19.double?>

Mentions: , p. 10

56. Les modules de peritext.

Mentions: , p. 18

Glossaire

contextualiseur

Mentions: , p. 21, p. 21, p. 22, p. 23, p. 23

CSS

Mentions: , p. 21, p. 21

format json

Mentions: , p. 11, p. 13

gabarit

Mentions: , p. 22, p. 21, p. 21, p. 50, p. 50, p. 50, p. 50, p. 21, p. 21

générateur

Mentions: , p. 22, p. 22, p. 22, p. 50

Michel-Ange

Mentions: , p. 25, p. 25

Très Grandes Infrastructures de Recherche

Mentions: , p. 12

Index des auteurs

Pierre-Laurent Boulanger

Mentions: , p. 57

Vannevar Bush

Mentions: , p. 19

Communauté p5.js

Mentions: , p. 44

Robin De Mourat

Mentions: , p. 34, p. 35, p. 34, p. 53, p. 53, p. 57, p. 57, p. 27, p. 48, p. 47, p. 47, p. 13, p. 10, p. 15, p. 21

Gayffier Eugène de

Mentions: , p. 10

Gary Hustwit

Mentions: , p. 33, p. 24

Internet Engineering Task Force

Mentions: , p. 13

Bruno Latour

Mentions: , p. 57

Le Signe

Mentions: , p. 57

Christophe Leclercq

Mentions: , p. 57

Donato Ricci

Mentions: , p. 34, p. 57, p. 57, p. 27, p. 24, p. 24, p. 24

Peritext - publication multimodale orientée contexte

Une introduction

, Robin De Mourat

En amont comme en aval de l'acte de publication, les textes produits et échangés par les chercheurs universitaires convoquent de manière de plus en plus intime les environnements numériques (au premier rang desquels le web) dans lesquels ils sont inscrits.

Il est donc nécessaire d'inventer de nouvelles manières de connecter les modes de production des documents académiques avec les contextes du web. Cette recontextualisation doit opérer dans les deux sens : les environnements numériques et toutes les ressources et formes qu'ils proposent doivent être plus significativement convocables dans les documents académiques, et les documents académiques doivent être plus significativement contextualisables dans les environnements numériques. C'est là le projet à la base de Peritext.