

Capture et modélisation de l'activité utilisateur pour l'évaluation d'applications d'analyse visuelle de données

Romain Vuillemot

Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205
F-69621, France
romain.vuillemot@insa-lyon.fr
<http://liris.cnrs.fr/romain.vuillemot>

Résumé. Nos travaux s'intéressent au développement et à l'évaluation d'applications d'analyse visuelle de données, basées sur le paradigme d'architectures orientée service. Dans cet article nous montrons comment ce type d'architecture permet de capter automatiquement l'activité utilisateur et de la modéliser sous forme de diagrammes d'état. Nous utilisons comme cas d'étude l'exploration multi-résolution de graphes, et discutons la généralisation de notre approche à d'autres tâches.

1 Evaluation des applications d'analyse visuelle de données

Le domaine de l'analyse visuelle de données (*Visual Data Analytics*) est apparu relativement récemment, et combine les méthodes de visualisation d'information, avec celles d'analyse automatique de données, afin de réaliser des tâches exploratoires. Les domaines d'application sont nombreux, et vont de l'intelligence économique, jusqu'à l'analyse de données biologiques. Le livre « *Illuminating the path* » de Thomas, J. J. & Cook, K. A. (2005) trace un agenda et identifie les principaux verrous à relever, dont celui que constitue l'évaluation des applications qui est considéré comme un pilier essentiel « *Visual analytics must develop meaningful and effective techniques to evaluate the actual value any specific visual analytic technique may provide* ».

La conception et l'évaluation d'applications d'analyse visuelle relèvent de compétences multiples et variées, car elles impliquent toute la chaîne de traitement de l'information, du modèle de données, en passant par les techniques d'analyse automatique, jusqu'à la visualisation et l'interaction qui doivent minimiser la surcharge visuelle (par exemple le nombre d'éléments présents à l'écran) et cognitive (par exemple le temps d'apprentissage) de l'utilisateur.

L'évaluation des interfaces, comme par exemple estimer leur capacité à résoudre une tâche donnée en un minimum de temps et en réduisant les taux d'erreurs, intervient à différents stades de développement de l'application. Comme toute interface utilisateur, la métrique d'évaluation est tout d'abord le respect des recommandations de design (types check-lists) qui sont souvent ouvertes et très peu formelles, et qui empêchent la mise en place d'une vraie démarche scientifique. Ensuite, l'évaluation peut impliquer directement l'utilisateur avec des méthodes dites centrées utilisateurs, en l'impliquant dans la boucle d'itération de chaque étape du développement et en lui demandant son avis. L'avis de l'utilisateur peut être qualitatif (réponse à des questions plus ou moins ouvertes) mais aussi quantitatif comme la comparaison d'indicateurs (taux d'erreur, vitesse de réalisation d'une tâche) face à des alterna-

tives de designs qui explorent plusieurs valeurs d'une variable (couleur, taille d'un élément, etc..). Cette approche permet, elle, une démarche scientifique : définition d'hypothèses, expérimentation et analyse statistique des résultats. Cependant elle ne met pas en situation réelle l'utilisateur, qui n'est pas dans son environnement quotidien de travail. Face cela, Shneiderman, B. & Plaisant, C. (2006) proposent d'effectuer des études à long terme et multi-variables, qui peuvent prendre plusieurs années. Ils recommandent également un traçage de l'activité de l'utilisateur afin d'évaluer son activité.

Nos travaux concernent l'évaluation quantitative de l'efficacité d'une interface. Nous pensons qu'il est nécessaire d'automatiser et de systématiser ce type d'évaluation, de manière fréquente et la moins visible possible pour l'utilisateur. Pour cela nous avons déjà montré qu'utiliser le paradigme d'architectures orientées service était adéquat en termes de mise en place d'indicateurs quantitatifs pour capter l'activité utilisateur (Vuillemot, R. (2009)).

Dans cet article nous rappelons tout d'abord les principes des architectures orientées services pour la visualisation (section 2) et ensuite illustrons la mise en place d'une capture de l'activité et de la modélisation de l'utilisateur sous forme de diagrammes d'états (section 3). Nous discutons les principales perspectives de nos travaux en conclusion (section 4), et leur généralisation à d'autres tâches.

2 Architecture orientée services pour la visualisation

Une architecture orientée service a pour principale caractéristique de rendre disponible, au travers du web, des applications (que nous appellerons *services*) qui ne sont visibles que par leurs entrées et leurs sorties. Ces services ont alors les caractéristiques suivantes :

- ils sont indépendants de l'environnement d'exécution;
- ils peuvent être mis à jour à distance sans déploiement chez le client.

Les services peuvent avoir un rôle fonctionnel très variable : du simple calcul arithmétique à la mise à jour de données dans des bases de données. Ils peuvent également être composés (*mashups*) entre eux et former des applications plus complexes. Cette composition est très souvent réalisée manuellement par un programmeur via une API, et parfois automatisable si les services sont décrits sous forme sémantique.

Exemple de ManyEyes. ManyEyes¹ d'IBM offre un service web (accessible via un site web) qui permet de créer en ligne des représentations visuelles, telles que des nuages de mots clés ou des graphes. Un utilisateur s'identifie sur le site, télécharge son jeu de donnée (texte, tables) et créé ensuite des visualisations multiples qu'il personnalise (taille, couleur) et qui sont disponibles sous forme d'URL. Cette URL est une vue sur les données qui peut être partagée, commentée et évaluée par d'autres utilisateurs (paradigme du Web 2.0). Cette approche est prometteuse car elle permet désormais une évaluation *sociale* par rapport à l'approche *cognitive* classique, si l'usage est massif. Mais le système ne permet ni d'offrir un cadre d'analyse de données sophistiqué, ni d'effectuer des tâches exploratoires.

Notre approche. Notre approche est assez similaire techniquement, étant donné que nous considérons la visualisation comme une vue accessible sous forme de ressource publiée sur le web (URL). Mais nous souhaitons cependant non pas uniquement publier la vue finale des

¹ <http://manyeyes.alphaworks.ibm.com/manyeyes/>

données (autrement dit la visualisation), mais également tout le processus qui y conduit, à savoir toutes les étapes de traitement (comme celles issues du modèle de référence). A tout moment l'utilisateur peut avoir accès aux étapes qui constituent la visualisation et ainsi connaître l'anatomie d'une visualisation (Vuillemot, R. (2009)).

La figure 1 illustre le résultat d'une composition de services web, dont le résultat est un nuage de mots-clés. Les étapes de traitement sont disponibles sous forme de services web, tels que l'analyse de texte ou l'association entre les données et les variables graphiques (Vuillemot, R et al. (2009)).



FIG. 1 – Voici un nuage de mot clé accessible sous forme de Service Web dont l'URL est : <http://HOST/posvis/tagcloud.php?chapter=1&collection=1&startSelectionA=0&endSelectioA=4892&startSelectionB=4893&endSelectionB=&orderBy=appearence&sizeBy=frequencyInB&colorBy=frequencyInA&minSize=50&maxSize=500&limitMinSize=50>

L'URL doit ensuite être intégrée dans une application interactive (par exemple une application JAVA SWING ou un navigateur web), et les widgets de l'application joueront le rôle de contrôleur et passeront les variables en paramètre de l'URL du service. Ainsi il sera possible d'effectuer des actions de filtrage, sélection ou de changement de modèle de données.

3 Capture et analyse de l'activité de l'utilisateur

Les services de visualisation permettent donc d'inclure dans n'importe quel type application interactive une représentation visuelle, qui est hébergée sur un serveur web et accessible via une URL. Nous allons désormais nous intéresser à la visualisation de graphes, et deux de ses principaux challenges que sont 1) trouver la bonne disposition des sommets et des arêtes et 2) utiliser la bonne stratégie d'exploration. Notre objectif sera ensuite de montrer comment

il est possible de capturer et d'analyser l'activité de l'utilisateur (à savoir ce qu'il réalise effectivement) dans l'exploration de ces graphes.

3.1 Application à l'exploration multi-résolution de graphes

Disposition du graphe. Concernant la disposition nous avons publié sous forme de services web différentes dispositions (*layout*) de graphes, en rendant la bibliothèque Graphviz disponible sous forme d'URL, à laquelle il faut passer des paramètres pour choisir le type de disposition et indiquer quel jeu de donnée utiliser. La figure 2 illustre les choix disponibles :

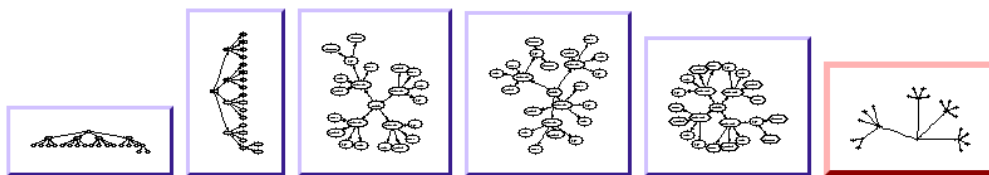


FIG. 2 – Dispositions de graphe accessible au moyen de services web.

Exploration multi-résolution. La disposition du graphe résulte en une image qu'il est parfois difficile d'explorer de part sa taille ou le grand nombre d'attribut de données. Une approche possible pour limiter la surcharge visuelle et le nombre d'interaction, est de montrer une vue globale pour ensuite prendre une décision et affiner le parcours jusqu'au niveau minimal de zoom et maximal de détails (Shneiderman, B. (1996)). Une contrainte majeure est de garder une certaine consistance dans les différents résolutions, nous avons pour cela gardé la même disposition quelque soit la résolution. L'utilisateur pourra donc zoomer en avant et en arrière sur le graphe comme une carte normale, et le niveau de détail variera, soit automatiquement selon le niveau de zoom, soit par filtrage manuel (via des listes). Nous avons déjà implémenté un prototype en utilisant Google Earth (dans lequel nous avons supprimé toute référence géographique) comme environnement interactif, et en changeant la résolution du graphe en fonction de l'altitude de l'utilisateur (Vuillemot, R. & Rumpler, B. (2008).)

3.2 Modélisation sous diagramme d'état l'activité de l'utilisateur

Les appels d'URL générés par l'application Google Earth, et captés par le serveur permettent désormais de construire un diagramme d'état qui représente l'activité de l'utilisateur. Cette étape est immédiate car cela revient à analyser des logs structurés du serveur qui héberge les graphes et qui nous donnent des informations d'accès aux services (identifiant utilisateur, identifiant application, timestamp, etc.).

La figure 3 montre une série de diagrammes automatiquement générés et mis à jour au fur et à mesure de l'activité d'exploration visuelle d'un graphe, à partir des logs du serveur web. Les états des diagrammes sont représentés par des carrés aux bords arrondis (qui correspondent à chaque fois à une image du graphe), et les transitions d'un état à l'autre par une flèche. Le nom des carrés est le nom associé aux vues sur les données (vue globale, zoom, détails). L'étiquette des flèches est le nombre de transition entre ces vues. A gauche, sont représentés les diagrammes qui synthétisent les activités de plusieurs utilisateurs, au cours de plusieurs sessions d'utilisation. A droite, ces diagrammes sont agrégés en un seul afin de faire émerger le motif général de la navigation.

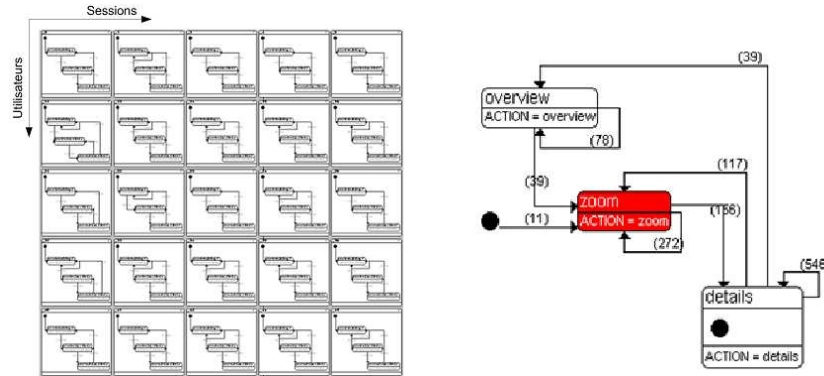


FIG. 3 – A gauche l’activité d’utilisateurs au travers de plusieurs sessions. A droite l’agrégation de ces utilisateurs et sessions en un seul diagramme. La zone rouge indique l’état actuel du système de traitement de logs

Résultat. Le principal résultat que nous pouvons mettre en avant est que les utilisateurs commencent l’application (rond noir foncé) à un état de zoom, alors qu’ils devraient commencer par la vue globale (selon Shneiderman, B. (1996)). Une deuxième remarque est que beaucoup d’utilisateurs repassent à la vue global directement sans passer par l’état de zoom, cela indique au designer qu’il serait utile de mettre en place un moyen rapide de passer entre ces deux étapes (sans passer par le zoom intermédiaire).

Le diagramme reste cependant une vue restreinte sur l’activité réelle, car 1) la vue est du côté serveur, et 2) chaque étape est étiquetée selon des sous-tâches principales par le programmeur (vues sur les données), alors que le rôle de ces étapes est très contextuel, et peut avoir un rôle différent selon l’étape précédente. Enfin, la granularité des appels n’est pas assez fine pour avoir une bonne connaissance de l’activité réelle de l’utilisateur sur une vue. Par exemple il peut effectuer des rotations au niveau du zoom, ou tout type d’interaction qui ne réalise pas d’autre appel au serveur, le diagramme restera à l’état de zoom.

4 Perspectives

Nous avons montré comment les applications d’analyse visuelle de données, basées sur une architecture orientée services, permettent de capter l’activité de l’utilisateur. Cette activité peut ensuite être modélisée sous forme de diagrammes d’état et être confrontée à la tâche utilisateur initialement prévue par le concepteur.

Notre principale perspective est de diversifier l’étude d’applications et de paradigmes d’interaction et de tâches. Par exemple, Keim et al. (2006) ont revu le mantra de Shneiderman en “*Analyse First - Show the Important - Zoom, Filter and Analyse Further - Details on Demand*” qui est plus approprié à l’analyse visuelle de données. A noter que pour être pertinent, nous devons analyser des tâches, utilisateurs et données réels.

Une autre perspective qui nous paraît très prometteuse est de passer d’une phase d’analyse d’applications, à une phase *générative* d’applications. Autrement dit, permettre aux designers de générer des « préconceptions » d’interfaces basées sur des analyses précédentes. Ces « préconceptions » devront valider des règles qui intègrent les tâches ou con-

Capture et modélisation de l'activité utilisateur pour l'évaluation d'applications...

signes de design. Autrement dit, inclure dans les diagrammes d'état des contraintes génériques qu'il est nécessaire de respecter. Ces contraintes peuvent être complexes à modéliser comme la consistance des interfaces ou leur universabilité. Ces règles devront pouvoir être exprimées et incluses dans un système de vérification de règles formelles, qui sera à définir et à intégrer à l'architecture.

Références

- Keim, D. A. & Mansmann, F. & Schneidewind, J. & Ziegler, H. (2006). Challenges in Visual Data Analysis IV '06: Proceedings of the conference on Information Visualization, IEEE Computer Society, , 9-16
- Shneiderman, B. (1996). *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages, IEEE Computer Society, 336
- Shneiderman, B. & Plaisant, C. (2006). *Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies*. BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors, ACM, 1-7
- Thomas, J. J. & Cook, K. A. (2005) *Illuminating the Path: The Research and Development Agenda for Visual Analytics National Visualization and Analytics*.
- Vuillemot, R. (2009). *Modèle et Architecture Orientée Services pour la Visualisation d'Information Interactive*. Rencontres Doctorales IHM'09.
- Vuillemot, R & Clement, T. & Plaisant, C. & Kumar, A. (2009). *What's Being Said Near "Martha"? Exploring Name Entities in Literary Text Collections*. IEEE Symposium on Visual Analytics Science and Technology (VAST).
- Vuillemot, R. & Rumpler, B. (2008). *Mapping visualization on-demand onto a virtual globe: an appealing complement to browser-based navigation*. HT '08, ACM, 249-250.