De l'apprentissage artificiel pour l'apprentissage humain : de la récolte de traces à la modélisation utilisateur

Marc Damez-Fontaine

UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

Institut Formation Doctorale

THESE DE DOCTORAT DE
L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité
Informatique

Présentée par

Marc Damez-Fontaine

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITE PIERRE ET MARIE CURIE

Sujet de la thèse :

De l'apprentissage artificiel pour l'apprentissage humain : de la récolte de traces à la modélisation utilisateur

soutenue le 18 Septembre 2008

devant le jury composé de

Bernadette Bouchon-Meunier (Directeur de recherche, CNRS)	Co-directeur de thèse
Charles Tijus (Professeur, Université Paris 8)	Co-directeur de thèse
Khaldoun Zreik (Professeur, Université Paris 8)	Rapporteur
Jean-Daniel Fekete (Directeur de recherche, INRIA)	Rapporteur
Brigitte Trousse (Chargée de recherche, INRIA)	Examinateur
Thierry Artières (Professeur, Université Paris 6)	Examinateur
Christophe Marsala (Maître de conférences, Université Paris 6)	Co-Encadrant

Résumé

Les interactions entre l'homme et l'ordinateur peuvent être étudiées sous plusieurs angles : certaines analyses s'attachent à décrire le comportement humain utilisant des outils informatiques, d'autres cherchent à optimiser l'ergonomie de l'interface, et enfin, d'autres cherchent à améliorer l'interactivité entre l'utilisateur et l'immense quantité d'informations véhiculées par le dispositif numérique. Après avoir récapitulé l'ensemble des propriétés nécessaires à la conception d'un système générique de récolte de données issues directement de l'interaction homme-machine, nous présentons des méthodes de visualisations de ces données permettant la mise à jour de propriétés utiles ainsi que des exemples d'analyses automatiques par des méthodes algorithmiques.

Nous proposons un nouveau formalisme générique de récolte de données provenant des interactions homme-machine. Ce modèle permet toutes les modélisations et l'application de toutes les méthodes évoquées précédemment. Nous présentons également des outils d'analyses, qui permettent la reconnaissance automatique de caractéristiques de comportements des utilisateurs, ainsi qu'un outil de visualisation de ces données qui permettent la remise en contexte de l'action de l'utilisateur ainsi que la comparaison de l'activité de plusieurs individus sur une ou plusieurs interfaces. Deux algorithmes sont introduits pour faciliter la lecture de la visualisation.

Ces travaux doivent s'appliquer dans un contexte scolaire et quelques théories pédagogiques élaborées par les psychologues cognitivistes spécialisés ce domaine ont été étudiées. Une plateforme permettant l'implémentation de méthodes d'apprentissage artificiel décrit les modalités d'application pour adapter automatiquement l'interaction et fournir une aide personnalisée. Des méthodes algorithmiques d'apprentissage artificiel sont également détaillées suivant une typologie des méthodes de traitement des données. Deux analyses effectuées sur des données récoltées lors d'expériences que nous avons menées, ont permis d'élaborer des méthodes de personnalisations automatiques de l'interaction homme-machine pour l'enseignement.

Mots-clés : interaction homme-machine, étude de traces, visualisation de données, apprentissage artificiel, analyse automatique, caractérisation de comportements.

Abstract

Human-computer interactions can be studied under several angles: some analyses attempt to describe human behavior using the computer as a tool, others seek to optimize the ergonomics of the interface, and finally, others seek to improve the interactivity between the user and the immense quantity of information conveyed by the numerical device. After having studied the properties necessary to design a generic system to gather data from human-computer interaction, we present methods of data visualization for the purpose of identifying useful properties, as well as examples of automatic analyses by algorithmic methods.

We propose a new generic formalization of data harvesting, designed for data derived from human-computer interaction. This formalization allows all modeling, and all applications of all the methods previously exposed. We also present tools for analysis which allow automatic recognition of behavior characteristics of users, as well as a tool for data visualization. Application of the data-visualization tool permits us to replace the global interaction of the user in context as well as comparing the activity of several users on one or more interfaces. Two algorithms are introduced to facilitate the reading of this representation.

As this work must be applicable in a scholastic context, we have studied various teaching theories elaborated by cognitivists. This has enabled us to develop a platform which allows the implementation of methods for machine learning and describes ways for adapting the interaction automatically to provide personalized help. Algorithmic methods of artificial intelligence are also classified by a typology of data processing. Two analyses carried out on data collected during experiments enabled us to work out methods that can be used for teaching by automatic personalization of human-computer interaction.

Keywords: human computer interaction, traces analysis, visualizing information, machine learning, automatic analysis, behavior characterization.

Remerciements

Table des matières

Introduction

Cadre de l'étude

Cette thèse entre dans le cadre de la *Convention Industrielle de Formation par la Recherche* (CIFRE) passée entre l'Université Pierre et Marie Curie (Paris VI) et l'entreprise Sejer (groupe Editis). A la suite du projet *Adaptation du Cartable Electronique à ses Divers Utilisateurs* (ACEDU) lancé par le *Réseau National des Technologies Logicielles* (RNTL), l'université et l'entreprise ont souhaité poursuivre l'étude initiée par l'un des sous projets d'ACEDU : l'étude des traces d'interaction homme-machine.

Lorsque Sejer a voulu s'enrichir de l'expertise théorique de l'université dans le domaine de l'analyse des utilisateurs du cartable électronique, l'université a saisi l'occasion d'obtenir un cadre d'expérimentation pratique pour les théories qu'elle développe.

Notre rôle a consisté à faire le lien entre les deux parties en menant des recherches expérimentales et théoriques. Pour ce faire, nous avons aussi été encadrés par le laboratoire de Cognition et Usage de l'Université de Paris VIII, qui a apporté son expertise dans la modélisation cognitive de l'utilisateur.

Problématiques

Les récentes avancées logicielles et l'augmentation continue de l'utilisation d'Internet dans le monde éducatif ont permis aux professionnels et aux chercheurs de concevoir de nouveaux supports d'enseignement. Ainsi, le suivi individuel de l'état d'avancement de l'apprentissage de l'élève par les professeurs a été rendu possible. Cependant, les problématiques que soulèvent ces nouvelles possibilités n'ont pas encore été totalement étudiées car les méthodes et les techniques mises en œuvre sont assez complexes. Toutes les questions, relatives à un domaine plus général d'étude des comportements d'utilisateurs dialoguant avec une interface, doivent pouvoir s'appliquer dans un contexte scolaire et trouver des réponses afin d'aider l'élève et son professeur : « Comment analyser l'impact d'une interface sur les comportements des utilisateurs ? » devient « Comment évaluer l'apport d'une méthode d'enseignement sur l'apprentissage des élèves ? »

C'est dans ce cadre, qui a été mis en évidence par les participants au projet ACEDU, que se pose alors la question de l'analyse des données provenant directement de l'interaction de l'élève avec ces nouveaux supports. L'analyse dialogue homme-machine pose alors toutes les questions relatives à l'interaction : Comment l'étudier de façon automatique pour produire un système utile à l'activité humaine ? Comment enregistrer les données d'une interaction ? Quelles données de l'interaction faut-il enregistrer ?

Ces questions sont depuis longtemps au cœur des préoccupations des scientifiques qui cherchent à étudier les comportements humains. Par ailleurs, les sujets d'une interaction doivent aussi être observés afin de les étudier empiriquement.

Lors d'un dialogue homme-machine, beaucoup de données sont naturellement échangées entre les deux parties. La plupart de ces données ne sont habituellement pas enregistrée et sont simplement utiles pour le fonctionnement de la communication. Cependant, leur étude peut révéler d'importantes différences dans le comportement des individus. En particulier, dans un contexte scolaire, une telle étude est capable notamment de mettre en évidence les disparités dans la façon d'appréhender une notion.

Les travaux présentés dans cette thèse portent sur l'étude des données directement issues de l'interaction homme-machine. Ces données sont objectives, car elles représentent le véritable dialogue tel qu'il est réalisé par l'homme et la machine, contrairement aux données issues directement des utilisateurs, telles que des questionnaires par exemple, qui révèlent souvent un caractère subjectif d'appréciation des questions posées et du contexte dans lequel elles le sont.

Plusieurs axes vont guider notre étude : Quelles données doit-on récolter en conservant un cadre générique pour l'amélioration globale du dialogue ? Quelles sont les contraintes à prendre en compte pour satisfaire l'utilisation de méthodes performantes sur de telles données ? Quelles méthodes automatiques d'analyse de données offrent une amélioration de la qualité du dialogue homme-machine ? Quelles améliorations concrètes sont envisageables dans un processus de personnalisation automatique de l'interaction ?

Un enregistrement de toutes les actions d'un utilisateur sur un ordinateur entraîne une surveillance de son activité. Cependant, dans le cadre d'une activité d'apprentissage, ce suivi du dialogue possède un caractère tutélaire, par définition voué à l'épanouissement individuel.

Nous présentons dans notre étude, des analyses qualitatives dont le but est d'offrir un suivi pédagogique personnalisé de l'élève par le professeur. Cette thèse analyse les données issues directement de l'interaction homme-machine (noté HM) dans un but de

personnalisation de l'interface. Il s'agit d'en extraire des informations pertinentes pour la personnalisation. Nos cas d'application portent sur les processus d'enseignement et d'apprentissage, et dans ce cadre, nous souhaitons fournir une analyse constructive et qualitative du dialogue homme-machine pour son amélioration et par conséquent celle du processus d'apprentissage également.

Organisation du document

La première partie de ce document présente un état de l'art des travaux dans le domaine de l'extraction d'information à partir de données issues d'une interface HM. Nous y présentons les différents types de modélisation qui vont de la caractérisation de l'utilisabilité de la machine, à la description du comportement humain en passant par l'adaptation du dialogue lui-même. En suivant une typologie des méthodes d'extraction d'informations à partir des données issues de l'interaction homme-machine, nous analysons les différentes propriétés importantes à inclure dans des traces qui autorisent un grand nombre de modélisations.

Ces données sont souvent représentées dans des visualisations construites par les chercheurs qui tentent de mettre à jour automatiquement des informations lors d'une simple lecture. Nous présenterons donc différentes visualisations tirées de la littérature qui, selon le niveau d'abstraction, mettent à jour de différentes propriétés caractéristiques du dialogue. Des techniques de fouilles de données issues du domaine de l'intelligence artificielle sont ensuite généralement mises en application pour l'automatisation du processus d'extraction d'informations. Nous présentons donc quelques exemples d'applications issus de la littérature scientifique qui traitent de notre problématique, avec en particulier le cas d'application scolaire.

La deuxième partie de ce document introduit un nouveau modèle de récolte de traces. Celui-ci inclut les recommandations formulées lors de l'état de l'art pour la mise en œuvre de l'ensemble des analyses présentées précédemment, centrées autour du dialogue homme-machine.

Nous présentons ensuite un nouveau système de visualisation des données précédemment décrites. Ce système a été conçu pour mettre à jour par simple lecture des propriétés caractéristiques dans l'interaction. Plusieurs outils pour la manipulation de cette visualisation font l'objet d'un développement.

Deux exemples d'analyses à partir de cette visualisation sont également présentés. Ils permettent d'une part, l'analyse de parcours d'utilisateurs sur différentes interfaces et d'autre part, la comparaison de l'accomplissement par différents utilisateurs de plusieurs tâches remises dans un contexte commun.

La troisième et dernière partie de ce document revient sur le cas d'application pratique précédemment évoqué et décrit un système d'aide à l'apprentissage humain. Nous présentons quelques théories issues de la pédagogie moderne qu'il nous paraît intéressant de connaître pour faire fonctionner le système.

Celui-ci est ensuite décrit de façon fonctionnelle pour sa mise en application informatique avec une première phase où le système doit apprendre à fonctionner, et une deuxième phase où il peut fonctionner en temps réel.

Nous revenons également sur les méthodes d'analyses issues de l'intelligence artificielle pour l'extraction d'informations pertinentes pour la compréhension des processus cognitifs lors de l'apprentissage humain.

Enfin, la dernière partie présente deux expériences que nous avons menées pour récolter des traces et les analyser avec des algorithmes statistiques. La première expérience a été effectuée dans un cadre relativement contraint afin de construire une expertise alors que la deuxième présente une analyse de traces relevées dans un cas réel d'enseignement en salle multimédia, accompagnée par un professeur.

Nous terminons en traçant un bilan et une conclusion de notre travail et nous présentons un ensemble de perspectives qui en découlent.

Partie 1 - Etat de l'art

"Some of the most crucial steps in mental growth are based not simply on acquiring new skills, but on acquiring new administrative ways to use what one already knows."

Seymour Papert.

Introduction de la première partie

Au début du XXI^{ème} siècle, l'ordinateur est encore au service de l'homme, son concepteur. La littérature scientifique établit des modèles d'utilisateurs pour améliorer l'interaction de l'homme avec la machine, et les résultats de ces analyses servent à la fois l'homme, la machine, et le dialogue homme-machine. Ce dialogue est au centre de beaucoup d'études qui constitue notre état de l'art de la modélisation utilisateur à partir de traces d'utilisation.

Les traces d'utilisation peuvent être de différentes natures et provenir de différentes sources. A ce jour, il n'existe aucun standard ou formalisme pour l'enregistrement des données provenant de l'interaction HM. Les expériences cherchant à modéliser le comportement utilisateur pour personnaliser l'interaction sont menées indépendamment les unes des autres. Pourtant les données automatiquement produites par ce dialogue forment un ensemble décrivant précisément les différentes caractéristiques de ce dialogue. Dans les années 2000, le développement et l'utilisation de plus en plus massive d'internet a encore largement contribué à la nécessité de personnaliser les interactions homme-machine.

Dans un premier chapitre, une analyse des différents objectifs poursuivis par les modèles centrés autour du dialogue homme-machine est présentée. Nous y détaillons les modèles servant à analyser le comportement humain, les modèles caractérisant l'interaction homme-machine afin d'optimiser le dialogue, et les modèles décrivant la machine et les caractéristiques optimales permettant sa meilleure utilisation. Ces trois points de vue sont intrinsèquement liés, et il va sans dire qu'un changement dans l'interface graphique d'un ordinateur change le comportement de l'homme manipulant cet ordinateur et par conséquence le dialogue HM.

Dans le deuxième chapitre, différents types de données relevées pour des analyses automatiques sur le dialogue HM sont présentés. Pour ce faire, nous suivrons la typologie des méthodes définie dans [Hilbert & Redmiles, 2000]. En effet, les auteurs de cet article présentent l'état de l'art de l'extraction d'informations à partir des évènements provenant de l'interaction HM. La typologie proposée permet un bon survol

des méthodes rencontrées dans la littérature pour récapituler l'ensemble des propriétés nécessaires à la caractérisation des traces issues de l'interaction HM pour l'extraction d'informations.

Dans le troisième chapitre, différentes visualisations de données pour la modélisation de l'interaction HM sont présentées. En effet, les données relevées pour ces analyses sont généralement trop abondantes et complexes pour une interprétation immédiate. Après avoir exposé les raisons de la nécessité d'une visualisation de ces données, quelques exemples permettant différentes analyses sont présentés.

Le quatrième chapitre de cette partie présente quelques exemples d'applications utilisant des méthodes de fouilles de données pour l'automatisation de l'analyse du dialogue HM. Ainsi, une application d'aide au travail collaboratif et plusieurs applications de construction et de caractérisation de profils d'utilisation sont présentées. Nous discutons ensuite d'un cas d'application où la nécessité d'une modélisation utilisateur pertinente est cruciale. Il s'agit des agents intelligents dans les systèmes éducatifs. En effet, afin de bien comprendre le cheminement de l'apprentissage chez l'élève, un suivi très attentif du comportement, ainsi qu'une analyse pratique des interfaces utilisées permettent d'éviter des mésinterprétations dans l'enseignement.

Chapitre 1. Modéliser

Toutes les personnes qui souhaitent analyser les dialogues homme-machine sont intéressées par les données provenant des évènements de l'interface. Ce sont en effet des données extractibles automatiquement et très représentatives car très proches de l'interaction HM. Seulement avant de chercher des données à analyser, il est bon de se poser des questions sur le modèle à explorer. Nous revenons donc sur les acteurs de ce dialogue. Plusieurs questions se posent : Quelles sont les entités qui interagissent ? Comment peut-on modéliser le comportement de chacun ? Les techniques d'analyses sont-elles les mêmes ? Produisent-elles des effets différents ?

Nous avons distingué trois types de modélisations formant une synthèse du dialogue. Ces trois points de vue servent trois buts différents : l'analyse psychologique des utilisateurs, l'analyse des adaptations relatives à une interface par catégorie d'utilisation, et l'analyse de l'ergonomie logicielle déterminant la bonne utilisabilité du système.

Chacune des entités homme-interaction-machine est perpétuellement soumise aux réactions des autres et l'analyse de l'une de ces parties pour sa personnalisation modifie l'intégralité du dialogue. Aussi, l'objet au centre de l'analyse n'est plus considéré en tant que tel, et nous tournons cette analyse vers les différentes personnalisations recherchées.

1. L'utilisateur

Chaque utilisateur est unique. Aussi, la psychologie cognitive analyse le comportement des utilisateurs en identifiant les processus impliqués dans l'accomplissement d'actions servant la connaissance humaine. Une fois un modèle de processus mental introduit, l'étude des psychologues se tourne vers la validation du modèle en y implémentant diverses catégories de comportements. Le modèle sera d'autant plus pertinent qu'il sera robuste aux différentes catégories d'utilisateurs. L'analyse permettra donc d'identifier des différences de comportements humains lors de

la réalisation d'un même processus. Une définition généralement acceptée par la communauté scientifique décrit les processus cognitifs en ces termes [Wiki, 2007] :

Les processus cognitifs sont les différents modes à travers lesquels les systèmes naturels ou artificiels traitent l'information en y répondant par une action. Les processus cognitifs sont :

- *perception, attention, sensation*
- *mémoire, représentation, langage*
- *raisonnement, catégorisation, prise de décision, reconnaissance*
- *apprentissage, émotion*
- *comportement individuel et collectif*

Théorie de modélisation cognitive et analyse de tâches

Un modèle d'analyse cognitive décrit les processus mentaux par lesquels l'homme acquiert des informations sur son environnement et les traite pour ajuster son comportement. Afin de comprendre comment modéliser l'activité cognitive d'un utilisateur sur une interface informatique, il faut bien comprendre le mécanisme du processus cognitif associé.

Les données récoltées lors de l'interaction HM ne nous renseignent que sur les actions et réactions des acteurs du dialogue. Pour une analyse des tâches en cours, il est intéressant de chercher le contexte de l'utilisateur. [Kobsa & al., 2001] proposent plusieurs exemples de questions pour mettre à jour ce contexte :

- Quel est le type de l'utilisateur ?
- Quelles sont ses connaissances a priori ?
- Quelles sont ses capacités et ses compétences ?
- Quels sont ses intérêts et ses préférences ?
- Quels sont ses buts ?

La communauté scientifique cherchant des modèles pour l'analyse des tâches utilise également ces informations pour formaliser l'accomplissement des tâches par des utilisateurs.

Nous dressons dans le tableau suivant quelques méthodes d'analyses de tâches sélectionnées dans la littérature :

HTA	Hierarchical Task Analysis [Annett & Duncan, 1967]
GOMS	Goals, Operators, Methods and Selection rules [Card & al., 1983]
UAN	User Action Notation [Hartson & al., 1990]
MAD	Méthode Analytique de Description de tâches [Scapin & Pierret-Goldbreich, 1989]

Figure 1. Quelques techniques d'analyse de tâches

a) L'évaluation d'une analyse HTA reste très subjective, car les principaux critères de validité sont la consistance (le nombre de sous-tâches décrites pour chaque tâche doit être homogène) et la validité empirique (un novice doit être capable d'apprendre la tâche décrite). On trouve dans cette définition la notion de hiérarchie, une tâche se définit par un ensemble de sous-tâches, et la notion d'ordonnancement, l'ordre des sous-tâches par rapport au même niveau hiérarchique et par rapport à d'autres niveaux.

b) La méthode GOMS a introduit la notion d'opérateur (outil pour faire la tâche), de méthode (comment utiliser l'outil), et l'idée d'une certaine optimisation en organisant des règles à appliquer sur les actions. Ces notions ont ensuite été largement reprises par la communauté et déclinées pour des applications plus spécifiques [John & Kieras, 1996]. Par exemple, le modèle KLM-GOMS (Keystroke-Level Model) propose quelques règles d'optimisation et d'organisation dans l'agencement des actions à faire pour de l'édition de texte. L'automatisation d'un autre modèle de la famille GOMS, CPM-GOMS (Cognitive Perceptual Motor-GOMS), est décrite dans [John & al., 2002]. Dans cette étude, les auteurs comparent le temps d'exécution d'une tâche par des utilisateurs avec celui produit par un modèle fonctionnant selon les principes GOMS. Les résultats s'avèrent tout à fait satisfaisants dans les cas de tâches simples, et les auteurs tentent alors de modéliser des tâches plus complexes. Une visualisation des différents niveaux cognitifs associés à la réalisation de cette tâche est détaillée au chapitre 4.

c) La méthode UAN est un langage de description qui permet une évaluation analytique du scénario de la tâche. Les éléments qui composent ce langage sont :

- les symboles et les opérateurs. Les opérateurs sont les différentes actions possibles sur une interface. Elles portent sur des symboles (généralement des menus, boutons, etc.).
- les conditions et les options. Ils sont les contraintes d'ordres logiques permettant l'agencement de la tâche.

- la table des actions utilisateurs, des réactions du système et des états du système. Pour chaque action décrite par les opérateurs, une liste des réactions et des états du système doit être fournie.
- les relations et les contraintes temporelles. Elles représentent les contraintes d'ordonnancement des actions.

d) La méthode MAD fournit un contexte d'application à la tâche, et permet de décrire chaque tâche de manière similaire pour les comparer entre elles. Chaque intervenant dans une tâche collective à réaliser peut ainsi trouver son modèle d'activité. Une nouvelle version de ce modèle, appelé maintenant MDA (*Modèle de Description de l'Activité*) a donné lieu à une implémentation informatique [Baron & al., 2006]. Cet outil possède plusieurs options intéressantes pour décrire la réalisation d'une tâche. On peut ainsi éditer *l'arbre de tâches*, *les caractéristiques de tâches* et *les objets manipulés par l'utilisateur*. Un outil permet d'interroger la cohérence, l'analyse statistique et la recherche approfondie dans le modèle et un autre outil permet de simuler l'exécution d'une tâche pour vérifier notamment le bon ordonnancement des sous-tâches.

Une taxonomie des différents modèles de description de tâche est présentée dans [Balbo & al., 2004]. Il nous semble que la description cognitive la plus approfondie est obtenue par la méthode GOMS qui permet de mettre en évidence chaque mouvement de l'utilisateur, et ainsi, le processus cognitif associé. Comme pour tous les modèles énoncés ci dessus, et contrairement à ce que l'on souhaiterait pour une analyse automatique, les tâches ne sont pas découvertes sur les données, mais directement implantées dans le modèle.

2. L'interaction

Les développeurs d'applications ou de systèmes d'informations conçoivent souvent ceux-ci en imaginant des tâches que l'utilisateur va effectuer. Pourtant, les utilisateurs peuvent vouloir accéder à la même information ou vouloir manipuler le même logiciel à des fins très différentes. Les concepteurs d'interface graphique se posent donc maintenant la question de savoir comment, à partir d'une même interface, fournir des outils de personnalisation permettant l'accomplissement des buts de chacun.

Pour les systèmes adaptatifs automatisés on distinguera deux types de personnalisation :

- celle basée sur la découverte de catégories d'utilisateurs, on parlera alors de méthode de clustering ou regroupement ;

- celle basée sur une connaissance préalable des utilisateurs, on parlera alors de méthode supervisée.

Dans tous les cas, il sera très important pour les concepteurs de systèmes adaptatifs de prêter attention à l'utilisation qui est faite de l'interface. Les données issues de l'interaction HM fourniront donc une très bonne source de données pour la modélisation de l'usage. En effet, les méthodes statistiques, les méthodes de détection de séquences, de comparaison de séquences et de caractérisation de séquences décrites ci après, fourniront un ensemble de descripteurs des différents types d'utilisation d'une interface.

Web Usage Mining

Cette terminologie, définie dans [Cooley & al., 1997], constituant un domaine de la recherche scientifique à part entière, a été admis par la communauté scientifique comme la discipline étudiant les fichiers de traces des serveurs Internet afin de déterminer les besoins des internautes et les éventuelles adaptations qu'un système pourrait produire. La personnalisation de la navigation d'un site et la recommandation lors de la recherche de documents sont les principales applications de ces systèmes. Les données construisant le modèle utilisateur sont ici très incomplètes car seules les requêtes de l'utilisateur sont analysées.

Les expérimentations menées dans ce domaine sont à rapprocher de celles effectuées pour le *Web Content Mining*. A la différence du précédent, les données traitées ici sont les pages internet du web. C'est notamment grâce à ces études que des systèmes de personnalisation dans la recherche d'information ont vu le jour. En effet, les systèmes couplant du *Web Usage Mining* et du *Web Content Mining* ont pu proposer des solutions hybrides de recommandations personnalisées [Delort, 2005].

Adaptive Hypermedia

[Brusilovsky, 2001] présente un état de l'art sur l'adaptabilité des systèmes hypermédia. Il expose les applications les plus sujettes à une forte adaptabilité comme les systèmes informatifs en ligne (presse, musée), les systèmes éducatifs, ou les systèmes de recherche de contenu. Il pose ensuite les questions essentielles de l'adaptabilité : A quoi doit-on adapter ? Qu'est-ce qui doit être adapté ? Bien que ces questions semblent évidentes lorsque l'on cherche à adapter, les réponses le sont beaucoup moins. On remarquera ainsi que l'on peut adapter du contenu informatif en fonction de la connaissance préalable de l'utilisateur, mais également en fonction du contexte de consultation de l'information par l'utilisateur. De la même façon, on peut personnaliser le contenu de l'enseignement au niveau de l'élève mais également au

contexte de l'apprentissage. Par exemple, l'apprentissage par cœur d'un poème ou d'une règle mathématique pourra être proposé à un élève lorsqu'il n'est pas à l'école, et un exercice de dictée en langue maternelle lorsqu'un professeur ou un ordinateur est présent pour le corriger.

3. La machine

La description du logiciel qui permettrait à un utilisateur donné de réaliser une tâche précise est une étape cruciale pour tout éditeur de logiciel. Ne pouvant répondre à toutes les catégories d'utilisateurs, ni à toutes les tâches courantes que ceux-ci souhaiteraient accomplir, les concepteurs ont fait appel à des sociétés de normalisation qui définissent les standards communs à tout logiciel. Ces standards sont amenés à évoluer en fonction des progrès matériels et des innovations techniques. Ainsi, les premières pierres posées par ces organismes de standardisation n'exigent aucune implémentation particulière, et définissent simplement un vocabulaire commun de référence.

La norme ISO 9241 définit l'*utilisabilité* de la façon suivante: "*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.*" (Source : ISO 9241 - Part 11). Cette notion comprend donc l'*efficacité* et l'*efficience* d'une interface : « efficacité » pour atteindre les objectifs fixés ou obtenir les effets attendus, « efficience » pour les atteindre ou les obtenir au moindre coût (temps passé, difficultés, aléas). Ces deux propriétés ont été approfondies par les ergonomes et les concepteurs d'interface, et l'on préfère maintenant définir la notion d'utilisabilité avec [Benard, 2001] :

- La facilité de prise en main. Un utilisateur doit être capable d'apprendre le fonctionnement de l'interface rapidement et de façon instinctive
- La facilité de perception du but du système
- La facilité de perception de l'état du système
- La facilité de repérage et de compréhension des commandes du système
- Une bonne prévisibilité des commandes du système
- La notion de prévention et gestion des erreurs
- La fiabilité du système

Les critères cités servent donc à concevoir et analyser une interface graphique d'ordinateur, et ne prennent pas en compte les différentes utilisations, ou les différents utilisateurs.

L'évaluation automatique de tous ces critères n'est évidemment pas une chose facile à réaliser. Néanmoins, beaucoup de travaux de recherche tentent d'y parvenir et on peut par exemple se référer aux travaux de [Farenc, 1997].

Les données de l'interaction HM, et surtout leur interprétation par les techniques présentées au chapitre suivant, seront très utiles pour la détection des mésinterprétations, ou les dysfonctionnements d'une interface. On pourra en particulier rechercher les méthodes de détection de séquences pour repérer automatiquement les difficultés des utilisateurs. Ces données pourront également fournir le scénario qui a conduit l'utilisateur à des difficultés.

4. Bilan sur la modélisation

Les trois points fondamentaux de l'analyse du dialogue HM apparaissent comme étant intrinsèquement liés. En effet, de part sa nature même, un dialogue est tributaires des réactions des autres entités dans la mesure où l'interaction elle-même peut être une source de modélisation et d'adaptation.

L'utilisateur est ainsi souvent modélisé à partir de connaissances théoriques et les outils décrits précédemment sont les plus courants lorsque l'on cherche à personnaliser le dialogue. Ces outils doivent leur performance à une certaine polyvalence dans leurs applications. Ils peuvent synthétiser un problème particulier que rencontre un utilisateur, aussi bien qu'un problème rencontré par une catégorie d'utilisateurs dont les caractéristiques sont données par le modèle.

L'interaction est elle-même source de personnalisation, principalement à cause de l'utilisation de connaissances de plus en plus complexes et variées. Sans entrer dans les considérations de modélisation de la connaissance, nous avons posé les questions relatives aux buts d'une éventuelle adaptation.

La machine, qui créée l'interface en suivant des implémentations humaines, réagit aux actions de l'utilisateur. Ces réactions sont en principe prévues par les concepteurs d'interfaces, et des outils d'aide à l'analyse de critères d'efficacité ont été élaborés spécialement pour l'utilisation de ces interfaces. Les perspectives de ces analyses sont multiples car elles font appel à un ensemble de critères permettant de décomposer la tâche d'un utilisateur. Ainsi, les concepteurs d'interfaces sont au plus près de l'activité de l'homme lors de la réalisation de tâches au quotidien.

Les adaptations de l'interaction homme-machine peuvent se faire à plusieurs niveaux. D'une part, il faut que l'homme utilise correctement ce pour quoi la machine a été conçue. Pour garantir cela, on peut détecter ses difficultés d'utilisation afin de

l'aider à mieux comprendre l'interface qui lui est proposée. D'autre part, nous pensons qu'il est possible d'aller plus loin dans l'interprétation de l'utilisation d'une machine pour effectuer des tâches. Ainsi, une adaptation réellement performante permettrait à l'homme de réaliser ce qu'il souhaite par l'intermédiaire de la machine (acquisition de connaissance ou apprentissage d'une notion par exemple).

Aussi, tous ces modèles servent différents buts et tous fonctionnent sur des données proches de l'interaction HM. Nous nous posons à présent la question des données nécessaires et suffisantes pour la modélisation du processus d'adaptation ou de personnalisation. Nous avons vu que connaître l'utilisateur peut être utile pour concevoir une interface appropriée. De même, connaître la tâche et les connaissances requises et disponibles pour l'effectuer constituent les fondements de toute modélisation de l'interaction. Nous allons maintenant présenter les méthodes que nous avons rencontrées dans la littérature, qui manipulent des données provenant directement de l'interaction homme-machine en précisant les besoins que les données doivent satisfaire pour aboutir un processus d'analyse automatisé.

Chapitre 2. Récolter les données

Les données générées par l'interaction homme-machine peuvent être nombreuses, de différentes natures et provenir de différentes sources. Ainsi, tout enregistrement de l'interaction elle-même, mais également, les annotations, les remarques expertes, et les connaissances abstraites sur le domaine sans oublier les connaissances sur les utilisateurs peuvent se révéler potentiellement utiles. Nous cherchons ici une méthode pour générer automatiquement des traces en respectant un formalisme pour représenter l'interaction. Nous ne considérons ici que les données issues directement de l'interaction homme-machine. Celles obtenues via des questionnaires ou des vidéos seront un plus pour une éventuelle analyse plus complète. Autrement dit, nous cherchons quelles sont les données que l'on peut récolter de façon automatique pour fournir une base de connaissances dont on pourra extraire de l'information pertinente. Nous nous plaçons donc au cœur de l'interaction pour tracer celle-ci et proposer un formalisme qui puisse servir l'analyse à la fois de l'utilisateur, de l'interaction et de la machine comme évoqué précédemment.

[Hilbert & Redmiles, 2000] ont réalisé un état de l'art des techniques relatives à l'extraction d'informations à partir de données provenant de l'interaction HM. Ils ont ainsi recensé un ensemble de méthodes couvrant la plupart des analyses modernes effectuées dans le domaine de la modélisation de l'interaction.

Nous reprenons cette typologie des méthodes en précisant les propriétés nécessaires que ces données doivent satisfaire.

1. Synchronisation et recherche

Lorsque l'on cherche à établir les propriétés d'un comportement d'un utilisateur, on est souvent amené à inférer les actions atomiques (soit les plus petites actions pouvant être décrites par des mots) en actions de plus haut niveau permettant une description globale de la tâche. Lorsqu'une description contextuelle précise de ces actions manque, une autre source de données est souvent recherchée. Ainsi, des modèles

ont été créés pour la synchronisation de données provenant directement de l'interface avec d'autres sources de données comme par exemple, les vidéos ou les observations notées à la main pendant des expérimentations. Les données provenant directement de l'interface sont alors utilisées comme indexation et servent pour retrouver les données provenant d'une autre source. Ainsi, un analyste comportemental peut chercher à retrouver les séquences d'une vidéo ou des observations d'utilisation faites à la main. Il peut par exemple être intéressant de retrouver les demandes d'aide ou les utilisations de certaines fonctions de l'interface par l'utilisateur. Au lieu de parcourir toute la vidéo ou toutes les notes d'observation, l'analyste peut formuler une requête sur les données provenant de l'interface et retrouver les données qui l'intéressent à partir de la date observée pour ces évènements. Ces modèles sont les plus simples présentés ici, mais sont souvent très efficaces.

Propriétés nécessaires des traces

Les données provenant de l'interaction HM représentant les actions des utilisateurs doivent donc être indexées selon le temps, et doivent comporter une sémantique du contexte de l'interaction. La mise en contexte par une action d'un utilisateur peut également être indexée selon le temps ce qui facilitera la complexité de calcul des algorithmes d'indexation.

Avantages

La force de ces systèmes est qu'ils permettent l'utilisation de différentes sources de données ayant chacune leurs avantages et leurs inconvénients. Ainsi, il est facile de retrouver les données provenant d'une source à partir des données provenant d'autres sources. Toutefois, le traitement automatique de différentes sources de données en même temps n'est pas chose facile, et il est souvent très coûteux de chercher à retranscrire ces informations automatiquement.

Inconvénients

La récolte de données provenant de différentes sources implique souvent un ensemble de données de différentes natures : vidéo, hypertexte, observations prises sur le vif d'une expérience à l'aide d'un dictaphone par exemple. Tous ces types de données nécessitent un traitement particulier, et les systèmes pour les traiter sont souvent très coûteux en fabrication et en utilisation. Le ratio de temps mis pour l'analyse des sessions observées, sur la durée des sessions effectuées, est généralement de l'ordre de 10:1 [Sanderson & Fisher, 1994].

2. Transformation du flot d'évènements

Ces techniques de transformation qui regroupent la sélection, l'abstraction, et l'encodage du flot de données facilitent la reconnaissance, la comparaison et la caractérisation de séquences ou la préparation des données pour l'implémentation de processus d'analyse automatique.

La sélection s'effectue en soustrayant un certain nombre d'évènements parmi la séquence complète, permettant l'émergence de données intéressantes non bruitées. Cette sélection implique de spécifier un certain nombre de contraintes sur les attributs des évènements que l'on veut observer ou ceux que l'on ne veut pas observer. Par exemple, on peut être intéressé par regarder uniquement les mouvements de souris, et dans ce cas, on supprimera tous les évènements ayant trait à d'autres types de périphériques comme la frappe au clavier.

L'abstraction constitue la synthétisation d'évènements de l'interface de bas niveau en évènements interprétables de haut niveau. On pourra par exemple remplacer les évènements du type « Souris Bouton 1 Enfoncé » immédiatement suivi d'un « Souris Bouton 1 Relâché » par un évènement du type « Souris Bouton 1 Cliqué ». Si ces deux premiers évènements ne se suivent pas suffisamment rapidement, il peut être intéressant de les conserver tels quels, car ils peuvent révéler un autre attribut du type « Souris Bouton 1 Cliqué hésitant ». On pourra également remplacer des évènements si l'on sait qu'ils interviennent tous pour la même tâche. Par exemple, lors de l'utilisation d'un navigateur Internet, il y a souvent toutes sortes d'évènements machine générés lors de l'affichage d'une nouvelle page. Ces évènements sont généralement dus à la création de l'objet DOM (*Document Object Model*, utilisé pour l'interprétation du fichier HTML représentant la page Internet) par le composant graphique du navigateur Internet. Ces évènements sont souvent très nombreux, et n'apportent aucune information sur l'action de l'utilisateur qui voit la page s'afficher. On pourra alors remplacer ces évènements par un évènement simple du type « Nouvelle page Internet affichée ».

Paradigme

L'encodage des évènements constitue un ensemble d'opérations du type sélection et abstraction. Par exemple, pour imprimer, il y a généralement plusieurs moyens d'actions mis à la disposition de l'utilisateur dans les logiciels de traitement de texte. On peut cliquer sur le menu fichier puis descendre jusqu'à l'item imprimer, cliquer sur le bouton de raccourci dans la barre des raccourcis, ou utiliser le raccourci « [Ctrl]-P » si on connaît cette fonction du logiciel. Certaines de ces actions vont engendrer d'autres actions utilisateurs si par exemple un menu de préférences d'impression apparaît. Dans

d'autres cas, l'impression sera lancée directement en appliquant les préférences enregistrées pendant les impressions précédentes. Cependant, le but de l'utilisateur était le même à chaque fois : l'impression. Il peut être utile de coder le flux d'évènements si seule la sémantique de l'action de l'utilisateur importe pour l'analyse. On pourra alors pour l'exemple substituer l'ensemble des évènements précédemment décrits pour un évènement du type « impression ».

Propriétés nécessaires des traces

La méthode de récolte de traces issues de l'interaction HM doit donc permettre à l'analyse de sélectionner, abstraire et encoder certaines données. Le format des données doit donc supporter ces opérations facilement par une sélection précise sur une propriété d'un évènement de l'interaction. De même, la méthode doit être paramétrable pour pouvoir effectuer ses opérations lors de la récolte de données.

Avantages

Le principal avantage de ce type d'approche vient du fait que l'on connaît précisément les évènements qui servent de support à la sélection, l'abstraction et l'encodage. Cette connaissance est en général très utile dans la préparation de la plupart des analyses. Une fois ces évènements de l'interface distingués du flot total d'évènements, on peut regarder le contexte de leurs apparitions et trouver des informations très intéressantes sur le comportement des utilisateurs.

Inconvénients

Le risque rencontré en influant directement sur la capture des données par les techniques de sélection, abstraction et encodage est de simplifier l'information de l'interaction HM, et ainsi de perdre des données qui peuvent être importantes pour une analyse plus précise de cette interaction. C'est particulièrement remarquable lorsqu'il s'agit d'évènements que l'utilisateur décide de sélectionner (dans les systèmes d'auto-évaluation par exemple).

3. Comparaison de séquences

Le but de ces méthodes (voir Figure 2 extrait de [Hilbert & Redmiles, 2000]) est de comparer une séquence d'actions produite par un utilisateur à une séquence d'actions modèles servant de référence. On distingue les méthodes qui cherchent à comparer les évènements d'un utilisateur à une séquence abstraite d'évènements (cas de droite sur la Figure 2) à celles qui cherchent à les comparer à une séquence concrète d'évènements

(cas de gauche sur la figure) obtenue par exemple à l'aide d'experts. Certaines comparaisons servent à établir différents critères de correspondance entre les séquences et d'autres tentent de trouver le meilleur alignement possible de séquences d'évènements. D'autres encore sont spécialisées dans la détection d'« incidents critiques » toujours par comparaison de séquences d'évènements modèles à une séquence d'évènements d'un utilisateur.

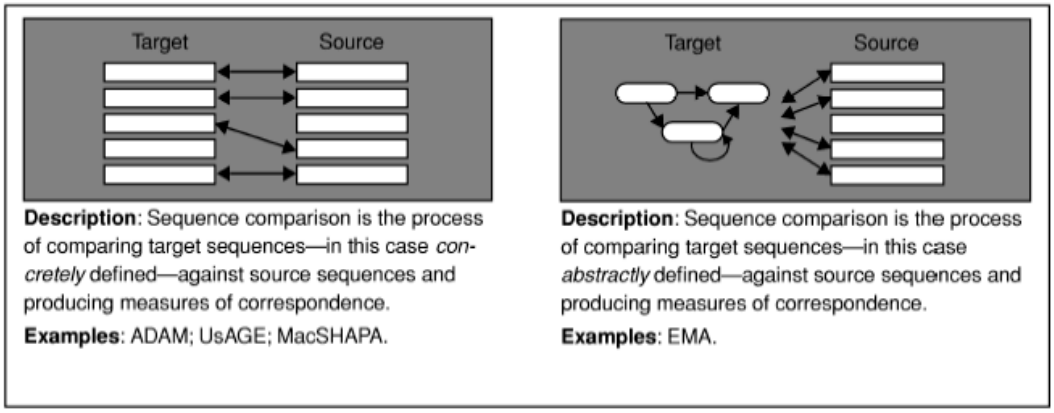

Figure 2. Comparaison de séquences [Hilbert & Redmiles, 2000]

Propriétés nécessaires des traces

De la même façon que pour les méthodes de synchronisation et de recherche évoquées précédemment, des indexations selon différents critères doivent pouvoir être facilement effectuées. Ces critères comprennent notamment le temps ainsi que la nature des évènements et leur contexte.

Une méthode de visualisation des données doit permettre de comparer les parcours de plusieurs utilisateurs avec des parcours de référence ou des parcours d'autres utilisateurs.

Avantages

L'avantage de ces méthodes est qu'elles permettent de comparer les évènements de l'interface effectivement produits par un utilisateur à un modèle de tâches dont on attend qu'il soit accompli par cet utilisateur.

On peut ainsi rendre compte de l'usage et de l'utilisabilité d'une interface en comparant l'activité de différents profils.

Inconvénients

Le principal inconvénient de ces méthodes qui comparent une séquence d'évènements de l'interface à une séquence d'évènements modèles est qu'elles supposent deux choses : d'une part le modèle de tâches ainsi que les évènements de l'utilisateur sont facilement comparables, et d'autre part l'ensemble des séquences d'évènements produits par différents utilisateurs possèdent des ressemblances raisonnables.

De plus, la sortie de ces modèles est généralement exprimée sous la forme d'un taux de correspondance entre la séquence produite par des utilisateurs et la séquence modèle introduite par un expert. Ce taux n'est ensuite interprétable que par un expert tant il dépend du type d'interface testé et du taux de divergence de l'ensemble des utilisateurs de l'interface.

Cet inconvénient est d'autant plus important que le modèle de tâches incorporé dans le système doit contenir tous les chemins possibles des utilisateurs sur l'interface ce qui est pratiquement infaisable pour une interface non triviale. Il faudra donc parvenir à éliminer tous les bruits survenant dans les traces utilisateurs comme les mouvements de souris qui ne peuvent correspondre parfaitement à celui produit par le modèle. Il sera alors impossible de faire une comparaison de la totalité de la séquence et certains systèmes s'intéresseront uniquement à des sous-séquences clairement identifiables pour calculer l'utilisabilité de l'interface.

4. Comptage et statistiques

Ces méthodes doivent par définition s'appliquer à des données volumineuses pour être utiles. Ainsi, les techniques d'apprentissage artificiel permettent de reconnaître automatiquement le type d'utilisateur que l'on est en train d'observer et peuvent prédire certaines actions en se référant à une base de données d'apprentissage. Les algorithmes utilisés par ces techniques sont très variés et le Chapitre 10 détaille un certain nombre d'entre eux. Ces techniques sont souvent employées dans l'étude de l'interaction HM (voir chapitre 2).

Propriétés nécessaires des traces

Les données récoltées retraçant l'activité de l'interaction HM doivent être suffisamment précises pour être abondantes, mais également homogènes pour pouvoir être statistiquement interprétables. La structure des données peut également être complexe, ce qui permet un traitement sémantique des données par les algorithmes.

Avantages

La nature séquentielle des données provenant de l'interaction HM peut être décrite précisément avec un grand nombre de statistiques sur les éléments composant ces séquences (fréquence d'apparition, position dans la séquence et dans le temps, durée de l'interaction, etc.). Un bilan très détaillé du type d'interaction HM peut être fourni. Il peut également être très utile d'inclure ce genre de comptage directement dans le logiciel à tester de façon à produire des bilans spécialisés pour chaque outil de l'interface.

Inconvénients

Ces statistiques sont souvent très utiles pour une analyse approfondie du comportement, cependant, il faut toutes les discerner pour parvenir à une description complète. Ces statistiques peuvent varier en fonction de l'expérience et de l'interface. Ainsi, nous pensons qu'il peut être utile d'offrir un modèle général de construction de statistiques autour de traces d'utilisation. Néanmoins, le nombre de statistiques descriptives d'activités sur une interface est considérable.

5. Détection et caractérisation de séquences

Les techniques de détection se focalisent sur des occurrences concrètes ou abstraites de séquences d'évènements de l'interface (voir Figure 3 extrait de [Hilbert & Redmiles, 2000]). Il s'agit d'une analyse exploratoire de données séquentielles. Certaines méthodes recherchent les enchaînements consécutifs d'évènements (comme *Maximal Repeating Pattern* [Siochi & Hix, 1991] ou les techniques de chaînes de Markov [Guzdial, 1993]), alors que d'autres recherchent les enchaînements non consécutifs d'évènements (comme Fisher's cycle [Fisher, 1991] ou Lag Sequential Analysis [Sanderson & Fisher, 1994]).

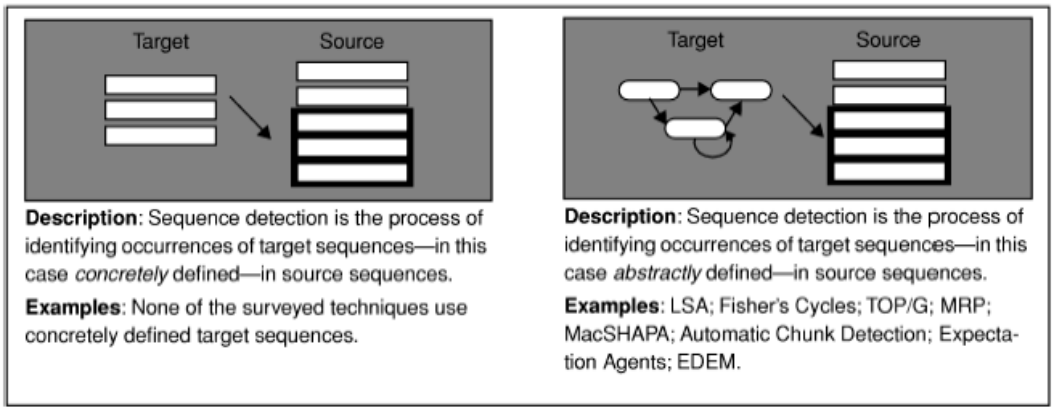

Figure 3. Détection de séquence concrète et abstraite [Hilbert & Redmiles, 2000]

Les méthodes de caractérisation de séquences mettent à jour des propriétés globales ou plus ciblées sur des sous-séquences, caractérisant des actions d'utilisateurs. Certaines méthodes calculent les probabilités de transitions entre les évènements de l'interface (voir Figure 4 extrait de [Guzdial, 1993]), et d'autres tentent de construire des grammaires structurées autour de ces séquences.

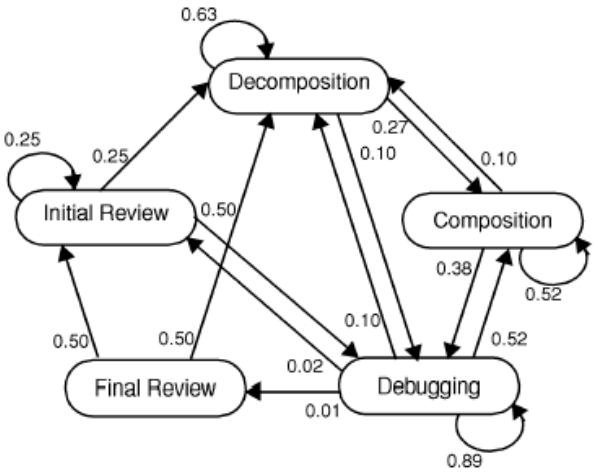

Figure 4. Schéma de caractérisation de comportements utilisateurs où les nœuds sont les étapes et les arcs représentent les probabilités de transitions entre ces étapes. [Guzdial, 1993]

Propriétés nécessaires des traces

Les techniques de détection et de caractérisation de séquences seront les plus utiles pour les analyses de comportement utilisateur. De plus, ce traitement pourra être effectué à différents niveaux de détails pour apporter différentes interprétations sémantiques, et le contexte lié à telle ou telle séquence d'actions de l'utilisateur sera important. La méthode de récolte des traces doit donc permettre une étude à différents niveaux sémantiques.

Avantages

Le principal avantage de ces techniques de détection de séquences est leur intérêt pour une séquence d'évènements et non plus pour un évènement en particulier dans une séquence. Il est ainsi possible de recoder la séquence d'évènements originaux en séquence d'évènements abstraits, où les séquences découvertes par les différentes techniques décrites précédemment sont remplacées par des évènements abstraits.

L'avantage des techniques de caractérisation de séquences est qu'elles peuvent aider les experts à déterminer des structures implicites d'une séquence et qualifier ces structures par des descripteurs abstraits.

Inconvénients

L'analyse exploratoire de données en vue de produire des motifs séquentiels a tendance à produire un très grand nombre de candidats difficiles à analyser et n'aboutissant pas forcément à un indicateur sur l'usage ou l'utilisabilité d'une interface.

La technique issue des méthodes markoviennes paraît être efficace pour la caractérisation de séquences bien qu'il faille inclure un nombre restreint d'états sous peine de modéliser le bruit présent dans les traces (voir chapitre 1). Les grammaires seront utiles pour traduire les évènements de l'interface en évènements abstraits. Pourtant, si elles permettent de caractériser les séquences, elles sont incapables de dire quelles sont les séquences les plus générales. Ainsi, les concepteurs d'interfaces qui voudront avoir un retour sur l'utilisation de leurs interfaces seront plus intéressés par les fréquences d'apparition de certains motifs d'évènements, reflétant des comportements utilisateurs, que par des grammaires résumant toutes ces pratiques.

6. Bilan sur les données

La classification des méthodes proposée ici couvre l'éventail des recherches qui sont menées sur l'extraction d'information à partir des évènements de l'interface. Bien que les recherches aient continué leurs évolutions dans ce domaine, cette typologie des

méthodes est toujours d'actualité [Kort & de Poot, 2005]. Des applications combinant ces différentes méthodes pour l'évaluation automatique de l'utilisabilité d'une interface sont décrites dans [Ivory & Hearst, 2001].

Nous avons donc présenté des exemples de techniques de synchronisation et de recherche, de transformation du flot d'évènements, de comparaison de séquences, de comptage et de statistique et de détection et caractérisation de séquences. De plus, un formalisme général permettant plusieurs types de modélisations comme ceux évoqués dans le Chapitre 1 doit comprendre :

- une indexation selon le temps ainsi que selon différents critères liés à la nature des évènements et à leur contexte,
- différents niveaux de description sémantique du contexte,
- une adaptabilité permettant d'effectuer directement des opérations simples comme la sélection, l'abstraction et l'encodage des données,
- une visualisation brute des données permettant la comparaison de l'utilisation de différentes interfaces et la comparaison de différents parcours d'utilisateurs,
- une description précise, pour être abondante, et homogène, pour être interprétable, dans une structure éventuellement complexe permettant des analyses sémantiques.

On peut également proposer d'autres classifications de ces méthodes : celles qui fournissent une indexation pour la recherche dans d'autres sources de données et celles qui permettent une étude directe, tant les techniques pour les analyser sont sophistiquées et la qualité et la quantité des données sont suffisantes. Ou bien : celles qui font appel à un traitement séquentiel des données pour y faire apparaître une propriété et celles qui cherchent à détecter des propriétés dans un ensemble de statistiques de traces d'utilisation.

Les chercheurs en fouille de données ont souvent besoin de savoir ce qu'ils sont en train de mettre à jour pour automatiser le processus de fouille. Pour ce faire, une première lecture *à la main* des données est souvent utile, mais l'interprétation de celle-ci est souvent difficile quand les données sont nombreuses ou complexes. Dans le chapitre suivant, nous présentons des visualisations de données issues de l'interaction que nous avons rencontrées dans la littérature.

Chapitre 3. Visualiser les données…

Dans le chapitre précédent, nous avons passé en revue les propriétés nécessaires des traces pour rendre compte de façon très précise de l'interaction homme-machine. Ces données sont complexes à analyser et il est donc naturel de chercher à les représenter. Les représentations proposées ne permettent pas de rendre compte de l'intégralité du dialogue homme-machine, soit parce que les données représentées ne contiennent pas toutes les informations, soit parce qu'elles cherchent justement à ne représenter qu'une vue explicite et interprétable des données.

Dans ce chapitre, nous présentons d'abord les propriétés introduites par les psychologues cogniticiens en visualisation de données, qui leur permettent de concevoir de nouveaux modèles de représentation des processus cognitifs impliqués dans la réalisation d'une tâche.

Nous présentons ensuite des exemples de visualisation issus d'applications du domaine du *Web Usage Mining*. Ce domaine de recherche a vu son intérêt croître rapidement avec l'essor de l'utilisation de l'Internet. Ces modèles présentent généralement des manques dans la représentativité de l'interaction, tant les données qui y sont représentées ne sont issues que d'une seule entité : la machine. Néanmoins, les modèles présentés proposent des analyses du comportement des utilisateurs ainsi que des propriétés liées à la présentation de l'information dans une page Internet.

Un modèle de représentation de l'interaction, théorique car non encore automatisé sur des données, est ensuite présenté. Ce modèle présente l'avantage de remettre en contexte l'intégralité d'une interaction pour mettre en évidence les propriétés liées à chaque utilisateur.

Nous présentons ensuite un modèle semi-automatisé, où des traces de l'interaction ainsi qu'une description formelle de l'activité permettent de représenter graphiquement l'ensemble des processus cognitifs impliqués dans la réalisation d'une tâche. Cette

représentation est donc automatisée et suit le formalisme d'une famille de modélisation cognitive théorique, le CPM-GOMS.

1. Les nécessités pour une analyse cognitive de parcours

Les problématiques des psychologues cognitivistes qui étudient les usages des outils numériques sont nombreuses. Pour commencer, il leur est nécessaire de distinguer l'origine des phénomènes observés, à savoir ce qui appartient au sujet et ce qui appartient au dispositif testé. Par exemple, les analyses cognitives de parcours peuvent porter sur de nombreux aspects de l'utilisation d'une interface. Cette première obligation nécessite de prendre en compte tous les paramètres observables lors d'une expérience, ce qui représente une grande quantité de données Les psychologues utilisent toutes les sources d'informations possibles (plans de caméra, audio, logiciel de traces, questionnaires, …). Cette profusion de données est en constante augmentation en raison du développement des interfaces numériques toujours plus complexes qui permettent des applications de plus en plus riches et engendrent des comportements toujours plus hétérogènes de la part des sujets dans leur interaction avec le dispositif. Lors de l'étude des données, les psychologues cognitivistes subissent ces mêmes phénomènes, c'est-à-dire que les outils de recueil de données qu'ils utilisent sont de plus en plus performants et diversifiés et les applications dédiées à l'analyse sont de plus en plus complexes. Par exemple, l'analyse des données issues de capteurs oculaires doit être faite avec précaution tant les erreurs de manipulations peuvent fausser les résultats.

Les traces permettent les analyses les très fines, mais elles sont difficiles à exploiter. Une des contraintes des psychologues est de manipuler ces grandes quantités d'informations pour repérer des phénomènes isolés ou répétés et de s'en faire une représentation. Ils ont besoin d'outils d'analyse de parcours souples et précis qui offrent une représentation visuelle. Les méthodes de traitement automatique produisent généralement des résumés numériques et l'interprétation que l'on doit en déduire ne reflète généralement qu'une partie de l'action des utilisateurs. Bien que soulevant les points cruciaux d'une analyse sur un comportement, les psychologues cognitivistes ont souvent besoin d'aller plus loin, en validant leurs hypothèses par une remise en contexte global de toute l'interaction. De plus, l'analyse d'actions élémentaires, comme une prise de décision simple (un clic de souris), nécessite une remise en contexte local par la représentation d'une sous-tâche particulière.

2. Web Usage Mining

Le *Web Usage Mining*, par opposition au *Web Content Mining*, est le terme choisi par [Cooley & al., 1997] pour qualifier l'étude de l'usage de l'Internet. Cela consiste à analyser les fichiers de traces des serveurs web pour construire des mesures d'audiences, des systèmes de recommandations voire même la réorganisation d'un site en fonction du comportement des internautes. Il existe plusieurs outils de visualisation de données d'usage de l'Internet dont nous présentons ici quelques exemples.

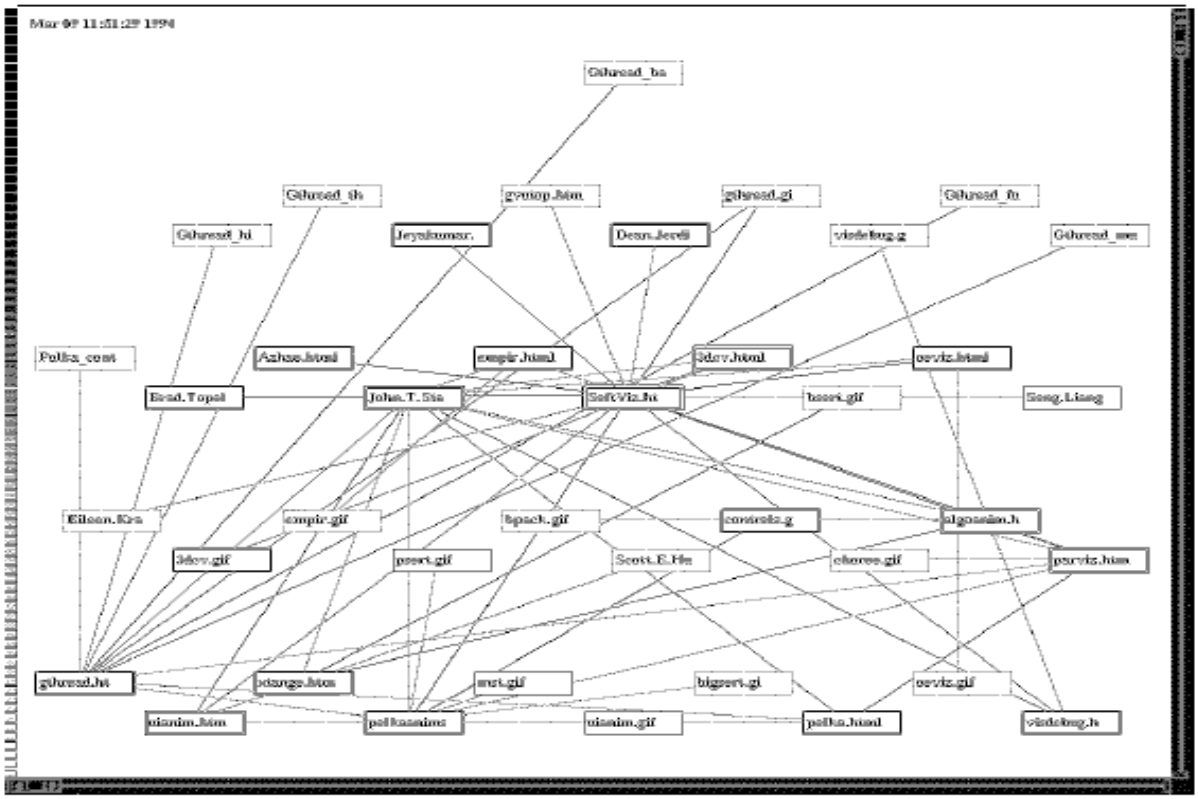

Figure 5. Extrait d'une visualisation de Webviz [Piktow & al., 1994]

Webviz permet de mettre en évidence des comportements d'internautes. Comme le montre la Figure 5 extrait de [Piktow & al., 1994], un site est représenté par un graphe où l'ensemble des nœuds forment l'ensemble des pages du site. Les liens entre les nœuds sont les liens entre les pages du site. L'outil propose alors de représenter la totalité des parcours possibles sur un site ou ceux effectués par certains utilisateurs seulement. La représentation utilisée nécessite la connaissance a priori de l'information recherchée et, par l'application d'un filtre sur la visualisation, la validation d'une

hypothèse peut être réalisée. On peut par exemple choisir de ne représenter que les parcours des personnes qui ont visité une page particulière. Par contre, les aspects temporels de la navigation et du comportement des utilisateurs sont difficilement visualisables dans leur ensemble avec cet outil. Ce type d'outil a été amélioré, et propose maintenant des visualisations dans des espaces tridimensionnels.

Wum [Spiliopoulou & Faulstish, 1998] utilise une représentation arborescente, qualifiée d'agrégée car manipulant des statistiques de navigation, pour mettre en évidence des comportements d'utilisateurs. Par contre, *Wum* ne donne que très peu de renseignements sur le contenu de l'interface utilisée.

Un logiciel commercial comme *Clicktracks* (voir Figure 6) utilise des données provenant des serveurs Internet et représente la probabilité de clic sur un lien d'une page. Dans ce cas, il y a une remise en contexte de l'utilisation de l'interface, mais sur un instant très précis de la navigation. On peut ainsi comparer l'effet que peut produire le changement d'apparence d'une page sur la navigation. Un outil permet également de représenter les probabilités qu'ont certains internautes particuliers de changer de parcours si l'on change l'interface. Ces internautes particuliers peuvent, par exemple, être tous ceux qui sont habitués à visiter certaines pages du site. Les statistiques fournies sont nombreuses et peuvent notamment se révéler très utiles pour l'optimisation de l'impact publicitaire de certains sites. Cependant, la définition des catégories d'utilisateurs doit être formulée par l'analyste et aucune étude sur le parcours d'un internaute n'est proposée.

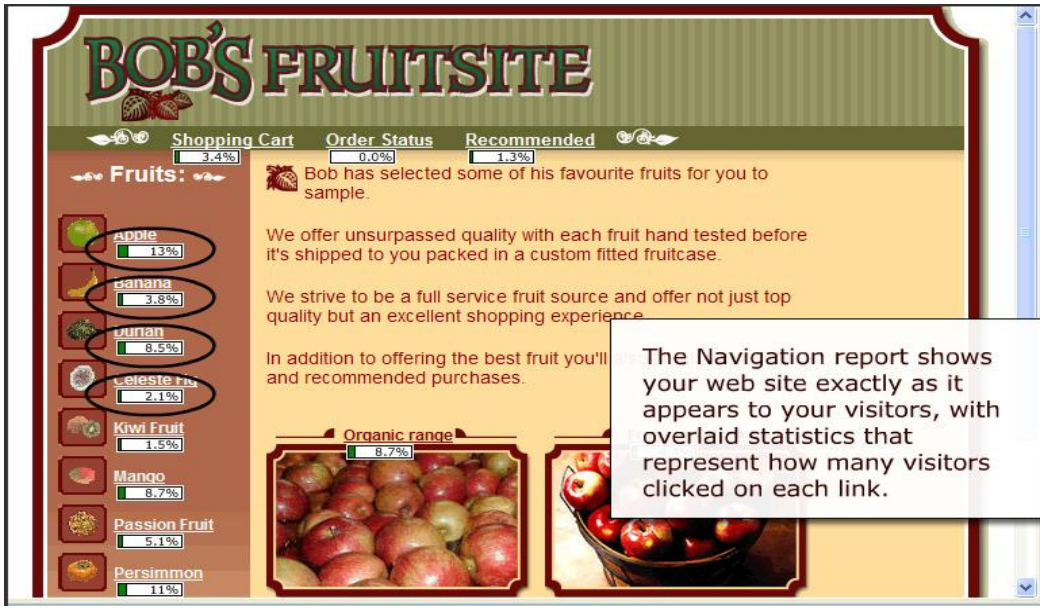

Figure 6. Démonstration du logiciel Clictraks

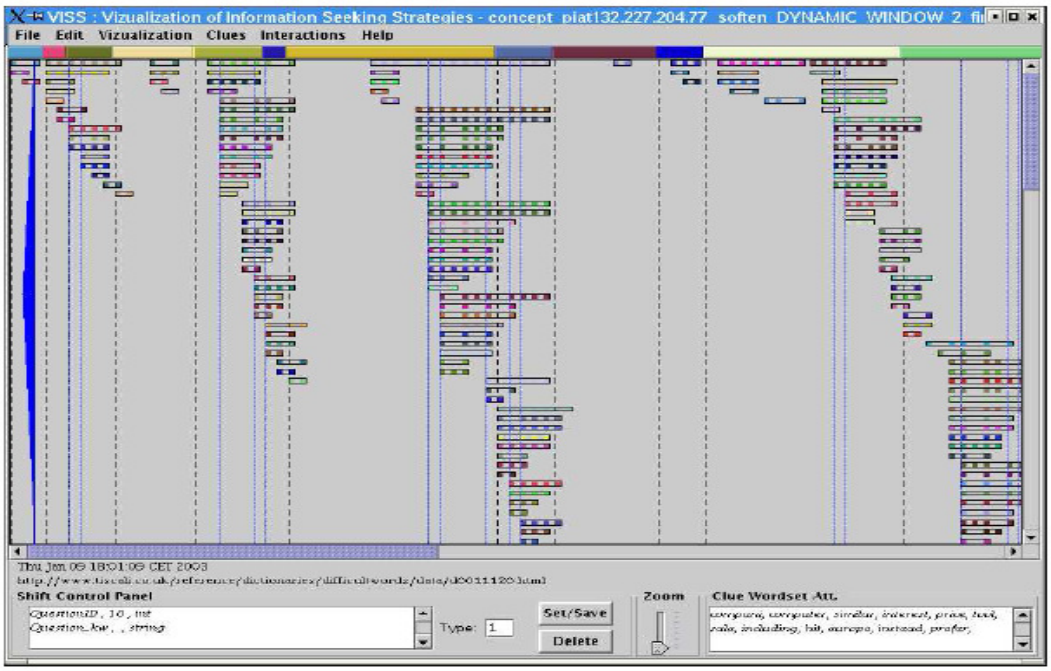

Figure 7. VISS un outil de visualisation des stratégies de recherche sur le web [Delort & al., 2003]

Une autre approche de modélisation utilisateur pour la recherche d'information sur le web utilisant un outil de visualisation est proposée par [Delort & al., 2003] et

présentée dans la Figure 7. Dans ce cas, l'étude est centrée sur le contenu des pages Internet visitées et des indices de changements de stratégies de navigation par l'utilisateur sont proposés. Ces changements de stratégie interviennent lors de la recherche de documents sur Internet. Cependant la visualisation ne renseigne pas sur le véritable comportement de l'utilisateur, car elle n'est qu'une représentation du parcours des contenus visités.

3. Les graphes contextuels

Une méthode de visualisation permet de rendre compte du contexte de la tâche réalisée lors du dialogue HM. Il s'agit des *graphes contextuels* présentés dans [Brézillon & Tijus, 2005].

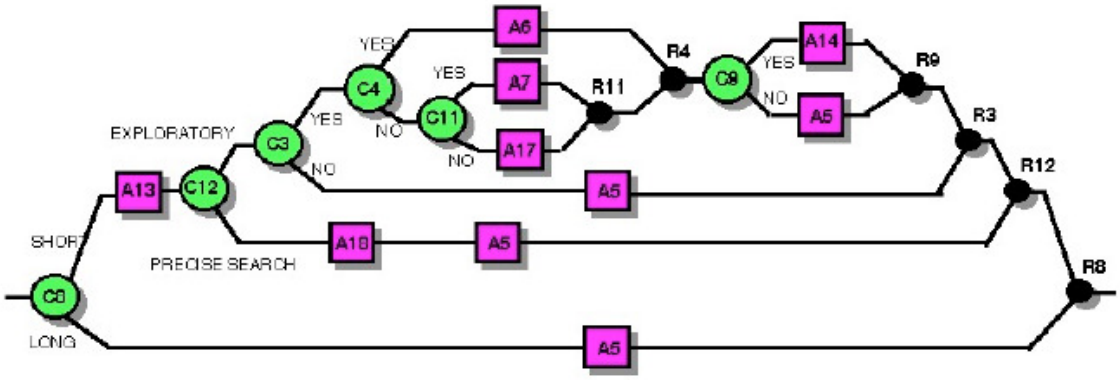

Figure 8. Exploitation des informations sur une page Web. Les carrés correspondent aux actions, les cercles aux éléments contextuels : les grands cercles aux nœuds contextuels et les petits cercles aux nœuds de recombinaison [Brézillon & Tijus, 2005]

Elément contextuel	
CE3	La page est-elle intéressante
CE4	Y a-t-il des figures à récupérer ?
CE8	Quel est le temps de chargement de la page ?
CE9	Tout le site est-il exploré ?
CE11	Est-ce que toute la page est intéressante ?
CE12	Type de recherche ?

Action	Définition
A5	Fermer la fenêtre
A6	Sauvegarder la page html
A7	Copier et coller le contenu de la page dans un éditeur
A13	Localiser les mots clés dans la page
A14	Aller à la page d'accueil du site
A17	Sélectionner la partie intéressante de la page
A18	Localiser l'item recherché (e.g. l'adresse)

Figure 9. Exploration de la page au bout du lien [Brézillon & Tijus, 2005]

La Figure 8 présente le graphe contextuel d'un utilisateur cherchant une information sur internet. Avec cette représentation, il n'est pas nécessaire de connaître le but précis de l'utilisateur, autrement dit, le but de sa recherche, pour faire apparaître des propriétés intéressantes sur son parcours sur une interface hypermédia. Ainsi, la représentation proposée fait apparaître au même niveau, les éléments de comportements et les éléments contextuels qui définissent leur validité (voir Figure 9). Cette méthode est applicable à beaucoup de domaines comme la psychologie, la sécurité informatique, la gestion d'incidents, le diagnostic médical, etc. mais présente l'important désavantage de n'être pas automatisable dans sa construction. En effet, seul un expert peut construire à la main ce type de graphe, tant la sélection parmi la multitude des paramètres de contexte qui peuvent être prise en compte pour décrire l'accomplissement d'une tâche n'est pas triviale.

Cet exemple montre à quel point il est difficile de réellement mettre en évidence automatiquement les processus cognitifs discriminants associés à la réalisation d'une tâche par un utilisateur. Dans la section suivante, un exemple d'automatisation d'un modèle d'analyse cognitive de tâches est détaillé.

4. Validation d'un modèle de description cognitive des tâches

[John & al., 2002] ont automatisé le processus de construction d'un modèle de la famille GOMS, appelé CPM-GOMS pour *Cognitive-Perceptual-Motor operation* in GOMS. Cette automatisation nécessite la création préalable de la séquence de sous-tâches à réaliser. Le graphe de la Figure 10 est ensuite créé automatiquement et permet de déterminer le temps d'exécution d'une tâche. Afin de valider le modèle, les auteurs comparent le temps calculé par le modèle à celui réellement effectué par des utilisateurs.

Les données issues de l'interaction avec l'interface sont récoltées et permettent d'affiner le modèle pour différentes catégories d'apprenants.

Cette technique de validation est souvent utilisée quand les modèles de comportement sont construits a priori car les données des interactions HM récoltées forment une description exacte des actions de l'utilisateur perçues par l'ordinateur. En plus de valider les hypothèses de fonctionnement des processus cognitifs, cette visualisation permet de mettre en parallèle la décomposition globale des processus cognitifs avec leurs agencements séquentiels hiérarchiques. Par exemple, les actions dont une activité cognitive de la vision est importante, comme le clic de souris, sont mises en valeur automatiquement. De même, il apparaît automatiquement que certaines actions (mouvements de souris) ne peuvent être accomplies tant que d'autres n'ont pas été terminées (frappe au clavier).

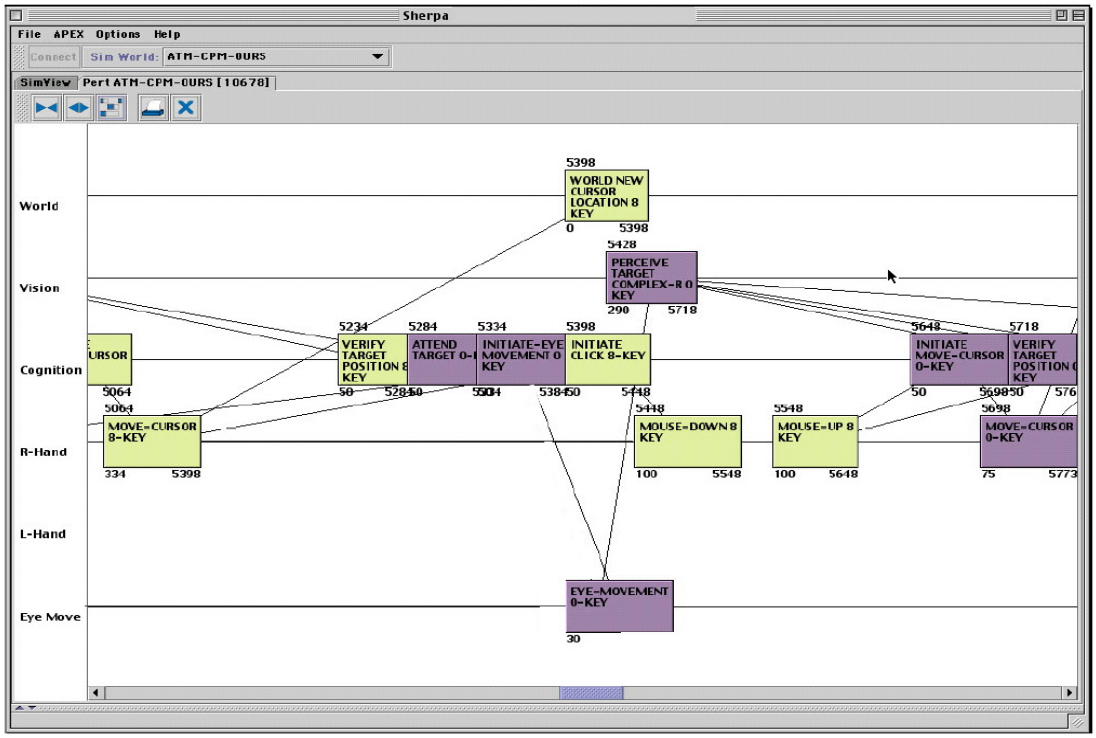

Figure 10. Modèle de déplacement attentif du curseur jusqu'à une cible et clic de la souris [John & al., 2002].

Ainsi, et comme le veut le modèle GOMS, il est possible de modéliser la performance d'une interface connaissant les caractéristiques cognitives de l'homme. L'extrait de la Figure 10 est issu d'une analyse portant sur la performance des automates

distributeurs d'argent. Les résultats sont présentés dans la Figure 11. Les temps calculés automatiquement par le modèle sont très proches de ceux réellement effectués par les utilisateurs (à 13% près selon les auteurs).

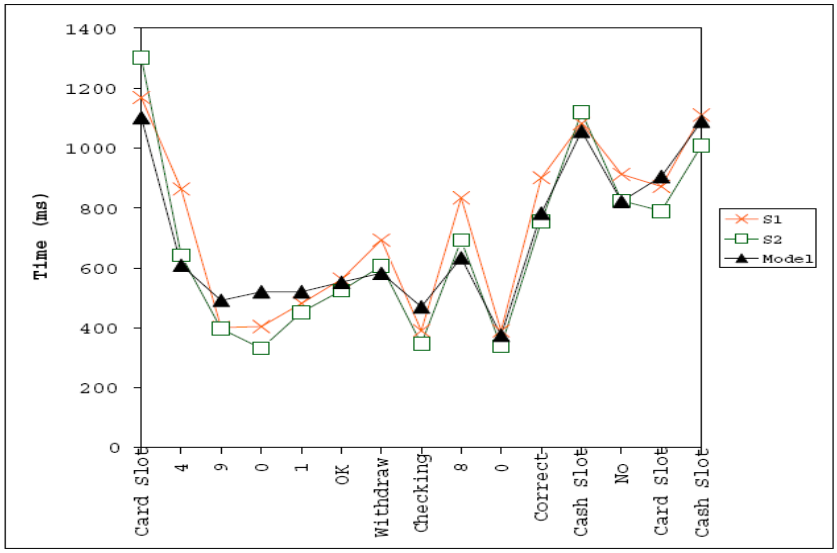

Figure 11. Prédiction du modèle GOMS et résultats de 2 utilisateurs [John & al., 2002]

Une autre expérience a été menée par la même équipe d'auteurs, et les résultats ont permis d'affiner les paramètres d'un autre modèle d'analyse cognitive de la famille GOMS, le KLM [Teo & John, 2006]. Ainsi, les constantes décrivant les processus cognitifs impliqués dans la réalisation d'une tâche ont été décomposé en plusieurs composantes permettant une utilisation plus approfondie du modèle.

5. Bilan sur la visualisation

Qu'elles soient théoriques, c'est-à-dire construites sur le savoir d'experts, automatiques, donc construites directement à partir de données, ou hybrides, les visualisations proposées dans ce chapitre sont des représentations interprétables directement par des analystes experts ou novices. Nous avons également présenté les nécessités d'une visualisation de données provenant de l'interaction HM.

Ces représentations de données sont de différentes natures, et proviennent de différentes sources. Elles sont généralement exploitées soit pour la compréhension du comportement de l'utilisateur, soit pour décrire les implications de fonctionnement de

différentes interfaces, soit pour mettre en évidence des besoins de changements dans l'interaction elle-même.

Les propriétés observables dans ces visualisations permettent ensuite aux chercheurs de concevoir des systèmes automatiques de détection de ces propriétés. Comme nous l'avons évoqué précédemment, les caractéristiques de comportement sont rarement découvertes avec sérendipité. La plupart des systèmes tente d'automatiser un processus observable par un expert, et les représentations de données permettent souvent la mise à jour de caractéristiques.

Chapitre 4. … et analyser les données

Ce chapitre présente des travaux de fouille de données pour l'implémentation de systèmes où la modélisation de l'utilisateur sert la personnalisation du dialogue HM. Les travaux de fouille de données (ou *data mining*) sur les données issues de l'interaction HM s'effectuent par un apprentissage par des systèmes automatiques qui reconnaissent automatiquement les caractéristiques du dialogue HM. Ces caractéristiques peuvent servir plusieurs buts différents. En effet, nous distinguons ceux qui améliorent l'interaction HM pour une navigation intelligente plus facile et plus rapide, de ceux qui modélisent le processus d'acquisition de la connaissance humain pour aider l'humain dans sa tâche. Ce dernier but est le plus proche de notre problématique générale et nécessite généralement un lourd processus préalable de modélisation de la connaissance. Néanmoins, et pour les deux approches, une analyse du comportement de navigation de l'utilisateur et donc une analyse des données provenant directement de l'interaction HM sont nécessaires.

Une classification des différents types d'apprentissage pour la personnalisation de l'interface HM est présentée dans la Figure 12 extraite de [Akoulchina, 1998].

	La même tâche	Une nouvelle tâche
Le même utilisateur	Modélisation dynamique et individuelle de l'utilisateur + Adaptation dynamique Apprentissage amnésique	Acquisition et transfert de connaissances à propos de l'utilisateur Apprentissage centré sur l'utilisateur
Un nouvel utilisateur	Acquisition et transfert de connaissances à propos de la tâche Apprentissage centré sur tâche	Adaptation et apprentissage impossibles

Figure 12. Types d'apprentissages pour la personnalisation d'interface HM [Akoulchina, 1998]

Cette classification fait apparaître très clairement le fait que la personnalisation automatique du dialogue HM ne peut se faire que sur des utilisateurs dont on a appris le comportement et/ou dont on connaît la tâche à réaliser.

Comme on l'a vu pour l'objet de la modélisation dans le Chapitre 1, ces notions sont intrinsèquement liées. En effet, ce qui caractérise un utilisateur est souvent sa façon de résoudre une tâche, et il peut aussi réaliser une tâche de plusieurs façons, selon le contexte.

Des exemples de types d'apprentissage centrés tantôt sur la tâche, tantôt sur l'utilisateur sont présentés ici. Le premier exemple présente une application de suivi de la tâche en cours, mise en œuvre dans une université et utilisée pour le travail collaboratif. Dans la section suivante, nous présentons et discutons plusieurs exemples de caractérisation de comportements pour différents hypermédias. Dans la dernière section, nous donnons une description plus approfondie d'une tâche en particulier : l'apprentissage humain, en détaillant brièvement les modèles de représentation des connaissances et les méthodes utilisées pour personnaliser l'enseignement.

1. Analyse de l'utilisation d'une interface pour le travail collaboratif

Le projet « Tasktracer » de l'Université d'Oregon aux Etats-Unis est une plateforme de travail collaboratif qui relève les données de l'interaction HM pour les affecter à une tâche. Ainsi, à chaque fenêtre, ou plus exactement à chaque contenu d'une fenêtre, est associée une tâche. Les utilisateurs du système peuvent ainsi consulter les documents et les connaissances vus par les autres collaborateurs réalisant la même tâche. Le système apprend donc le comportement de chaque utilisateur et prédit les documents consultés.

Une méthode de reconnaissance de la tâche est décrite dans [Shen & al., 2006]. L'application proposée relève les mots traités dans divers documents tels que les rapports, les mails, les notes, etc. et l'utilisateur doit les associer à une tâche. Cette association est ensuite apprise par le système à l'aide d'un réseau bayésien. Les auteurs ont ainsi pu implémenter un système d'aide à l'organisation des documents. En observant le rangement de fichier sur un disque, ou le tri des mails par un utilisateur, le système est capable de proposer un répertoire, ou une catégorie auquel un nouveau document fait référence.

Le système de récolte de traces de Tasktracer a également permis de concevoir un outil de suivi d'historique internet. Cette application apprend une sélection de sites

Internet qu'un utilisateur souhaite conserver pour l'accomplissement d'une tâche future ou le suivi d'une recherche en cours [Lettkeman & al., 2006]. Les auteurs comparent les performances des méthodes telles que les machines à vecteurs de support (SVM) et les réseaux bayésiens pour effectuer la classification des sites à conserver.

Ce projet attire beaucoup les regards de la communauté [ScienceDaily, 2005] car les ambitions de leurs auteurs semblent à la hauteur de leurs résultats. Il est néanmoins difficile de réellement éprouver l'apport d'un tel système. En effet, la classification, correspondant à la phase d'apprentissage de la tâche par le système, est faite par l'utilisateur. Celui-ci doit renseigner un champ, chaque fois qu'il change de tâche ou de sous-tâche. De plus, cette notion de hiérarchie des tâches n'est pas clairement définie par les auteurs et, pour l'instant, une tâche n'est affectée qu'à une fenêtre de l'interface et à son contenu.

2. Construction et caractérisation de profils d'utilisation

Dans [Bidel & al., 2003], les auteurs comparent différentes méthodes d'apprentissage artificiel pour la classification de comportements utilisateurs de produits hypermédias. Le but de cette expérience est de savoir s'il est possible de classifier un utilisateur en cours d'utilisation afin de personnaliser l'interface. Le système présenté doit donc fonctionner en ligne et l'apprentissage du comportement se fait à chaque visite d'une page de l'hypermédia. Les données utilisées sont issues directement des interactions HM et sont formalisées pour l'apprentissage de l'expérience. Ainsi, ils définissent une « trame » comme étant la visite d'une page Internet et étant formée d'un vecteur de 8 variables calculées à partir d'un traitement spécifique sur les données de l'interaction. Ces 8 caractéristiques se répartissent en 3 groupes selon le type d'informations qu'elles reflètent :

Indicateur de lecture	- Taux de lecture (rapport de temps passé sur le temps de référence) sur le paragraphe introductif des pages - Taux de lecture sur le reste de la page - Taux d'activité correspondant au nombre d'évènements d'interaction survenus sur les pages lues
Indicateur sur la ressource	- Pourcentage de temps passé sur les articles hypermédia - Pourcentage de temps passé sur les pages sommaires de l'hypermédia - Pourcentage de temps passé sur le moteur de recherche de l'hypermédia
Indicateur sur les concepts	- Similarité moyenne entre deux pages successives accédées par l'utilisateur - Similarité moyenne entre deux pages successives accédées par l'utilisateur mais pondérée par le temps passé sur chaque concept

Ils relèvent quatre comportements à discriminer : *survol* (acquisition d'une vision globale panoramique d'un thème), *lecture approfondie* (l'utilisateur veut se documenter de façon précise sur un thème), *recherche* (d'un document ou d'une information précise), et *papillonnage* (parcours désordonné sans stratégie ou but particulier). Les résultats montrent qu'il est possible de classifier correctement un utilisateur en ligne en mode supervisé à 80%, alors qu'en mode non supervisé les résultats ne dépassent pas 61%.

La classification effectuée dans cette expérience semble effectivement la plus appropriée dans une perspective d'adaptation de l'interface multimédia (ici une encyclopédie en ligne). Cependant, il est dommage que les auteurs n'aient pas cherché à entrer plus en profondeur dans la description de la classification effectuée. L'effet « boîte noire » des méthodes d'apprentissage artificiel utilisées (réseaux de neurones, les chaînes de Markov, ou les modèles de Markov cachés) est ici regrettable, car ils rendent impossible la personnalisation de l'interface d'un type de navigation en fonction des critères utilisés par la classification.

Une nouvelle théorie, issue des recherches menées sur la sécurité des réseaux établit un concept d'*identité* lié à la façon qu'ont les utilisateurs de naviguer sur des logiciels ou des sites Internet. Ainsi, chaque utilisateur possèderait une empreinte reconnaissable à son style de navigation. [Padmanabhan & Yang, 2006] proposent le terme *Clickprint* pour qualifier cette empreinte. Les vecteurs de données utilisés pour catégoriser les utilisateurs ont été formatés pour représenter des notions simples du comportement. Ainsi, la durée, le nombre de pages vues dans une session, la moyenne de temps passé sur chaque page, l'heure de la consultation et le jour de la semaine sont les seules caractéristiques qui servent à décrire une session. Afin d'obtenir plus de détails sur ces variables, la moyenne, la médiane, la variance, la valeur maximum et la valeur minimum des 4 premières caractéristiques ont été extraites pour qualifier le comportement de chaque utilisateur.

La méthode d'apprentissage artificiel employée (arbre de décision) a été empiriquement éprouvée et les résultats sont donc très satisfaisants. Néanmoins, le protocole expérimental de l'expérience est très contestable car en réalité, 8 catégories d'utilisateurs sont identifiées et seulement les utilisateurs très réguliers du site (plus de 100 visites) sont reconnus.

L'émergence de nouvelles pratiques sur Internet a permis à la communauté scientifique de proposer de nouveaux descripteurs de comportements. Ainsi, [Anjo &

Efimova, 2006] analysent les caractéristiques d'une communauté d'internaute écrivant des blogs (sorte de carnet personnel et partagé en ligne, intime ou professionnel). Cinq dimensions sont mises à jours :

- Les documents : ils constituent la base de publication d'un membre de la communauté.
- Les termes : ils correspondent aux concepts significatifs utilisés par un ou plusieurs membres de la communauté. Ces concepts sont présents dans les documents.
- Les personnes : un membre (auteur de blog) de la communauté.
- Les liens : une référence d'un document à un autre document, et donc implicitement, d'un auteur à un autre auteur.
- Le temps : la date de publication d'un document.

[Anjo & Efimova, 2006] posent ensuite quelques questions relatives à la compréhension de la communauté : Quels sont les principaux concepts dont parle la communauté ? Comment ces concepts sont-ils liés entre eux ? Ces concepts évoluent-ils au cours du temps ? Qu'est ce qui différencie un membre de la communauté des autres ? Quelles sont les principales conversations de la communauté ?

Les techniques d'analyses de texte classique, tels que celles basées sur l'utilisation de *tfidf* et ses variantes, sont mises en œuvre pour l'analyse des documents et des concepts traités. Les réponses aux questions posées semblent pertinentes. Cependant, il n'y a aucune proposition afin d'améliorer l'interaction des utilisateurs avec la communauté.

Une autre modélisation est proposée dans [Agichtein & al., 2006] où les auteurs étudient les liens entre les préférences de recherche et le comportement d'interaction des utilisateurs avec l'interface. L'hypothèse des auteurs est que la position du document le plus pertinent dans le résultat que produit une requête dépend du profil d'interaction de l'internaute. Les auteurs proposent 28 caractéristiques réparties dans 3 grandes catégories :

Caractéristique liées à la requête	Intersection des mots de la requête et du titre
	Intersection des mots de la requête et du résumé
	Intersection des mots de la requête et de l'URL
	Intersection des mots de la requête et du domaine
	Nombre de mots dans la requête
	Moyenne de l'intersection des mots avec ceux de la requête suivante
Caractéristique liées à la navigation	Durée de pause sur la page
	Durée cumulée sur la page
	Durée dans le domaine
	Durée sur le site
	Résultat trouvé par le lien
	Le lien renvoie au site
	Le lien est redirigé
	Le lien permet de trouver la page
	Nombre de clics pour trouver la réponse
	Durée moyenne sur une page
	Variance de la durée moyenne sur la page
	Variance de la durée cumulée sur la page
	Variance de la durée sur le domaine
	Variance de la durée sur le site
Caractéristiques de parcours	Position du lien cliqué dans la page de résultat
	Nombre de clic pour la requête
	Fréquence relative des clics pour la requête
	Variance du nombre de clic pour la requête
	Le lien suivant est cliqué
	Le lien précédent est cliqué
	Un clic au dessus
	Un clic en dessous

Les auteurs utilisent un réseau de neurones pour détecter les comportements prédominants des utilisateurs selon leurs tendances à cliquer sur le 1er, le 2^{ième}, le 3^{ième},… lien du résultat de la requête. L'entraînement intensif que permet cette technique statistique permet aux auteurs de publier de bons résultats de reconnaissance de profil. Néanmoins, aucune possibilité d'interprétation cognitive n'est possible car les paramètres utilisées ainsi que la méthode ne permettent pas la personnalisation de l'interface ou de l'interaction.

3. Cas d'application : les agents intelligents pour les systèmes éducatifs

Les systèmes dédiés à l'apprentissage humain ou plus particulièrement à l'apprentissage scolaire sont des systèmes d'e-*learning*. Ils sont généralement conçus pour l'apprentissage d'un domaine en particulier, et dans ce cas ils sont appelés *tuteurs intelligents*. Il est alors généralement nécessaire de connaître l'état de connaissance de l'utilisateur sur un domaine avant de vouloir personnaliser l'apprentissage de celui-ci.

1. La représentation des connaissances

La plupart des hypermédias éducatifs représentent les connaissances de l'utilisateur dans un *modèle en couche* (ou *overlay model*). Le domaine de connaissance est alors représenté par une sorte de réseau sémantique de concepts. A chaque concept du modèle de domaine est associée une valeur d'estimation des connaissances de l'utilisateur. Selon les approches, cette estimation peut être une valeur binaire « acquis / non acquis » ou qualitative « bien / satisfaisant / insuffisant ». L'ensemble des couples *concept - estimation* forme alors le modèle de couche assez puissant et flexible pour mesurer indépendamment les connaissances de l'utilisateur à propos de sujets différents du domaine à étudier [Akoulchina, 1998]. L'élève/utilisateur est donc modélisé ici à l'aide de données fabriquées sur la notion à apprendre.

On peut ici s'interroger sur l'intérêt de modéliser les concepts. En effet, les supports éducatifs hypermédias sont généralement des documents. Ceux-ci peuvent être formés de plusieurs concepts, organisés pour formuler des connaissances sur un domaine. Le modèle en couche peut tout de même servir à la modélisation de la connaissance des documents, mais il faudra alors permettre à un outil d'établir les bonnes corrélations concepts-documents. Cette idée sert maintenant de base à tous les systèmes d'EIAH (Environnement Intelligent pour l'Apprentissage Humain). Les questions d'adaptation de présentation de l'information hypermédia doivent donc à la fois prendre en compte les concepts à apprendre et les documents servant à cet apprentissage. La notion de graphe conceptuel a été introduite pour définir à la fois ce qu'est un concept, un nœud d'un graphe, et les relations conceptuelles, les liens entre les nœuds.

La représentation des connaissances est encore largement un domaine à explorer. Nous présentons dans le dernier chapitre de ce mémoire, un exemple de modélisation où seule l'interaction HM est utilisée pour modéliser implicitement les connaissances d'un utilisateur.

2. La personnalisation des tuteurs intelligents

[Beck & Woolf, 2000] analyse le comportement d'élèves étudiant l'arithmétique au collège. Un tuteur intelligent connaît le niveau de chaque élève et peut proposer des exercices plus ou moins difficiles à réaliser dans des temps plus ou moins importants. Un élève peu enthousiaste pour les mathématiques se verra demander des exercices rapides adaptés à son niveau. Un élève passionné aura au contraire des exercices plus difficiles à réaliser, en fonction du temps qui lui est imparti. Dans ce cas, le modèle de

l'élève/utilisateur est utilisé pour personnaliser l'activité proposée. L'interaction est au centre de l'étude et de l'adaptation pour la pédagogie.

Une autre approche de classification d'utilisateurs est présentée dans [Renaudie, 2003]. Ici les connaissances sur le domaine sont découvertes sur les évènements survenant dans le dialogue HM et sont analysées pour catégoriser des élèves apprenant l'algèbre. Les résultats produits renseignent sur le niveau des élèves, selon le protocole de description de la tâche utilisée, et sur les « groupes d'erreurs » que peuvent faire certains élèves avec des difficultés. Dans cette analyse, les données utilisées pour la classification des individus sont la connaissance experte du développeur du système, et on ne pourra pas exploiter directement ce modèle pour d'autres domaines de connaissance. De même que précédemment, l'interaction entre l'élève et la machine est au centre de l'étude qui permet la personnalisation de l'enseignement.

4. Bilan sur l'analyse

Les applications utilisant des analyses de données issues de l'interaction sont nombreuses. Nous avons présenté des applications utilisées pour aider le travail collaboratif dans une équipe de recherche. Nous avons ensuite présenté des travaux de détection en ligne des comportements d'utilisateurs de produits hypermédias. Ces deux approches utilisent les évènements de l'interface comme moyen pour construire différents indices de comportement. La première s'attache aux contenus des interfaces et les interprète comme représentation de la connaissance d'un utilisateur et de sa tâche courante. La seconde prête plus d'attention aux parcours d'un utilisateur pour définir des vecteurs de données caractéristiques du comportement, tirés directement des évènements de l'interface. Ces deux méthodes sont proches par les données qu'elles manipulent et pourtant différentes par la façon qu'elles ont de les manipuler.

Nous avons ensuite présenté un cas d'application plus précis : l'apprentissage humain. Les systèmes actuels d'aide à l'apprentissage humain sont pour la plupart des tuteurs intelligents. Ils manipulent des représentations de la connaissance humaine et observent les interactions de l'élève avec le système pour adapter ce dernier efficacement. Ces systèmes sont performants, mais ont le défaut d'être dédiés, c'est-à-dire difficilement exportables pour différents domaines d'enseignement. Ces systèmes demandent donc un apport très important de l'expert enseignant.

Conclusion de la première partie

La modélisation utilisateur à partir de traces d'utilisation d'une interface graphique est donc très présente dans la littérature. Nous avons vu que plusieurs types de modélisation existent. Certains s'attachent à comprendre le comportement psychologique d'utilisateurs et d'autres cherchent des automatisations pour la construction ergonomique d'interfaces. Entre les deux, des modèles personnalisent le dialogue HM en essayant, par exemple, de personnaliser l'information lors de la recherche de document sur Internet.

Aussi, dans la plupart des cas présentés, les données relevées proviennent directement de l'interaction homme-machine et sont formatées pour chaque application. Afin de mettre à jour l'ensemble des propriétés nécessaires pour la modélisation, nous avons recensé les méthodes et techniques d'analyse généralement employées en suivant la typologie décrite dans [Hilbert & Redmiles, 2000].

Des systèmes de visualisation de ces données ont ensuite été présentés. Après avoir mis en évidence la nécessité de telles visualisations pour faire des analyses, nous avons présenté des systèmes automatiques permettant notamment d'expliquer l'interaction de l'homme avec Internet. D'autres formalismes s'attachent à trouver la meilleure représentation d'une tâche en la décomposant selon différents contextes d'interactions. Enfin, des systèmes mixtes, basés sur une connaissance de la tâche, permettent de construire automatiquement, à partir de l'observation de l'activité, des représentations complètes des processus cognitifs humains.

Des applications utilisant des méthodes de fouilles de données ont également été présentées. Des applications semi-automatiques, car nécessitant une aide de l'utilisateur, permettent de concevoir une collaboration intelligente entre différents individus travaillant ensemble. Nous avons également présenté des systèmes capables de catégoriser automatiquement des comportements d'utilisateurs manipulant des hypermédias.

Enfin, nous nous sommes attachés à un cas d'application de systèmes utilisant les données provenant de l'interaction HM : les systèmes éducatifs. Nous avons distingué deux types d'applications : celles basées sur la représentation des connaissances de l'apprenant, et celles qui observent le comportement des élève pour personnaliser l'enseignement.

Toutes ces applications n'ont pas de base commune de travail et chacun essaie de son côté de trouver les meilleures méthodes de récolte de traces, de visualisation de données et d'analyse.

Aussi, nous nous sommes posé la question de savoir comment récolter des traces d'interactions de l'utilisateur avec la machine qui puissent servir à toutes ces modélisations et applications. La partie suivante s'attache donc à répondre à la question : Comment peut-on rendre compte, de façon complète et automatique, de l'interaction homme-machine pour automatiser les processus de personnalisation d'interface ?

Partie 2 - Nouvelle méthode de récolte de traces d'interaction homme-machine et outils d'analyses

« La vraie nouveauté naît toujours dans le retour aux sources. »

Edgar Morin

Introduction de la deuxième partie

L'interaction homme-machine peut être étudiée comme tous les autres types d'interactions. Les données à récolter pour une étude empirique sont naturellement produites par les acteurs du dialogue : l'homme et la machine. Aussi, ces deux sources produisent une très grande quantité de données récoltables à partir desquelles de nombreux scientifiques ont cherché à modéliser les comportements des utilisateurs. Ces données peuvent être classées selon deux catégories : les données objectives et les données subjectives. Ces dernières sont généralement issues d'observations d'experts qui y apportent une interprétation. Au contraire, les données objectives sont celles qui sont naturellement produites par les acteurs de l'interaction pendant le dialogue.

Lorsque deux personnes parlent, les mots prononcés forment une quantité de données objectives qui peuvent être simplement transcrites mais ne rendent pourtant pas compte de l'intégralité de l'interaction. En effet, le simple fait d'étudier le seul dialogue de deux personnes par une transcription de celui-ci masque le contexte dans lequel la conversation a eu lieu. D'autres données seront également occultées comme les différentes intonations que les intervenants choisissent de prononcer. La transcription d'une conversation représente donc réellement et objectivement l'interaction, mais de façon incomplète. Cependant, lorsque l'analyse de la conversation porte sur le fond de la discussion plus que sur la forme, la transcription du dialogue est souvent suffisante.

Qu'en est-il pour les interactions homme-machine ? Quelles sont les données objectives que l'on doit récolter pour une étude cherchant à améliorer cette interaction automatiquement ? Ces données seront nécessairement incomplètes et la sélection opérée lors de la récolte des données doit être étudiée. Une partie du contexte de l'interaction peut-elle être enregistrée pour faciliter l'analyse ? Le contexte d'un évènement de l'interaction peut-il être représenté dans des traces ?

Comme nous l'avons vu dans notre état de l'art, les études scientifiques portent sur de multiples améliorations possibles de l'interaction homme-machine et les données utilisées pour modéliser le comportement des utilisateurs peuvent provenir de différentes sources et être de différentes natures. Pour les analyses automatiques, il faut

convenir d'un format de données auquel chacun peut faire référence. Que ce soit pour la visualisation ou pour la fouille de données, nous présentons ici une nouvelle méthode pour la génération automatique de traces d'utilisation d'interfaces graphiques. Les évènements générés naturellement par une interface graphique sont une trace réelle et objective de l'activité de l'utilisateur telle qu'elle est perçue par l'ordinateur et ils nous semblent être le meilleur compromis pour rendre compte automatiquement du dialogue homme-machine.

Dans une interface, chaque objet possède plusieurs propriétés particulières que nous n'énumérerons pas ici, mais il est tout de même possible de rendre compte de leur apparence et de leur fonctionnalité de façon automatique. Nous avons montré, dans la partie précédente, les propriétés nécessaires pour des traces se voulant génériques pour différents types de modélisation centrés autour du dialogue homme-machine. Aussi, nous introduisons dans cette partie une nouvelle méthode de récolte de traces satisfaisant ces conditions.

Le premier chapitre de cette partie présente notre méthode de récolte de traces d'utilisation d'une interface graphique. Nous présentons d'abord de façon générale, la structure en arbre que nous avons adoptée pour la récolte de traces d'interaction. Nous expliquons ensuite comment l'impact cognitif lié à la structure visuelle d'une interface peut être mise en parallèle avec la réalisation d'une tâche par un utilisateur. Un exemple de cette méthode est présenté sur un cas simple de navigation sur un hypermédia ainsi que quelques recommandations pratiques pour l'implémentation de cette méthode.

Nous avons présenté dans le Chapitre 3, quelques exemples d'outils d'analyses et de visualisation de données modélisant les interactions HM. Ainsi, il a semblé difficile de visualiser l'intégralité du comportement des utilisateurs sur une interface, ce qui est dû notamment à l'abondance des données et à leur incapacité à représenter l'évolution du dialogue. Nous décrivons ci-dessous les outils logiciels que nous avons implémentés pour analyser les traces générées par notre méthode. Nous présentons également, une nouvelle méthode de visualisation à partir des données décrites précédemment et qui permet de rendre compte très précisément de l'activité du dialogue HM. Les représentations graphiques de ces données offrant plusieurs types d'analyse du dialogue HM sont difficiles à mettre en œuvre. Aussi, en nous inspirant des spécifications scénario de UML [Rumbaugh & al., 2004], nous proposons une visualisation de ces données pour une analyse cognitive de l'interaction HM.

Dans le Chapitre 7, deux exemples d'analyse de cette visualisation sont proposés : l'analyse de parcours sur différents types d'interfaces hypermédias d'un manuel scolaire

électronique, et l'analyse de parcours d'un groupe d'utilisateurs pour mettre en évidence des comportements en fonction des contextes d'utilisation.

Chapitre 5. Traces d'interaction homme-machine

Les interactions homme-machine peuvent être étudiées comme n'importe quel autre type d'interactions. Par exemple, pour qu'une conversation entre deux individus fonctionne, il faut que les deux interlocuteurs se répondent afin de donner un sens au dialogue. Le sujet de la conversation est alors le sujet de l'interaction, et ne dépend pas directement des intervenants même si, évidemment, ce sont eux qui choisissent le sujet de la conversation et décident d'en changer quand bon leur semble. Ceci dit, lorsque les deux parties parlent à propos d'un sujet, seules leurs idées portant sur le sujet leur sont propres. L'énonciation d'une opinion constitue alors une action propre à l'interlocuteur. Hors du contexte de la conversation, le thème du dialogue est déjà existant avant que les interlocuteurs ne se l'approprient. Le sujet de la conversation appartient donc au contexte de l'interaction, alors que les énonciations des opinions des participants constituent les différentes actions.

Alors comment distinguer ce qui est du domaine du contexte (tel que le sujet de la conversation) de ce qui est du domaine de l'intervention sur ce contexte (tel que l'énonciation d'une opinion), propre aux interlocuteurs ? Pour l'analyse d'une conversation, cette distinction est difficile à établir. En effet, chaque intervention modifie le contexte de l'interaction. C'est précisément cette idée qui différencie l'interaction de l'action.

Dans ce qui suit, le premier paragraphe présente notre modèle de récolte de traces d'une interaction en général. Nous avons choisi une structure hiérarchique et ordonnée, proche de la description des processus cognitifs impliqués dans la réalisation de tâches (voir Chapitre 1.1). Nous présentons ensuite notre modèle sur les interactions homme-machine plus précisément, et comparons la structure visuelle des interfaces graphiques modernes avec la structure hiérarchique précédemment évoquée. Nous détaillons un exemple de récolte de traces ainsi que des recommandations pratiques pour l'implémentation de cette méthode.

1. La récolte de traces d'interaction

La Figure 13 montre un schéma représentant la structure des traces telles qu'elles peuvent être générées directement en format XML. Cette structure en arbre stratifié ordonné (ASO) permet de réduire la redondance inévitable dans la génération de traces. En effet, on ne peut espérer rendre compte de tous les évènements d'une interaction ainsi que de l'impact qu'ils produisent sur le contexte sans un minimum de redondance. Ceci donne naturellement la faculté aux traces de pouvoir être compressées de façon très efficace. En effet, le niveau de détail des traces générées peut être paramétré (le nombre de couche descriptive de contexte, cible, évènements peut être limité) et la nature textuelle des traces générées autorise un stockage informatique optimisé par les algorithmes de compression classiques tels que LZM [Welch, 1984].

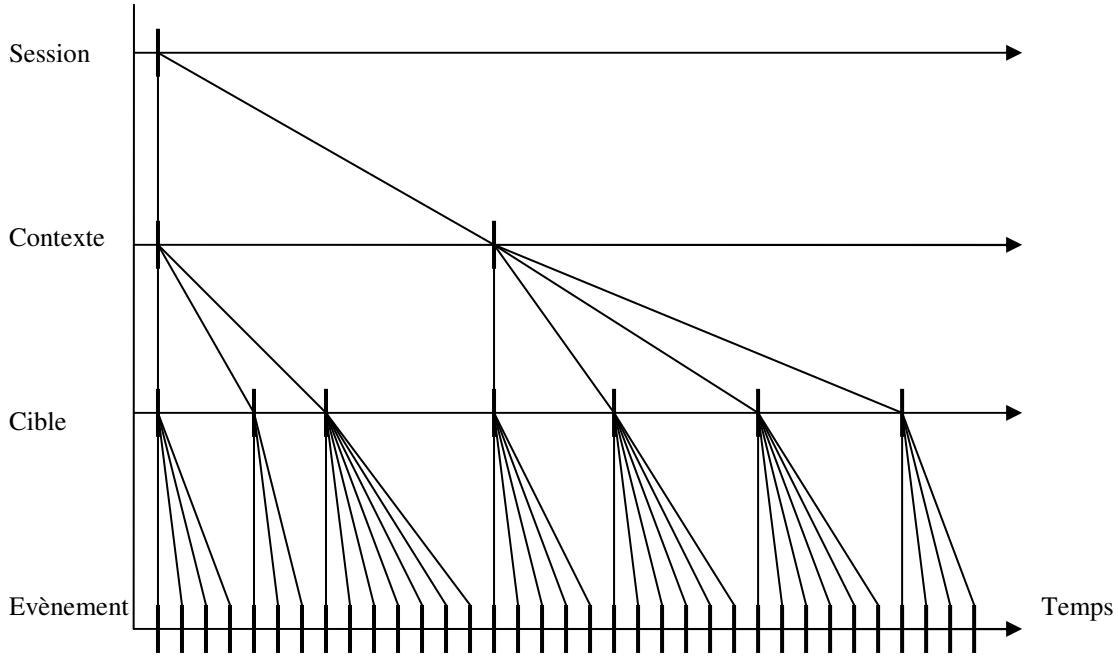

Figure 13. Structure arborescente des traces

Cette structure en arbre est organisée comme suit :

- La racine de l'arbre contient toutes les informations relatives à la session en cours : nom de l'utilisateur, date de début. Il s'agit donc du niveau « Session ». Il est très important de renseigner au maximum les informations de session dont on dispose. En effet, si l'on connaît la tâche générale en cours dans cette session, on pourra la noter sous une forme textuelle narrative. Comme nous l'avons vu précédemment (voir Figure 12), les adaptations d'une interaction ne sont envisageables que si l'on connaît la tâche en cours et/ou l'utilisateur. C'est à ce niveau qu'il faut renseigner ces champs de façon à diriger l'analyse, tantôt sur la tâche, tantôt sur l'utilisateur.
- Le plus haut niveau de l'arbre sous la racine concerne le descripteur le plus général de l'évènement. Il s'agit du conteneur de la cible de l'évènement. Nous l'appellerons « Contexte ». Ce niveau contient la sémantique de contexte la plus générale, et il nous parait essentiel dans la mesure où il rend la trace plus lisible à un œil averti. Les différentes ressources manipulées seront notées à ce niveau, avec le maximum d'attributs descriptifs. Il ne s'agit pas de noter tout un texte présent dans une ressource, ni d'y inclure une image manipulée pendant

l'interaction, cependant, plus le niveau contexte sera décrit, plus les analyses pourront être pertinentes. Dans le cas d'étude d'utilisation de ressources complexes, comme des livres contenant des images, des textes et des illustrations ce niveau pourra être amené à être multiplié. On pourra ainsi trouver plusieurs contextes permettant de bien rendre compte de la situation de l'évènement en cours.

- Le niveau suivant est le niveau « cible » (ou « target »). C'est sur ce niveau que porte l'évènement. Il s'agit par exemple du papier sur lequel sont inscrites les notes d'une réunion, ou la région d'une image sur laquelle porte un commentaire. La distinction entre le niveau cible et le niveau contexte doit être hiérarchique dans le sens où la cible est toujours contenue dans le contexte. De plus, il est essentiel encore une fois de rendre compte le plus précisément possible des attributs permettant de décrire la cible. Sans dresser la panoplie des évènements qui y sont applicables, nous indiquons ci-après, une méthode pour noter les attributs les plus pertinents pour une interaction homme-machine.
- Le dernier niveau, qui constitue les feuilles de l'arbre, contient l'« évènement » proprement dit. Sous forme d'attributs, on y notera le type, la date mais aussi l'attribut spécifique de l'évènement. Il s'agira donc d'une action décrite par un verbe. Lors d'une discussion par exemple, lorsque l'un des participants interpelle un autre participant, l'action peut être « prendre la parole ». Si le sujet traité lors de cette interpellation ne change pas, l'action portera sur la même cible et donc sur le même contexte également.

Ce modèle général de récolte de données provenant d'une interaction peut être utilisé pour de nombreux cas. Ainsi, lors de la prise de note pendant une réunion, ce modèle peut être respecté. La sémantique du discours des divers interlocuteurs ne sera alors que peu prise en compte, mais le déroulement de la réunion sera bien restitué.

Remarque

Comme il est souvent relevé dans les méthodes d'analyse des processus cognitifs présentées précédemment dans la Figure 1, la distinction entre la cible et le contexte n'est pas toujours triviale. Aussi, nous présentons ci-après un exemple de récolte de traces à partir d'un hypermédia qui permet d'établir facilement la distinction entre ces deux niveaux. Dans le cas des interfaces graphiques modernes, l'implémentation des logiciels peut effectivement aider à établir cette distinction.

2. Interface graphique et modèle de tâche

La plupart des interfaces graphiques modernes sont conçues à partir d'un outil générique utilisant des Graphical User Interface (GUI). Aussi appelé WIMP, acronyme anglais pour Windows (fenêtres), Icons (icônes), Menus (menus) and Pointing device (dispositif de pointage), ce type d'interfaces graphiques a été inventé par la firme Xerox et rendu célèbre par le Macintosh de la société Apple. Dans ces interfaces, plusieurs contrôles graphiques sont couramment utilisés pour interagir avec l'utilisateur comme les boutons, les menus, les listes déroulantes, les ascenseurs, etc.

Lorsque l'utilisateur manipule un périphérique tel que le clavier ou la souris, le système génère des évènements qui sont interprétés. Cette interprétation se traduit sous deux angles. Le premier est de rendre compte de la manipulation de l'utilisateur sur le système. Le deuxième est de rendre compte de la réaction du système en modifiant son état. Ce changement d'état est notifié à l'utilisateur par un autre des périphériques disponibles : écran, enceinte, périphérique haptique, etc.

Par exemple, le mouvement physique de la souris, initié par un processus cognitif de l'utilisateur, est généralement représenté à l'écran du système par un mouvement du curseur que l'on qualifie d'évènement utilisateur. De plus, les composants présents dans l'interface graphique peuvent être programmés pour réagir à ce mouvement de souris, auquel cas, le système modifiera l'état d'un des périphériques à sa disposition. Il s'agira là d'un évènement système. Ainsi, les évènements produits lors du dialogue homme-machine sont générés de façon naturelle par le système et il est donc possible de les récupérer pour nos études.

Les évènements utilisateurs sont en général bien moins nombreux que les évènements systèmes. Cela s'explique très simplement : lors du lancement d'un logiciel ou de l'ouverture d'une nouvelle page Internet par exemple, le système produit un grand nombre d'évènements car il effectue beaucoup de changements dans l'interface graphique. Chaque composant d'un logiciel qui se met en exécution crée plusieurs évènements comme son apparition, sa mise en page, la vérification de la disposition graphique, etc. De la même façon, lorsqu'une nouvelle page Internet s'affiche dans un navigateur, tous les composants formant la page génèrent des évènements d'apparition, de mise en page, etc. Par exemple, on peut estimer que le ratio du nombre d'évènements utilisateurs sur le nombre d'évènements machines lors d'une utilisation normale du traitement de texte *Gedit* sous linux est d'environ 1/25. En effet, certaines commandes, comme la vérification des conflits d'affichages, génèrent beaucoup d'évènements, notamment lors d'opérations comme l'application d'un style à un paragraphe ou l'ajout

d'une figure dans un texte. Toutefois, ce ratio varie beaucoup en fonction du logiciel tracé et des actions de l'utilisateur.

1. La structure WIMP

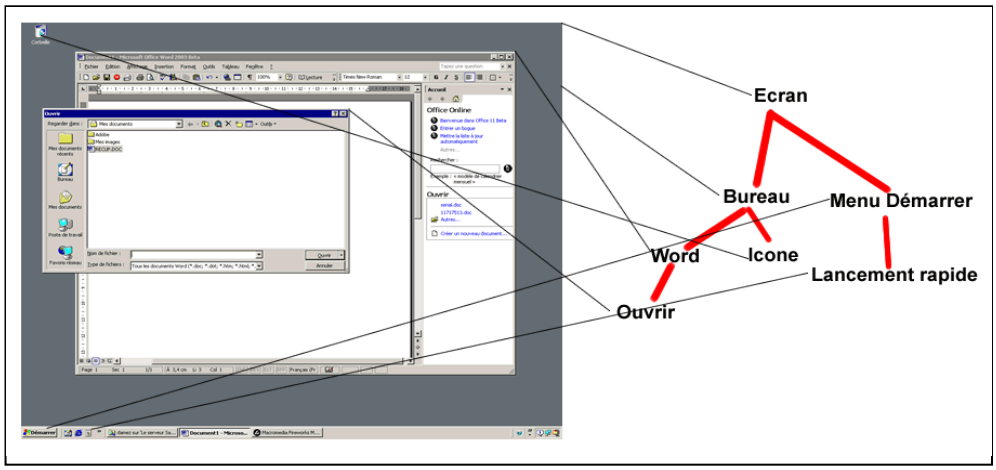

Figure 14. L'arbre de l'interface.

La représentation des traces que nous avons choisie est issue d'une observation des interfaces WIMP (*Windows, Icons, Menus and Pointing device*). En effet, il nous est apparu qu'une telle interface fait référence à une structure arborescente. En réalité, seule l'apparence peut être représentée comme un arbre comme le montre la Figure 14. L'écran constitue alors le nœud racine de l'arbre. Dans les interfaces modernes, une barre correspondant à un menu « démarrer » sert de principal objet amenant aux logiciels et autres outils d'un système d'exploitation. Le bureau, l'espace non occupé par le menu « démarrer », contient généralement des icônes utilisées pour accéder à des applications. On peut très bien considérer ces icônes comme filles du bureau. En effet, il faut d'abord prendre connaissance de l'environnement bureau, explicitement ou non, pour utiliser une icône. L'ouverture d'un logiciel, comme un traitement de texte sur la Figure 14, par le menu « démarrer » ajoutera un nouveau fils au bureau. Cette hiérarchie est donc une vue de l'esprit qui reflète tout de même l'apparence d'une interface. Nous pensons que cette apparence a un impact cognitif important dans la réalisation d'une tâche par un utilisateur.

Les fonctions des logiciels affichées sur des interfaces ne sont pas nécessairement organisées sous forme d'arbres, pourtant on voit naturellement que l'homme s'est inspiré de cette hiérarchie pour nommer les procédures qu'il voulait implémenter. Ainsi, il est facile de constater qu'un menu « Outils » n'a pas sa place dans une barre des

menus de par le simple fait que tout logiciel est un outil. Pourtant, on peut constater que l'homme adapte sa connaissance à son contexte, et on ne sera plus surpris, après usage, de trouver dans ce menu la fonction « option » bien que ce terme n'évoque aucune sémantique de possibilité d'action.

2. Les tâches

Comme le montre la Figure 15 (extraite de [Hilbert & al, 1997]), l'accomplissement d'actions sur une interface est hiérarchiquement structuré et peut être représenté sous forme d'arbre. En effet, un ensemble d'actions réalisées dans un même contexte fera apparaître une tâche sur un seul domaine, soit « *Domain/Task-Related* ». Par exemple, cela traduit le phénomène : 'on manipule un courriel avec un logiciel de courriel'. Un ensemble d'actions sur un même objet de l'interface sera d'un même niveau d'abstraction : « *Abstract Interaction Level* ». Cela traduit par exemple le fait qu'on édite généralement du texte, dans une zone de saisie de texte. La hiérarchie représentée dans cette figure permet donc d'établir qu'une zone de saisie de texte est nécessaire dans un logiciel de courriel.

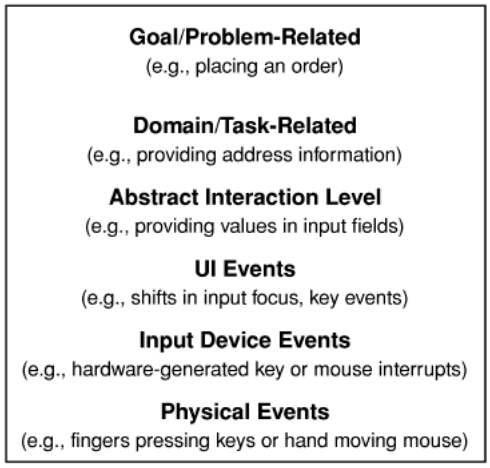

Figure 15. Niveaux d'abstractions des interactions humaines sur une interface machine [Hilbert & al, 1997]

Cette représentation confirme les hypothèses formulées par les psychologues cogniticiens dans leurs propositions de modèle pour décrire la réalisation d'une tâche. Comme nous l'avons vu dans l'état de l'art, les modèles utilisés pour analyser la réalisation de tâche comportent les notions de hiérarchie et de séquençage des actions d'un utilisateur.

En comparant ces deux structures hiérarchiques qui reflètent tantôt la structure visuelle et organisationnelle d'une interface tantôt la représentation cognitive de l'accomplissement d'une tâche, nous proposons une méthode de récolte de traces d'interactions qui vérifie ces propriétés.

3. Exemple d'implémentation pour un hypermédia

Pour illustrer le modèle présenté précédemment, nous présentons un exemple d'une trace récoltée lors de la l'utilisation d'un hypermédia (voir Figure 16).

```
<Session id="utilisateur1">-
  <context role="HTML" uri="file:///D:/Expelip6/Experience_fichiers/question.html">
   - <target role="INPUT" value=" Show first question ">
      - <eventMotif type="MouseClic" duration="130" time="51153532">
            <event type="mousedown" time="51153532" />
            <event type="mouseup" time="51153652" />
            <event type="click" time="51153662" />
        </eventMotif>
     </target>
   - <target role="TEXTAREA">
      - <eventMotif type="MouseClic" duration="120" time="51156697">
            <event type="mousedown" time="51156697" />
            <event type="mouseup" time="51156817" />
        </eventMotif>
     </target>
...</context>
</Session>
```

Figure 16. Extrait XML d'une trace issue de l'observation d'utilisation d'un hypermédia

La structure en arbre est organisée comme suit :

- Le niveau « Session », racine de l'arbre, contiendra toutes les informations relatives à la session en cours. En plus de ceux évoqué précédemment, on pourra noter les attributs spécifiques à l'utilisation d'un hypermédia soit : le système d'exploitation, les dimensions de l'affichage, la résolution de l'interface graphique, etc. (Ce niveau n'est pas représenté dans la Figure 16.)
- Le plus haut niveau de l'arbre est le niveau « contexte ». Dans notre exemple, il s'agit de la page web visitée : *context role="HTML" uri="file:///D:/Expelip6/Experience _fichiers/question.html"*. Dans le cas de la manipulation de plusieurs pages web, ou de plusieurs logiciels simultanément, c'est lui qui va bâtir le parcours de l'utilisateur. C'est d'ailleurs en général ce niveau qui est utilisé en modélisation de recherche d'informations. Dans le cas

d'étude d'utilisation de logiciels complexes (comme les *Multiple Document Interface*), ce niveau pourra être amené à être multiplié.

- Le niveau suivant est le niveau « cible » : *target role="INPUT" value=" Show first question "*. Il est essentiel encore une fois de rendre compte le plus précisément possible des attributs qu'il possède. Dans le cas d'un hypermédia, le nom, la description, mais aussi, la référence de l'objet en question sont des renseignements très utiles pour étudier le comportement des utilisateurs.
- Un niveau intermédiaire peut être ajouté si la manipulation de l'interface conduit à beaucoup d'éléments du dernier niveau. Ce niveau correspond à la sémantique de l'« action » de l'utilisateur. On y regroupera sous des types génériques (clavier, souris) les évènements qu'il supporte au niveau inférieur, exemple : *eventMotif type="MouseClic" duration="130" time="51153532"*. Il servira notamment pour les études statistiques à effectuer lors de l'interprétation de ces traces. On pourra y spécifier la durée et la date de commencement de l'action. Toutefois, il faudra prêter une attention particulière à ce niveau si l'on souhaite avoir une description très précise des actions. Comme évoqué au paragraphe 2, c'est à ce niveau que l'on pourra différencier le clic normal du clic hésitant.
- Le dernier niveau contient l'« évènement » : *event type="mousedown" time="51153532"*. Sous forme d'attributs, on y notera le type, la date mais aussi l'attribut spécifique de l'évènement. Pour un évènement de type « scroll » par exemple, on notera le niveau auquel l'ascenseur de défilement est situé. Pour un évènement faisant apparaître un popup, on y notera le texte du popup en question. Pour obtenir ce dernier sans connaître toutes les propriétés de tous les types d'évènements, on peut les comparer à la valeur qu'il avait précédemment. Si celle-ci est différente, alors il faut la noter, sinon l'oublier.

De façon générale, chacun des nœuds de l'arbre décrit dans le fichier correspond à un objet de l'interface et possède plusieurs propriétés. Certaines révèlent une information importante sur l'action de l'utilisateur, d'autres moins. Les propriétés les plus importantes sont celles qui évoluent au cours du temps. On peut donc comparer la valeur d'une propriété à celle qu'elle avait précédemment pour savoir s'il faut en tenir compte. Notons également que les propriétés généralement appelées *Id, label* ou *nom* sont très importantes. Ce sont en effet les seules qui puissent assurer qu'un nœud de l'arbre (soit un objet de l'interface) n'est pas confondu avec un autre qui aurait pris les mêmes propriétés.

4. Recommandations pour la récolte de données

1. L'accessibilité

Les professionnels de l'informatique ne se sont pas préoccupés de concevoir des systèmes d'accessibilité générique avant 1997 avec le WAI (Web Accessibility Initiative) organisé par le W3C. Ainsi, le premier standard officiel n'est apparu qu'en décembre 2002 avec le « User Agent Accessibility Guidelines ».

Initialement pensé pour la traduction d'interfaces visuelles vers un support à destination d'handicapés visuels, la couche d'accessibilité permet également de rendre compte de tous les évènements susceptibles d'intervenir dans le dialogue HM. Les spécifications par des organismes professionnels sont destinées aux développeurs d'applications, mais il faut reconnaître qu'il incombe à ces derniers la responsabilité de renseigner la « couche accessibilité » dans le code de l'application. Ces modules d'accessibilité permettent aux constructeurs de supports alternatifs d'accéder aux fonctions de l'interface, et de rendre compte de toutes les modifications qui y sont apportées en temps réel.

L'interrogation de cette couche d'accessibilité nous paraît être le meilleur moyen de tracer l'activité d'un utilisateur sur une interface graphique. En effet, lorsque celle-ci est bien renseignée, elle permet l'accès de façon générique aux différents rendus et aux différents types d'implémentation mis en œuvre dans les interfaces graphiques modernes.

2. Le flot d'évènements

Il convient de noter qu'observer un évènement peut corrompre cet évènement. Dans notre cas, l'évènement n'est pas véritablement observé mais le fait de vouloir le noter dans un fichier peut ralentir son exécution. En effet, les mouvements de souris par exemple, généralement assez nombreux dans les interfaces graphiques, ne peuvent être écrits sur un fichier en même temps qu'ils sont effectués sans perturbation de l'environnement. Certaines machines très puissantes pourront le faire dans le cas de mouvements de souris. Mais il est apparu pendant nos tests que d'autres types d'évènements peuvent survenir de manière bien plus rapide que les mouvements de souris. Par exemple, lors d'étude portant sur les hypertextes, au cours de la mise en page, le système utilise les fonctions propres à l'hypertexte. Ainsi, lorsque l'arbre de l'interface se crée, chaque nœud génère un grand nombre d'évènements correspondant à son apparition : insertion du nœud dans l'arbre, mise en place des dimensions, vérification des conflits d'affichage, affichage, etc.

Pour remédier à ces difficultés, il est possible de ne noter que les évènements correspondant réellement à des évènements d'utilisateurs. Ainsi, un clic sur un lien d'une page Internet sera un évènement utilisateur. Par contre, les évènements qui en résultent, soit l'affichage de la page Internet, sa mise en page, etc. seront considérés comme des évènements systèmes et ne seront pas relevés.

Nous avons donc classé l'ensemble des évènements DOM décrits dans les spécifications du *World Wide Web Consortium* [W3C, 2000] en séparant les évènements utilisateurs pertinents des évènements systèmes n'apportant pas d'information utile sur le comportement de l'utilisateur manipulant un hypermédia (voir Figure 17).

	Evènement utilisateur	Evènement clavier	Evènement souris	Evènement HTML	Evènement de mutation
Evènement utilisateur	DOMFocusIn, DOMFocusOut	keydown keyup keypress	click, mousedown, mouseup, mousemove	select, submit, resize, scroll, focus	
Evènement système	DOMActivate		mouseover, mouseout	load, unload, abort, error, change, reset, blur	DOMSubtreeModified, DOMNodeInserted, DOMNodeRemoved, DOMNodeRemovedFromDocument, DOMNodeInsertedIntoDocument, DOMAttrModified, DOMCharacterDataModified

Figure 17. Classification des évènements de l'interface pertinents pour une analyse.

Cette sélection faite, il se peut que le nombre d'évènements à récolter soit toujours trop grand pour ne pas entraver le bon déroulement du programme. Aussi, nous recommandons de ne pas écrire directement les évènements utilisateurs quand ils sont reçus, mais de les garder dans une mémoire tampon, qui pourra se vider quand le rythme des évènements survenant diminuera.

Si on désire rendre compte de l'état de l'arbre de l'interface à tout moment, on peut recréer cet état par des logiciels qui exécuteront la trace de l'utilisateur. Il sera même possible de diminuer le rythme des traces pour bien noter les différentes actions qui sont effectuées sur l'interface. Si cette solution n'est pas satisfaisante et que l'on connaît les évènements de l'interface qui la transforment radicalement, la couche d'accessibilité

évoquée précédemment peut être interrogée. L'arbre complet de l'interface peut ainsi être retracé.

Dans les interfaces moderne, de type Web2.0, l'arbre de l'interface modélisé dans le DOM peut parfois être très important. Par exemple, la profondeur peut atteindre plusieurs dizaines de niveaux, ce qui créé dans notre méthode de récolte, autant de contexte. Des filtres prédéfinis doivent alors être développés afin de ne récolter que les informations réellement pertinentes pour l'étude de l'interaction.

Le formalisme XML est maintenant largement utilisé dans la communauté informatique. Aussi, nous présentons ici quelques recommandations qui nous paraissent importantes pour l'exploitation des traces.

5. Bilan sur la méthode de récolte de traces d'interactions

Nous avons présenté dans ce chapitre un formalisme de récolte de données pour tous les types d'interactions. Cette méthode s'inspire des méthodes d'analyse de l'activité cognitive d'individus en situation d'interaction. En replaçant celle-ci dans le contexte de l'interaction homme-machine, nous avons proposé un modèle permettant de rendre compte du contexte de l'interaction sur une interface. Nous l'avons pensé de façon à pouvoir modéliser les trois composantes du dialogue décrites dans l'état de l'art : l'homme, l'interaction et la machine. Plusieurs recommandations pour l'implémentation de cette méthode ont été présentées.

De très nombreuses études peuvent porter sur ces traces. Les applications développées lors de ces études peuvent être par exemple :

- La classification du comportement des utilisateurs en ligne. Très utile pour le ciblage de la consommation afin d'orienter le consommateur vers des produits proches de ses habitudes ou détecter des besoins particuliers.
- La détection d'action critique pour la conception de système automatique d'avertissement ou d'aide à la compréhension/orientation.
- La conception de système d'aide automatique comme les compagnons logiciels de plus en plus sollicités dans les interfaces complexes.
- La caractérisation de la navigation, souvent utilisée pour la refonte ergonomique et la simplification des interfaces.
- Les tests d'utilisabilité, parmi les premières utilisations des traces d'interactions. Ainsi, les interfaces innovantes peuvent être directement étudiées pendant leurs phases de conception.

Comme nous l'avons présenté dans l'état de l'art, les experts qui analysent les données pour produire ces modèles ont souvent besoin de visualisations qui permettent la remise en contexte de l'utilisation ainsi que la comparaison des différents usages. Ainsi, des visualisations bien construites aident les experts à diriger ce qui doit être automatisé dans une analyse par fouille de données.

Aussi, dans le chapitre suivant, nous présentons un outil que nous avons implémenté pour analyser et visualiser les données issues de notre modèle de récolte. Cet outil s'inspire de différentes recommandations tirées de la littérature, et permet différentes analyses dont des exemples sont présentés par la suite.

Chapitre 6. Exploitation des traces

Les traces générées par la méthode décrite précédemment ont une structure complexe et sont très détaillées, car toutes les actions de l'utilisateur y sont relevées. Cherchant une méthode générale pour l'étude de ces traces, nous avons développé une plateforme permettant d'analyser plusieurs caractéristiques. Comme nous l'avons montré précédemment, ces traces peuvent servir à toutes sortes d'analyses telles que la description des processus cognitifs ou l'explicitation des propriétés liées à l'utilisation d'une interface. Les outils présentés ici permettent toutes ces analyses à des degrés de précision variable.

Dans la première partie, nous présentons les principales fonctionnalités qui nous ont permis d'extraire des indices de comportements intéressants pour l'analyse cognitive. Certaines permettent d'analyser l'ensemble du parcours sur un seul graphique, alors que d'autres retracent l'activité des utilisateurs sur plusieurs graphiques successifs.

Dans la seconde partie, nous décrivons une nouvelle méthode de représentation des données issues de l'interaction homme-machine. Nous avons implémenté cette méthode qui permet la construction automatique d'un graphe à partir des données récoltés par notre méthode et facilite ainsi la lecture des traces pour des analyses cognitives de comportement utilisateur. Nous présentons également les fonctionnalités que nous avons implémentées pour l'aide à l'analyse. En effet, la visualisation représentant fidèlement et entièrement les traces très abondantes, l'analyse doit être aidée par un ensemble d'outils de sélection et de zoom performant.

1. Outils d'analyse

1. Techniques d'implémentations

Selon le niveau de modélisation que l'on souhaite faire de l'utilisateur, une quantité plus ou moins importante de traces est générée. Dans le cas de modélisation cognitive par exemple, à un niveau de traces très détaillées, des fichiers très volumineux

seront générés et leur exploitation en sera plus difficile. Aussi, nous proposons une technique de stockage en base de données classique qui permet de ne pas perdre l'avantage de la structure en arbre.

Structure en base de données

Afin de permettre des études sur les différents niveaux de traces, un stockage de niveau par table offre un bon compromis. On aura ainsi une table « session » contenant les descriptions globales décrites précédemment. De même, une table « contexte », une table « cible », une table « action » et une table « évènement » pourront être créées, contenant respectivement les descripteurs décrits précédemment. Il faudra prêter une attention particulière à la table « évènement » qui peut devenir très importante. Un ensemble de références croisées permettront de retrouver les correspondances entre les évènements et les cibles sur lesquelles ils portent. Ainsi, la table la plus volumineuse possèdera tous les identifiants nécessaires pour retrouver le contexte auquel il appartient (identifiant session, identifiant contexte, identifiant cible, identifiant action).

Une telle base de données possèdera une redondance naturelle de par le fait que plusieurs utilisateurs peuvent manipuler des écrans identiques. Elles connaîtront donc une optimisation par compression importante.

Organisation des requêtes

Les tables de données ainsi implémentées permettent d'utiliser au maximum l'avantage que procurent certaines bases de données, comme SQL, par rapport à la gestion de requêtes croisées. Ainsi, on pourra créer un ensemble de requêtes préformées pour relier les tables entre elles. Une requête sur un évènement particulier observé dans plusieurs traces devra contenir les identifiants de session, de contexte et de cible.

De même une requête sur une période particulière se fera à l'aide de la table des évènements (seule table possédant les dates précises des évènements). Cette requête, intervenant sur un niveau particulier de l'arbre de l'interface, se fera donc sur la table adéquate et la requête croisée se fera par les identifiants décrits précédemment. Une méthode simple permettra de donner la syntaxe SQL reliant les différents niveaux de tables entres elles.

D'autres bases de données intègrent directement des formats arborescents. Cependant, l'étude de ce type de traces pouvant s'effectuer à différents niveaux sémantiques, il n'est pas nécessaire de connaître toutes les propriétés des nœuds parents/fils pour analyser le parcours d'un utilisateur. Par exemple, pour qualifier la vitesse de frappe au clavier d'un utilisateur, il n'est pas nécessaire de connaître l'endroit

où le texte est saisi. Il en est de même pour des statistiques comportementales globales comme le nombre d'objets avec lesquels l'utilisateur a eu une interaction.

Mise en œuvre des techniques classiques sur notre modèle.

Le standard XML est maintenant largement répandu dans la communauté informatique et des outils très performants, comme le format XSLT et les parseurs DOM et SAX, permettent aux analystes de traces de concevoir des routines préprogrammés. Pour récapituler l'inventaire des analyses observées dans la littérature (voir Chapitre 2) et y ajouter notre modèle de traces nous pouvons faire les recommandations suivantes :

Modèle de synchronisation et de recherche : La structure XML que nous proposons conserve les propriétés temporelles linéaires liées aux évènements de l'interface. Il est donc facile de parcourir ces fichiers de traces et d'y rechercher des indices sur les plus hauts niveaux de description des évènements.

Nous avons testé la synchronisation des traces récoltées pour la recherche dans d'autres sources de données. La Figure 1 en annexe est un extrait d'une vidéo montrant un utilisateur réalisant un exercice de mathématique sur un manuel scolaire électronique. Sur la gauche de la figure, deux images issues de deux caméras montrent la scène sous deux angles différents. L'image principale, au centre de la figure, montre le parcours de l'utilisateur sur l'interface. Nous avons pu constater que la synchronisation et la recherche par élément atomique dans des traces est effectivement facile à mettre en œuvre. En sélectionnant dans la vidéo, les périodes de nos traces qui comprenaient l'évènement « *mousemove* », nous avons pu constater qu'elles correspondaient dans 60% des cas à des moments où l'utilisateur posait sa tête sur sa main, comme on le voit sur la figure.

Modèles de transformation : La structure en arbre stratifié ordonné pour enregistrer les évènements de l'interface constitue une transformation du flot d'évènements. En effet, le niveau que nous avons qualifié de « sémantique de l'action » constitue une sélection et une abstraction des évènements bruts de l'interface. De plus, notre approche de traces indépendantes du type des évènements permet d'effectuer des sélections préalables sur l'information que l'on souhaite conserver pour analyse.

L'aspect temporel d'apparition des évènements de l'interface a été conservé et nous y avons incorporé une structure permettant de mettre en évidence le contexte des

évènements. Cette mise en contexte peut très largement aider à combler le manque d'information dans le paradigme de l'impression évoqué dans le Chapitre 2.2.

Méthode de comparaison de séquences : La technique de visualisation proposée dans le Chapitre 6 constitue une comparaison de séquences puisque l'on place sur le même graphe plusieurs traces d'utilisations d'une interface. Les outils proposés apportent des possibilités supplémentaires pour comparer plus précisément les séquences.

Méthodes de comptage et de statistique : Il est très important pour effectuer les comptages et les statistiques de bien situer l'objet que l'on cherche à décrire. Notre structure propose trois descripteurs d'évènements de l'interface (contexte, cible et évènement). Il suffira de choisir le niveau de description (soit la table de la base de données correspondante) pour effectuer tous les calculs relatifs aux comptages et aux statistiques.

L'étude décrite dans [Damez & al. 2005] est un exemple utilisant des méthodes de comptage et de statistique. En effet, les descripteurs cognitifs qui y sont décrits sont issus de comptage et de statistique. Nous avons par exemple:

- Les temps : Combien de temps est nécessaire pour aboutir à une tâche, comme par exemple spécifier les arguments d'une commande ?
- Le déplacement de la souris : La distance entre deux clics de souris est-elle excessivement élevée ?
- La fréquence des commandes : Quelles sont les commandes les plus fréquentes et celles qui ne le sont pas du tout ?
- L'association des commandes : Quelles sont les commandes qui sont toujours associées à une autre commande ? Peuvent-elles être combinées ou rapprochées pour faciliter leur utilisation ?
- Les annulations : Quels sont les fenêtres de dialogue ou les actions qui sont fréquemment annulées ?
- Les fréquences de changement de périphérique : L'utilisateur est-il constamment en train de changer de périphérique, souris ou clavier, de façon inappropriée ? Quels sont les outils de l'interface associés à ces changements ?

Nous avons également implémenté deux outils de visualisation de statistiques. Les courbes de la Figure 2 en annexe représentent les traces de différentes personnes ayant toutes réalisé la même expérience. Il est alors très facile de distinguer les différents

types d'utilisateur, tant par la durée qu'ils ont mise pour faire l'expérience que par le nombre d'évènements que leur utilisation de l'interface a générés. On notera également que ce type de graphe peut servir à faire apparaître les temps morts dans l'activité de l'utilisateur. En effet, les deux courbes noire et jaune au bas de la figure montrent des activités d'utilisateurs particulièrement inactifs dans la période centrale de l'expérience.

En découpant les traces globales pour y faire apparaître les différentes tâches que les utilisateurs ont accomplies, nous pouvons également chercher à représenter et à comparer les caractéristiques de ces tâches. Ainsi, la Figure 3 en annexe représente les temps d'exécution des tâches les unes en fonction des autres. Notre implémentation permet de sélectionner un graphe représentant les temps de réalisation de deux tâches particulières et de zoomer afin de retrouver un ou plusieurs utilisateurs.

Méthodes de détection et caractérisation de séquences : La lecture d'un fichier XML en utilisant les outils des parseurs SAX permet de détecter très rapidement des sous-séquences prédéterminées. Faisant office de filtre sur les évènements, le parseur SAX ne parcourra qu'une fois le fichier de traces pour trouver des séquences. Les méthodes de caractérisation de séquences sont pour la plupart purement automatiques ou purement manuelles.

2. Fonctionnalités

Nous présentons ici les outils statistiques que nous avons développés et qui nous ont permis de constituer l'essentiel des propriétés remarquables à relever dans les traces pour nos expérimentations.

Suivi de l'activité

En utilisant les techniques de transformation du flot d'évènements décrit précédemment (voir Chapitre 2.2), nous avons implémenté une représentation des traces qui permet de visualiser le taux d'activité de l'interaction. Un exemple de cet outil est présenté dans la Figure 2 en annexe. Nous définissions le taux d'activité comme la fréquence d'apparition des évènements générés lors de l'interaction. Ce taux peut être général ou décliné selon le type de l'évènement. Une sélection préalable des évènements dont il faut relever les taux doit être effectuée.

Pour ce type d'analyses, deux catégories d'évènements peuvent être distingués : ceux qui signifient un changement d'état d'un élément de l'interface, comme *resize*, *mousemove*, *scroll*, etc. et les autres qui signalent une création ou une suppression d'un

élément dans l'interface. Les évènements de la première catégorie comportent une information supplémentaire qui doit figurer sur le nœud supérieur dans notre modèle de récolte de trace décrit au Chapitre 5. En effet, la cible de l'évènement doit changer son état suite à ce type d'évènements et un attribut doit nécessairement changer de valeur. Par exemple, l'évènement *resize* change les attributs *mousePosX* et *mousePosY*.

Les évènements propres à un périphérique peuvent se révéler intéressants pour une analyse cognitive. Ainsi, nous avons implémenté des mesures de vitesse de frappe au clavier, de distances parcourues avec la souris, et de nombre de défilement (évènements *scroll*) de la page opéré à l'aide de la molette de la souris, ou à l'aide des différentes touches du clavier.

D'autres indicateurs peuvent également se révéler significatifs d'une activité particulière d'un individu, comme le rapport entre le taux d'activité du clavier et le taux d'activité de la souris. Ces indicateurs peuvent être étudiés sur différentes périodes d'un parcours. Ces intervalles de temps seront d'autant plus pertinents s'ils correspondent à des évènements sémantiquement corrélés, significatifs de l'accomplissement d'une tâche. Un outil permettant de telles études est présentés ci-après.

Nous avons également utilisé cet outil pour représenter l'évolution de certaines mesures en fonction du temps. Ainsi, des similarités calculées sur les séquences d'actions des utilisateurs peuvent être représentées tout au long de l'interaction.

Graphique de comparaison 2 à 2

Pour étudier les techniques de comparaison de séquences et de comptage et statistiques évoqués précédemment (voir Chapitre 2.3), nous avons développé un outil permettant de comparer des caractéristiques des séquences. Ainsi sur un graphique à deux dimensions, nous pouvons comparer l'activité de plusieurs utilisateurs sur deux caractéristiques descriptives, ou sur deux périodes de temps particulières. C'est principalement cette dernière fonctionnalité qui nous permet d'obtenir une description particulièrement fine des comportements des utilisateurs.

La Figure 3 en annexe présente plusieurs graphiques qui permettent de comparer les temps d'accomplissement des sous-séquences représentant les sous-tâches d'une activité. Dans la figure présentée, chaque utilisateur est représenté sur chaque graphique par un point de couleur. La lecture de ce graphique nous a permis de caractériser les tâches particulièrement discriminantes, notamment par le temps que mettent les utilisateurs pour la réaliser.

L'outil que nous avons développé permet également de comparer deux caractéristiques d'un ensemble d'individus. Nous pouvons ainsi visualiser et comparer les taux d'activités du clavier et de la souris de tous les utilisateurs à un instant donné.

Clustering

Afin d'étudier le comportement des utilisateurs pendant leurs utilisations de l'interface, nous avons voulu représenter les traces traitées par des méthodes d'apprentissage artificiel incrémentales. Parmi ces méthodes, celles qui traitent les données deux à deux, sont généralement incrémentales. Cette démarche consiste à comparer chaque individu à tous les autres à chaque changement d'état. Dans notre cas, un changement d'état se manifeste par l'apparition d'un élément dans une séquence, et nous pouvons ainsi, en conservant les calculs effectués par la programmation dynamique, suivre les évolutions de chaque utilisateur les uns en fonction des autres.

Figure 18. Dendrogrammes créés de façon incrémentale

Pour représenter cette information, nous avons implémenté la méthode de représentation des données généralement utilisée pour le clustering hiérarchique ascendant décrit dans l'Algorithme 6 (cf. p.142). Les dendrogrammes ainsi créés à chaque temps (voir Figure 18) montrent l'évolution des proximités entre les individus selon une mesure de similarité ou de distance. Pour cette application, nous avons utilisé les similarités construites par l'algorithme du string kernel (cf. p131). Cette mesure permet de trouver les individus dont les séquences accomplies ont été globalement les

plus proches. En représentant cette information à chaque changement dans une des séquences et son impact sur les mesures de similarités avec toutes les autres séquences, les changements de structures de dendrogramme permettent de trouver les évènements qui montrent un changement de comportement.

Par exemple, si un individu effectue une série d'actions qu'aucun autre individu n'a encore effectuée, sa similarité avec les autres individus va baisser. Sur le dendrogramme, le regroupement avec les autres individus va apparaître plus tard (plus près de la racine du dendrogramme). Si au contraire, un individu fait une séquence d'actions qu'un des autres individus à déjà faite, sa proximité avec cet individu va augmenter, et sur le dendrogramme, le regroupement va apparaître plus tôt (plus près des feuilles sur le dendrogramme).

2. La visualisation de parcours

Nous proposons une visualisation en 2 dimensions des données récoltées dans les traces d'interaction présentées au Chapitre 5. Les évènements survenant dans l'interaction sont relevés directement dans une structure arborescente décrivant le contexte pendant lequel les actions sont effectuées. Cette représentation est automatiquement construite à partir des traces générées par les évènements de l'interface, les aspects temporels et séquentiels des parcours sont conservés de façon stricte. En nous inspirant des spécifications scénario UML décrites dans [Rumbaugh & al., 2004], nous proposons une approche qui permet de représenter l'ensemble des informations récoltées :

- Les objets de l'interaction sont représentés sur l'axe des ordonnées. Seuls les objets qui ont été manipulés par l'un des acteurs de l'interaction sont représentés.
- Les enchaînements des actions des utilisateurs sont représentés par des flèches. Celles-ci joignent deux objets cibles, signifiant que l'interaction passe d'un évènement produit sur l'objet à l'origine de la flèche à une interaction avec l'objet à la fin de la flèche. Ceci permet de suivre le parcours des interactions.
- Le temps pendant lequel se déroule l'interaction est représenté sur l'axe des abscisses. La longueur d'une flèche projetée sur l'axe horizontal représente la durée séparant les débuts des interactions entre l'objet à l'origine de la flèche et l'objet à la fin de la flèche.

La Figure 19 illustre la méthode de projection de l'interface tracée. Elle y présente également le parcours de deux personnes (l'un en trait plein, l'autre en trait pointillé). Lorsque l'interface change, suite à une navigation hypermédia par exemple, des nouveaux objets sujets à interactions peuvent être ajoutés sur l'axe des ordonnées. L'outil de visualisation effectue une projection des objets de l'écran en deux dimensions, sur un axe vertical en une dimension.

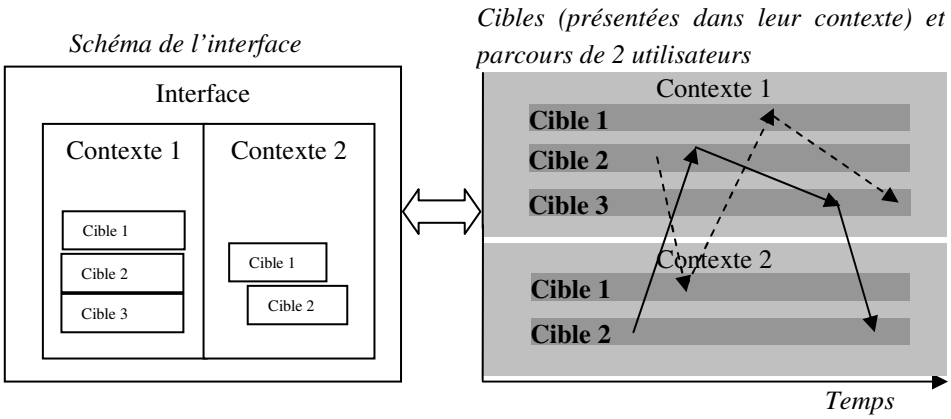

Figure 19. Méthode de projection de la visualisation

De la même façon que nous avons présenté notre modèle de récolte de traces sur des activités d'interactions générales (voir Chapitre 5), la visualisation proposée ici peut être utilisé de façon générale pour analyser des activités d'interactions.

Cette représentation des données est relativement simple, mais son implémentation peut poser quelques problèmes dus notamment à la quantité de données à traiter. Dans notre cas d'application portant sur l'étude des comportements utilisateurs sur une interface graphique, cet outil se révèle particulièrement performant. Nous présentons ci-après un extrait d'une visualisation pour compléter la description de l'outil de visualisation.

1. Exemple

Sur la Figure 20, les barres horizontales représentent les « Graphical User Interface » (GUI) qui ont fait l'objet de manipulation de la part des utilisateurs. Les barres en gris clair avec un label en italique sont les conteneurs (ou « contextes » décrits précédemment) ici les pages Internet :

- « HTML uri=file:///D:/Expelip6/Exp%E9rience_fichiers/jour0528.htm, » et

- « HTML uri=file:///D:/Expelip6/Exp%E9rience_fichiers/jour0308.htm, ».

Les barres de couleur gris foncé avec le label en gras sont les cibles qui ont fait l'objet d'une manipulation par les utilisateurs :

- « HTML uri=file:/// D:/Expelip6/Exp%E9rience_fichiers/jour0528.htm, »
- « TD »
- « A href=http://www. herodote.net/histoire05100.htm, »
- « FONT »
- « CENTER »
- etc.

Les flèches numérotées représentent les parcours sur ces cibles à raison d'un utilisateur par couleur de flèche et/ou par numéro.

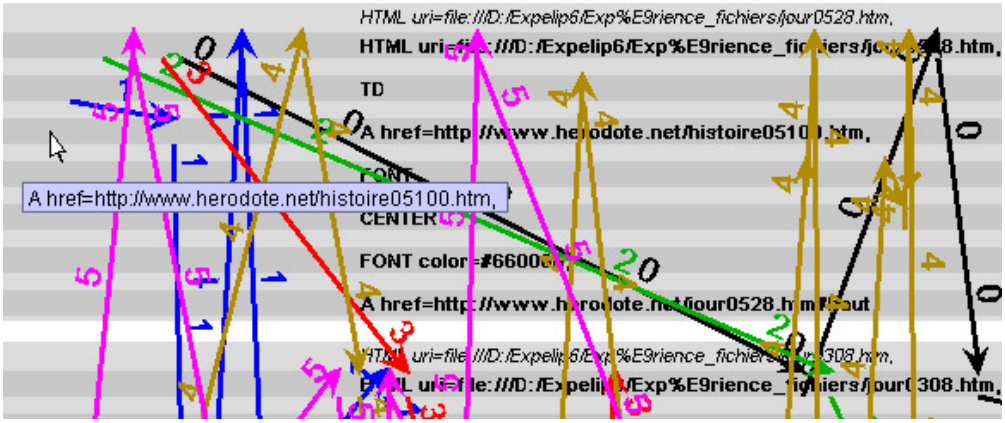

Figure 20. Extrait d'une visualisation

2. Liste des fonctionnalités implémentées

Une visualisation à partir des évènements de l'interface est souvent difficile à mettre en œuvre. En effet, les éléments de l'interface manipulés sont souvent nombreux, et une visualisation de données souple et adaptable est souvent demandée par les analystes. Nous avons donc implémenté plusieurs outils offrant une meilleure navigation dans la visualisation.

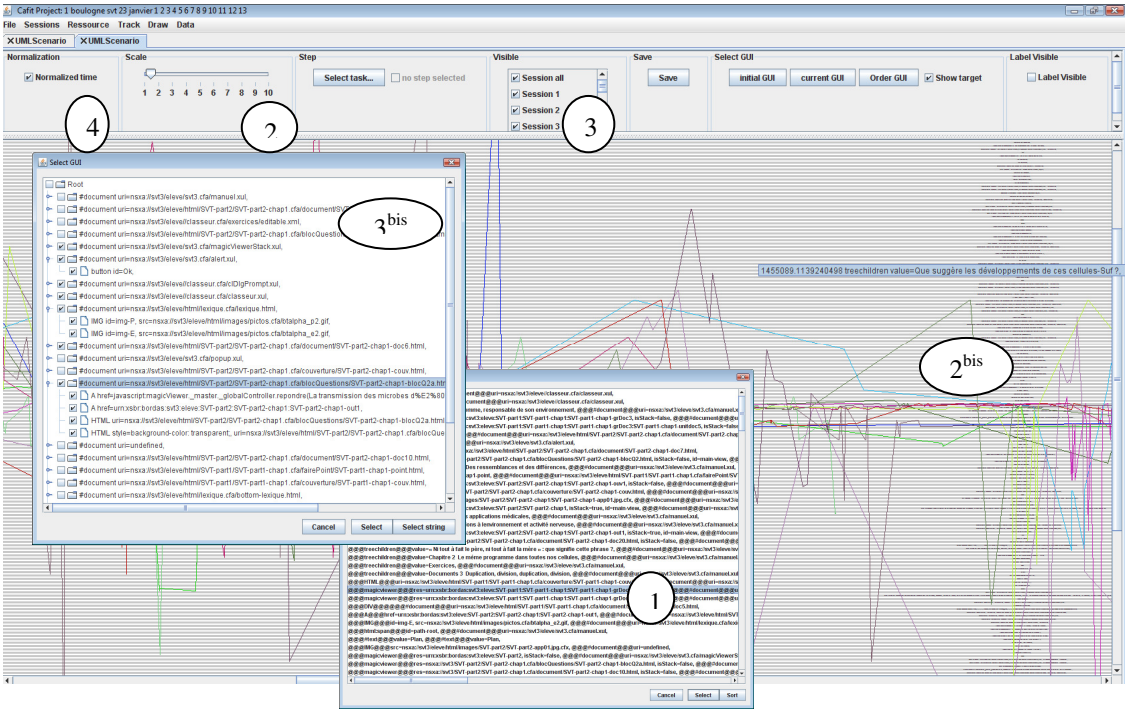

Figure 21. Logiciel de visualisation de parcours illustrant les fonctionnalités

L'ordonnancement

Comme nous l'avons décrit précédemment, il est possible de représenter l'utilisation de deux ou trois interfaces dans la même visualisation. Le nombre d'objets représentés est alors plus conséquent et la lecture plus difficile. Nous avons donc implémenté un outil pour ordonner manuellement les objets (GUI) apparaissant dans la visualisation (voir (1) de la Figure 21). Ceci permet la construction de graphes bien plus lisibles, comme le montre les analyses cognitives de parcours présentées dans le chapitre suivant qui ordonnent les cibles de plusieurs interfaces sur le même schéma.

En plus de l'outil manuel, nous proposons un algorithme pour ordonner de façon automatique les objets à représenter dans la visualisation. Cette problématique générale d'organisation linéaire minimum (*minimum linear arrangement*) d'un graphe est NP-Complet [Diaz & al., 2002]. Il s'agit de trouver l'organisation linéaire optimale des sommets $E = \{1,2,...,n\}$ d'un graphe rapprochant globalement, par une permutation π, les sommets i et j dont les arrêtes ont les poids w_{ij} les plus importants. C'est-à-dire celle qui minimise la somme :

$$\sum_{i,j \in E} w_{ij} \left| \pi(i) - \pi(j) \right|$$

Dans ce qui suit, par analogie, nous considérons un graphe dont les sommets sont donnés par les éléments des séquences. L'algorithme que nous proposons et qui est présenté ci-dessous peut s'appliquer sur plusieurs types de matrices de transition ou de similarité entre les sommets du graphe. On note *i* et *j* des éléments des séquences :

- matrice de transition conservant l'ordre de la séquence $M = (m_{i,j}) = |E_{i \to j}|$ avec $E_{i \to j} = \{(i,j) \in E \mid i \text{ est suivi de } j\}$, ie : chaque élément de la matrice contient le nombre d'occurrence de sous séquence *ij*.
- matrice de transition ne conservant pas l'ordre de la séquence $M' = (m'_{i,j}) = |E_{i \leftrightarrow j}|$ avec $E_{i \leftrightarrow j} = \{(i,j) \in E \mid i \text{ est suivi de } j \text{ ou } j \text{ est suivi de } i\}$ ie : chaque élément de la matrice contient le nombre d'occurrence des sous séquences *ij* ou *ji*.
- matrice binaire de transition $N = (n_{i,j}) = (if\,(m'_{i,j}))$, ie : chaque élément de la matrice contient 1 ou 0 selon si les sous séquences *ij* ou *ji* sont observées ou non.
- matrice de probabilité de transition $Q(i,j) = (q_{i,j}) = \left(m_{i,j} \Big/ \sum_j m_{i,j} \right)$, ie : chaque élément de la matrice contient la probabilité que l'élément *i* soit suivi de l'élément *j*.
- matrices de similarités issues de traitements séquentiels $K = (k_{i,j})$, ie : chaque élément de la matrice contient une valeur représentant une proximité séquentielle des éléments *i* et *j*.

Dans notre l'algorithme, l'ordre des n cibles est calculé avec une complexité en $O(n^3)$. Un des principaux avantages de cette méthode est dans le paramétrage de la relation d'ordre R qui permet de prendre en compte la structure arborescente des cibles sur l'interface tracée.

Entrée

- Matrice carrée symétrique $M = (m_{i,j})_{\substack{1 \le i \le n \\ 1 \le j \le n}}$ *de transition ou de similarité*

- R une relation d'ordre permettant de trier les colonnes $m_{i,*}$ *de M.*

On note $L_M = \{1,...,n\}$ *l'ensemble des valeurs d'indices possibles pour référencer un élément dans M.*

On note $\max_{L_M}(M)$ *la valeur maximale de M pour les indices couverts par* L_M*, c'est-à-dire :* $\max_{L_M}(M) = \left\{ m \in M \,\middle|\, \forall i \in L_M, \forall j \in L_M, m_{i,j} \le m \right\}$

Soit L la liste ordonnée de sortie de taille |L|

Initialisation:

$L = \left\{ i \in L_M \middle| \exists j \in L_M, m_{ij} = \max_{L_M}(M) \text{ ou } m_{ji} = \max_{L_M}(M) \right\}$

Tant que $|L| \neq |L_M|$

Soit $L' = \left\{ i \in L_M / L \middle| \exists j \in L_M, m_{ij} = \max_{L_M}(M) \text{ ou } m_{ji} = \max_{L_M}(M) \right\}$

Soit L_α et L_ω deux listes ordonnées

Pour chaque $i \in L'$

Si $m_{i,*}$ est plus proche de l'ensemble $\left\{ m_{j,*} \middle| \exists j \in L, j \leq |L|/2 \right\}$ *selon R*

Alors mettre i dans L_α

Sinon mettre i dans L_ω

Fin pour

Trier L_α (respectivement L_ω) selon R, de façon à avoir les éléments les plus forts en début (respectivement en fin) de liste

$L \leftarrow L_\alpha + L + L_\omega$

Fin Tant que

Algorithme 1. Algorithme d'ordonnancement

Les ordonnancements que produisent les différentes matrices en entrée de l'algorithme sont souvent très différents et permettent généralement différentes interprétations. L'ordonnancement des cibles représentées modifie alors la structure des contextes représentés dans la visualisation. En effet, l'ordre des cibles peut ne pas respecter celui induit par la structure arborescente de l'interface tracée. L'importante information que représente l'affichage des contextes nous a conduit à implémenter une solution faisant apparaître plusieurs fois le même contexte sur la visualisation. Une propriété logique peut alors s'appliquer : « Si deux cibles ordonnées côte à côte appartiennent au même contexte, alors ce contexte *parent* peut être représenté ». Cette propriété peut également être directement implémentée dans l'algorithme. La relation d'ordre *R* est alors fonction de la structure hiérarchique H des éléments représentés et des valeurs de *M* représentant les transitions ou les proximités entre ces éléments :

$$xRy \Leftrightarrow f\left(\Delta_H(x, y), d(m_{x,*}, m_{y,*})\right)$$

Où : $\Delta_H(x, y)$ *représente la distance entre les éléments x et y dans la structure H et* $d(m_{x,*}, m_{y,*})$ *la distance entre les individus x et y dans M*

Cette fonction peut s'appliquer dans le cas où plusieurs *contextes* englobant les uns les autres (comme dans les *Multiple Document Interface*) sont présents dans l'interface

tracée. Ces notions dépendent largement du type d'interface et de l'information que l'on souhaite représenter et l'analyste doit pouvoir personnaliser la structure affichée.

La sous séquence commune

La description des processus cognitifs impliqués dans la réalisation d'une tâche est l'un des principaux outils utilisés pour la conception de logiciels. Comme nous l'avons décrit dans le chapitre 1, la modélisation de tâches réalisées par des utilisateurs est souvent décrite par une décomposition en sous-tâches. Pour le cas de sous-tâches réalisées séquentiellement, nous avons implémenté un outil de sélection de séquences pour comparer visuellement la réalisation de ces sous-tâches (voir Figure 22).

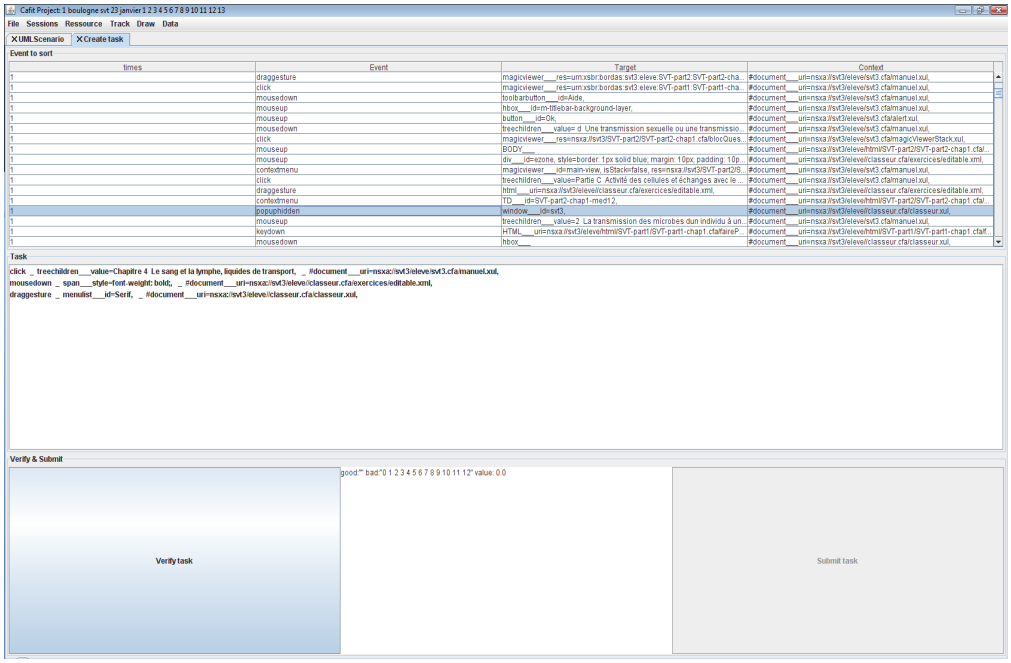

Figure 22. Outil de création d'une sous séquence commune

Cet outil énumère l'ensemble des objets de l'interface et permet de sélectionner les interactions avec les objets marquant le passage d'une sous-tâche à une autre. On peut ainsi comparer les différentes façons qu'ont les utilisateurs de réaliser une même sous-tâche. Un exemple de visualisation faisant apparaître ces sous-tâches est présenté dans la Figure 25 en annexe.

Le zoom

Les études portant sur les processus cognitifs impliqués dans la réalisation de tâches peuvent durer longtemps (plusieurs minutes à quelques heures). Aussi, afin

d'éviter une surcharge des nombres d'évènements apparents, deux fonctions de zooms ont été implémentées. La première permet d'étendre l'échelle de temps (voir (2) de la Figure 21). Les flèches représentant les parcours peuvent ainsi être plus horizontales, et la lecture du cheminement des actions plus efficace.

La seconde fonction est un zoom classique (voir (2^{bis}) de la Figure 21) qui permet tantôt de remettre l'ensemble de la visualisation visible à l'écran, tantôt une zone sélectionnée bien précise. Dans ce cas, la souris est le centre de la cible et le zoom s'opère sur la zone où est présent le curseur.

Le filtrage

Le nombre d'objets représentés dans la visualisation est généralement assez important. En effet, toutes les actions de tous les utilisateurs ainsi que tous les objets manipulés sont représentés. Pour observer plus précisément certains utilisateurs ou les interactions avec certains objets de l'interface, un outil de sélection a été implémenté. Cet outil permet de sélectionner quelques utilisateurs et/ou quelques objets de l'interface (voir (3) et (3^{bis}) de la Figure 21). Les éléments non sélectionnés sont alors opacifiés pour rendre la lecture et l'analyse plus facile. Cette option est comparable au filtre implémenté dans [Piktow & al., 1994] pour la visualisation de sessions d'internautes.

La normalisation du temps

Les séquences d'actions des utilisateurs peuvent également fournir de précieuses informations. On cherche alors à comparer le parcours global de chaque utilisateur pour en tirer des informations du type : « L'activité de la personne X est plutôt intense au début de l'expérience, alors que les autres utilisateurs ont une période dense vers la fin de leur session » ou « La personne X a consulté le document A au début de sa session, alors que la personne Y a consulté le même document A mais à la fin de sa session ». Un outil permettant de comparer les séquences d'actions a été implémenté à cet effet (voir (4) de la Figure 21). Le temps est alors normalisé et toutes les traces sont représentées sur tout l'axe horizontal de la visualisation.

Bilan sur l'outil d'analyse et de visualisation de traces

Ce chapitre a présenté des outils d'analyse, ainsi qu'un outil de visualisation de traces d'interactions homme-machine respectant le formalisme de récolte de traces décrit au Chapitre 5. Les outils d'analyse sont de complexité variable et des statistiques simples de comptage sont disponibles, autant que des méthodes d'apprentissage artificiel permettant des interprétations plus profondes des traces. La méthode de visualisation de données provenant de l'interaction homme-machine est directement exploitable pour des analyses cognitives. L'ensemble des données y est représenté automatiquement et sans prétraitement, contrairement à la plupart des systèmes de visualisation (voir Chapitre 3). Plusieurs outils ont été implémentés pour faciliter la lecture et l'analyse d'une telle représentation. En effet, les données représentées sont très nombreuses, car l'on peut y comparer différents utilisateurs, ainsi que l'utilisation de différentes interfaces. Le chapitre suivant propose deux analyses. La première compare les implications de l'utilisation de différentes interfaces pour la réalisation d'une même tâche. La seconde remet en perspective l'ensemble des interactions de plusieurs utilisateurs pour une analyse comparative des processus cognitifs.

Chapitre 7. Analyses à l'aide de l'outil de visualisation

Nous présentons ici deux expériences qui ont permis d'analyser précisément les processus cognitifs impliqués dans la réalisation de tâches en situation réelle. Ces analyses ont pu être réalisées à l'aide de la méthode de visualisation présentée précédemment au Chapitre 6.2. Dans les graphes présentés dans les parties suivantes, nous avons choisi de montrer la totalité du parcours, car ils sont plus compréhensibles pour l'interprétation cognitive.

1. Analyse de parcours sur différents types d'interfaces hypermédias d'un manuel scolaire numérique

Dans cette partie, nous présentons une expérience qui a été réalisée pour mesurer l'impact d'une nouvelle méthode de navigation dans les hypermédias proposée par [Renaud et al., 2006]. Le protocole de l'expérimentation est décrit en annexe (cf. Expérimentation 2 : « Etude de l'impact cognitif de l'utilisation de la commande sémantique »). Par notre méthode, la visualisation est construite automatiquement la visualisation à partir de données issues de l'interaction homme-machine. Elle permet d'établir rapidement un diagnostic de l'utilisabilité de cette nouvelle méthode de navigation.

1. Un nouvel outil de navigation dans un hypermédia

Dans le cadre d'un projet d'analyse sémantique des contenus pour un hypermédia pédagogique, un nouveau type de navigation a été conçu. La navigation dans les manuels scolaires numériques est encore fortement influencée par la présentation des manuels papier : elle s'opère de page en page selon un parcours organisé par les concepteurs (auteur et éditeur), dans le but de fournir des supports de cours en classe

pour le professeur. Ainsi, l'hypertextualité est réduite à l'arborescence des manuels papier, les documents étant organisés en chapitres et sous-chapitres.

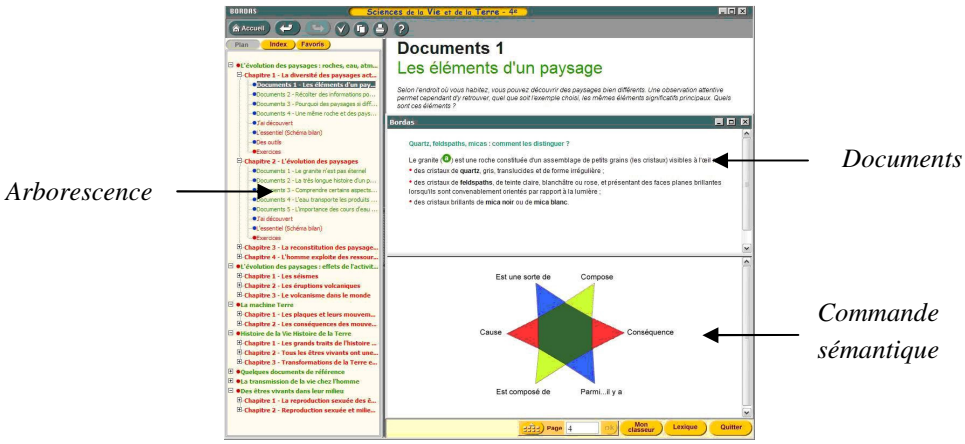

Figure 23. Illustration de la commande sémantique

Dans le nouvel outil de navigation (Figure 23), le parcours se fait de document en document, l'organisation de ces derniers étant calquée sur celles des concepts. Cette organisation est obtenue à partir des critères de la typologie des relations conceptuelles décrite par [Vignaux, 2002]. Trois types de relations ontologiques adaptées pour notre expérience (temporelle, catégorielle et granulaire) organisent les concepts sur 3 axes bijectifs, ce qui permet 6 types de déplacement qui sont à la fois des articulations logiques et langagières. Ainsi, les déplacements dans l'hypermédia sont à la fois des changements de documents et des « pas » conceptuels. L'espace de navigation peut être considéré comme un « espace problème » où chaque déplacement est révélateur d'un type d'activité cognitive de l'utilisateur (compréhension, désorientation, confrontation d'informations, etc.). Dans ce type d'expérimentation où le matériel cognitif est orienté vers l'apprentissage, les sujets font de nombreux traitements de hauts niveaux (mémorisation, inférences, déductions). Leur charge mentale est souvent saturée. Il est donc nécessaire de concevoir et de choisir un module de navigation qui offre à la fois les performances d'apprentissage les plus intéressantes et les utilisations les plus intuitives ou les plus économiques du point de vue des ressources attentionnelles. Ce module de navigation, « la commande sémantique », a été décliné sous 5 versions afin d'en évaluer chacune des composantes. Les méthodes classiques d'analyse des psychologues cognitivistes permettent facilement, dans ce type d'expérimentation, d'obtenir une estimation du gain global offert par le dispositif sur l'apprentissage. Par

contre, il est long et délicat d'arriver à une analyse fine sur les erreurs et styles de navigation des individus n'adoptant pas le nouveau système. Pour que les déplacements puissent être interprétés, il est nécessaire qu'ils soient contextualisés dans un parcours, ce qui nous ramène à la visualisation du protocole dans son ensemble, pour en saisir la dynamique.

2. Expérimentation

Les concepts de deux chapitres de géologie du manuel de « Sciences de la Vie et de la Terre » de 4ème des éditions Bordas ont été organisés selon quatre relations sémantiques : les relations temporelles (cause / conséquence), les relations catégorielles (est une sorte de / est composé de) et les relations granulaires (parmi … il y a / compose). Plusieurs versions du manuel ont été présentées à environ 20 étudiants : quatre versions contenant des commandes sémantiques avec différents types de ressources pédagogiques et de liens, et une version originale, sans commande sémantique. Le protocole de cette expérience est décrit précisément en annexe (cf. Protocoles d'expérimentations).

3. Résultats

La Figure 24 représente les parcours de sept sujets ayant réalisé la recherche d'informations avec deux interfaces différentes. On peut ainsi comparer plusieurs traces et observer les similitudes et différences dans les navigations des participants sur deux interfaces différentes.

Sur le schéma, les cibles et contextes du haut de la figure (zone A et D) représentent les pages visitées par les élèves manipulant la première interface. Les cibles et contextes du milieu (zone B), les plus larges, représentent les objets communs aux deux interfaces des manuels.

Ils représentent les menus, les outils, et l'arbre de la structure générale d'un manuel. Les cibles et contextes du bas de la figure (en C et E) représentent les pages visitées par les élèves manipulant la deuxième interface.

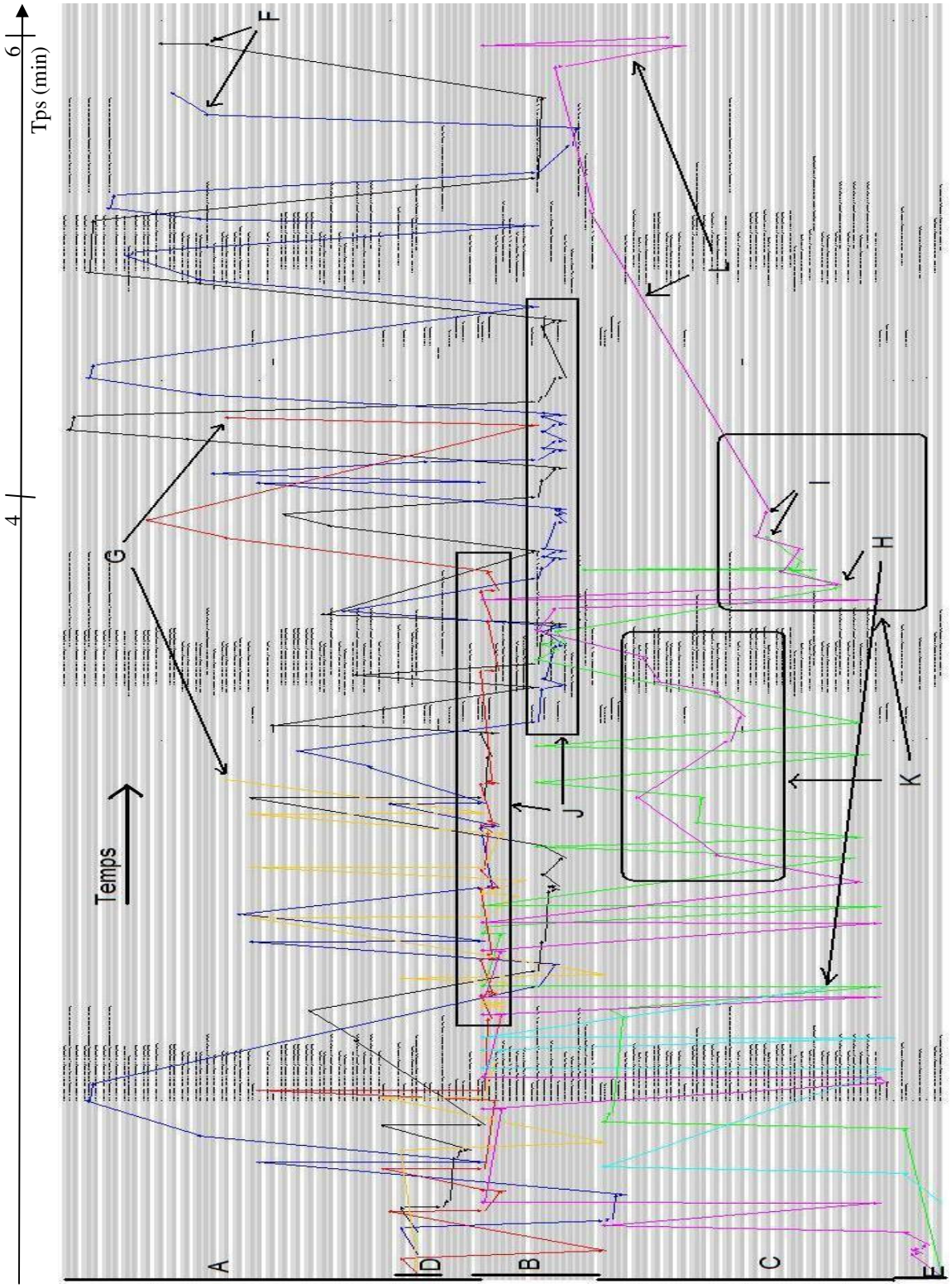

Figure 24. Visualisation des traces de 7 utilisateurs sur 2 interfaces différentes

La visualisation de l'ensemble des parcours permet de repérer rapidement (sans traitement, ni analyse des données) les évènements clés, les épisodes et les types de navigation :

- On voit que la deuxième interface est plus efficace que la première (temps total de réalisation de 4 minutes contre 6) et qu'elle réduit les passages dans l'arborescence (en B).
- Les élèves manipulant la première interface ont beaucoup plus interagi avec l'ancienne méthode de navigation que les élèves possédant la commande sémantique (en J)
- Deux participants de la première interface terminent le test (en G), en finissant sur la même page qui donne une information non pertinente pour la résolution de l'exercice.
- Deux autres parcours de la première interface menant à la réussite (en F) mettent un temps supérieur à celui des participants utilisant de la deuxième interface qui réussissent également (en H).
- Ces derniers réussissent ensemble en H (première question) et en I (deuxième question) après une utilisation du nouveau module de navigation découvert en K.
- Les erreurs de manipulation lors de l'expérimentation, telles que les résidus logiciels de trace non coupées en fin de passation (en L), sont également facilement détectables.

La visualisation des parcours met en évidence les phénomènes et les résultats remarquables des passations, ce qui facilite et accélère l'analyse qualitative. En plus de donner des informations sur la temporalité, les pages visitées et revisitées, elle permet d'étudier les phases des parcours, les types de parcours, leurs similitudes (ici les duos des participants présents en F et I). Cela permet d'envisager une nouvelle approche de détection et de caractérisation des styles cognitifs de navigation. Autre grand intérêt de cette représentation, elle permet de sélectionner les épisodes intéressants pour l'observation de données d'autres types, comme les vidéos et les analyses qualitatives les plus fines. Pour les psychologues cognitivistes, l'utilisation de ce logiciel offre un confort d'analyse sans précédent en permettant de multiplier les analyses très rapidement.

La visualisation présentée dans le Chapitre 6 a permis, avec cette expérimentation, la comparaison de l'utilisation de différentes interfaces. L'exemple présenté ici a mis en évidence l'effet d'un nouvel outil dans la navigation dans les hypermédias éducatifs.

L'analyse a permis de préciser les caractéristiques de comportements à détecter pour une fouille de données sur un plus grand nombre d'utilisateurs. Ces caractéristiques ont pu être mises à jour grâce à la comparaison avec l'utilisation de l'ancien système de navigation. Ici, à la fois l'interface, l'usage et le comportement des utilisateurs ont pu être représenté pour l'analyse.

2. Analyse de comportement et remise en contexte de l'utilisation de l'interface

Dans le cadre de tâches à réaliser sur un hypermédia clos, c'est-à-dire où chaque utilisateur doit impérativement accomplir une action, représentant la fin d'une sous-tâche et le début d'une autre, il est possible de visualiser des différences de comportements dues notamment au contexte créé par la réalisation des sous-tâches précédentes. Ainsi, nous appelons hypermédia clos, un environnement informatique hypertextuel limité à l'expérimentation. Afin de bien comparer le contexte d'utilisation d'une interface et d'isoler les comportements cognitifs discriminants, nous avons imposé une consultation dirigée. Autrement dit, l'utilisateur est seul face à la machine et il n'y a pas d'intervention extérieure pendant l'expérimentation. De plus, les tâches à réaliser sont prédéfinies et de nature séquentielle facilement discernable. Le protocole d'expérimentation est décrit en annexe (cf. Expérimentation 1 : « Consultation dirigée d'un hypermédia clos »).

[Brézillon & Tijus, 2005] ont proposé la représentation de « graphe contextuel » (voir chapitre 3) pour la mise en évidence de la granularité du contexte d'application de la tâche, et la mise au même niveau des éléments de comportement et des éléments contextuels. La visualisation proposée ici s'inspire de cette idée et automatise la construction d'un graphe.

1. Expérimentation

Afin de comparer les utilisations d'une interface par un groupe d'utilisateurs dans un hypermédia clos (c'est-à-dire où la navigation est contrainte à un espace défini), nous avons mis au point une expérience de consultation de deux pages Internet sur un navigateur, pour répondre à des questions. Le test est composé de 4 questions et s'accomplit en 6 sous-tâches séquentielles : la première des sous-tâches est la consultation libre des documents, les quatre suivantes sont l'affichage d'une question et la saisie de sa réponse, la dernière étant la fermeture de la page. Cette séquence définit donc la tâche à exécuter et le contexte de l'activité effectuée. Des captures d'écran de

l'interface sur laquelle s'est déroulée l'expérience sont présentées dans les Figure 1 etFigure 2 du protocole d'expérimentation 1 détaillé en Annexe.

Les questions ont été étudiées pour faire appel à différents processus cognitifs. Certaines questions demandent des renseignements très précis sur le contenu d'une des deux pages, alors que d'autres demandent une réponse portant sur le contenu ou la forme générale des pages. Une trentaine d'utilisateurs de différents niveaux d'expertises informatiques ont effectué cette expérience.

2. Résultats

Cette représentation doit être manipulée avec précaution tant les modifications qui y ont été apportées pour faire apparaître les sous-tâches transforment la lecture. En effet, l'échelle temporelle (horizontale) est ici différente pour chacune des sous-tâches, et la comparaison entre les traces ne doit pas se faire par une analyse des rythmes ou de la rapidité dépendant du temps. En conséquence, seul l'aspect séquentiel des actions réalisées par un utilisateur est conservé, et on pourra analyser le rythme d'une action sur une sous-tâche en le comparant avec le rythme des actions réalisées dans la même sous-tâche par le même utilisateur.

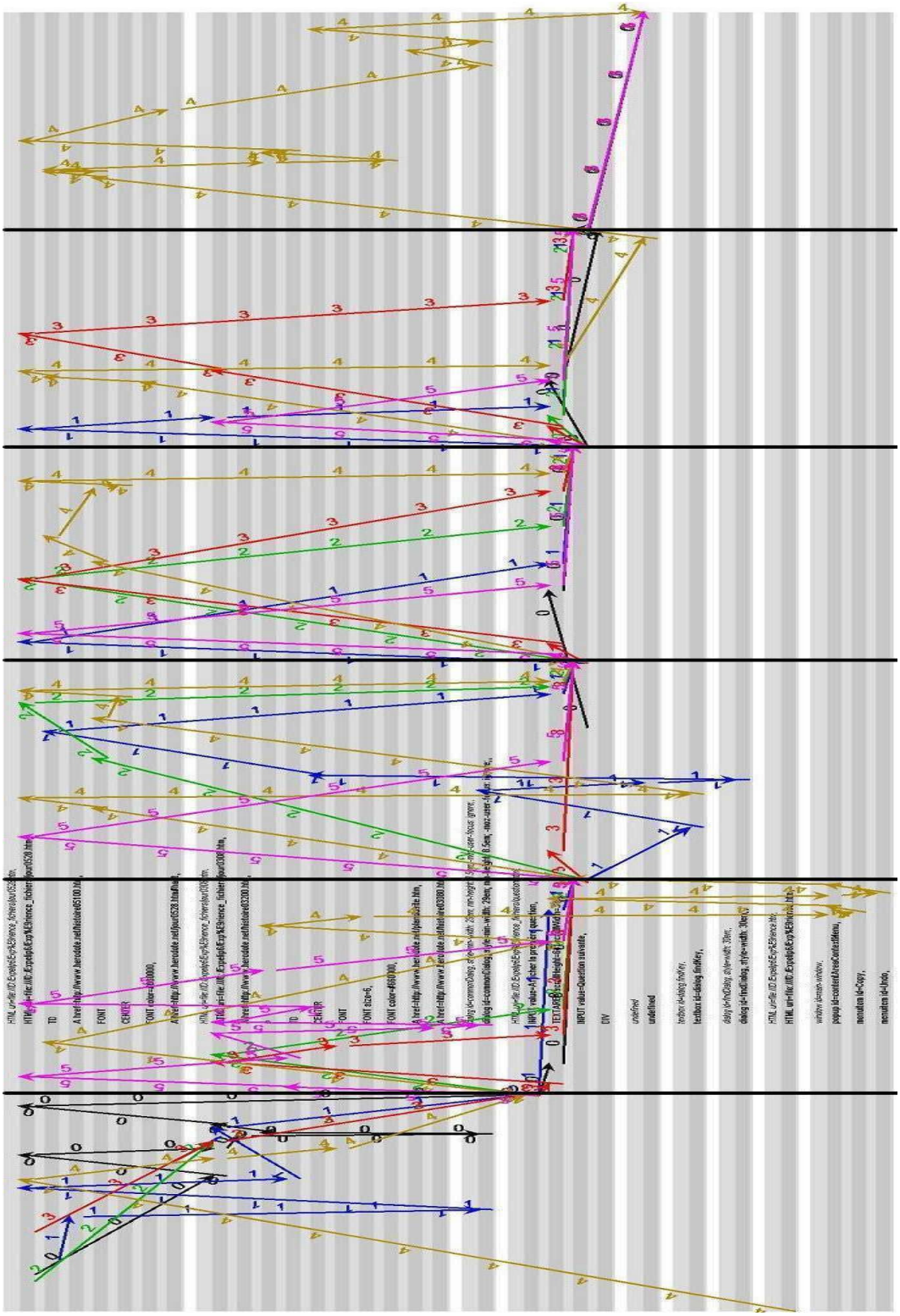

Figure 25. Visualisation du parcours de 6 utilisateurs sur un hypermédia clos représentant l'accomplissement de 6 sous-tâches

La visualisation proposée comportant la représentation de l'accomplissement des sous-tâches est présentée dans la Figure 25, chaque sous-tâche étant délimitée par les traits verticaux. Ces six traces d'utilisateurs font apparaître plusieurs propriétés :

- L'utilisateur 5 est le seul qui n'a pas consulté les documents avant l'affichage de la première question.
- L'utilisateur 4 est le seul qui a continué à lire les documents après avoir répondu à toutes les questions.
- Cet utilisateur semble également expert dans la manipulation de l'interface, puisqu'il a utilisé des outils, les cibles du bas de la Figure 7, qui représentent les fonctions de copier/coller (2ème étape) et de recherche (3ème étape) dans les menus du navigateur internet.
- L'utilisation de cette dernière fonction n'a d'ailleurs pas été fructueuse, puisqu'elle est suivie d'une deuxième consultation de la page, où la réponse a finalement été trouvée.
- D'une façon générale, beaucoup de clics ont été effectués par l'ensemble des utilisateurs sur des objets non interactifs de l'interface. La présence des cibles FONT, DIV, CENTER, etc. (objets du Document Object Model présentés dans la Figure 6) est le signe d'une volonté de sélection du contexte de ces objets, pour pouvoir accéder à la fonctionnalité de la molette de la souris, qui produit un déroulement de la page.
- La lecture détaillée de cette représentation a permis de constituer la liste des propriétés utilisées par un arbre de décision pour la classification automatique du niveau d'expertise dans la manipulation de l'interface (Damez et al., 2005) :

Général	- Durée - Nombre de changements de périphérique (clavier<-> souris) - Nombre de cibles visitées et nombre de changements de cibles - Nombre de contextes visités et nombre de changements de contextes
Souris	- Longueur de la trajectoire - Durée d'utilisation de la molette - Durée d'interaction avec les ascenseurs de navigation - Utilisation de fonctionnalité(s) avec le menu - Utilisation de fonctionnalité(s) avec le menu contextuel
Clavier	- Durée d'utilisation des flèches de navigation - Utilisation du copier/coller et des raccourcis - Vitesse de frappe

3. Bilan des analyses à l'aide de l'outil de visualisation

Les besoins des psychologues en matière d'outils de visualisation de données issues directement de l'interaction sont importants. Les possibilités d'analyse lors de la consultation d'un parcours entier d'un utilisateur sur une interface graphique sont nombreuses (cf. p.41). Dans ce chapitre, deux exemples d'analyse cognitive de parcours proposent des interprétations de cette visualisation. Les résultats de la première expérience ont permis de comparer l'utilisation de deux types d'interface et de concevoir un nouvel outil pour la pédagogie sur manuel scolaire numérique. L'exemple présenté ici a permis de préciser les caractéristiques de comportements à détecter pour la fouille de données sur un plus grand nombre d'utilisateurs. Ces caractéristiques ont pu être mises à jour grâce à la comparaison avec l'utilisation de l'ancien système de navigation hypermédia. Ici, les interfaces, les usages et les comportements des utilisateurs ont pu être représentés en même temps pour l'analyse.

L'autre expérience remet en contexte les actions des utilisateurs par une description séquentielle de la tâche effectuée. Une lecture approfondie a permis de proposer des propriétés cognitives intéressantes pour caractériser le niveau d'expertise de l'utilisateur et un système de prédiction de ce niveau a pu être implémenté (cf. p.154). Cette représentation peut être comparée à celle des graphes contextuels, car elle met en parallèle l'accomplissement d'actions effectuées par différents utilisateurs dans un même contexte. Nous avons pu représenter le contexte de la réalisation d'une tâche en fonction de la réalisation des tâches précédentes.

Conclusion de la deuxième partie

La personnalisation automatique de l'interaction HM doit être réalisée à partir de données représentant le plus fidèlement possible les actions et réactions de chacune des deux parties. La méthode de récolte de traces d'utilisation d'interfaces graphiques proposée construit automatiquement des données structurées et peut satisfaire tous les modèles utilisateurs évoqués dans l'état de l'art. Nous avons vu qu'elle est issue d'une observation directe des interfaces graphiques modernes, et sa structure permet de rendre compte aussi bien de la hiérarchie des processus cognitifs impliqués dans la réalisation d'une tâche que de la composition d'une interface. Les analyses applicables à ce type de données regroupent toutes celles utilisées précédemment lors d'analyses ponctuelles sur le comportement d'utilisateurs. De plus, des outils de visualisation de données sont souvent très utiles aux experts pour concevoir les outils de traitement automatiques.

Nous avons également présenté un outil d'analyse de ces traces qui comporte notamment une visualisation de parcours d'utilisateurs sur plusieurs interfaces graphiques, construite automatiquement à partir de données provenant de l'interaction HM. Plusieurs fonctionnalités qui permettent de faciliter la lecture de cette visualisation affichant l'ensemble des données récoltées ont également été présentées.

Deux exemples d'analyses cognitives de parcours proposent des interprétations de cette visualisation. Les résultats ont permis de comparer l'utilisation de deux types d'interface et de montrer les avantages d'un nouvel outil pour la pédagogie sur un manuel scolaire numérique. Une autre interprétation propose une remise en contexte de l'action par une description séquentielle de la tâche et permet facilement la reconnaissance de phénomène atypique ou isolé.

Ainsi, la détection de caractéristiques de comportements par une visualisation est réalisable. L'expert humain lit une représentation et cherche à qualifier ces caractéristiques. L'automatisation complète de ces processus est difficile dans la mesure où, potentiellement, l'intégralité de l'interaction homme-machine peut être caractérisée par un expert. Néanmoins des algorithmes d'analyse de données permettent de guider

les recherches. Nous présentons dans le chapitre suivant, des méthodes d'apprentissage artificiel pour la détection automatique de caractéristiques pertinentes du comportement qui pourraient amener à une qualification pertinente, par l'expert humain, de propriétés permettant la personnalisation de l'interaction HM.

Partie 3 - De l'apprentissage artificiel pour l'apprentissage humain

« L'esprit du débutant contient beaucoup de possibilités, celui de l'expert en contient peu »

Koan Zen

Introduction de la troisième partie

Peut-on utiliser les techniques d'apprentissage artificiel pour améliorer la qualité de l'enseignement ? Il ne s'agit pas de trouver le meilleur moyen d'enseigner telle ou telle discipline à des étudiants. Cette tâche est déjà remplie par les tuteurs intelligents qui s'appliquent à trouver les meilleurs formalismes pour l'apprentissage humain d'une discipline particulière. Il s'agit ici d'établir automatiquement les différences de comportements d'étudiants possédant différentes capacités d'apprentissage. Ces différences mises à jour, les enseignants peuvent personnaliser leurs cours à l'aide des caractéristiques de chaque étudiant fournies par le système.

Dans cette partie, le premier chapitre présente quelques notions de théorie pédagogique générale, pour une meilleure compréhension des implications de l'utilisation d'interface personnalisable pour l'enseignement. Ainsi, le design pédagogique adopté par le professeur doit être au centre de la personnalisation de l'enseignement. La notion d'affordance élaborée par [Gibson, 1977] aide à de mettre en évidence les mésinterprétations qu'une interface peut produire. Nous présentons ensuite quelques notions relatives aux styles d'apprentissage ainsi que les différences psychologiques entre des apprenants experts et des apprenants novices. Enfin, quelques standards élaborés par les professionnels de l'éducation sont présentés, car il nous semble judicieux de pouvoir les inclure dans le modèle de traces proposé précédemment.

Le deuxième chapitre expose les recommandations nécessaires pour l'implémentation d'un système de personnalisation automatique d'interface à vocation pédagogique. Ce système se décompose en deux phases, l'une d'apprentissage artificiel par le système fonctionnant « hors ligne » et l'autre permettant l'adaptation automatique de l'interface « en ligne ». La première phase est bien sûr la plus complexe et une méthodologie de reconnaissance de la tâche effectuée par l'utilisateur y est présentée.

Le dernier chapitre de cette partie revient sur les algorithmes d'apprentissage artificiel pour la mise en évidence de comportements d'utilisateurs apprenants, ainsi que leurs catégorisations ou leurs regroupements de façon automatique. Sans chercher à

construire un récapitulatif de toutes les méthodes existantes, nous proposons une taxonomie des méthodes applicables pour notre problématique et nous les présentons sous une forme simple pour leurs implémentations.

Nous présentons également deux analyses des données issues de la nouvelle méthode de récolte de traces d'interaction homme-machine présentée au Chapitre 5. Ces analyses explorent différents types de méthodes d'analyses de données et proposent des applications de personnalisation automatique d'interface. Nous avons particulièrement traité le cas de l'apprentissage sur une interface numérique et les adaptations permettant d'améliorer la pédagogie.

Chapitre 8. Théories pédagogiques

Les théories accompagnant les enseignants pour la conception de leurs enseignements sont nombreuses. Généralement issues de l'expérience, elles concernent à la fois la forme et le fond d'un cours. Comme nous l'avons vu précédemment dans notre état de l'art (voir Chapitre 4.3), la plupart des systèmes informatiques d'aide à l'enseignement sont dédiés à une discipline particulière et les concepts généraux de la pédagogie, tels qu'ils sont couramment dispensés dans l'enseignement classique, sont omis. Nous présentons donc ici quelques théories qu'il nous semble important d'inclure dans un système général de suivi pédagogique.

Nous commençons par présenter le design pédagogique qui constitue la partie la plus importante de la pédagogie. Il s'agit de l'organisation des contenus qui vont être présentés à l'apprenant. Cette notion ne peut pas être dissociée de l'affordance du contenu, présentée par la suite. L'affordance est une notion qui n'a pas été décrite spécialement pour l'éducation, et qui traite plus généralement des propriétés de l'interaction d'un objet avec son utilisateur. Nous pensons que cette notion est primordiale dans la conception d'un cours et plus particulièrement pour un système pédagogique informatique.

Les études menées par les psychologues cognitivistes depuis une vingtaine d'années ont permis la construction de modèles de comportement qualifiant le style d'apprentissage des individus. Plusieurs exemples de classification de ces styles sont présentés.

Une autre notion importante et plus générale est développée en psychologie cognitive. Il s'agit de la différence de comportement observable entre les experts et les novices, dont nous décrivons ici les propriétés que nous souhaitons reconnaître automatiquement.

Enfin, la dernière section présente des standards de communication que la communauté informatique a cherché à formaliser autour de la pédagogie. Cette notion

nous semble très importante à inclure dans un système informatique d'aide à la pédagogie.

1. Le design pédagogique

Le design pédagogique est le choix des méthodes d'apprentissage et de leur mise en application dans un contexte pédagogique dynamique. Cette préparation de l'enseignement se caractérise par l'application de la méthode scientifique, l'utilisation de techniques éprouvées lors de la réalisation des étapes du processus de planification de l'enseignement et l'application de principes de psychologie de l'apprentissage démontrés scientifiquement dans le ce but [Brien, 1981]. Le premier support de design pédagogique actuel est le manuel scolaire. En effet, l'auteur et l'éditeur sélectionnent dans un ouvrage les composants indispensables à l'apprentissage d'une notion par l'apprenant. Cette démarche doit prévaloir dans tous les types d'interfaces adaptables que l'on souhaite concevoir. On pourra ainsi personnaliser l'enseignement en modifiant le design pédagogique. Cette démarche est déjà adoptée par les systèmes de tuteurs intelligents évoqués précédemment. Elle nécessite une préparation très longue et complète des experts (professeurs) pour pallier toutes les éventuelles mésinterprétations d'un apprenant (élève).

Aussi, la démarche proposée ici, suppose la remise en contexte de l'apprentissage à l'école, où l'apprenant acquiert des connaissances au sein d'une classe. De même que pour le travail collaboratif présenté au chapitre 1, nous pensons que les mésinterprétations de certains élèves peuvent être corrigées et/ou rattrapées en observant le parcours pédagogique d'autres élèves ayant suivi le même cours par le même professeur, mais ayant eu un processus d'apprentissage réussi. La reconnaissance pertinente et de façon automatique de ces observations permettra aux professeurs de véritablement personnaliser l'enseignement.

2. L'affordance

L'affordance est le terme choisi par le psychologue Gibson [Gibson, 1977] pour définir *l'état latent de l'interface permettant à l'utilisateur de s'approprier plus rapidement la maîtrise de la connaissance*. Plus l'interface du support pédagogique est intelligente et moins les apprenants perdent leur temps à s'approprier l'outil dans le but de l'utiliser pour l'acquisition et la construction de connaissances. L'exemple classique est celui de la porte d'entrée unidirectionnelle. Si on doit pousser une porte pour

l'ouvrir, une plaque suffit. S'il faut ouvrir la porte en tirant, une poignée est nécessaire. Pourtant, certains établissements placent des poignées sur les deux cotés de la porte en y ajoutant les panneaux « Poussez » et « Tirer ». On pourra dire que les gens qui tirent la porte alors que le panneau « Tirer » est de leur coté présument de l'affordance de la porte (la poignée aura prévalu sur le panneau).

Remarque

Les évènements de l'interface récoltés lors de l'interaction HM nous apportent des informations très précises sur l'affordance. Ce concept a été inventé pour identifier les propriétés actionnables entre le monde et l'utilisateur. Pour Gibson, les affordances sont des relations. Lorsque nous mesurons les interactions d'un utilisateur avec une interface à partir des évènements de l'interface, nous mesurons l'affordance de l'interface telle que l'utilisateur la connaît.

Ainsi, à la fin d'un cours, si on récupère toutes les affordances de l'interface, on mesurera un taux d'interaction pour un élève. Il nous apparaît difficile de comparer différents taux d'interactions de manière globale, ou du moins, nous pensons que l'étude qui en sortira ne nous renseignera pas sur les réelles difficultés qu'a rencontrées un élève lors de l'apprentissage d'une notion. Par contre, si on observe le taux d'interaction d'un élève ayant des difficultés sur une notion, et celui d'autres élèves sur cette même notion, nous pensons que cela nous aidera à mieux comprendre ces difficultés. En effet, les composants de navigation dans une interface graphique doivent pouvoir présenter les contenus appropriés dans les meilleures conditions d'apprentissage. Si la manipulation de l'interface conduit à des erreurs de présentation, le taux d'interaction mesurant l'affordance peut nous renseigner sur les difficultés d'un élève. Cependant, ces indices peuvent révéler différentes propriétés du comportement. Par exemple, un mouvement précipité et intense de la souris peut signifier plusieurs choses, comme une incompréhension de l'état de l'interface (ordinateur bloqué), ou un désaccord avec le contenu d'une interface (l'utilisateur exprimant ainsi son sentiment d'insatisfaction).

3. Les styles d'apprentissages

Il existe plusieurs classifications des styles d'apprentissage, et la plus reconnue d'entre elle est celle de l'apprentissage par l'expérience, développée par [Kolb, 1976]. Elle distingue quatre styles d'apprentissage définis autour de quatre affinités de comportement pour l'acquisition de connaissances que sont l'expérience concrète (EC), l'expérimentation active (EA), la conceptualisation abstraite (CA) et l'observation réfléchie (OR). Ces quatre styles sont les suivants :

- Le divergent : EC & OR. Il préfère l'observation de situations concrètes sous des angles différents. Son mode de raisonnement est intuitif et pragmatique (génère des solutions). Il apprécie les séances de remue-méninges et le travail de groupe.
- L'assimilateur : OR & CA. Il s'approprie une quantité importante d'informations et les organise sous forme concise et logique. Il possède plus d'affinités avec les concepts et les idées qu'avec les personnes. Il est intuitif et réfléchi. Il préfère les lectures, les conférences et la réflexion approfondie.
- Le convergent : CA & EA. Il excelle à mettre les idées et les théories en pratique. Il préfère s'occuper des tâches et des problèmes plutôt que des questions sociales et interpersonnelles. Il est méthodique et réflexif, et préfère planifier et décider.
- L'accommodateur : EC & EA. Il effectue un apprentissage par l'expérience et la pratique en s'impliquant personnellement dans de nouveaux défis. Plutôt guidé par l'intuition que par la logique, il préfère travailler sur le terrain avec d'autres personnes.

Cette classification a été mise à jour sur des individus adultes en situation d'apprentissage. Les expériences menées sur des jeunes étudiants n'ont pas été aussi probantes que celles menées sur des adultes. En effet, consciemment ou non, l'homme apprend à apprendre au cours de son existence et il n'est pas surprenant que les styles d'apprentissage soient plus apparents chez les adultes que chez de jeunes étudiants.

La communauté s'est ensuite interrogée sur les différents styles d'enseignements appropriés à chaque classe de comportement [McCarthy, 1987]. Ainsi, la Figure 26 propose une catégorisation du type d'enseignement en fonction du type d'apprentissage. La représentation montre bien que les limites entre les styles d'apprentissage ou d'enseignement sont très floues et interdépendantes.

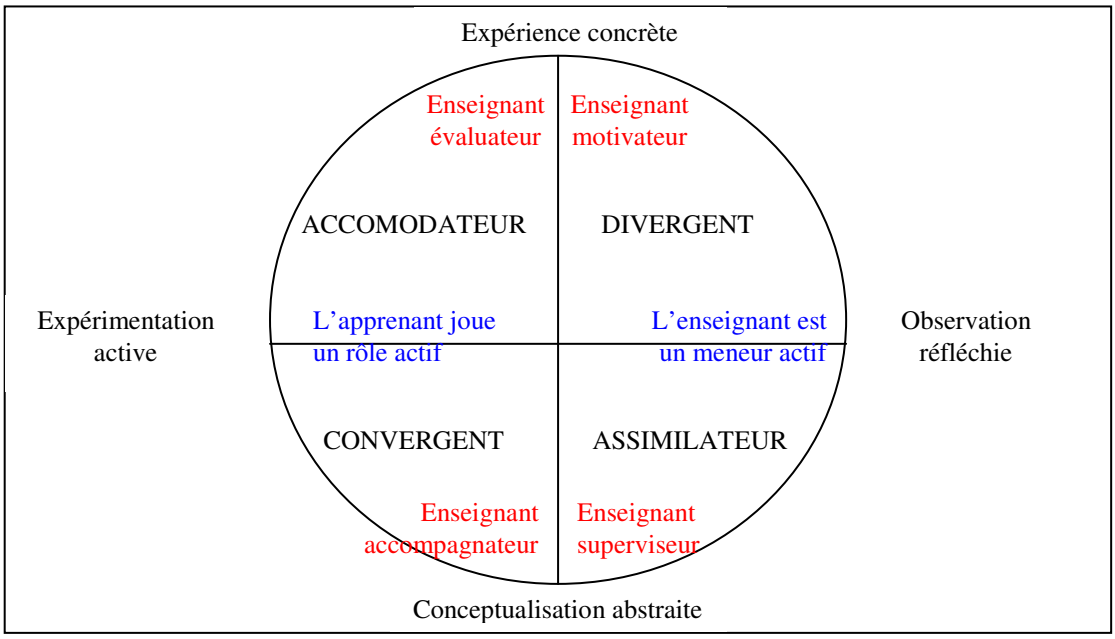

Figure 26. Styles d'apprentissage et stratégies d'enseignement [Kolb, 1976]

Une autre classification couramment employée pour décrire les préférences d'apprentissage a été présentée dans [Felder & Silverman, 1988]. Celle-ci a ensuite été révisée pour ne tenir compte que des aspects présents dans la Figure 27.

Style d'apprentissage		Style d'enseignement	
Sensoriel Intuitif	Perception	Concret Abstrait	Contenu
Visuel Auditif	Entrée	Visuel Verbal	Présentation
Inductif Déductif	Organisation	Inductif Déductif	Organisation
Actif Réfléchie	Traitement	Actif Passif	Participation de l'élève
Séquentiel Global	Compréhension	Séquentiel Global	Perspective

Figure 27. Les dimensions de l'apprentissage et de l'enseignement [Felder & Silverman, 1988]

Ces différentes caractéristiques établissant différents styles d'apprentissage ne doivent pas être confondues avec les conclusions des études cognitives menées sur le fonctionnement de la mémoire. En effet, certaines expériences ont prouvé que la façon dont la connaissance est manipulée influe directement sur sa mémorisation par le sujet.

Ces différentes affinités dans la manipulation regroupent généralement le visuel, l'oral, le verbal, le physique, le logique, le social et le solitaire.

4. L'expert et le novice

Dans un processus de transmission des connaissances, il existe toujours deux entités, un émetteur et un récepteur, l'expert et le novice. « Les experts classent les problèmes et s'appuient en général sur des principes de haut niveau correspondant à des relations fondamentales [Genter & Ratterman, 1991] qui les orientent vers la solution du problème. Ils disposent de connaissances vastes et organisées, alors que les novices, faute de connaissance et d'organisation entre celles-ci, s'appuient plutôt sur des similarités de surface » [Chi & al., 1981].

En cherchant à isoler la différence de comportement cognitif entre l'expert et le novice face à une tâche similaire, on pourra fournir une description précise sur les connaissances et les relations fondamentales mises en jeu dans la résolution du problème. Le fait de se référer à un tel bloc de connaissances structurées réduit le coût de traitement et permet d'effectuer un « bond cognitif » [Holyoak & Thagard, 1995].

[Bransford & al, 1999] ont établi six principales distinctions entre les experts et les novices dans l'accomplissement d'une tâche :

- Les experts relèvent les propriétés et les structures intéressantes de l'information que les novices ne relèvent pas.
- Les experts ont acquis une grande maîtrise sur la connaissance, qui est structurée de façon à refléter une profonde compréhension sur le sujet.
- Le savoir des experts ne peut pas être réduit à une liste de faits ou de propositions mais possède une représentation de l'applicabilité dans un contexte : le savoir est conditionné par un ensemble de circonstances.
- Les experts sont capables de retrouver les aspects importants de leurs connaissances de façon flexible avec peu d'effort.
- Bien que les experts connaissent leurs disciplines complètement, ceci ne garantit pas qu'ils puissent l'enseigner à d'autres.
- Les experts ont plusieurs niveaux de flexibilité dans leurs approches de nouvelles situations.

Fournir une analyse des processus cognitifs des utilisateurs est une préoccupation actuelle pour les chercheurs travaillant sur la personnalisation des interfaces adaptatives. Aussi, pour notre problématique, nous pensons qu'une observation très attentive des interactions HM pourra fournir les indicateurs de différences de comportements entre

les experts en apprentissage -les bons élèves- et les novices en apprentissage -les élèves ayant des difficultés-. Une automatisation de ce processus peut ainsi permettre de guider le novice par les bonds cognitifs qu'effectue l'expert.

5. Les standards

Plusieurs communautés scientifiques se sont penchées sur la question de la description de l'enseignement, et trois standards semblent subsister [Pernin, 2006]. Ainsi, une communauté informatique issue du traitement des données pour l'indexation et la recherche d'information a défini les propriétés des contenus pédagogiques afin d'aider les professeurs dans leur recherche de supports pédagogiques. Le standard LOM (pour *Learning Object Metadata*) fournit une taxonomie générale des contenus à vocation pédagogique.

Le standard SCORM pour (*Sharable Content Object Reference Metadata*) a été conçu par des industriels souhaitant fournir un support à l'enseignement à distance. Ainsi, cette norme permet de renseigner les propriétés des supports de la pédagogie comme l'utilisation faite par les élèves ou l'agencement possible de granules pédagogiques entre eux. Les acteurs de la pédagogie, élèves et professeurs, ainsi que leurs activités respectives sont décrits et l'on peut utiliser le LOM pour la description des objets manipulés lors de l'apprentissage.

Le dernier standard est apparu récemment, suite à des critiques signalant les manques de cohérence entre les deux précédents et l'enseignement tel qu'il est vécu par le professeur. Ainsi, le standard IMS-LD est soutenu par l'organisme *IMS Global Design Consortium* (précédemment appelé *Instructional Management Systems*), et définit le *Learning Design*. Cette norme établit les prérequis indispensables à la conception du design pédagogique évoqué précédemment. L'activité est mise au centre du processus d'apprentissage et permet la définition d'un objectif pédagogique précis, effectué par différentes personnes ayant différents rôles dans un certain environnement et en s'appuyant sur un certain nombre de ressources pédagogiques.

Ces standards ne sont, pour l'instant, que très peu implémentés dans des cas réels mais il nous semble important de les présenter car ils doivent, à terme, trouver toute leur place dans l'étude de traces d'apprenants dialoguant avec une interface HM personnalisable.

6. Bilan sur les théories pédagogiques

Nous avons décrit quelques recommandations générales pour la conception d'un système informatique à vocation pédagogique. Celles-ci ne sont pas toutes directement implémentables tant elles demandent encore un formalisme fonctionnel informatique. Comme nous l'avons vu dans le Chapitre 4, des catégories de comportement sont reconnaissables automatiquement. Les études portant sur la reconnaissance automatique en situation réelle du style d'apprentissage ne sont pas très abouties à ce jour, et nous participons à cette recherche dans [Damez & al. 2005].

Elles doivent néanmoins être présentes dans l'esprit de l'analyste cherchant à décrire le comportement d'apprenant. Elles doivent également être manipulées avec la plus grande précaution. En effet, les efforts fournis par la communauté scientifique pour décrire le fonctionnement d'un apprentissage réussi ne doivent pas cacher le danger que leurs conclusions peuvent apporter. Les champs d'exploration sont encore très vastes, et nous pensons qu'il serait une erreur de croire qu'un système informatique puisse toutes les implémenter. De plus, un processus d'aide à l'apprentissage automatisé ne doit pas être intrusif dans l'interaction HM. Nous pensons qu'il peut guider l'enseignant dans sa conception de cours personnalisé, mais ne peut en aucun cas se substituer totalement à lui.

Chapitre 9. Une plateforme pour la pédagogie

Les systèmes informatiques capables de s'observer et d'auto-évaluer l'utilisation de leurs usagers sont extrêmement complexes. En effet, la personnalisation automatique d'un logiciel nécessite la prise en compte de beaucoup de paramètres difficiles à modéliser. Les aspects liés au type d'utilisateur actuellement en train d'interagir, ainsi que l'activité modélisée sous forme de tâches doivent être pris en compte. Pour notre cas d'application, nous souhaitons que la personnalisation de l'interface soit essentiellement basée sur une étude des traces générées lors de l'interaction HM.

Ce chapitre présente quelques recommandations d'architecture logicielle pour l'implémentation d'un système automatique d'aide à la pédagogie. Le but de ce système est de fournir aux professeurs les moyens de personnaliser leurs enseignements sans leur ajouter de contraintes supplémentaires. En incluant les remarques précédemment formulées, nous souhaitons reconnaître automatiquement des caractéristiques de comportements liés au design pédagogique ou à l'affordance de l'interface. Ces caractéristiques doivent être liées aux styles d'apprentissage ainsi qu'aux succès de l'élève dans son processus d'apprentissage. Dans le cas d'un apprentissage non réussi, nous souhaitons fournir à l'enseignant les moyens de personnaliser son enseignement en lui fournissant une synthèse des caractéristiques de comportement observées sur des élèves ayant réussi leur apprentissage.

Nous décomposons le fonctionnement d'un tel système en deux phases : l'apprentissage hors ligne et la détection en ligne. Il s'agit donc dans un premier temps de construire un module capable de reconnaître le comportement d'élèves en situation d'apprentissage. Deux types de descripteurs sont alors utiles : ceux permettant de mettre en contexte la tâche en cours et ceux permettant de reconnaître les caractéristiques de comportement. Une fois la phase hors ligne réalisée, le programme de détection en ligne peut fonctionner en temps réel pour assister un élève dans son apprentissage ou un professeur dans son enseignement.

1. Apprentissage hors ligne

Une plateforme qui peut automatiquement personnaliser l'interface à un apprenant dans le but de participer à l'amélioration de la pédagogie doit être capable de traiter les traces issues des interactions HM. Plusieurs caractéristiques doivent être extraites de ces traces. Ainsi, il nous paraît important de connaître le niveau de l'utilisateur que l'on souhaite aider car nous pensons qu'il n'est pas utile d'adapter l'interface à un élève qui réussit d'ores et déjà son acquisition des compétences. Il faut donc être capable de discriminer les différents niveaux des apprenants sur des indices de comportement comparables. Une fois ce discernement effectué, il faut être capable de discriminer automatiquement les différences de comportement dans une acquisition des compétences, efficace par rapport à celui correspondant à acquisition inappropriée. Tous ces indices doivent être extraits d'un ensemble de traces d'utilisation de l'interface servant de base pour les méthodes d'apprentissage automatique (voir Figure 28).

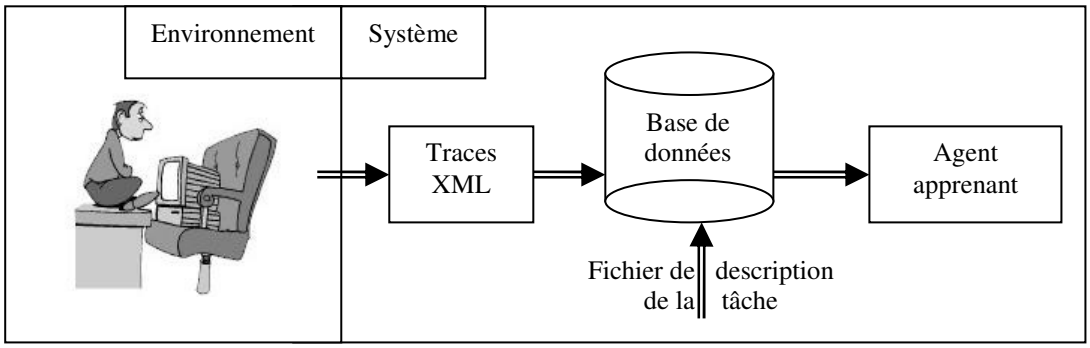

Figure 28. Phase d'entraînement hors ligne

Il est donc nécessaire d'avoir des descripteurs très précis de ces processus, et pour ce faire, nous proposons d'étudier les traces présentées dans le Chapitre 5.

1. Extraction des descripteurs cognitifs

Il est nécessaire de connaître les intentions de l'utilisateur pour mesurer son implication cognitive dans l'interaction HM. Au niveau le plus abstrait de description, on sait que l'utilisateur se sert de l'outil informatique pour accomplir une tâche : l'apprentissage d'une notion. Aussi est-il indispensable de connaître ces tâches et, lorsqu'elles ont le même but, de comparer les évènements qu'elles génèrent sur l'interface. Nous avons distingué trois principales méthodes pour l'extraction de tâches. Cet apprentissage artificiel sur un ensemble de traces d'utilisateurs fournira les descripteurs cognitifs nécessaires pour la génération des conseils.

Analyse des tâches

Cette analyse constitue un apprentissage artificiel qui regroupe les séquences d'évènements de l'interface. Ces regroupements constituent les tâches que l'utilisateur accomplit, à un niveau de détail plus ou moins élevé. Ce niveau de détail permettra de définir les tâches, les sous-tâches, et les sous-sous-tâches dont l'utilisateur n'a pas forcément conscience lors de la manipulation de l'interface. Le but de cet apprentissage est de trouver le plus petit niveau de détail de ces tâches commun à tous les utilisateurs. Ce n'est qu'en prêtant attention à ce plus petit niveau de tâche que l'on pourra mesurer les implications cognitives distinctes de chaque utilisateur. Nous avons distingué trois types d'apprentissage pour faire apparaître ces tâches dans un ensemble d'évènements de l'interface.

Méthode sémantique

Dans notre cas, les données des traces sont des évènements de l'interface, et comme décrit précédemment, elles sont sémantiquement très renseignées sur leur nature et leur source. Comme le fait remarquer très justement [Hilbert & Redmiles, 2000] dans son état de l'art sur l'extraction d'informations à partir d'évènements de l'interface, plusieurs actions peuvent avoir les mêmes buts, tout en étant de nature différente. Si on reprend le paradigme de l'impression présenté au Chapitre 2.2, des évènements différents peuvent représenter la même tâche de l'utilisateur. Comme les données que nous utilisons sont issues de nos traces sémantiquement enrichies, il est possible d'étudier cette sémantique pour connaître automatiquement les différentes façons d'atteindre un but. Les implications cognitives de chacune des méthodes employées pour réaliser une impression pourront être mises à jour dans le contexte global de la réalisation de la tâche par l'utilisateur.

Méthode séquentielle

Pour connaître les intentions d'un utilisateur, on peut aussi comparer ses actions à celles effectuées par un autre utilisateur ayant les mêmes intentions. Dans ce cas, si le scénario des actions des utilisateurs est le même, on pourra comparer les évènements issus de l'interface qu'ils manipulent pour trouver les actions communes et les différents processus cognitifs mis en œuvre pour résoudre la tâche.

Aussi, et comme suggéré dans [Ganascia, 2002], nous recommandons l'utilisation de la structure des traces pour extraire les motifs récurrents dans une forêt d'arbres stratifiés ordonnés. On peut également calculer le parcours minimal obligatoire en cherchant l'arbre couvrant maximum commun à toutes les traces (voir Algorithme 3).

En effet, les évènements de l'interface étant très nombreux, et les traces très volumineuses, il nous semble judicieux de profiter de la structure en arbre plutôt que chercher directement les motifs communs dans des séquences linéaires (la complexité algorithmique étant nettement inférieure dans l'utilisation des arbres stratifiés ordonnés).

Comme nous l'avons remarqué précédemment lors de la description des traces, les évènements de l'interface peuvent être contextualisés de façon précise en ajoutant plusieurs niveaux de « Contexte » dans la sémantique de l'évènement. Ceci ajoutera des nœuds dans les arbres de traces et la complexité de l'algorithme en sera encore réduite. Il est donc très important de bien prêter attention à l'ergonomie de l'interface, et s'assurer que le modèle d'« arbre de l'interface » n'est pas remplacé par d'autres mises en forme de représentation.

Cette méthode est la plus efficace, mais elle comporte un gros risque. En effet, si pendant la phase d'entraînement tous les utilisateurs font la même séquence de tâches, il est possible que les utilisateurs de la phase opérationnelle n'effectuent pas systématiquement ces mêmes tâches. Le système sera alors perdu dans la reconnaissance de l'activité en cours et ne fonctionnera plus correctement. Il est donc très important d'entraîner le système sur un ensemble très représentatif d'utilisateurs.

Méthode experte

Cette méthode est la plus facile à mettre en œuvre, mais elle a l'inconvénient de ne pas être automatique. Afin de connaître les différentes tâches des utilisateurs, on demande à un expert de regarder les traces, et/ou la tâche globale à effectuer. Cet expert doit pouvoir fournir un maximum de précisions sur les sous-tâches que les utilisateurs sont en train d'effectuer. Renseignées sémantiquement et/ou séquentiellement, ces tâches pourront alors être retrouvées dans les traces utilisateurs de notre ensemble d'apprentissage. Cette méthode est utilisée dans [Druganov & al., 2005] et permet, selon ses auteurs, la plus grande fiabilité d'utilisation.

Descripteur cognitif

Une fois le modèle de tâches correctement extrait de l'ensemble d'apprentissage, il faut approfondir l'étude des évènements générés par l'interface lors de l'accomplissement de ces tâches. Ces évènements peuvent être regroupés en motifs. Les propriétés liées à ces motifs, telles que leurs fréquences d'apparition, leurs dates d'occurrence dans une séquence, leurs niveaux de pertinence sémantique, entre autres, seront utiles. Les motifs sont construits automatiquement par une analyse globale des

évènements. A chaque évènement sont attribués au minimum un type, une cible et un contexte. Les motifs sont formés par regroupements successifs des évènements selon ces trois propriétés. Les motifs sont également formés des séquences d'évènements redondants. Nous ne notons alors que les séquences qui paraissent caractéristiques, et pour ce faire, le support de ces séquences doit être supérieur à un seuil jugé pertinent par un expert (la définition d'un support est donnée au chapitre 3). L'ensemble des attributs servant à notre module d'apprentissage contiendra les propriétés de ces motifs. On notera également pour ces motifs s'ils sont présents ou non dans la trace. C'est ce dernier attribut qui permettra de distinguer les descripteurs globaux des descripteurs particuliers.

Les descripteurs globaux seront ceux que l'on peut observer dans un très grand nombre de traces quel que soit le type de l'individu. Ils permettront, pendant la phase opérationnelle, de catégoriser l'utilisateur en cours d'utilisation de l'interface. Comme [Brusilovsky, 2001], nous pensons que tout utilisateur exprime son niveau d'expertise de façon naturelle lorsqu'il utilise une interface.

Les descripteurs particuliers auront la propriété de n'être présents que dans les traces dont on sait que l'utilisateur qui les a générées est expert. Ce sont eux qui vont, après interprétation, permettre de donner un conseil aux utilisateurs dont on sait qu'ils sont novices.

Les motifs extraits n'auront évidemment pas tous la même importance pour notre classificateur, et il pourra être judicieux d'étudier la quantité d'informations présente dans toutes ces variables.

2. Le module d'apprentissage

Le module d'apprentissage doit être capable de discriminer les utilisateurs novices des experts. Il doit également être capable de fournir une description sur la façon dont cette discrimination a été faite. Cette description sera ensuite utilisée lors de la phase en ligne pour détecter s'il y a lieu de fournir une aide et si cette aide sera effectivement appropriée. Par exemple, [Damez & al. 2005] proposent l'utilisation des arbres de décision comme type d'agent apprenant. Nous pensons qu'il est aussi important de faire intervenir un maximum de descripteurs cognitifs dans cet agent. En effet, il s'agit de ne pas classifier trop précipitamment notre utilisateur sous peine de nuire à l'accomplissement de la tâche en fournissant des aides non appropriées. De par ses facultés de souplesse et de précision, il nous paraît judicieux d'utiliser les propriétés qu'apporte le flou à la logique traditionnelle [Zadeh, 1965]. Comme nous le décrivons dans le Chapitre 11, nos expérimentations ont été réalisées en collaboration avec Thanh

Ha Dang et son système de construction d'arbres de décision flous [Dang, 2007], et il apparaît que les résultats sont souvent meilleurs qu'avec des arbres de décisions classiques [Damez & al. 2005].

Toutefois, ce type d'algorithme d'apprentissage artificiel n'est pas très approprié au traitement des données séquentielles, telles que celles issues des interactions HM. Il faut au préalable construire les motifs à la main, puis former une matrice des propriétés comme décrit au Chapitre 10.2. Les algorithmes décrits dans le Chapitre 10.1 semblent plus appropriés lorsque l'on souhaite automatiser le processus d'extraction de descripteur.

2. Détection en ligne

Nous savons, pendant la phase d'entraînement, qu'un certain nombre de descripteurs cognitifs seulement entre en jeu lors de la discrimination de l'utilisateur. C'est pendant la phase opérationnelle qu'il s'agit d'observer très attentivement ces descripteurs. En effet, pour produire un système « en ligne », autrement dit qui fonctionne en temps réel, le système doit être capable de classifier l'utilisateur le plus tôt possible de façon à ne pas fournir un indice de résolution de problèmes trop tard.

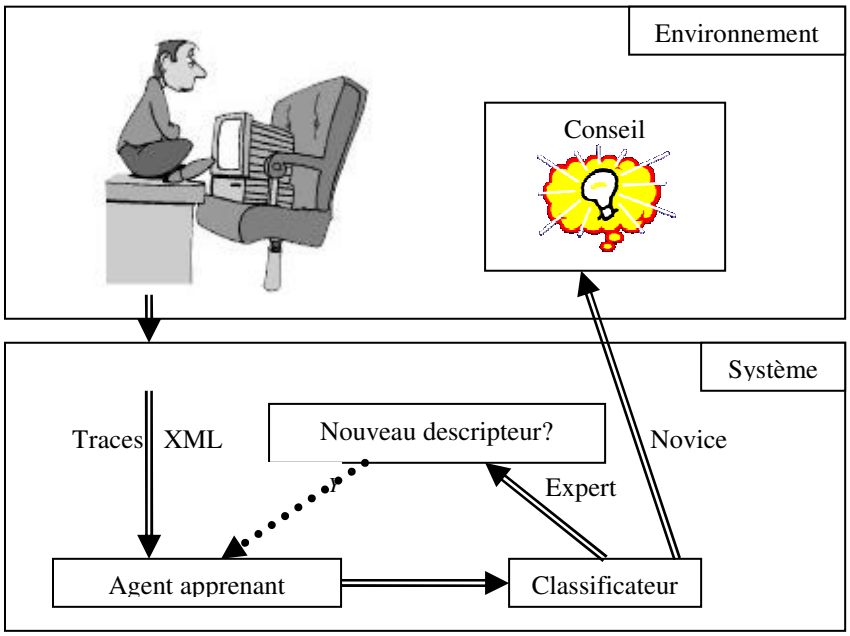

Figure 29. Phase opérationnelle en ligne

Notre système de récolte de traces ne fonctionne plus alors de la même façon que précédemment (voir Figure 29), et le traitement des données doit se faire au fur et à mesure qu'elles arrivent. De même, les conseils à donner qui sont découverts lors de la phase d'entraînement doivent être prêts à être fournis lors de l'utilisation de l'interface.

On pourra également, pendant la phase opérationnelle, continuer à récolter des traces d'utilisation et activer des indices de pertinence des conseils. Ces traces pourront servir à rebâtir le module d'apprentissage, et si c'est le cas, trouver de nouveaux descripteurs et de nouveaux conseils à donner. Il faut alors prêter une attention particulière à recenser les conseils donnés par le système dans le fichier de traces. On pourra ainsi véritablement mesurer l'impact de celui-ci. Il faut alors vérifier dans la trace s'il a été effectivement suivi et s'il a effectivement fonctionné. Dans le cas contraire, l'indice de pertinence du conseil pourra être diminué. Une base de données d'indice de pertinence des conseils promulgués pourra ainsi être construite pour chaque utilisateur ou chaque catégorie d'utilisateurs.

3. Discussion et bilan sur la plateforme pour la pédagogie

Nous avons présenté une plateforme pour la pédagogie permettant l'automatisation d'un processus d'aide à l'apprentissage. Ce système est collaboratif puisqu'il est capable d'utiliser automatiquement les caractéristiques de comportements d'individus réussissant leur apprentissage pour les reformuler sous forme de conseils aux élèves éprouvant des difficultés.

Aussi, nous avons décrit ici un système entièrement automatique. Il nous paraît évident qu'un tel système ne peut pas fonctionner entièrement automatiquement sans un certain entraînement (l'apprentissage hors ligne doit être important). De plus, plusieurs fonctionnalités doivent encore être étudiées. Par exemple, il faut préciser les modalités de transformation d'une observation d'action experte, qualifiée de descripteur particulier, en conseil utilisable par un novice. Une classification des différents types de conseils envisageables et leurs pertinences relatives dans le parcours pédagogique seraient également une étude intéressante. Dans le cas de motifs séquentiels par exemple, il est important de savoir exactement quand il faut fournir le conseil sous peine d'entraver le parcours de l'élève. Des critiques ont été formulées quand à la distinction entre la reconnaissance de la tâche en cours (à l'aide des méthodes sémantique, séquentielle et experte) et la formulation des descripteurs cognitifs nécessaires pour la classification de l'utilisateur. Cette remarque est également apparue

dans les modèles de tâche utilisés en psychologie cognitive, comme GOMS, et la distinction entre un but et une tâche est laissée à l'appréciation de l'expert. Autrement dit, cette différence représente le degré de détail que l'on souhaite modéliser. Nous pensons qu'il en va de même pour nos descripteurs cognitifs. L'élève ne représentant pas une bonne activité d'acquisition des compétences ne pourrait-il pas simplement se tromper de tâche ?

L'enseignement et l'apprentissage sont des activités cognitivement extrêmement complexes et nous pensons qu'un système automatisé qui permettrait à une classe d'effectuer un apprentissage personnalisé ne peut être efficace que si le système a été entraîné sur un même design pédagogique et sur une même interface, sur une grande quantité d'apprenants. Ceci est pratiquement irréalisable. Cependant, et comme nous l'avons introduit dans ce chapitre, le professeur cherchant à personnaliser son enseignement peut, à l'aide de ce système, consulter des propositions de remédiations générées automatiquement. Son œil de pédagogue expert peut alors être utilisé pour sélectionner la personnalisation qu'il souhaite.

Dans le chapitre suivant, nous présentons des algorithmes d'apprentissage artificiel qui nous paraissent appropriés pour la reconnaissance du comportement des utilisateurs à partir de traces de l'interaction HM.

Chapitre 10. Apprentissage artificiel

L'apprentissage artificiel est le terme désignant la reconnaissance automatique par une machine, de concepts généralement décrit par l'homme. Par exemple, un ordinateur peut différencier automatiquement une chaise d'un fauteuil si les caractéristiques fournies pour décrire ces deux objets sont suffisamment pertinentes (seul le fauteuil possède des accoudoirs latéraux). Les techniques utilisées sont de plusieurs formes et manipulent souvent des données de types différents. Le cadre de cette étude est l'analyse des fichiers de traces décrites précédemment (voir Chapitre 5). Ces traces sont générées automatiquement lors de l'utilisation d'une interface, et forment un historique du dialogue HM. Nous présentons donc ici des techniques permettant d'analyser ces données pour reconnaître automatiquement des caractéristiques d'utilisations.

Plusieurs familles de techniques existent à ce jour, et nous rappelons ici le vocabulaire usuel. L'utilisation de ces termes amène encore la communauté scientifique à les réviser. Aussi, les termes utilisés dans ce mémoire respectent les définitions suivantes :

Individu : Elément que l'on cherche à qualifier. Dans notre cas, il s'agit d'une trace d'un utilisateur manipulant une interface.

Propriété : Caractéristique descriptive d'un individu. En général, les propriétés étudiées sont les mêmes pour tous les individus de façon à comparer des choses comparables.

Classe : Ensemble d'individu partageant des propriétés. Le nom donné à une classe permet la description synthétique d'un concept.

Classification : Assignation d'un individu à une classe. Une classification suppose que l'on connaisse, avant traitement, la classe à reconnaître. On parle également de classification supervisée.

Clustering ou regroupement : Assignation d'un individu à un groupe d'individus partageant les mêmes propriétés. Le clustering souligne le fait que l'on ne connaît pas avant le traitement des données, les propriétés partagées. On parle également de classification non supervisée.

Les traces sont des données de nature séquentielle. Cependant et comme pour le traitement automatique du texte, les approches modernes utilisent un traitement des

données sous la forme des « sacs de mots », c'est-à-dire que les propriétés inhérentes aux séquences sont ignorées et le traitement des données est purement statistique. Nous avons également observé cette évolution pour les méthodes d'analyse de traces dans la littérature. Nous avons donc réparti les méthodes en deux familles distinctes. Celles qui manipulent les données en incluant la structure séquentielle et celles qui traitent des données matricielles. La liste d'algorithmes présentée ici n'a pas prétention à être exhaustive. Il s'agit des méthodes qui nous paraissent les plus appropriées pour être implémentées dans le système décrit dans le Chapitre 9 et répondre aux exigences évoquées dans le Chapitre 8.

1. Algorithmes avec traitement séquentiel des données

Les techniques algorithmiques qui manipulent directement les séquences sont maintenant nombreuses, et sont, pour la plupart, applicables à notre cas. Nous proposons ici trois catégories : celles qui parcourent chaque séquence individuellement, celles qui comparent les séquences par paires, et celles qui parcourent toutes les séquences plusieurs fois.

Quelle que soit la technique employée, l'utilisation de la structure en arbre stratifié ordonné des traces décrite dans le Chapitre 5 permet une réduction de la complexité algorithmique. En effet, toutes les techniques décrites ci-après sur les séquences sont applicables directement aux nœuds de plus haut niveau de l'arbre. Que les traitements soient individuels, deux à deux, ou sur un ensemble de séquences, la comparaison des éléments composant les séquences est utilisée. Si cette comparaison se révèle positive, autrement dit les éléments comparés sont identiques, alors le traitement peut être itéré sur les fils des nœuds comparés, eux-mêmes séquentiellement ordonnés [Ganascia, 2002]. Si la comparaison se révèle négative, autrement dit les éléments comparés sont différents, alors le traitement n'a pas besoin d'être effectué sur les fils des nœuds comparés. Des calculs inutiles sont ainsi évités.

1. Traitement individuel des séquences

Nous appelons « traitement individuel » sur les séquences, les méthodes algorithmiques qui ne nécessitent pas plusieurs passages sur les éléments composant les séquences. Chaque séquence est ainsi parcourue individuellement, et lors de ce passage, des informations pertinentes sont relevées. Certaines méthodes demandent un travail manuel préalable important, comme pour les automates qui nécessitent un paramétrage

par un expert. D'autres sont plus souples pour des utilisations automatiques, mais agrègent l'information sous une forme qui n'est plus intelligible, comme les modèles de Markov à états cachés.

Automate fini

Introduits pour la reconnaissance de langage formel, les algorithmes utilisant les automates finis sont très bien adaptés aux traitements de détection et de caractérisation de séquences.

Un automate fini est constitué d'un ensemble fini d'états et d'un ensemble de transitions possibles entre ces états. La variable d'entrée est un mot (séquence de caractères) que l'automate parcourt en passant d'état en état et suivant certaines transitions. Ainsi, un automate fini forme généralement un graphe orienté étiqueté dont les états sont les sommets et dont les transitions sont représentées par les arêtes étiquetées (voir Figure 4 du paragraphe 5).

Définition :
Un automate fini est un quintuplet $(Q, \Sigma, \delta, Q_0, F)$ où Q est un ensemble fini d'états, Σ est un alphabet fini, δ est une fonction de transition, c'est-à-dire une application de Q x $\Sigma \rightarrow 2^Q$, $Q_0 \subseteq Q$ est l'ensemble des états initiaux et $F \subseteq Q$ est l'ensemble des états finaux ou d'acceptation.

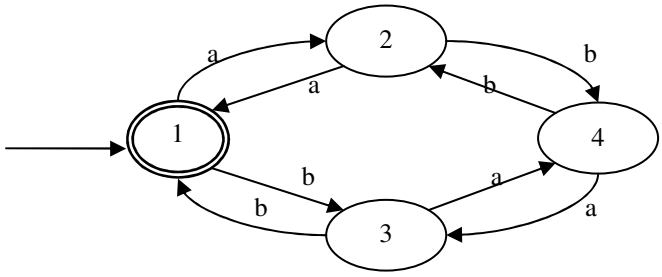

Figure 30. Un automate fini acceptant les phrases composées d'un nombre pair de a et pair de b.

Il existe plusieurs types d'automates finis permettant la mise à jour de différentes propriétés sur les séquences. Ainsi, on distingue les automates finis déterministes (DFA pour *Deterministic Finite-state Automata*) des automates finis non déterministes (NFA pour *Non-Deterministic Finite-State Automata*). Les premiers ont une vocation d'identification d'une propriété sur des séquences (voir Figure 30), alors que les seconds, plus souples, permettent la reconnaissance d'un langage, ensemble de propriétés que peuvent vérifier plusieurs séquences.

Beaucoup de résultats théoriques ont émergé suite aux travaux effectués sur la manipulation des graphes. Une synthèse est décrite dans [Cornuéjols & Miclet, 2002]. Cependant, l'apprentissage automatique sur des séquences n'est possible que dans les cas supervisés et pour des classements binaires.

Ainsi, [Jéron & al., 2006] propose un cadre formel permettant de décrire de manière uniforme une classe importante de problèmes de diagnostic. Ces systèmes sont tout à fait adaptés à la détection d'anomalies de comportement dans les séquences d'évènements utilisateur.

Modèles de Markov Cachés

Un modèle de Markov Cachés (MMC), noté $\Lambda = (A, B, \pi)$, se définit par :

- *Ses n états qui composent l'ensemble* $S = \{s_1, s_2, ..., s_n\}$. *L'état où se trouve le modèle à l'instant t est noté* $q_t \quad (q_t \in S)$
- *M symboles observables dans chaque état. L'ensemble des observations possibles (l'alphabet) est noté* $V = \{v_1, v_2, ..., v_M\}$. $O_t \in V$ *est le symbole observé à l'instant t.*
- *Une matrice A de probabilité de transition entre les états :* a_{ij} *représente la probabilité que le modèle évolue de l'état i vers l'état j :*

$$a_{ij} = A(i, j) = P(q_{t+1} = s_j \,|\, q_t = s_i) \;\forall i, j \in [1...n] \quad \forall t \in [1...T]$$

$$avec : a_{ij} \geq 0 \;\; \forall i, j \;\; et \sum_{j=1}^{n} a_{ij} = 1$$

- *Une matrice B de probabilité d'observation des symboles dans chacun des états du modèle :* $b_j(k)$ *représente la probabilité que l'on observe le symbole* v_k *alors que le modèle se trouve dans l'état j, soit :*

$$b_j(k) = P(O_t = v_k \,|\, q_t = s_j) \;\; 1 \leq j \leq n, \; 1 \leq k \leq M$$

$$avec : \; b_j(k) \geq 0 \;\; \forall j, k \;\; et : \sum_{k=1}^{M} b_j(k) = 1$$

- *Un vecteur* π *de probabilités initiales :* $\pi = \{\pi_i\}_{i=1,2,...,n}$. *Pour tout état i,* π_i *est la probabilité que l'état de départ du MMC soit l'état i :*

$$\pi_i = P(q_1 = s_i) \;\; 1 \leq i \leq n$$

$$avec : \; \pi_i \geq 0 \;\; \forall i \;\; et : \sum_{i=1}^{n} \pi_i = 1$$

Les MMC ont trois principales applications pour la caractérisation de séquences :

- L'évaluation de la probabilité d'observation d'une séquence. Ainsi, dans un problème de classification, on attribue une séquence à la classe que modélise le MMC le plus probable [Baum & al., 1970].
- La recherche du chemin le plus probable. L'algorithme de *Viterbi* permet de calculer la séquence d'évènements la plus probable étant donné un MMC ayant appris une ou plusieurs séquences [Rabiner, 1989].
- L'apprentissage automatique d'un MMC, par l'algorithme de *Baum-Welch*, soit l'apprentissage des paramètres (A, B, π) à partir d'une séquence ou d'un ensemble de séquences [Baum & al., 1970].

Les travaux décrits dans [Bidel & al., 2003] proposent plusieurs variantes de cette méthode pour reconnaître automatiquement les comportements de navigation d'internautes. La notion de flux modélise alors l'indépendance entre les plusieurs groupes propriétés descriptives.

Bilan sur le traitement individuel des séquences

Le traitement individuel des séquences possède théoriquement le meilleur rapport entre la complexité de calcul et les résultats escomptés. Néanmoins, ces méthodes ne sont pas toutes automatiques et nécessitent souvent la définition de paramètres non triviaux. Par exemple, le nombre d'états d'un MMC représenté dans une suite d'actions par un utilisateur sur une interface ne nous semble pas définissable a priori dans notre contexte.

2. Traitement des séquences deux à deux

Dans ce chapitre, nous présentons trois algorithmes qui prennent en entrée deux séquences. Cette approche de traitement de séquences par paires a été introduite par [Levenshtein, 1965] puis améliorée par [Wagner & Fisher, 1974] lors de la description de la programmation dynamique. Le traitement des données par paires permet la construction de matrices triangulaires (de similarité ou de distance) où chaque individu est comparé à tous les autres. Les individus ne possèdent plus de caractéristique absolue. Pour les séquences par exemple, la longueur sera prise en compte dans la comparaison, et ne sera pas une donnée traitée à part entière. Chaque individu n'est caractérisé que par rapport aux autres individus. Pour notre cadre d'application, nous considérons cette approche comme la plus pertinente. En effet, les processus cognitifs impliqués dans le cadre de l'apprentissage humain sont beaucoup trop nombreux et non forcement significatifs pour être décrits dans leur totalité.

Cette notion de traitement par similarité est maintenant largement répandue dans la communauté scientifique et l'on peut par exemple se référer aux « méthodes à noyau » qui produisent les meilleurs résultats dans la fouille de données moderne [Ng & al., 2001].

Distance d'édition

Introduite par [Levenshtein, 1965], la distance d'édition est définie comme le nombre minimal d'opérations d'édition nécessaire pour passer d'une séquence à une autre. Les opérations d'édition classiques sont : la suppression, l'ajout et la substitution d'un élément de la séquence. Le principe du calcul de la distance d'édition repose sur l'alignement dynamique des séquences, aussi appelé « programmation dynamique ».

L'algorithme initial permettant de calculer cette distance est présenté ci-dessous :

```
Entrées : Séquence x de longueur m, séquence y de longueur n

Initialiser une matrice Distance[m+1][n+1] remplie de 0 sauf
la première ligne qui contient les indices des colonnes
la première colonne qui contient les indices des lignes.

Remplissage de la matrice Distance :
Pour i de 1 à m faire
        Pour j de 1 à n faire
                Soit coût = 1
                Si x[i] égale à y[j] alors coût = 0
                Distance[i][j] = min ( Distance[i-1][j]+1,
                                       Distance[i][j-1]+1,
                                       Distance[i-1][j-1]+coût )

La distance d'édition entre la séquence x et la séquence y est le score inscrit dans la
case Distance[m][n]
```

Algorithme 2. Calcul de la distance d'édition entre deux séquences.[Levenshtein, 1965]

		c	h	i	e	n	s
	0	1	2	3	4	5	6
n	1	1	2	3	4	5	6
i	2	2	2	2	3	4	5
c	3	2	3	3	3	4	5
h	4	3	2	3	4	4	5
e	5	4	3	3	3	4	**5**

Figure 31. Calcul de la distance d'édition entre les mots « chiens » et « niche »

La complexité $\Theta(mn)$ de cet algorithme est relativement élevée car il est nécessaire de parcourir n fois la séquence x. Cette complexité a été très nettement améliorée par la suite avec [Wagner & Fisher, 1974], qui ont établi un lien très important entre la distance d'édition et la longueur de la plus longue sous-séquence commune à deux séquences.

La distance d'édition pour les arbres stratifiés ordonnés a été introduite par [Ganascia, 2002] et suit le même principe. Dans ce cas, le coût des opérations d'éditions classiques est calculé sur les séquences des nœuds fils de façon récursive. Ainsi, l'exploration en largeur des fils de 2 nœuds appariés permet une simplification de la complexité inhérente à la longueur des séquences. Il n'est plus nécessaire de parcourir l'ensemble de la séquence que constituent les feuilles de l'arbre. Par exemple, l'implémentation de cette technique a permis la caractérisation du style de narration d'auteurs littéraires [Ganascia, 2002].

Plus longue sous-séquence commune

La détermination de la plus longue sous-séquence commune à deux séquences correspond, dans notre cas, à la reconnaissance automatique de la tâche effectuée par deux individus lorsque celle-ci est séquentiellement définie. Toutefois, une contrainte sémantique (voir paradigme de l'impression au Chapitre 2.2) apparaît lorsqu'il s'agit de traiter directement les traces décrites dans le Chapitre 5. Il faut alors effectuer un prétraitement sur les données et construire un alphabet permettant d'associer différemment deux actions enrichies sémantiquement par les traces correspondant à une même tâche.

De nombreux algorithmes (dont les complexités sont présentées notamment dans [Paterson & Dancik, 1994]) pour trouver la plus longue sous-séquence commune à deux séquences (LCS pour *Longest Common Subsequence*) ont été proposés. En effet, il est apparu que la complexité de ce calcul pouvait être très significativement diminuée en fonction du résultat escompté. On peut ainsi choisir une méthode plus rapide si l'on

s'attend à obtenir une taille de LCS relativement petite par rapport aux tailles des séquences initiales.

Tous ces algorithmes ont pour but la détermination d'une des plus longues sous-séquences communes à deux séquences. Cependant, lors d'une étude comme celle portant sur des données issues des interactions HM, il est fréquent d'obtenir plusieurs sous-séquences communes de tailles maximales.

Nous proposons un nouvel algorithme (l'Algorithme 3 ci-dessous) qui introduit la notion de treillis pour parcourir la matrice (fonction *chercherSuivant*) remplie par la programmation dynamique (fonction *remplissage*). Notons que l'implémentation d'un treillis stratifié telle que celui décrit dans l'algorithme est relativement simple. Il suffit d'implémenter un objet treillis contenant un ensemble de matrices binaires. Chaque matrice représente alors les liaisons d'une couche à une autre couche du treillis. Les booléens de la matrice représentant la présence ou l'absence de liaison entre les éléments des couches. L'algorithme est étendu à la recherche du plus grand arbre couvrant deux arbres stratifiés ordonnés, suivant les remarques formulées précédemment pour la distance d'édition. Cette méthode a été implémentée pour le modèle de visualisation de traces décrit dans le Chapitre 5.

Entrées :
Deux Arbres stratifiés ordonnés X et Y de nœuds racines respectifs x_r et y_r
On note fils(x[i]) la liste des fils du nœud x[i], |fils(x[i])| la longueur de cette liste et val(x[i]) la valeur du nœud x[i]

Initialiser une matrice Indice[fils(x_r)+1][fils(y_r)+1] remplie de 0

Remplissage de la matrice Indice :
Fonction remplissage (Indice, x_r, y_r)
Pour i de 1 à |fils(x_r)|+1 faire
Pour j de 1 à |fils(y_r)|+1 faire
    Soit le nœud x[i] (respectivement y[i]) le ième (respectivement jième) fils de x_r(respectivement y_r)
    Si val(x[i-1]) égale val(y[j-1]) alors
        Indice[i][j] = Indice[i-1][j-1]+1
        Initialiser une matrice IndiceFils[|fils(x[i-1])|+1][|fils(y[j-1])|+1] remplie de 0
        remplissage (IndiceFils, fils(x[i-1]), fils(y[j-1]))
    Sinon Indice[i][j] = Max(Indice[i-1][j], Indice[i][j-1])
    Fin Si
Fin Pour
Fin Pour
Fin fonction

Soit T treillis de sommet A et de feuille terminale Z. Chaque nœud G de T est défini par un nom (représentant un nœud du sous arbre ordonné cherché) et un triplet d'entiers positifs (p0, p1, p2)

Soit L une liste de paires d'entiers positifs

On définit : A (Indice[|fils(x_r)|+1][|fils(y_r)|+1]+1, |fils(x_r)|+2, |fils(y_r)|+2) et Z (0,0,0)

```
Fonction chercherSuivant(i, j, Indice)
Si (i>0 et j>0 et (i, j) n'appartient pas à L) Alors Ajouter (i, j) à L
        Si x[i]=y[j] faire
                Soit G un nœud de T de nom x[i] et de propriété (Indice[i+1][j+1],i+1,
                j+1)
                Ajouter G aux nœuds R de A tel que T(p0)=R(p0)+1 et T(p1)>R(p1) et
                T(p2)>R(p2)
                chercherSuivant(i-1, j-1, Indice)
                chercherSuivant(|fils(x[i])|, |fils(y[j])| , IndiceFils)
        Sinon faire chercherSuivant(i-1, j, Indice) et chercherSuivant(i, j-1, Indice)
        fin si
fin si
fin fonction
```

L'exécution de chercherSuivant(|fils(x_r)|+1, |fils(y_r)|+1) construit le treillis T contenant toutes les plus longues sous-séquences communes à 2 séquences.

Algorithme 3. Algorithme d'extraction de l'ensemble des arbres stratifiés ordonnés communs à 2 arbres stratifiés ordonnés

Nous avons également implémenté une généralisation de cet algorithme pour la détection des LCS communes à plusieurs séquences. Malheureusement, l'exécution d'un tel programme est d'une complexité espace/temps désastreuse [Rick, 2000]. En effet, ce problème est connu dans la littérature sous le nom d'alignement multiple et est reconnu comme étant un problème NP-Difficile.

Pour calculer la plus longue sous-séquence commune à un ensemble de séquences, une étude sur la taille de la plus longue sous-séquence attendue est nécessaire. La construction de la matrice *Indice* par la fonction *remplissage* décrite dans l'algorithme permet de connaître la taille maximale de la sous-séquence sans la construire. Nous avons alors cherché la paire d'arbres responsables de cette taille. Nous avons ensuite généré toutes les sous-séquences de la taille attendue en comparant la paire d'arbres identifiée à l'ensemble des séquences à analyser. Cette opération ne garantit pas l'exploration totale de l'ensemble des solutions, mais une complexité abordable pour notre problème. Un expert a donc dû vérifier la validité du résultat.

Remarque :

La distance d'édition entre deux séquences a connu la même évolution dans la complexité de son calcul que celui de la plus longue sous-séquence commune à deux séquences. En effet, il a été montré par [Wagner & Fisher, 1974] que la distance d'édition et la longueur de la plus longue sous-séquence commune à deux séquences sont deux valeurs intrinsèquement liées par la relation :

$$\rho(x,y) = m + n - 2\,\sigma(x,y)$$

Où : $\rho(x,y)$ = distance d'édition entre la chaîne x et la chaîne y

m = longueur de la chaîne x

n = longueur de la chaîne y

$\sigma(x,y)$ = longueur de la LCS

Méthodes à noyau : String Kernels

Les méthodes à noyau constituent ces dernières années les avancées algorithmiques majeures en apprentissage artificiel [Shawe-Taylor & Cristianini, 2004Shawe-Taylor & Cristianini, 2004Shawe-Taylor & Cristianini, 2004]. Le principe de base de fonctionnement de toute méthode à noyau est la projection implicite des données initiales X dans un autre espace F, généralement de dimension plus grande, qui permet à un classificateur linéaire d'opérer simplement : $\phi : X \to F$ (un classificateur linéaire est une fonction affine permettant de couper l'ensemble des exemples d'apprentissage selon leurs classes). L'espace F, appelé *espace des caractéristiques,* ou *feature space,* doit être un espace de Hilbert, c'est-à-dire accepter le produit scalaire. Le second principe de base est que le calcul effectué par l'algorithme de classification n'est pas appliqué dans ce nouvel espace mais dans l'espace produit : $k : X \times X \to \Re$. On parle alors de *l'astuce noyau* ou du *kernel trick* :

$$\forall x, y \quad \langle \phi(x), \phi(y) \rangle = k(x, y)$$

La fonction noyau k permet d'effectuer le produit scalaire des données projetées dans le nouvel espace, généralement de dimension plus grande, sans réellement effectuer ce calcul. Elle donne ainsi des renseignements très importants sur la similarité : plus la valeur du noyau est grande, plus les individus comparés sont similaires. Les données n'ont plus besoin d'être représentées dans leur espace d'origine et la définition de l'algorithme d'apprentissage utilisé n'est plus nécessaire.

La méthode appelée *String Kernel* a été développée par [Lodhi & al., 2002] pour la classification textuelle. Cette technique introduit la notion de traitement séquentiel des données par les méthodes à noyau. La projection implicite effectuée est présentée dans

la Figure 32. Il s'agit de projeter la séquence initiale sur l'ensemble des sous-séquences de longueur *n* donnée formées par les caractères de cette séquence. Un paramètre *dégradant* λ est introduit pour inclure la plus ou moins grande importance que l'on veut accorder aux espaces dans les sous-séquences projetées. Dans l'exemple présenté ci-dessous, les séquences de caractères cat, car, bat et bar sont formées par des combinaisons des sous séquences de caractères ca, ct, at, ba, bt, cr, ar, br. Seules ces sous séquences de longueur deux sont utilisées pour composer les séquences initiales, et il n'est pas utile de projeter une séquence initiale sur la sous séquence rb.

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
ϕ (cat)	λ^2	λ^3	λ^2	0	0	0	0	0
ϕ (car)	λ^2	0	0	0	0	λ^3	λ^2	0
ϕ (bat)	0	0	λ^2	λ^2	λ^3	0	0	0
ϕ (bar)	0	0	0	λ^2	0	0	λ^2	λ^3

Figure 32. Projection des séquences {cat, car, bat, bar} dans un espace de plus grande dimension {ca, ct, at, ba, bt, cr, ar, br}

Ici, le noyau non normalisé entre *car* et *cat* est K(car, cat) = λ^4, et la version normalisée du noyau est obtenu par : K(car, car) = K(cat, cat) = $2\lambda^4+\lambda^6$ et donc K(car, cat) = $\lambda^4/2\lambda^4+\lambda^6 = 1/(2+\lambda^2)$.

Une implémentation directe de ce calcul implique une complexité $O(|\sum|^n)$ en espace et temps, étant donné le nombre de sous-séquences projetées. Ceci est pratiquement infaisable. Les auteurs proposent donc une fonction intermédiaire *K'* pour le calcul récursif du noyau qui calcule la longueur d'une sous-séquence particulière à partir de son début et jusqu'à la fin des séquences.

$$K_0'(s,t) = 1, \text{ pour tout } s,t,$$
$$K_i'(s,t) = 0, \text{ si } \min(|s|,|t|) < i,$$
$$K_i(s,t) = 0, \text{ si } \min(|s|,|t|) < i,$$

Pour i = 1, …, n-1 si x n'apparaît pas dans u, $K_i''(sx,tu) = \lambda^{|u|} K_i''(sx,t)$

$$\text{Sinon } K_i''(sx,tx) = \lambda\left(K_i''(sx,t) + \lambda K_{i-1}'(s,t)\right)$$
$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} K_{n-1}'(s,t[1:j-1])\lambda^2$$

Algorithme 4. Définition de la fonction récursive du string kernel

Une fonction de normalisation est nécessaire car un biais naturel va se créer dû notamment à la longueur des séquences traitées. De la même façon que l'on peut normaliser un vecteur dans l'espace des caractéristiques par $\hat{\phi}(s) = \frac{\phi(s)}{\|\phi(s)\|}$, le string kernel peut être normalisé par :

$$\hat{K}(s,t) = \left\langle \hat{\phi}(s) \cdot \hat{\phi}(t) \right\rangle = \left\langle \frac{\phi(s)}{\|\phi(s)\|} \cdot \frac{\phi(t)}{\|\phi(t)\|} \right\rangle = \frac{1}{\|\phi(s)\| \ \|\phi(t)\|} \langle \phi(s) \cdot \phi(t) \rangle = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}$$

La complexité est en $O(|s||t|)$ pour le calcul de $K_i^{"}(s,t)$, et la complexité totale du noyau devient $O(n|s||t|)$ en temps. Il revient ensuite à l'utilisateur de choisir la pondération qu'il souhaite accorder à chacune des longueurs de sous-séquences projetées pour calculer le noyau global $K(s,t)$.

Bilan sur le traitement deux à deux des séquences

Issue de la programmation dynamique, ces méthodes sont incrémentales. C'est-à-dire que le score calculé pour le n^{ième} élément de la séquence est utilisé pour calculer le score du (n+1)^{ième} élément. Pour notre cas d'application et lorsque les scores intermédiaires sont conservés, cette technique permet d'étudier le comportement des individus tout au long de leur parcours.

Toutes ces méthodes construisent des matrices de similarités globales où chaque individu est comparé à tous les autres. Dans ce cas, l'apprentissage automatique n'est pas formellement terminé puisque les données ne sont pas encore affectées à une classe. Néanmoins, la description des individus est complète puisque toutes les différences et les ressemblances avec les autres individus sont mises à jour. L'affectation d'un individu à une classe caractéristique d'un groupe peut ensuite être réalisée à l'aide des algorithmes décrits dans le paragraphe 2 ci-dessous.

Notons que les algorithmes utilisant des fonctions noyaux pour le traitement des séquences connaissent actuellement un essor important. Ainsi, le *mismatch string kernel* et le *spectrum kernel* sont des algorithmes qui peuvent fournir une description différente des séquences. Le lecteur intéressé peut se référer à [Leslie & al., 2002] et [Leslie & al., 2004]

3. Traitement sur un ensemble

L'alignement multiple de séquences est un problème connu NP-complet. La communauté cherchant à qualifier directement tous les individus s'est orientée vers une

méthode appelée *a priori*. Cela consiste en la génération d'un candidat possible *a priori*, et en la vérification *a posteriori* de la validité de ce candidat. Cette approche fut présentée par [Agrawal & al., 1995] puis largement étudiée par la suite. L'avantage de cette méthode provient du fait que les ressemblances entre les individus vont forcement être explicites, car elles vont supporter les séquences candidates. L'inconvénient majeur reste la complexité de calcul qui est trop grande quand la taille des individus ou le nombre des individus considérés augmente. Nous présentons ici quelques algorithmes qui dérivent de cette technique en suivant la typologie présentée dans [Masseglia & al., 2004]. Les algorithmes présentés dans cet article ont initialement été conçus pour traiter les bases de données de transactions. Ces dernières sont similaires aux bases de séquences, sauf que les modifications apportées sur les séquences peuvent aussi porter sur les éléments composant les séquences. Nous ne tiendrons pas compte de ce cas de figure dans ce chapitre.

A priori et améliorations

Elaboré par [Agrawal & al., 1995] la détection de motifs séquentiels a d'abord été étudiée comme un cas particulier des règles d'associations, avec une contrainte séquentielle. Il s'agit dans ce cas de trouver toutes les séquences candidates possibles possédant un seuil supérieur à $\mu \in]0,1]$ fixé par l'utilisateur

Définition : Soit $\sum$ *un ensemble de séquences contenant* $s = \{s_1, s_2, ..., s_{|s|}\}$ *et* $s' = \{s'_1, s'_2, ..., s'_{|s'|}\}$. *On dit que s est incluse dans* s' *si et seulement s'il existe* $i_1 < i_2 < ... < i_{|s|}$ *entiers tels que* $s_1 = s'_{i_1}, s_2 = s'_{i_2}, ..., s_{|s|} = s'_{i_{|s|}}$. *Une séquence candidate* s' *supporte une séquence s si* s' *est incluse dans s. Le support de la séquence candidate* s' *est calculé comme étant le pourcentage de séquences de* $\sum$ *qui supportent* s'.

L'algorithme présenté, appelé GSP pour *Generalized Sequential Patterns*, utilise deux fonctions, l'une pour calculer les supports des séquences candidates, et l'autre pour générer de nouveaux candidats à partir des séquences dont le support satisfait μ. Cette dernière fonction ajoute généralement un élément à chaque séquence de façon incrémentale. Ainsi, l'ensemble des séquences est parcouru à chaque recherche d'un motif séquentiel de longueur supérieure. Il faut s'attendre à devoir générer, uniquement pour la recherche de motifs de longueur 2, pas moins de $n^2 + \frac{n(n-1)}{2}$ candidats à partir des *n* items fréquents trouvé lors du premier passage dans la base.

Amélioration du stockage : SPADE et PSP

L'amélioration SPADE apporté par [Zaki, 2001] réduit l'espace de recherche des candidats en regroupant les motifs séquentiels déjà extraits par catégorie. Ce regroupement peut en effet être optimisé en préservant les supports et les préfixes des candidats déjà générés. Ainsi, les motifs fréquents sont regroupés par préfixe commun, et la génération de candidats se fait par groupe, ce qui divise le problème principal en sous-problèmes traités plus rapidement.

Ensuite, [Masseglia & al., 1998] propose PSP, pour *Prefix tree for Sequential Pattern*, qui optimise le stockage des motifs séquentiels en arbres dont chaque parcours de la racine aux feuilles représente un motif et inversement, tout motif possède un chemin de la racine aux feuilles dans l'arbre. Cette représentation en mémoire améliore considérablement la représentation utilisée dans GSP. En effet, GSP n'utilise la représentation arborescente que pour classer les motifs générés (sorte de fonction de hachage).

Amélioration de la complexité : PréfixSpan et Spam

L'algorithme *PrefixSpan* présenté dans [Pei & al, 2001] suit le même principe de « génération Candidat » / « comptage Support » mais propose de réduire le nombre de candidats générés, en utilisant une projection de la base totale des séquences traitées sur les motifs fréquents déjà extraits. Ainsi, l'espace de recherche lors de la génération de candidats est réduit aux séquences projetées de la base et non aux motifs fréquents déjà créés lors des itérations précédentes. Les auteurs proposent également une méthode de stockage des bases de données successives pour le traitement d'un grand nombre de séquences dont la duplication peut poser un problème de mémoire.

La méthode *Spam*, présentée dans [Ayres & al., 2002], propose de représenter les opérations de génération de candidats par des opérations d'ajouts d'items dans une séquence et de les stocker sous forme de vecteurs de bits. Ainsi, le calcul du support pour chaque motif est immédiat car il suffit de compter les bits positionnés à 1 dans la structure.

Séquences fermées

Afin de simplifier encore l'espace de recherche lors de la génération de candidats, la notion de séquences fermées a été introduit comme suit :

Définition : Soit minSup, le support minimum et FS l'ensemble des motifs séquentiels fréquents correspondants. L'ensemble des motifs séquentiels fermés CS est défini comme :

$CS = \{ s / s \in FS \text{ et } \nexists\ s' \text{ telle que } s' \subseteq s \text{ et } support(s') = support(s) \}$

Suite à cette définition les auteurs de [Yan & al., 2003] proposent de ne pas faire grandir tous les motifs précédemment générés, mais seulement ceux dont une génération ajoutera une information. En effet, les auteurs remarquent que si on découvre un ordre systématique du type *« a apparaît toujours avant b »* dans la base de données de séquences, alors il arrive un moment dans la génération des candidats où les motifs créés à partir de <*préfix_b*>*b* seront aussi créés à partir des motifs générés par <*préfix_a*>*a*. Il n'est donc pas utile de continuer la génération de candidats à partir de <*préfix_b*>*b*. La mise en évidence d'une règle d'ordre systématique dans une base de données n'étant pas chose facile, les auteurs présentent un théorème très utile pour la génération de candidats dans les bases de données volumineuses :

Théorème : Soient s et s' deux séquences telles que $s \subseteq s'$*. Soient* D_s *et* $D_{s'}$ *les bases de données projetées selon les préfixes s et s'. Soit L(D) le nombre total d'éléments dans une base de donnée D. Alors on a :*

$$D_s = D_{s'} \Leftrightarrow L(D_s)=L(D_{s'})$$

Ainsi, il suffit de comparer les bases de données projetées des différents motifs fréquents et dès que deux bases projetées sont similaires une relation d'ordre systématique apparaît et l'exploration d'une seule des deux bases pour la génération de candidats reste utile.

Recherche incrémentale

Afin d'éviter de parcourir à nouveau la base de données des séquences lorsque de nouvelles séquences sont ajoutées ou lorsque de nouveaux éléments apparaissent dans les séquences, la communauté s'est orientée vers la conception de méthodes incrémentales de détection des motifs fréquents. Il s'agit alors de conserver en mémoire une information pertinente sur la façon dont les motifs fréquents sont extraits d'une base de données de séquence *DB*.

Ainsi, [Masseglia & al., 2003] propose de ne conserver que les motifs fréquents obtenus précédemment et de reconsidérer la méthode générer-élaguer du GSP dans ce contexte. L'algorithme *ISE* décrit comporte deux étapes. La première consiste à chercher les éléments fréquents qui appartiennent aux nouvelles séquences d'une base *db* à traiter. Un ensemble, appelé *source* est construit, composé des motifs fréquents déjà extraits auxquels un élément fréquent de *db* est ajouté. Si la séquence qui résulte de cette opération est fréquente sur *U=DB+db*, alors elle est ajoutée à *source*. Ces

éléments fréquents de *db* vont constituer les extensions fréquentes des motifs de *source*. A chaque passe sur *U* les nouveaux motifs fréquents de taille *j* et les nouvelles extensions fréquentes de taille *j+1* sont calculés.

L'algorithme *ISM* proposé par [Parthasarathy & al., 1999] introduit la notion de *bordure négative* dans l'espace de recherche des candidats. Une *bordure négative* est l'ensemble des séquences qui ne sont pas fréquentes mais dont les sous-séquences qui l'ont générée sont fréquentes. Il s'agit alors de supprimer de l'ensemble des séquences fréquentes, celles qui ne le sont plus après l'ajout de *db*. On peut ainsi mettre à jour la *bordure négative* en prenant en compte le nouveau support des motifs et déterminer les séquences de la bordure négative qui vont devenir des motifs fréquents.

Bilan sur le traitement d'un ensemble de séquences

Les évolutions de l'algorithme initial traitant les séquences dans leur ensemble sont nombreuses. Toutes ces différentes versions cherchent à former des motifs qui explicitent le regroupement des individus. La conception incrémentale a permis d'établir des systèmes d'extraction d'informations fonctionnant en temps réel. Néanmoins, les coûts de calcul sont souvent élevés, et l'espace mémoire requis pour les faire fonctionner correctement peut être important. Ainsi, ces méthodes seront privilégiées lorsque le nombre de séquences à traiter est volumineux, mais la longueur des séquences relativement faible. Par exemple, ces méthodes sont très utilisées pour l'analyse de fichiers logs issus des serveurs internet.

4. Etude comparative

Les méthodes d'apprentissage artificiel qui traitent directement les données dans un format séquentiel sont nombreuses. Nous les avons présentées en suivant une typologie décrivant les différentes façons qu'ont ces méthodes de traiter les données en entrée. Ainsi, selon l'apprentissage que l'on souhaite effectuer et en tenant compte de la quantité et de la qualité de données présentes, on peut choisir la méthode la plus adaptée (voir Figure 33).

Les résultats produits par ses différentes méthodes sont également différents. Ainsi, les méthodes de traitement individuel sont généralement utilisées avec des probabilités. Celles-ci fournissent de précieuses informations sur la structure des séquences, mais ne permettent pas d'interprétation directe. L'interprétation n'est possible que lorsque l'on compare les résultats du traitement avec ceux sur d'autres séquences.

Méthodes de traitement des données en entrée	Individuel	Par paire	Sur un ensemble
Séquences longues et peu nombreuses	+	+++	
Séquences longues et nombreuses	++	+	
Séquences courtes et peu nombreuses		++	++
Séquences courtes et nombreuses			+++

Figure 33. Récapitulatif des méthodes à traitement séquentiel en fonction des données à traiter

Au contraire, les méthodes qui traitent les séquences par ensemble conservent l'intelligibilité des résultats fournis. C'est-à-dire qu'ils sont en général directement exploitables, notamment lors de l'extraction de motifs séquentiels. Le traitement par ensemble nécessite tout de même une implémentation rigoureuse tant les complexités d'espace peuvent être importantes. Le traitement par paire de séquences connaît la même difficulté. La complexité devient trop importante lorsque le nombre de séquences à traiter augmente.

Ces méthodes ne sont pas toutes équivalentes lorsque l'on cherche à construire un système d'apprentissage incrémental. Un ensemble de données séquentielles peut subir deux sortes de modifications :

- Une nouvelle séquence correspondant à un nouvel individu, ou à une nouvelle session est ajoutée à la base,
- Les séquences déjà présentes évoluent en s'ajoutant des éléments (analyse de comportement en temps réel).

Dans le premier cas, le traitement individuel nous semble le plus approprié car il ne tient pas compte des autres séquences de la base. De même, des propositions ont été formulées pour le traitement incrémental sur un ensemble de séquences. Pour le second cas, le traitement par paire est un bon compromis car ces méthodes connaissent généralement un processus itératif qui peut suivre l'incrémentation.

Le modèle de plateforme pour la pédagogie décrite au Chapitre 9 autorise l'utilisation de tous les algorithmes d'apprentissage qui fournissent un résultat interprétable. Les méthodes décrites dans ce chapitre ne fournissent pas toutes une interprétation utile pour la personnalisation de l'interaction. Par exemple, les modèles markoviens sont capables de fournir une description de la classification opérée mais cette description n'est généralement pas interprétable pour la personnalisation. Au

contraire, les méthodes d'extraction de motifs (traitement sur un ensemble) peuvent fournir les sous-séquences communes à un ensemble d'individus facilitant l'interprétation du comportement par un expert. Les méthodes de traitement des séquences par paire proposent différentes propriétés permettant une interprétation. En effet, la plupart de ces méthodes sont itératives et offrent une caractérisation des éléments composant les séquences. Ces caractérisations par éléments peuvent ensuite se révéler interprétables.

2. Algorithmes avec traitement matriciel des données

Les données de nature séquentielle n'ont pas toujours été traitées en incluant cette notion de séquence. Ainsi, on peut remarquer qu'avant d'être en format séquentiel, les éléments qui les composent sont d'abord présents chez un individu. Cette remarque a ainsi mené les chercheurs en *text mining* (fouille de données textuelles) à ne considérer que des statistiques sur les éléments composant une séquence (les mots).

Pour le traitement des traces décrites dans le Chapitre 5, la nature temporelle des données ne peut pas être facilement incluse dans les algorithmes de traitement séquentiel. Par contre, elle est facilement incorporable dans une représentation matricielle où la durée des évènements peut constituer une propriété.

Les algorithmes présentés dans ce paragraphe considèrent que chaque donnée possède certaines propriétés qui prennent différentes valeurs selon les individus. Ainsi, une matrice où les individus sont représentés dans les lignes et les attributs dans les colonnes peut être formée. Ces propriétés peuvent être de différents types et les valeurs peuvent appartenir à des ensembles différents selon les attributs. On s'assure dans ces algorithmes que les propriétés, relatives à une même caractéristique de l'individu, sont comparées entre elles.

On peut également comparer les individus entre eux. Dans ce cas, la distance ou les similarités de chaque individu avec tous les autres individus constitue les propriétés traitées par l'algorithme. La distance ou la similarité considérée prendra alors en compte les caractéristiques partagées par les individus comparés.

1. Du séquentiel au matriciel, l'exemple TFIDF

Nous présentons ici l'algorithme TFIDF [Salton & Buckley, 1988] car il constitue, de par son utilisation, un cas d'application pour des données séquentielles traitées en format matriciel. En effet, dans ce cas, les propriétés inhérentes à la séquence sont omises, et seules sont prises en compte des statistiques de fréquences d'apparition en

fonction du corpus de documents considéré. Le principe suit la loi de Zipf : *« Un mot est informatif dans un document s'il y est présent souvent mais s'il n'est pas présent trop souvent dans les autres documents du corpus »*.

Principalement utilisé pour le *text mining*, cet algorithme est performant quand il s'agit de traiter des séquences longues, voire très longues, comme des phrases ou des textes et dont les éléments qui les composent se répètent. Les variantes de cet algorithme sont nombreuses et la formule la plus couramment utilisée est celle-ci :

$$d_{ik} = tf_{ik} \times \log\left(\frac{N_D}{n_k}\right)$$

avec : d_{ik} = poids du terme k dans le document i
tf_{ik} = occurrence du terme k dans le document i
N_D = nombre de documents dans la collection
n_k = nombre de documents possédant le terme k

Ainsi, chaque document de la collection D est décrit selon les pondérations affectées aux termes d_{ik} de toute la collection. Une fois cette matrice obtenue, l'apprentissage artificiel n'est pas fini puisque les documents ne sont pas encore regroupés, et aucun concept général ne peut les décrire. Les données, initialement sous forme de séquence, ont simplement changé de représentation, et l'application de méthodes comme celles décrites ci-après permettent la véritable reconnaissance de descripteurs.

Les méthodes présentées dans le paragraphe 2 sont également des méthodes qui changent la représentation des données. Dans ce cas, les données sont représentées en fonction d'une ou plusieurs propriétés partagées par deux séquences. Par exemple, un individu peut être représenté en fonction du nombre de sous-séquences de longueur maximale que l'on peut former avec chacune des autres séquences [Banerjee & Ghosh, 2001].

2. Arbres de décision et arbres de décision flous

Les arbres de décision forment une très grande famille des algorithmes d'intelligence artificielle car ils offrent de multiples avantages, notamment pour la classification supervisée. En effet, ils ne souffrent pas de la sémantique des données car ils peuvent traiter aussi bien les données symboliques que les données numériques définies dans un espace discret ou continu. Le processus classique de construction des arbres de décision a été présenté dans [Quinlan, 1986] et amélioré dans [Quinlan, 1993] et est appelé la méthode *Top Down Induction*. Cela signifie que la construction de

l'arbre, qui constitue l'apprentissage par la machine de la classification des utilisateurs, se fait à partir d'exemples et de façon incrémentale sur la structure de l'arbre : on construit d'abord les feuilles près de la racine de l'arbre, puis les feuilles les plus éloignées de la racine de l'arbre. Ainsi, la décision prise par un arbre lors de la classification d'un nouvel individu est précise, robuste et facilement interprétable par l'humain.

Les arbres de décision flous sont une extension des arbres de décisions classiques. Ils ont été décrits par [Ramdani, 1994] et incorporent la théorie des ensembles flous définie par [Zadeh, 1965]. Ils offrent plus de souplesse dans le traitement des données numériques en incorporant leurs éventuelles imprécisions. Leur construction suit généralement le même principe que celui des arbres de décisions classiques et nécessite une discrétisation floue des attributs.

Discrétisation floue des attributs numériques

A chaque nœud, avant la sélection du meilleur attribut, tous les attributs numériques sont discrétisés en intervalles. Les points de coupures de ces discrétisations sont définis par un triplet de réels $[c-\delta, c, c+\delta]$ choisi pour maximiser l'homogénéité des partitions floues. Un point de coupure d'un domaine de valeur d'un attribut en deux partitions floues D_L et D_R de l'ensemble des exemples dont les degrés d'appartenance sont définis dans la Figure 34 ci-dessous.

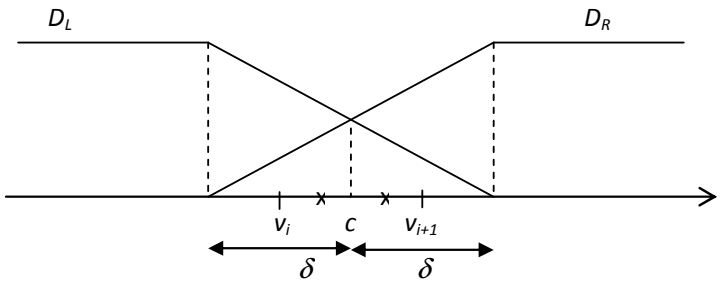

Figure 34. Point de coupure floue

Le point de coupure est celui qui minimise $I^*(D/A_k)=\frac{|D_L|}{|D|}I^*(D_L)+\frac{|D_R|}{|D|}I^*(D_R)$ où $I^*(X)=-\sum_{i=1}^{m}p^*(C_i)\log p^*(C_i)$ est l'entropie floue et $p^*(C_i)=\frac{\sum_{e_j\in c_i}p_j\mu_j}{\sum_1^n p_j\mu_j}$ la probabilité floue p_j d'un évènement e_j qui a un degré μ_j d'appartenir à la classe C_i [Ramdani, 1994].

Choix du meilleur attribut pour une discrimination par une mesure floue

Après la discrétisation, le gain d'information de chaque attribut flou A_k est donné par $Gain(A_K)=I^*(L)-I^*(L/A_K)$. L'attribut le plus discriminant est celui qui maximise ce gain d'information.

Entrée : langage de description ; échantillon S

Initialiser à l'arbre vide ; la racine est le nœud courant

Répéter tant qu'il y a des nœuds inexplorés
    Si le nœud courant est terminal alors
        Affecter une classe
    Sinon
        Sélectionner le meilleur attribut pour partitionner le nœud courant en sous-nœuds à traiter
        Créer le sous-arbre
    FinSi
    Passer au nœud suivant non exploré s'il en existe
Fin

Algorithme 5. Algorithme général de création d'un arbre de décision

Une description des arbres de décision flous est présentée dans [Damez & al. 2005]. Cette méthode a été implémentée et les résultats sont présentés dans le Chapitre 11.

3. Clustering hiérarchique

Les algorithmes de regroupements hiérarchiques peuvent être catalogués en deux catégories : le clustering hiérarchique ascendant où l'on construit de nouveaux clusters à chaque itération, et le clustering hiérarchique descendant où l'on détruit d'anciens clusters pour en former de nouveaux. Le *dendrogramme* d'un clustering hiérarchique est la représentation graphique des partitions/regroupements successifs que subissent les

données. Cette méthode de représentation est illustrée dans la Figure 18 du Chapitre 6.1.2.

Le clustering hiérarchique ascendant (CHA) est très largement utilisé dans la littérature scientifique. Son utilisation nécessite tout de même la connaissance de certains problèmes inhérents à son fonctionnement.

Entrée : Données X = {x_1, x_2, …, x_n} ainsi qu'une mesure de distance entre elles.

Initialisation : Affecter un cluster à chaque donnée.
Itérer jusqu'à obtenir un seul cluster
Fusionner les clusters dont la distance est minimale

Le dendogramme de sortie représentant les fusions successives des clusters forment généralement une bonne représentation du partitionnement de l'espace des données traitées

Algorithme 6. Clustering hiérarchique ascendant

Il existe donc plusieurs types de CHA qui dépendent de la façon dont on fusionne les clusters. Plusieurs opérateurs d'agrégation de données qui conservent les propriétés de distances souhaitées peuvent être utilisés pour fusionner les clusters. Les plus connues sont le *minimum* (stratégie de *chaînage simple*), le *maximum* (*chaînage complet*), la *moyenne* (*chaînage moyen*), et le critère de [Ward, 1963] défini par :

$$d(A,B) = \frac{|A||B|}{|A|+|B|}\left\|\bar{x}_A - \bar{x}_B\right\|^2$$

Où : $\bar{x}_A$ est la moyenne du groupe *A* et |*A*| le nombre d'individu du groupe *A*.

Ce chaînage consiste à fusionner les groupes qui minimisent l'augmentation de la variance intraclasse. Cette dernière mesure permet notamment d'éviter *l'effet de chaînage* qui consiste à fusionner dans le même cluster des données non directement similaires entres elles mais liées par un ensemble de données intermédiaires qui autorisent les fusions successives.

4. Les K-moyennes et C-moyennes floues

L'algorithme des *K-moyennes* [MacQueen, 1967] est parmi les plus simples et les plus efficaces des algorithmes de regroupement. Le but est de calculer les centres d'éventuels clusters dont le nombre est fixé a priori. L'implémentation de cette technique est relativement simple, mais son application doit être utilisée avec la plus grande précaution. En effet, les résultats dépendent beaucoup de l'initialisation. Le

principe est la mise à jour de façon itérative des centres de clusters initialement affectés aléatoirement. Ces méthodes constituent la famille des algorithmes à *centres mobiles*. Cette mise à jour du centre est effectuée en calculant la distance de celui-ci avec l'ensemble des données. Pour les K-moyennes, une attribution binaire du type « appartient / n'appartient pas » à ce centre est produite.

L'algorithme des C-moyennes floues [Bezdek 1981] fonctionne sur le même principe sauf pour l'attribution d'une donnée à un centre de cluster, qui est floue du type : « appartient à un degré μ » à un cluster. Le flou apporte ici une souplesse dans l'affectation des centres aux groupes. Celle-ci sera ainsi moins sujette aux initialisations et convergera plus rapidement.

Entrée : $k > 1$ *le nombre de clusters souhaités*
m *degré flou de partitionnement*
$X = \{x_1, x_2, \ldots, x_n\}$ *l'ensemble des individus*

Initialiser les centres des clusters $W = \{w_1, w_2, \ldots, w_k\}$ *aléatoirement.*

Itérer jusqu'à stabilisation (les centres ne subissent plus de nouvelles affectations)
Affecter les données au centre le plus proche en définissant les variables floues u_{ik} *tel que :*

$$u_{ik} = \frac{1}{\sum_{s=1}^{k} \left(\frac{\|x_i - w_k\|}{\|x_i - w_s\|} \right)^{\frac{2}{m-1}}}$$

Recalculer les coordonnées des centres par :

$$w_k = \frac{\sum_{i=1}^{n} u_{ik}^{\,m} w_k}{\sum_{i=1}^{n} u_{ik}^{\,m}}$$

Les centres des clusters forment un partitionnement de l'espace des données traitées

Algorithme 7. C-moyennes floues

De nombreuses variantes de cet algorithme ont été proposées. En effet, certains défauts de cet algorithme apparaissent lors de la présence d'individus limites. Ceux-ci sont de deux sortes, limite entre deux classes, et limite du type exceptionnel (l'individu forme une classe à lui tout seul). Dans le premier cas, l'algorithme aura tendance à converger vers un optimum local frontière, ne permettant pas de délimitation significative entre les groupes. D'autre part, les individus du type exception auront tendance à créer des centres de groupes « éloignés ». Pour pallier ce problème, [Lesot &

Bouchon-Meunier, 2004] proposent un algorithme de regroupement pour prendre en compte ces difficultés en couplant l'algorithme des C-moyennes floues avec un algorithme de classification hiérarchique.

5. Spectral Clustering

Le clustering spectral a été initialement introduit pour la segmentation des images [Shi & Malik, 2000]. Les excellents résultats produits dans ce domaine ont permis à la communauté d'étendre cette méthode à d'autres types de tâches comme le clustering. Les auteurs de [Yu & Shi, 2003] décrivent un algorithme de partitionnement normalisé en *k* classes.

Soit un graphe $G = (\nu, \varepsilon, A)$, avec ν l'ensemble des points du graphe à partitionner, ε l'ensemble des arêtes du graphe reliant tous les points de ν, et A la matrice de similarité, symétrique et à valeurs positives, de ces arêtes.

On définit : $links(\alpha, \beta) = \sum_{i \in \alpha, j \in \beta} A(i, j)$ avec $\alpha, \beta \subseteq \nu$,

que l'on peut normaliser par $normlinkratio(\alpha, \beta) = \frac{links(\alpha, \beta)}{links(\alpha, \nu)}$.

Le partitionnement normalisé en *k* classes consiste à minimiser le poids du cluster formé par α relativement au poids total formé par toutes les partitions possibles de ν.

Le problème se résume donc par : $minimize \frac{1}{k} \sum_{j=1}^{k} normlinkratio(\nu_j, \nu \setminus \nu_j)$

Les auteurs de [Yu & Shi, 2003] introduisent ici la relaxation spectrale pour résoudre ce problème : soit *D* la matrice diagonale où la valeur en (*i*, *i*) est la somme de toutes les valeurs de la ligne *i* dans la matrice *A*. Alors, la minimisation du critère du partitionnement normalisé ci-dessus est équivalent à :

$$maximize\left(\frac{1}{k} trace(Z^T A Z)\right),$$

où $Z = X(X^T D X)^{-1/2}$, et X est la matrice $n \times k$ de partitionnement. En introduisant $\tilde{Z} = D^{1/2} Z$, et en relâchant la contrainte d'avoir X comme matrice de partitionnement, le problème se formule comme étant la maximisation de la trace de $\tilde{Z}^T D^{-1/2} A D^{-1/2} \tilde{Z}$ où la contrainte sur $\tilde{Z}$ s'exprime par $\tilde{Z}^T \tilde{Z} = I_k$. Une solution connue pour résoudre ce problème [Dhillon & al., 2004] consiste à former $\tilde{Z}$ à partir des k vecteurs propres de la matrice $D^{-1/2} A D^{-1/2}$. Ainsi, ces vecteurs propres sont utilisés pour le partitionnement.

Notation : *Individu* $p \in \{1,...,P\}$

$w_{i,j}$ *similarité entre l'individu i et l'individu j*

Entrée : *Matrice de similarité* $W \in \Re^{P \times P}$

R nombre de clusters

1. Calculer les R premiers vecteurs propres de $U = D^{-1/2}WD^{-1/2}$ *avec D la matrice diagonale formée par* $d_{i,i} = \sum_i w_{i,j}$ *, soit* $D = diag(W1)$ *où* 1 *désigne un vecteur de* 1.

2. Soit $U = (u_1,...,u_p) \in \Re^{R \times P}$ *la matrice formée par l'agrégation des R premiers vecteurs propres orthogonaux et* $d_p = D_{pp}$

3. K-moyennes pondérées : On initialise une matrice d'affectation A au hasard. Tant que A n'est pas stationnaire

a. Pour tout r, on calcule le centre $\mu_r = \frac{\sum_{p \in A_r} d_p^{1/2} u_p}{\sum_{p \in A_r} d_p}$

b. Pour tous les individus p, on affecte p à A_r *tel que* $\left\| u_p d_p^{-1/2} - \mu_r \right\|$ *soit minimum.*

Sortie : matrice d'affectation A, mesure de distorsion $\sum_r \sum_{p \in A_r} d_p \left\| u_p d_p^{-1/2} - \mu_r \right\|^2$

Algorithme 8. Spectral Clustering

L'Algorithme 8, extrait de [Bach & Jordan, 2004], apporte plusieurs grandes améliorations par rapport aux méthodes à centres mobiles ou hiérarchiques. L'utilisation des vecteurs propres permet une plus grande interprétation des comportements des individus les uns par rapports aux autres. Ainsi, on forme des clusters dont les individus sont proches les uns des autres mais qui, en plus, ont le même comportement par rapport aux autres données (la dissimilarité des données est prise en compte). De plus, l'utilisation d'une matrice de similarité comme entrée permet un prétraitement des données par les méthodes à noyau, particulièrement performante dans la description des individus.

6. Etude comparative

Les algorithmes présentés dans cette partie fonctionnent selon le même principe de traitement des données. Ainsi, les individus doivent être décrits selon les mêmes critères, et le traitement qui permet l'apprentissage artificiel d'un concept sur ces

individus est possible. Cet apprentissage est statistique et il est évident que plus la masse de données est représentative de la réalité, plus les résultats seront pertinents. Néanmoins, et particulièrement dans notre cadre d'application, la description des individus peut être très riche et la représentation des données évoquée dans le paragraphe 1 permet l'extraction d'informations pertinentes.

Les arbres de décision forment un ensemble de techniques très utilisées en apprentissage artificiel dans un contexte supervisé. Le traitement des données dans leur ensemble et attribut par attribut permet une interprétation des résultats de façon immédiate. Cependant, l'utilisation de cette technique nécessite un très gros travail préalable pour la conception d'un système de classification automatique : il faut fournir un ensemble d'attribut. Ces attributs n'ont pas besoin d'être tous pertinents car l'algorithme les filtre naturellement et seuls les attributs utiles seront conservés. Cependant, le nombre d'attributs décrivant une séquence d'actions dans le temps est potentiellement infini. L'étude présentée dans [Damez & al. 2005] en propose un certain nombre pour reconnaître automatiquement le niveau d'expertise informatique d'un utilisateur dans sa façon de naviguer.

Le clustering hiérarchique tel qu'il a été présenté ici n'inclut pas de filtre lors du calcul de la distance entre individus. L'inconvénient majeur réside donc dans la possibilité de noyer l'information pertinente, décrite par certains attributs seulement. Cependant, cette méthode est très efficace lorsque l'on dispose d'une matrice de similarité (ou distance), qui compare tous les individus à tous les autres, comme lorsque l'on utilise un algorithme avec traitement séquentiel deux à deux. Une autre propriété de cette technique est qu'elle ne nécessite pas de connaissance a priori sur ce que l'on souhaite découvrir. Le nombre de classes optimal décrivant une population peut être choisi selon ce que l'on cherche à décrire.

Les algorithmes classiques de regroupement (K-moyennes et C-moyennes floues) possèdent la même propriété. Les données sont regroupées autour de centres, dont le nombre est à fixer a priori, et peuvent définir des prototypes de classes. Ces prototypes définissent des individus dont les caractéristiques sont très représentatives des individus de leur classe. Ainsi, les valeurs des attributs des prototypes sont proches de celles des attributs des individus de la classe.

Le clustering spectral est également un algorithme de regroupement. Son principal avantage réside dans le fait que le nombre de groupes optimal formé par un ensemble d'individus, selon des critères de compacité et de séparabilité comparables, n'est pas une connaissance à définir a priori. Par ailleurs, les groupes formés ne le sont pas seulement par des individus qui sont similaires, mais par des individus qui possèdent

également le même comportement de dissimilarité par rapport aux données des autres groupes.

3. Bilan sur l'apprentissage artificiel

Les méthodes de fouilles de données pour l'extraction d'informations sont très nombreuses. Nous avons présenté dans ce chapitre celles qui nous semblent correspondre avec le traitement des données issues des interactions HM. La classification des méthodes proposées ici suit les recommandations évoquées dans le Chapitre 8 et le Chapitre 9 pour la conception d'un système automatique à vocation d'aide pédagogique. Ainsi, les descripteurs cognitifs permettant la classification d'un élève peuvent être de nature séquentielle ou matricielle. Par exemple, lors de la consultation d'un cours sur une interface hypermédia, plusieurs types de descripteurs sont possibles. Le professeur peut vouloir que l'élève consulte le document X avant le document Y (traitement séquentiel des données), ou bien vouloir que l'élève passe au minimum 10 min dans la section exercices pour réviser telle notion (traitement matriciel des données).

La plupart de ces méthodes nécessite un paramétrage qui n'est généralement pas très intuitif. Ceci confirme les remarques formulées dans la discussion du Chapitre 9 sur l'automatisation du processus d'aide pédagogique.

Chapitre 11. Expériences de fouille de données sur les traces

Nous avons mené plusieurs expériences afin de récolter des traces suivant la méthode décrite dans le Chapitre 5. Nous les avons ensuite étudiées, tantôt avec l'outil de visualisation proposé dans le Chapitre 6, tantôt avec des méthodes d'apprentissage artificiel décrites dans le Chapitre 10. Des exemples d'analyse avec l'outil de visualisation de données provenant des interactions HM ont été présentés dans le Chapitre 7.

Nous présentons ici les résultats des analyses effectuées avec des méthodes d'apprentissage artificiel. La principale activité à analyser pour notre cas d'application est la navigation sur un hypermédia. Celle-ci est sujette à beaucoup de processus cognitifs [Carmel & al, 1992] en particulier lors de la recherche d'informations, de l'apprentissage de connaissances et la restitution de celles-ci pour l'évaluation. Comme nous l'avons présenté dans l'état de l'art, cette activité peut être modélisée sous trois points de vue qui forment une synthèse du dialogue HM : l'individu, l'interaction et la machine.

Le premier exemple analyse le comportement d'utilisateurs dans un milieu connu d'expérimentation favorable : l'interface est contrôlée et la navigation guidée. Nous y présentons une étude de classification d'utilisateurs selon leurs niveaux d'expertises informatiques pour la recherche sur hypermédia. Dans le second exemple, les données sont issues d'un cas réel d'application. Les utilisateurs observés sont des élèves d'une classe de 4^{ième} lors d'un cours de Science de la Vie et de la Terre. Le professeur a donné plusieurs consignes pour occuper les élèves pendant un cours de deux heures environ. Toutefois, nous tenons à faire remarquer que les élèves observés ainsi que le professeur n'avaient pas particulièrement l'habitude d'utiliser des dispositifs informatiques lors d'un cours. De ce fait, les comportements ont été plus ou moins précis et l'accès à d'autres types de médias tels que des jeux ou l'Internet ont été possibles pendant le cours.

1. Consultation dirigée d'un hypermédia clos

Le protocole expérimental de cette expérience comprenant un descriptif des méthodes et outils, du déroulement de l'expérimentation ainsi qu'un résumé des données récoltées est décrit en Annexe à l'Expérimentation 1.

1. Problématiques et hypothèses

Chercher de l'information dans un environnement hypertexte est sujet à beaucoup de processus cognitifs [Carmel & al, 1992]. Comme nous cherchons à modéliser ces processus, l'interface de notre expérience est une page Internet. Nous avons demandé à un panel d'utilisateurs d'accomplir un certain nombre de tâches sur une interface web. Nous avons qualifié cette consultation de dirigée car, l'expérience se révélant d'une durée assez courte, les utilisateurs n'ont pas été perturbés lors de leur passation et la consultation des documents a été attentive.

Cette expérience a été conduite auprès de 2 types d'utilisateurs : ceux maîtrisant facilement et régulièrement une interface de type web, et des utilisateurs novices étant peu familiers avec un environnement informatique. C'est justement cette différence d'habileté qui doit être caractérisée par le système. Une trentaine de personnes ont participé à notre expérience, et un peu moins de la moitié d'entre eux étaient de véritables novices.

2. Etudes et Résultats

Nous avons pratiqué deux méthodes d'apprentissage des tâches de l'utilisateur. Nous avons implémenté la méthode séquentielle et la méthode experte, décrites dans le Chapitre 9.

Pour la méthode experte, nous avons choisi d'établir cinq tâches correspondant aux quatre clics obligatoires que l'utilisateur doit effectuer sur les boutons « Afficher la première question » (1 fois) et « Question suivante » (3 fois). Nous avons créé un fichier au format XML décrivant ces quatre tâches et une simple routine linéaire utilisant ce fichier et le comparant à celui des traces a permis de faire apparaître les tâches dans les actions des utilisateurs.

La méthode séquentielle a également été appliquée sur l'ensemble des traces, car l'expérience révèle, de par sa nature, une séquence de tâches à accomplir. L'analyse par cette méthode a fait apparaître trois tâches supplémentaires correspondant aux clics sur la zone de saisie permettant à l'utilisateur de taper la réponse au clavier. Nous nous attendions à trouver quatre clics sur la zone de saisie, correspondant aux quatre réponses

que l'utilisateur devait fournir pour mener à bien notre expérience. Mais nous nous sommes aperçu qu'un utilisateur n'avait pas répondu à une question car il avait accidentellement double-cliqué sur le bouton « Question suivante » et n'avait donc pas saisi quatre réponses. Comme nous l'avons remarqué précédemment (Chapitre 9.1.1), l'utilisation de la méthode séquentielle pour l'apprentissage des tâches des utilisateurs peut donc être risquée. Le tableau suivant décrit les deux modèles de tâches étudiés :

1ière étape	1ière étape	Bouton « Afficher la première question »
	2ième étape	Clic dans la zone de saisie de la réponse
2ième étape	3ième étape	Bouton « question suivante »
	4ième étape	Clic dans la zone de saisie de la réponse
3ième étape	5ième étape	Bouton « question suivante »
4ième étape	6ième étape	Bouton « question suivante »
	7ième étape	Clic dans la zone de saisie de la réponse
5ième étape	8ième étape	Validation de la réponse et fermeture du logiciel

Nous avons voulu comparer l'efficacité de deux entropies pour notre système : l'entropie de Shannon et l'entropie floue définie dans [Ramdani, 1994]. Pour cette expérimentation, nous avons effectué une validation croisée à 10 blocs sur l'ensemble des données recueillies. Cela signifie que nous avons séparé notre ensemble d'apprentissage en 10 sous ensembles de taille égale, et que nous avons vérifié le taux moyen de bonne classification sur chacun de ses 10 sous ensembles. Cette opération réalisée 10 fois pour 10 découpages différents de la base permet d'estimer l'erreur de généralisation de la classification de l'algorithme (en supposant que l'ensemble des données servant de base d'apprentissage soit représentatif des données que l'on souhaite classifier dans une application réelle). Les résultats du taux de classification de l'arbre de décision flou à chaque tâche sont présentés dans la Figure 35.

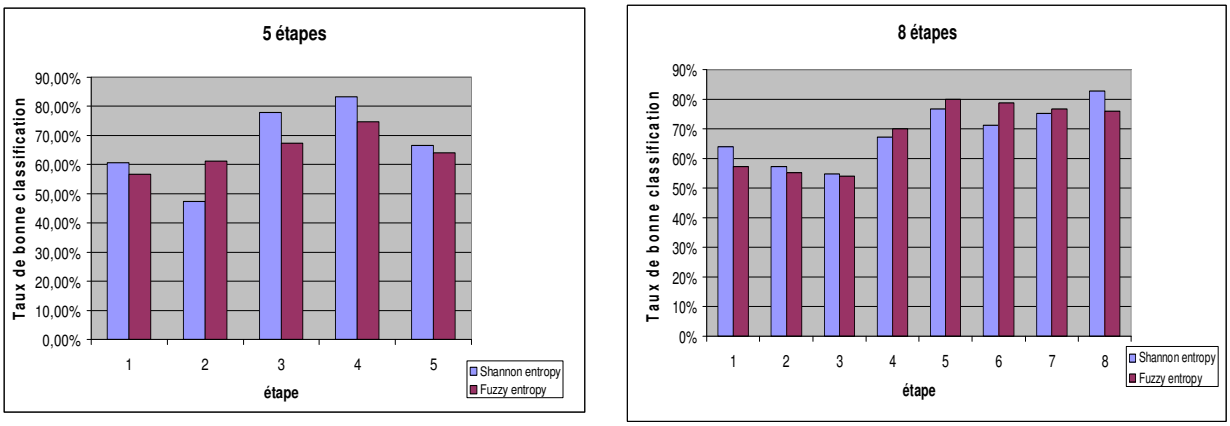

Figure 35. Résultats de la classification des utilisateurs lors de la consultation dirigée d'un hypermédia clos.

Ces résultats sont meilleurs dans le cas où la description de la tâche est en 8 étapes (extraction de la tâche par la méthode séquentielle). En effet, les descripteurs cognitifs généraux, qui sont les mêmes tout au long de l'expérience, décrivent de manière plus précise les actions liées à l'accomplissement des tâches. Il est tout à fait logique et intuitif que plus les étapes contiennent d'actions, plus la description cognitive est imprécise.

Notons que nous avons incorporé dans nos descripteurs cognitifs les conseils que l'on cherche à promulguer aux utilisateurs novices, à savoir utiliser la fonction « recherche » du navigateur. Autrement dit, les descripteurs cognitifs particuliers décrits dans le Chapitre 9 forment un sous ensemble des descripteurs cognitifs globaux utilisés pour la classification. Notons également que les étapes 3 et 4 dans la description à 5 tâches (à gauche), et les étapes 5, 6 et 7 dans la description à 8 tâches (à droite), donnent de meilleurs taux de classification. Ceci s'explique très simplement par le fait que c'est à ces étapes que les experts ont fait appel à leurs compétences et ont utilisé les fonctions de recherche. La classification est donc plus facile pour l'arbre de décision à ces étapes.

La Figure 36 montre un exemple d'arbre de décision flou construit pour la discrimination d'utilisateur novice et expert. L'utilisation des arbres de décision comme algorithme pour classifier les individus fait ici office de sélection d'attribut. En effet, dans l'exemple présenté, seuls 3 descripteurs globaux sont utilisés pour la discrimination, alors que 26 attributs ont été extraits des traces par l'expert.

La discrétisation floue des attributs assouplit la rigidité d'une coupe franche sur des valeurs numériques. Ainsi, lorsqu'un point de coupure c est calculé sur un domaine de valeurs $\{v_1, \ldots, v_n\}$ d'un attribut, deux sous ensembles flous lui sont associés. La valeur

v_i la plus proche du point de coupure c est utilisée pour définir la fonction d'appartenance des sous ensembles flous correspondant. Un sous ensemble flou se définit ainsi par deux point de coupure c_a et c_b respectivement proche de valeurs v_i et v_j :

$$\left\{c_a - 2\left|c_a - v_i\right|, c_a + 2\left|c_a - v_i\right|, c_b - 2\left|c_b - v_j\right|, c_b - 2\left|c_b - v_j\right|\right\}$$

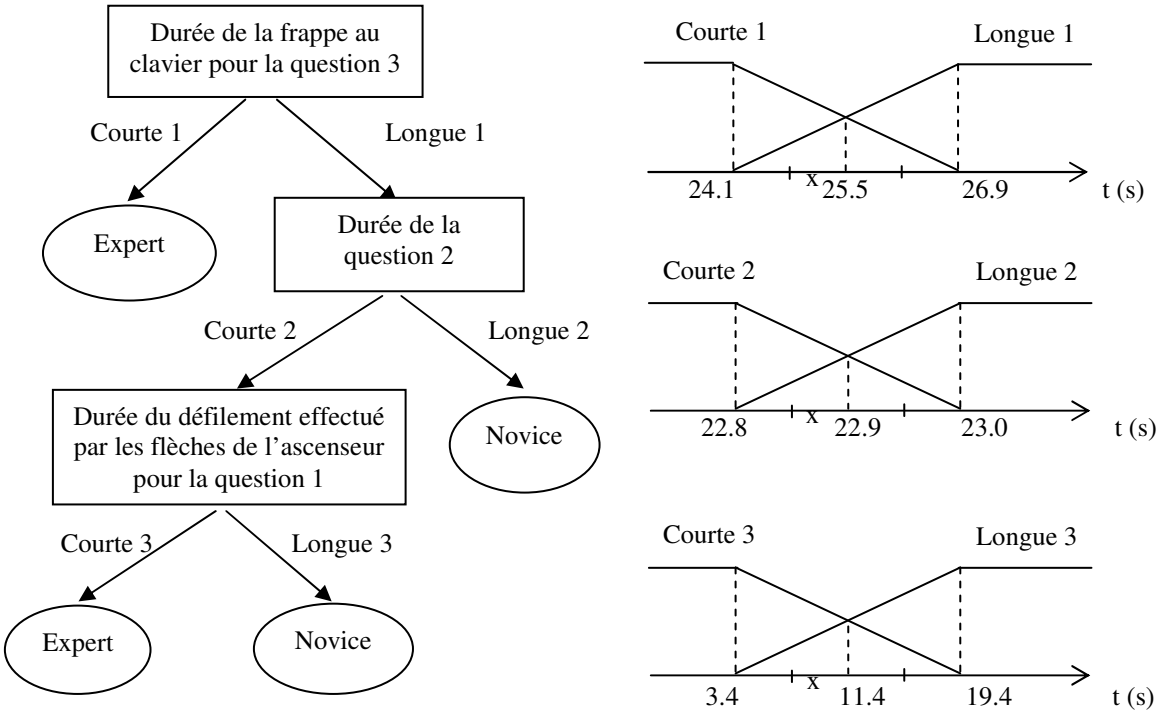

Figure 36. Exemple d'arbre de décision flou pour la classification d'utilisateurs en ligne

La plateforme pour la pédagogique présentée dans le Chapitre 9 peut utiliser des arbres de classification tels que celui-ci pour déterminer si l'individu en interaction avec l'interface nécessite un conseil. L'arbre présenté si dessus présente plusieurs règles pour déterminer si un individu est un expert ou un novice. Par exemple, si la durée de la frappe au clavier pour la question 3 est longue (supérieure à 25,5 sec.) et que la durée globale pour répondre à la question 2 à également été longue (supérieure à 22,9 sec), alors nous pouvons en déduire à 80% que l'individu est un novice dans l'utilisation d'un navigateur internet. Un conseil correspondant à la réalisation de la tâche 3 peut alors se révéler utile, comme l'utilisation de la fonction de recherche pour répondre à la question.

2. Consultation semi-ouverte d'un hypermédia clos

Le protocole expérimental de cette expérience comprenant un descriptif des méthodes et outils, du déroulement de l'expérimentation ainsi qu'un résumé des données récoltées est décrit en Annexe à l'Expérimentation 3.

1. Problématiques et hypothèses

Afin d'expérimenter un apprentissage artificiel sur des données issues d'un cas réel d'utilisation de ressources pédagogiques en situation d'apprentissage humain, nous avons élaboré une expérimentation pour récolter des données lors d'un cas réel d'apprentissage. Suite à une observation attentive du comportement global d'une classe dans la salle multimédia d'un collège, nous nous sommes aperçus que plusieurs paramètres précédemment utilisés pour calculer des statistiques de comportements n'étaient pas exploitables dans un cas réel.

En effet, il apparaît que l'utilisation du dispositif numérique n'est pas régulière et l'utilisateur n'est pas constamment attentif aux réactions de l'interface de l'ordinateur. Ainsi, le professeur a dû intervenir régulièrement pour plusieurs raisons. Il a dû rappeler les consignes de l'exercice, présenter une fonctionnalité du manuel scolaire numérique, ramener de l'autorité dans la classe ou interroger les élèves individuellement. Le professeur a également utilisé un vidéoprojecteur afin de montrer à l'ensemble de sa classe son interaction avec le manuel numérique mis à disposition de la classe.

De la même façon, les élèves ont parfois accompli des actions parfois déconnectées de leur apprentissage. D'autres logiciels, comme les messageries instantanées, les navigateurs Internet ou d'autres jeux du système d'exploitation ont été exécutés pendant le cours. Les élèves ont également dialogué entre eux, tantôt à propos de leurs apprentissages ou de la manipulation du logiciel, tantôt pour d'autres raisons extrascolaires. Ils ont également conservé l'utilisation de leur cahier de classe traditionnel, sur lequel le professeur leur à demandé de prendre des notes.

La connexion Internet de l'établissement s'est parfois révélée défectueuse, ainsi que certains ordinateurs, qui ont dû être redémarrés parfois plusieurs fois pendant la séance. Ceci explique notamment que toutes les traces ne représentent pas des interactions de durées équivalentes.

Afin de ne pas être influencé par des connaissances extérieures à celles relevées par notre logiciel de récolte de traces, nous avons volontairement choisi de ne pas assister au cours de la session analysée et présentée ici. Les données récoltées pendant

cette expérience ont donc été extrêmement difficiles à analyser, tant les conditions matérielles durant cette expérience ne reflètent pas réellement des conditions idéales.

2. Etudes et Résultats

Afin de pallier toutes les difficultés d'expérimentations présentées précédemment, nous avons choisi d'analyser les données à l'aide d'algorithmes permettant de multiplier les comparaisons séquentielles, qui nous semblent être le modèle le plus exploitable. Nous avons donc étudié les similarités créées par l'algorithme du String Kernel illustré au Chapitre 10 par la Figure 32. Cette technique calcule l'apport, en termes de similarité, par rapport aux autres individus, d'un évènement particulier sur une interaction globale. Elle s'implémente avec la technique de la programmation dynamique, ce qui permet d'observer son évolution de façon incrémentale. Elle est donc utilisable par un système fonctionnant en temps réel.

Nous avons étudié les similarités du String Kernel créées avec différents paramètres. Le facteur dégradant λ a été décliné sur l'ensemble {0.1, 0.5, 0.9} et nous avons étudié plusieurs longueurs de sous-séquences, variant de 2 à 6 avec un pas de 1. La définition de cet algorithme implique logiquement que l'information contenue dans une similarité calculée sur une longueur de sous-séquence égale à *n* est comprise dans l'information de la similarité calculée sur une longueur de sous séquence égale à *n-1*.

$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2$$

L'information la plus représentative de l'intersection de l'ensemble des sous-séquences communes à deux séquences est donc calculée pour une longueur de sous-séquence égale à 2. Afin d'étudier les résultats et d'extraire l'information la plus significative de cet algorithme, nous avons principalement exploré les résultats produits avec un facteur dégradant de 0.9, qui produit les similarités les plus élevées.

Lorsqu'une similarité entre deux individus augmente du fait d'une action d'un des deux utilisateurs, cela signifie que cette action rapproche globalement le comportement de ces individus. Cette technique permet de caractériser les éléments des séquences d'action, et lorsque l'on regarde attentivement les séquences des élèves ayant des facilités d'interaction et/ou d'apprentissage, par rapport aux élèves éprouvant des difficultés, les similarités évoluent de façon particulière et des indices concernant les éléments de ces séquences peuvent être extraits.

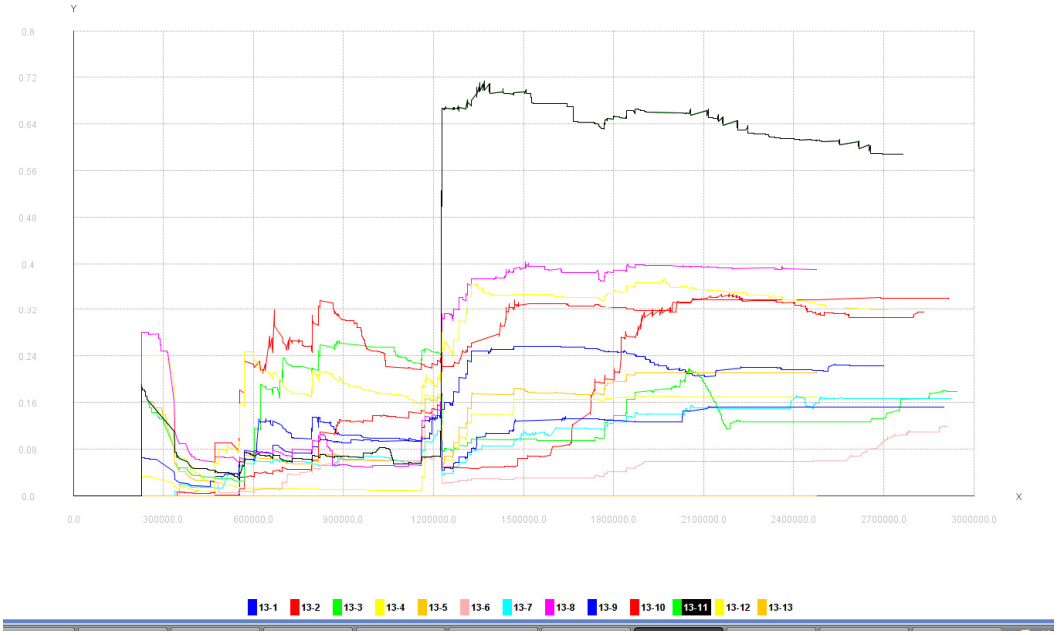

Figure 37. Evolution de la similarité entre tous les individus et un individu de référence en fonction du temps

La Figure 37 présente l'évolution de la similarité, en fonction du temps, entre tous les individus de l'expérience et un individu de référence. Il apparaît que quelques évènements particuliers de la séquence de référence créent un bond dans sa similarité par rapport à un autre individu (au milieu de la figure, la similarité passe de 0,08 à 0,66 en quelques évènements). Cet indice de comportement des utilisateurs est repérable automatiquement. Ces évènements tracent une activité désordonnée de la part des deux individus qui ont, de façon répétée, interagit avec deux objets du manuel électronique :

	Contexte	Cible
Objet 1	#document uri=/svt3/manuel.xul,	Magicviewer res=SVT-part2-chap1-doc6.html
Objet 2	#document uri=/SVT-part2-chap1-doc6.html,	P id=SVT-part2-chap1-p42,

Les interactions avec l'objet 1 sont le signe d'une manipulation du conteneur de document, appelé *magicViewer*. Alors que la description de l'objet 2 prouve que l'interaction a eu lieu avec le document lui-même (l'objet *P* représentant un paragraphe du texte du document). La distinction entre le conteneur de document et le document ne semble donc pas être très claire dans l'esprit des utilisateurs. A une échelle moins remarquable que celle présentée dans la Figure 37, ces évènements et l'évolution des similarités correspondantes, sont repérables dans d'autres séquences et traduisent un

défaut d'affordance dans l'interface proposée. En effet, les actions effectuées pendant ses périodes sont de multiples évènements de clic de souris ainsi que des évènements de glisser/déposer qui n'ont manifestement pas été réalisées de façon aisée tant ils ont été répétés. Ils correspondent à une volonté de mettre dans une disposition particulière les documents présentés dans un conteneur, et cette manipulation ne semble pas facile à mettre en œuvre.

Ces *sauts* dans la similarité peuvent être repérés automatiquement, et une alerte peut être lancée permettant au professeur d'intervenir ou à un système automatique de proposer son aide.

Grâce à une enquête menée au début du cours nous avons récolté quelques informations sur le profil des élèves de la classe. Ainsi, les élèves des premiers rangs sont globalement les élèves ayant des facilités dans l'acquisition de connaissance, et ce sont également ces élèves qui ont un ordinateur à la maison et se servent régulièrement de l'Internet. En suivant la description de la plateforme présentée au Chapitre 9 et les observations sur les théories pédagogiques distinguant les comportements des experts par rapport aux novices (cf. Chapitre 8.4), nous avons cherché les évènements qui rapprochent globalement le comportement d'un individu aux élèves ayant des facilités.

L'utilisation de l'arbre de navigation semble être privilégiée par les élèves ayant des facilités dans l'utilisation de dispositif hypermédia (Objet 3 ci-dessous). En effet, il apparaît qu'aux débuts des sessions de ces élèves, un nombre d'actions importantes ont été effectuées sur cet outil car les similarités correspondant aux instants où ces outils ont été utilisés augmentent plus facilement.

La mesure de similarité du string kernel a initialement été introduite pour la classification de documents textuels. Nous avons présenté précédemment (voir Chapitre 10.2.1) la loi de Zipf qui cherche à trouver parmi un ensemble de mot, ceux qui sont les plus représentatifs d'un document tout en n'étant pas représentatif du corpus de documents traités. Le string kernel vérifie naturellement cette propriété.

Ceci nous apporte une information très importante lorsque l'on regarde les évolutions des similarités au cours de la séquence. En effet, lorsqu'un individu fait une série d'actions que personne n'a encore effectuée, sa similarité avec les autres individus va décroître. Au contraire, si un individu fait une série d'actions déjà effectuées par d'autres, sa similarité avec ces individus augmentera. Ces actions caractéristiques ne sont pas nécessairement continues.

En relevant les interactions qui créent une augmentation de la similarité chez tous les individus, on peut mettre en évidence les éléments qui forment les passages obligés dans le parcours des élèves. L'ordre des objets ayant été sujet à interactions présenté dans le tableau ci-dessous montrent que les élèves ont dû d'abord parcourir le manuel, puis ont cliqué sur un bouton correspondant à une activité, et enfin ont enregistré leurs travaux du classeur :

	Contexte	Cible
Objet 3	#document uri=/svt3/manuel.xul,	treechildren value=…
Objet 4	#document uri=SVT-part2-chap1-blocQ2.html,	A href= magicViewer.repondre(Les voies de pénétration des microbes dans l'organisme - Activités);
Objet 5	#document uri=nsxa://svt3/eleve//classeur.cfa/classeur.xul,	menuitem id=Enregistrer Sous...,

Les difficultés rencontrées dans la récolte des traces pour cette expérience, ne nous ont pas permis d'obtenir des résultats satisfaisants permettant de classifier automatiquement les individus selon leur expertise en utilisant les mesures de similarités calculées. Nous avons implémentés un clustering hiérarchique ascendant incrémental et sa visualisation en dendrogramme associée, qui nous semble être un algorithme offrant une grande souplesse dans son interprétation et son utilisation. Le comportement extrêmement hétérogène des élèves tout au long de la séance de cours ne nous a pas permis de trouver des critères satisfaisant pour une catégorisation automatique par une étude séquentielle des données.

3. Bilan sur nos expériences de fouilles de données

Nous avons présenté dans ce chapitre deux expériences de fouilles de données sur des traces issues des interactions homme-machine. Ces deux expériences sont de natures différentes et nous ont permis d'accumuler des connaissances sur la façon d'établir un protocole d'expérimentation dans un contexte semi-simulé et pour une application réelle. Les données ainsi récoltées révèlent des propriétés différentes dans les analyses que nous avons effectuées. Elles nous ont permis de proposer deux cas d'applications pour faciliter l'apprentissage humain.

Nous avons présenté dans l'état de l'art (cf. Chapitre 4.2) quelques exemples d'applications où la fouille de données est utilisée au cas par cas. La première expérience présentée ici fait de même et nécessite un entrainement du système sur une base de données représentative des cas réels d'application. Après une phase de préparation et de paramétrage, le système peut alors fonctionner comme celui décrit au

Chapitre 9 et présente des performances intéressantes. La phase d'entrainement permet de surveiller le bon comportement du système, mais peu se révéler difficile à implémenter. Nous avons appliqué plusieurs algorithmes, certains nécessitant une connaissance experte des données à traiter, comme l'extraction de la plus longue sous séquence commune, et d'autres plus généraux, comme les arbres de décisions flous, qui offrent notamment une grande souplesse dans la catégorisation des descripteurs cognitifs.

La deuxième expérience ne nécessite pas de connaissance a priori sur le système ou sur la tâche à réaliser et peut fonctionner en temps réel sans entrainement. De la même façon que pour l'expérience précédente, l'application repose sur une connaissance des utilisateurs et utilise le savoir de certains individus, les plus performants, pour le faire profiter à d'autres. Basé uniquement sur le traitement et l'analyse séquentielle des données, cette méthode nous semble appropriée pour la conception d'aide automatique appliqué notamment dans le contexte d'une classe scolaire. Néanmoins, des précautions dans l'implémentation du système devront être prises car, le système fonctionnant en temps réel peut vouloir conseiller un mauvais comportement observé chez un bon élève.

Conclusion de la troisième partie

En nous plaçant dans le cadre d'une analyse automatique de traces issues des interactions homme-machine, nous avons étudié les méthodes et modèles qui permettent de concevoir des systèmes de personnalisation de l'enseignement sur un dispositif numérique. Les modèles issues des théories pédagogique en psychologie cognitive, comme le design pédagogique, l'affordance des outils numériques, les styles d'apprentissages et notamment les différences de comportements cognitifs entre les utilisateurs apprenants experts et les novices, nous ont permis de proposer des méthodes d'analyses permettant d'automatiser la personnalisation de l'enseignement.

Nous avons proposé des recommandations pour l'implémentation de ces méthodes dans un système de personnalisation automatique des interfaces qui permet d'adapter l'interaction à chaque individu. Des processus d'entrainement « hors ligne » permettent de paramétrer et contrôler précisément ces adaptations. Le système décrit est collaboratif, c'est-à-dire qu'il cherche à faire profiter certains utilisateurs de l'expérience d'autres utilisateurs. Les adaptations proposées ne sont donc pas crées, mais extraites automatiquement et le système permet à un concepteur de qualifier leur pertinence. Le fonctionnement du système « en ligne » permet d'enrichir et améliorer le système d'entrainement en temps réel tout en proposant des conseils tirés de ces expériences précédentes.

Ce système doit permettre l'instanciation de méthodes d'analyses de données que nous avons également étudiées. Ces techniques, appartenant à la famille des méthodes d'apprentissage artificiel, sont aussi nombreuses que complexes, et nous nous sommes attachés à les présenter dans un but pratique pour leurs utilisations dans un système réel. Ainsi, nous les avons présentées dans une taxonomie permettant de traiter différemment les données issues de l'interaction en précisant les avantages et inconvénients de chacune.

Le dernier chapitre de cette partie présente deux analyses sur des données issues d'expériences que nous avons menées. Nous avons ainsi pu appliquer des méthodes

comparables à celles présentées dans l'état de l'art, en conservant l'intelligibilité de la personnalisation de l'interaction. En nous plaçant dans le cadre d'un apprentissage scolaire réalisé dans une classe, notre système permet au professeur de suivre l'activité de ses élèves et personnaliser son enseignement en respectant le profil d'apprenant de l'élève. Ces deux expériences nous ont donné l'occasion d'implémenter et tester un grand nombre d'algorithmes présentés dans le chapitre précédent, et nous nous sommes attachés à présenter les résultats des applications décrites dans notre plateforme pour la pédagogie.

Conclusions générales et perspectives

L'étude des données provenant directement de l'interaction homme-machine n'a pas encore atteint son apogée. Les outils et les études existant dans la littérature permettent certes la conception de modèles théoriques très séduisants, mais cette discipline à part entière n'est pas encore arrivée à maturité.

Dans cette thèse, nous avons présenté dans la première partie, un état de l'art de l'avancement des recherches scientifiques dans ce domaine. Ainsi, nous avons pu remarquer que la caractérisation et la détection automatique d'informations pertinentes sur le dialogue homme-machine sont des tâches réalisables. Cependant, peu des modèles existants permettent la personnalisation de l'interaction, réalisable de façon générique, après la phase de reconnaissance du comportement.

Aussi, nous avons choisi de replacer les données analysées au centre de l'étude. En étudiant les méthodes de récolte et de traitement des données issues de l'interaction, nous avons recensé les propriétés essentielles pour un grand nombre de modélisation. Cela nous a amené à produire un nouveau formalisme de génération de traces centrées sur les différentes entités du dialogue homme-machine : une structure hiérarchique sémantiquement enrichie qui rend compte de l'état de la machine observée, les séquences d'actions des utilisateurs qui organisent en reflétant des caractéristiques essentielles du comportement humain -selon les modèles d'analyses en psychologie cognitive- et ceci en traçant l'intégralité de l'interaction pour rendre compte du dialogue de façon objective. Le formalisme que nous proposons est d'une complexité faible en implémentation et respecte les standards de données actuelles facilitant leurs traitements.

Pour analyser ces données structurées, nous avons proposé une visualisation ouvrant de nouvelles perspectives. En replaçant l'activité des utilisateurs dans l'intégralité de leurs parcours, notre représentation des données facilite l'analyse pour la caractérisation cognitive de comportements humains. Cette méthode représente la totalité de l'information présente dans les traces de notre modèle. Plusieurs outils facilitant la lecture et l'interprétation utilisent de nouveaux algorithmes pour automatiser leurs fonctionnements. Deux exemples d'analyses ont été présentés. En comparant l'utilisation de deux interfaces, la première analyse évalue l'impact cognitif d'un nouvel outil de navigation hypermédia. Le second exemple met en évidence, dans le contexte intégral de l'interaction, une séquence de tâche permettant de caractériser chacun des utilisateurs.

Le cadre d'application de nos analyses étant le milieu scolaire et l'apprentissage humain, nous avons présenté quelques notions théoriques qu'il nous paraît important de connaître pour l'implémentation d'un système d'assistance à la pédagogie. Ainsi, les

différences de comportement entre un individu expert et un novice qui peuvent être reconnues automatiquement par un système intelligent, permettent de concevoir une personnalisation d'aide automatique performante. Suivant une typologie de traitement de données, nous avons également décrit des méthodes algorithmiques pour l'extraction d'informations pertinentes sur les données issues des interactions homme-machine.

Afin de mettre ces méthodes en pratique, nous avons effectué plusieurs expérimentations d'apprentissage simulés et réelles. Deux exemples d'analyses par fouille de données sur notre modèle de traces ont été présentés. Une première analyse utilise des arbres de décisions floues pour catégoriser des descripteurs cognitifs décrits dans un espace vectoriel alors que la seconde étude présente une analyse des similarités créées par une fonction noyau traitant les individus en séquences. Ces études montrent qu'il est possible de personnaliser automatiquement l'interaction HM dès lors que l'on sait si une personne est un expert ou un novice dans la tâche qu'il est en train d'effectuer.

Perspectives et travail futur

Notre travail ouvre de nouvelles perspectives pour l'analyse de comportements d'utilisateurs et particulièrement d'utilisateurs en situation d'apprentissage dans un milieu scolaire.

Les applications de travaux collaboratifs, et plus particulièrement d'apprentissages collaboratifs, trouvent naturellement un support commun permettant de nombreuses perspectives. Nous avons mis l'accent sur une analyse qualitative des traces des interactions hommes-machine pour l'extraction d'informations pertinentes afin d'apporter une aide à la réalisation de tâches en général, et à la réalisation de la tâche d'apprentissage en particulier.

D'autres outils offrant d'autres types d'interprétations de la visualisation proposée dans le Chapitre 6 peuvent être ajoutés. La lecture d'un plus grand nombre de traces et une représentation non contraignante pour les traces longues ou les interfaces complexes peut encore être facilitée. Ainsi, les approches interactives telles que les fonctions de zooms améliorés en *fisheye* peuvent être envisagées.

Les méthodes de fouilles de données pour l'apprentissage automatique de concept sont très nombreuses, et nous avons tenté ici d'en présenter un certain nombre, bien adaptées à notre problématique. Cette liste n'est évidemment pas exhaustive, et d'autres méthodes peuvent être utilisées pour modéliser les processus cognitifs impliqués dans la

résolution d'une tâche par un individu (algorithme génétique, réseaux de neurones, système expert, etc…)

De la même façon, la description du modèle de tâche peut être enrichie, par exemple en fournissant une typologie des aides à détecter. Une étude sur la sémantique des objets manipuler doit permettre d'automatiser certains processus de description cognitive de l'activité humaine.

Perspectives pour l'entreprise

Cette thèse ouvre de nombreuses perspectives de développement pour l'entreprise. En effet, nous montrons dans ce mémoire que de multiples techniques d'intelligence artificielle existent pour modéliser le comportement de l'apprenant. Nous avons présenté quelques exemples issus de la littérature et de nos propres expériences pour la reconnaissance des comportements humains en vue d'une personnalisation. Nous avons également exposé quelques théories des psychologues cognitivistes pour la conception de nouveaux outils logiciels pour tous les acteurs du monde éducatif.

Le professeur peut maintenant bénéficier d'un environnement de suivi de l'activité d'une classe avec un logiciel de traces et son outil de visualisation. Une aide à la planification du cours en fonction de l'avancement de la classe peut être proposée au professeur afin de suivre plus attentivement ses élèves et ainsi personnaliser son enseignement. La caractérisation automatique des élèves fournit également une aide à la conception des évaluations selon les styles d'apprentissage ou le niveau d'expertise.

L'analyse des traces liées à l'utilisation des ressources numériques est généralement utilisée pour l'étude des usages et de l'utilisabilité des outils. Notre étude montre qu'elle autorise également la personnalisation automatique de l'interaction HM selon le type d'utilisateur. Plus particulièrement dans le cas de l'apprentissage humain, nous avons montré que l'apprentissage artificiel aide à la personnalisation de l'apprentissage.

Les éditeurs de ressources numériques pédagogiques ont donc de beaux jours devant eux. En effet, si la personnalisation de l'interaction coté humain est possible, seul l'auteur de la ressource prévoit les modalités de personnalisation de l'interaction coté machine.

Cette étude montre que le rôle du professeur est central dans le système éducatif moderne et qu'un système automatique observant le comportement des élèves en situation d'apprentissage peut fournir une aide à la personnalisation de son

enseignement. Ainsi, nous pensons qu'une pédagogie personnalisée efficace doit s'envisager avec un support logiciel important.

Table des figures

Bibliographie

Agichtein & al., 2006	Agichtein E., Brill E., Dumais S., Ragno R. 2006. Learning User Interaction Models for Predicting Web Search Result Preferences. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. Seattle, Washington, USA. Session*: *User behavior and modelling. p.3-10.*
Agrawal & al., 1995	Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T. 1995. The Quest Data Mining System. *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, Montreal, Canada. August.*
Anjo & Efimova, 2006	Anjewierden A. and Efimova L. 2006. Understanding weblog communities through digital traces: a framework, a tool and an example. *In Proceedings International Workshop on Community Informatics (COMINF 2006), pp. 279-289, Montpellier, 2006 (November). Springer, LNCS 4277.*
Annett & Duncan, 1967	Annett J., and Duncan K. D. 1967. Task analysis and training design. *Journal of Occupational Psychology, 41, 211-221.*
Akoulchina, 1998	Akoulchina I. 1998. SAGE un agent intelligent d'interface pour un hypermédia à base de connaissance taxinomiques fonctionnant dans l'environnement du Web. *Thèse soutenue au Lip6 UPMC.*

Ayres & al., 2002	Ayres J., Gehrke J., Yiu T. and Flannick J. 2002. Sequential Pattern Mining Using Bitmap Representation, *Proocedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, July.*
Bach & Jordan, 2004	Bach F.R. and Jordan M.I. 2004. Learning spectral clustering. *In S. Thrun, L. Saul, and B. Schoelkopf (Eds.), Advances in Neural Information Processing Systems (NIPS) 16, (long version).*
Balbo & al., 2004	Balbo S., Ozkan N., Paris C. Choosing the right task-modeling notation: A taxonomy. *The Handbook of Task Analysis for Human-Computer Interaction, 2004*
Banerjee & Ghosh, 2001	Banerjee A. and Ghosh J. 2001. Clickstream clustering using weighted longest common subsequences. *In Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining, Chicago, April.*
Baron & al., 2006	Baron M., Lucquiaud V., Autard D., Scapin D.L. *Proceedings of the 18th international conference on Association Francophone d'Interaction Homme-Machine. Montreal, Canada, p. 287 - 288*
Baum & al., 1970	Baum L.E., Petrie T., Soules G. and Weiss N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist., vol. 41, no. 1, pp. 164—171.*
Beck & Woolf, 2000	Beck J. and Woolf B.P. 2000. High-Level Student Modeling with Machine Learning. *Intelligent Tutoring Systems 2000: 584-593*
Benard, 2001	Benard V. 2001. Utilisabilité, usabilité: définition. *http://www.veblog.com/fr/2001/1021-usability-indepth.html.*
Bezdek 1981	Bezdek J. C. 1981. Pattern Recognition with Fuzzy Objective Function Algoritms, *Plenum Press, New York.*
Bidel & al., 2003	Bidel S., Lemoine L., Piat F., Artières T. and Gallinari P. 2003. Apprentissage de comportements utilisateurs de produits hypermedias. *In Extraction et Gestion des Connaissances. Lyon.*
Bransford & al, 1999	Bransford J.D., Brown A.L., and Cocking R.R., 1999. How People Learn: Brain, Mind, Experience, and School, *edited by Committee on Developments in the Science of Learning. Commission on Behavioural and Social Sciences and Education National Research Council.*

Brézillon & Tijus, 2005	Brézillon P., Tijus C. 2005. Une représentation basée sur le contexte des utilisateurs à travers leurs pratiques. *Atelier sur la modélisation utilisateurs et personnalisation de l'interaction homme-machine. EGC 2005. Paris.*
Brien, 1981	Brien R. 1981. Design pédagogique. *Press de l'Université du Quebec, Québec.*
Brusilovsky, 2001	Brusilovsky P. 2001. Adaptive Hypermedia. *User Modeling and User Adapted Interaction, 11, 87-110.*
Card & al., 1983	Card S. K., Moran T. P., and Newell A. 1983. The psychology of human computer interaction. *Lawrence Erlbaum.*
Carmel & al, 1992	Carmel, E., Crawford, S., and Chen, H. 1992. Browsing a hypertext: a Cognitive Study. *IEEE transaction on systems, Man and Cybernetics, 22(5), 865-883.*
Chi & al., 1981	Chi M. T. H., Feltovich J. P. and Glaser R. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5, 121-152.*
Cooley & al., 1997	Cooley R., Mobasher B., Srivastava J. 1997. Web Mining: Information and Pattern Discovery on the World Wide Web. *In: Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI97), Newport Beach, CA, USA, Nov.*
Cornuéjols & Miclet, 2002	Cornuéjols A. & Miclet L. 2002. Apprentissage artificiel : Concepts et algorithmes. *Eyrolles. Août. 638p. ISBN: 2-212-11020-0*
Damez & al. 2005	Damez M., Dang T.H., Marsala C., Bouchon-Meunier B. 2005. Fuzzy Decision Tree for User Modeling From Human-Computer Interactions. *In Fifth International Conference on Human System Learning. Marrakech. Maroc. Novembre, p.22-25.*
Damez & Renaud, 2006	Damez M. et Renaud S. 2006. Visualisation de navigation sur interface graphique pour l'analyse cognitive de parcours. *Atelier de visualisation de données EGC 2006. Lille.*
Damez, 2006	Damez M. 2006. Méthode de récolte de traces de navigation sur interface graphique et visualisation de parcours. *Article pour démonstration de logiciel, EGC 2006. Lille.*
Dang, 2007	Dang T.H. 2007. Mesures de discrimination et leurs applications en apprentissage inductif. *Thèse soutenue à l'Université Pierre et Marie Curie, Paris.*

Delort & al., 2003	Delort J.Y., Bouchon-Meunier B., Rifqi M. 2003. VISS : A Tool for Visualizing Clues about the Users' Information Needs and Their Information-Seeking Tactics. *Poster Proceedings of The Fourteenth International ACM Conference on Hypertext and Hypermedia. Nottingham, United Kingdom.*
Delort, 2005	Delort J.Y. 2005. CONQUERIES: An Agent that Supports Query Expansion. *In Proceedings of the Fifth International Conference on Web Engineering*.
Dhillon & al., 2004	Dhillon I.S., Guan Y., and Kulis B. 2004. Kernel k-means: spectral clustering and normalized cuts. *In Proceedings of the Tenth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Seattle, WA, USA, August 22 - 25, 2004). KDD '04. ACM Press, New York, NY, p.551-556.*
Diaz & al., 2002	Diaz J., Petit J., Serna M. 2002. A Survey of Graph Layout Problems. *In Journal of ACM Computing Surveys. Vol. 34. Issue 3. p.313-356.*
Druganov & al., 2005	Druganov A.N., Dietterich T.G., Johnsrude K., McLaughlin M., Li L., and Herlocker J.L. 2005. TaskTracer: A Desktop Environment to Support Multitasking Knowledge Workers. *International Conference on Intelligent User Interfaces, San Diego, California, USA, 75-82.*
Farenc, 1997	Farenc C. 1997. Ergoval: Une méthode de structuration des règles ergonomiques permettant l'évaluation automatique d'interfaces graphiques. *Thèse soutenue à l'Université de Toulouse I, Toulouse.*
Felder & Silverman, 1988	Felder R.M. and Silverman L.K. 1988. Learning and Teaching Styles in Engineering Education. *Engr. Education, 78(7), 674-681.*
Fisher, 1991	Fisher C. 1991. Protocol Analyst's Workbench: Design and Evaluation of Computer-Aided Protocol Analysis. *Carnegie Mellon University, Pittsburgh.*
Ganascia, 2002	Ganascia J. G. 2002. Extraction of recurrent patterns from stratified ordered trees. *European Conference on Machine Learning, 167-178.*
Genter & Ratterman, 1991	Gentner D., and Ratterman M. J. 1991. Language an the carrer of similarity. *Perspective on thought and language: Interrelations in development, S. A. G. e. J. P. Byrnes, ed., Cambridge University Press., Cambridge, England, 225-277.*

Gibson, 1977	Gibson E. J. 1977. How perception really develops: A view from outside the network. *Basic processes in reading: Perception and comprehension, Erlbaum, ed., 155-173.*
Guzdial, 1993	Guzdial M.J. 1993. Deriving software usage patterns from log files. *Georgia Institute of Technology. GVU Center Technical Report. Report #93-41.*
Hartson & al., 1990	Hartson H.R., Siochi A.C. and Hix D. 1990. The UAN: a User Oriented Representation for Direct Manipulation Interface Design. *ACM Transaction on Information Systems, 8(3), 181-203.*
Hilbert & al, 1997	Hilbert D.M., Robbins J.E., and Redmiles D.F. 1997. Supporting Ongoing User Involvement in Development via Expectation-Driven Event Monitoring. *Tech Report UCI-ICS-97-19, Dept. of Information and Computer Science, Univ. of California, Irvine.*
Hilbert & Redmiles, 2000	Hilbert D.M., Redmiles D.F. 2000. Extracting usability information from user interface events. *ACM Computing surveys, Vol. 32, No. 4, Decembre, pp384-421.*
Holyoak & Thagard, 1995	Holyoak K. J., and Thagard P. 1995. Mental leaps: Analogy in creative thought. *MIT Press, Cambridge, MA, USA.*
Ivory & Hearst, 2001	Ivory, M. Y. and Hearst, M. A. 2001. The State of the Art in Automating Usability Evaluation. *ACM Computing Surveys 2001, Vol. 33, No. 4, 470-516.*
Jéron & al., 2006	Jeron T., Marchand H., et Cordier M.O. 2006. Motifs de surveillance pour le diagnostic de systèmes à événements discrets. *RFIA 2006 (Congrès Reconnaissance des formes et Intelligence Artificielle), Tours, France, janvier.*
John & Kieras, 1996	John B.E. & Kieras D.E. 1996. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Transaction on Computer-Human Interaction, Vol. 3, No. 4, December, p. 320-351.*
John & al., 2002	John B., Vera A., Matessa M., Freed M., Remington R. 2002. Automating CPM-GOMS. *SIGCHI Conference on Human factors in computing systems, Minneapolis, USA.*
Kobsa & al., 2001	Kobsa A., Koenemann J., and Pohl W. 2001. Personnalised hypermedia presentation techniques for improving online custumer relationships. *The knowlegde engineering, 16(2), 111-155.*

Kolb, 1976	Kolb D.A. 1976. The Learning Style Inventory: *Technical Manual, Boston, Ma.: McBer.*
Kort & de Poot, 2005	Kort J. & de Poot H. 2005. Usage Analysis: Combining Logging and Qualitative Methods. *Conference on Human Factors in Computing Systems. Portland, Oregon, USA.*
Leslie & al., 2002	Leslie C., Eskin E. and Noble W.S. 2002. The Spectrum Kernel: A String Kernel for SVM Protein Classification. *In Proceedings of the Pacific Symposium on Biocomputing (PSB 2002). Kaua'i, Hawaii: January 2-7.*
Leslie & al., 2004	Leslie C., Eskin E., Cohen A., Weston J. and Noble W.S. 2004. Mismatch String Kernels for Discriminative Protein Classification. *Bioinformatics, 20:4, pp. 467-476.*
Lesot & Bouchon-Meunier, 2004	Lesot M.-J. & Bouchon-Meunier B. 2004. Cluster characterization through a representativity measure. *Flexible Query Answering Systems, FQAS'04, pages 446-458, Lyon, France.*
Lettkeman & al., 2006	Lettkeman A.T., Stumpf S., Irvine J., Herlocker J.. 2006. Predicting Task-Specific Webpages for Revisiting. *21st National Conference on Artificial Intelligence (AAAI-06), Boston, Ma, July 16-20.*
Levenshtein, 1965	Levenshtein V.I., Methods for obtaining bounds in metric problems of coding theory. 1976. *In IEEE-USSR Joint Workshop on Information Theory, New York, 126-143. (Première publication de cette découverte de 1965 en anglais)*
Lodhi & al., 2002	Lodhi H., Saunders C., Shawe-Taylor J., Cristianini N. and Watkins C. 2002. Text Classification using String Kernels. *Journal of Machine Learning Research 2 419-444*
McCarthy, 1987	McCarthy B. 1987. The 4 MAT System: Teaching to Learning Styles with Right/Left Mode Techniques. *Barrington, IL, EXEL, Inc.*
MacQueen, 1967	MacQueen J. B. 1967. Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297*

Masseglia & al., 1998	Masseglia F., Cathala F., Poncelet P. 1998. The PSP Approach for Mining Sequential Patterns. *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, Vol. 1510, Nantes, France, September, p. 176-184.*
Masseglia & al., 2003	Masseglia F., Poncelet P., Teisseire M. 2003. Incremental Mining of Sequential Patterns in Large Databases. *Data and Knowledge Engineering, vol. 46, no 1, p. 97-121.*
Masseglia & al., 2004	Masseglia F., Poncelet P., Teisseire M. 2004. Extraction de motifs séquentiels : problèmes et méthodes. *In Ingénierie des Systèmes d'Information (ISI), numéro spécial "Extraction de motifs dans les bases de données". Volume 9, n° 3-4. pp 183-210.*
Ng & al., 2001	Ng A. and Jordan M. and Weiss Y. 2001. On spectral clustering: Analysis and an algorithm. *Proceedings of Advances in Neural Information Processing Systems 14.*
Padmanabhan & Yang, 2006	Padmanabhan B. and Yang Y. 2006.Clickprints on the Web: Are there Signatures in Web Browsing Data? *(October 27, 2006). Available at SSRN: http://ssrn.com/abstract=931057*
Parthasarathy & al., 1999	Parthasarathy S., Zaki M., Ogihara M., Dwarkadas S.1999. Incremental and Interactive Sequence Mining. *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99), Kansas City, MO, USA, November, p. 251-258.*
Paterson & Dancik, 1994	Paterson M.S. and Dancik V. 1994. Longest Common Subsequences. *Mathematical Foundations of Computer Science. p.127-142*
Pei & al, 2001	Pei J., Han J., Mortazavi-asl B., Pinto H., Chen Q., Dayal U., Hsu M. 2001 PrefixSpan : Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. *Proceedings of 17th International Conference on Data Engineering (ICDE'01).*
Pernin, 2006	Pernin J.P. LOM, SCORM et IMS-Learning Design : ressources, activités et scénarios. 2004. *Babel - edit -, L'indexation des ressources pédagogiques numériques (journée d'étude du 16/11/2004). ENSSIB - janvier 2006 http://babel.enssib.fr/document.php?id=63*
Piktow & al., 1994	Pitkow J., Bharat K. 1994. Webviz: A tool for world wide web access log analysis. *Advance Proceedings First International World-Wide Web Conference, p. 217-277, May. Geneva, Switzerland*

Quinlan, 1986	Quinlan J.R. 1986. Induction of decision trees. *In Machine Learning 1, pp. 81-106.*
Quinlan, 1993	Quinlan J. R. 1993. C4.5: Programs for Machine Learning. *Morgan Kaufmann, San Mateo, CA, 1993.*
Rabiner, 1989	Rabiner L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77(2):257–286, February.*
Ramdani, 1994	Ramdani M. 1994. Système d'induction formelle à base de connaissances imprécises. *PhD thesis, University Paris VI, France.*
Renaudie, 2003	Renaudie D. Classification automatique de comportements d'élèves apprenant l'algèbre. 2003. *Actes de la conférence Majecstic'2003, Marseille, Octobre*.
Rick, 2000	Rick C. 2000. Efficient Computation of All Longest Common Subsequences. *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory. p 407 – 418.*
Rumbaugh & al., 2004	Rumbaugh J., Jacobsen I., Booch G. UML 2.0. *Campus Press. Décembre 2004.*
Salton & Buckley, 1988	Salton G. & Buckley C. 1988. Term weighting approaches in automatic text retrieval. *Information Processing and Management, 24, 513-523.*
Sanderson & Fisher, 1994	Sanderson P. M. & Fisher C. 1994. Exploratory sequential data analysis: foundations. *Human-Computer Interaction, 9, 251-317.*
Scapin & Pierret-Goldbreich, 1989	Scapin D. L., and Pierret-Golbreich C. 1989. MAD: Une méthode analytique de description de tâches. *Colloque sur l'ingénierie des Interfaces Homme-Machine, Sophia-Antipolis, France, 131-148.*
ScienceDaily, 2005	http://www.sciencedaily.com/releases/2005/01/050111162359.htm
Shawe-Taylor & Cristianini, 2004	Shawe-Taylor J. & Cristianini N. 2004. Kernel Methods for Pattern Analysis. *Cambridge University Press, ISBN: 0521813972*
Shen & al., 2006	Shen J., Li L., Dietterich T.G., Herlocker J.L. 2006. A hybrid learning system for recognizing user tasks from desk activities and email messages. *International Conference on Intelligent User Interfaces. Sydney, Australia, January 26-February 2.*

Shi & Malik, 2000	Shi J. and Malik J.2000. Normalized cuts and image segmentation. IEEE *Trans. Pattern Analysis and Machine Intelligence, 22(8):888-905, August.*
Siochi & Hix, 1991	Siochi A. C. and Hix D. 1991. A study computer supported user interface evaluation using maximal repeating pattern analysis. *CHI 91, New Orleans, Louisiana, United States, 301-305.*
Spiliopoulou & Faulstish, 1998	Spiliopoulou M. and Faulstich L.C. 1998 WUM: a Web Utilization Miner. *Workshop on the Web and Data Bases (WebDB98), p. 109-115. Valencia, Spain. March.*
Teo & John, 2006	Teo L. & John B.E. 2006. Comparisons of keystroke-level model predictions to observed data. *Conference on Human Factors in Computing Systems CHI '06 extended abstracts on Human factors in computing systems table of contents. SESSION: Work-in-progress table of contents. Montréal, Québec, Canada p 1421 – 1426.*
Vignaux, 2002	Vignaux G. 2002. Les relations sémantiques et cognitives appliquées aux notions dans un texte. *http://www.colisciences.net/pdf/Typologie des relations.pdf*
W3C, 2000	World Wide Web Consortium. Document Object Model (DOM) Level 2 Events Specification. *http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/.*
Wagner & Fisher, 1974	Wagner R.A. & Ficher M.J. 1974. The string to string correction problem. *Journal of the Association for Computing Machinery, Vol. 21, No. 1, January, p. 168-173.*
Ward, 1963	Ward J. H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association, 58:236-244.*
Welch, 1984	Welch T.A. 1984. A Technique for High-Performance Data Compression. *Computer. Vol. 17, pp. 8-19.*
Wiki, 2007	Wikipédia, l'encyclopédie libre. Dernière modification de l'article en 2007. *http://fr.wikipedia.org/wiki/Processus_cognitifs*
Yan & al., 2003	Yan X., Han J., Afshar R. 2003. CloSpan: Mining Closed Sequential Patterns in Large Databases. *SDM'03, San Fransisco, CA, May.*
Yu & Shi, 2003	Yu S.X. and Shi J. 2003. Multiclass spectral clustering. *In International Conference on Computer Vision.*
Zadeh, 1965	Zadeh L. A. 1965. Fuzzy sets. *In Information and Control, vol. 8, pp. 338-353*

Zaki, 2001 Zaki D., SPADE: An Efficient Algorithm for Mining Frequent Sequences. 2001. *Machine Learning, vol. 42, 2001, p. 31-60, Kluwer Academic Publishers.*

Annexes

Protocoles d'expérimentations

Expérimentation 1 : « Consultation dirigée d'un hypermédia clos »

But

Cette expérimentation fut la première entreprise pour récolter des traces d'interactions. Nous avons donc choisi un cas simple permettant la récolte de résultats facilement interprétables. Notre problématique d'étude principale étant l'apprentissage humain, nous avons choisi d'étudier le processus de recherche d'information, accompagné du processus de restitution d'information. En effet, ces deux processus sont couramment utilisés lors de l'apprentissage humain et en particulier lors de l'évaluation de cet apprentissage.

Nous avons choisi de mesurer les différences de comportements entre des personnes habituées aux interfaces hypermédia et des personnes novices, utilisant peu l'Internet par exemple. En particulier, nous avons cherché à déterminer ces différences de comportement avec des critères intelligibles qui permettent une interprétation directe facilitant sa description.

Méthodes & Outils

Nous avons conçu une interface simple contenant deux pages Internet dans lesquelles des informations sont présentées. La première ligne de texte de l'interface demande à l'utilisateur de bien vouloir prendre connaissance des deux pages web présentées. Des liens sont présents dans les deux pages, mais le travail est effectué hors ligne (une fenêtre de dialogue signalant que le lien n'est pas accessible apparaît). Il est précisé que les pages resteront apparentes tout le long de l'expérience. Quand il est prêt, l'utilisateur est prié de bien vouloir répondre aux questions qui lui sont posées en commençant par cliquer sur le bouton « Afficher la première question » (voir Figure 1)

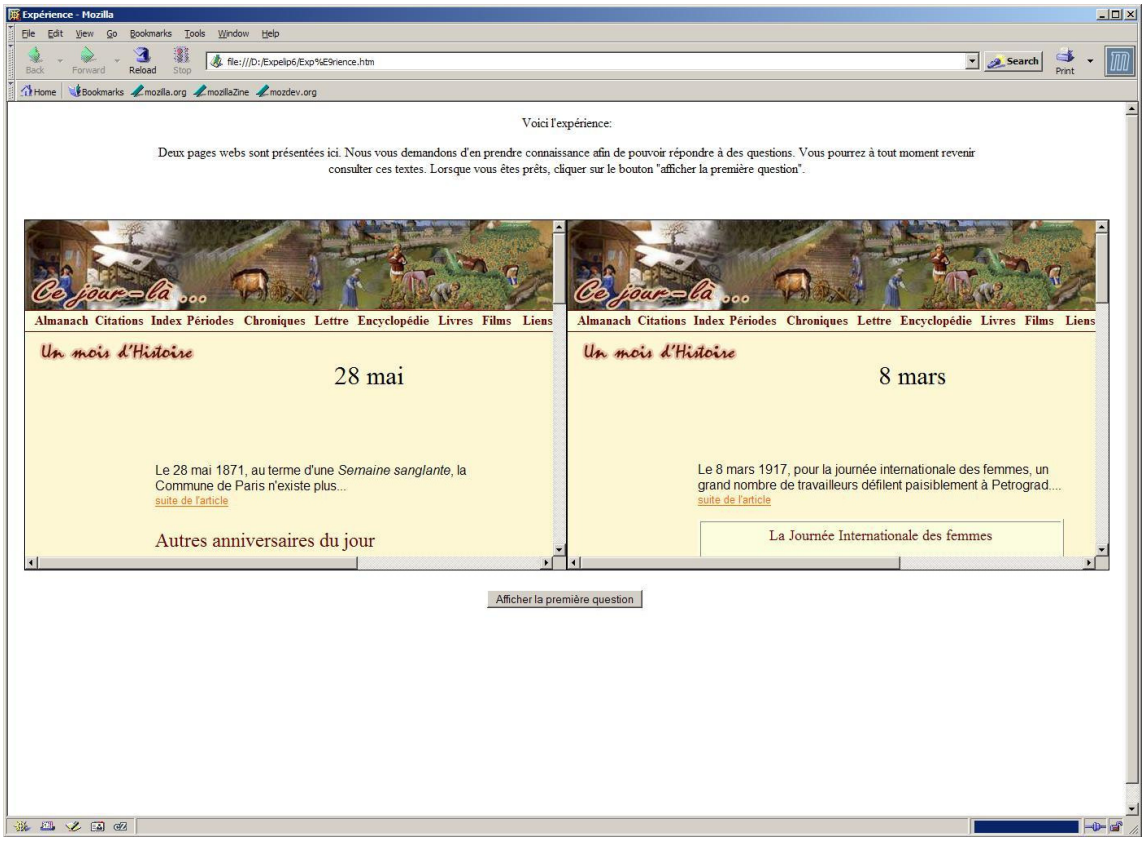

Figure 1. Interface de présentation de l'expérience

Une question, une zone de texte pour saisir la réponse et un bouton « Question suivante » apparaissent. Ainsi de suite, quatre questions sont posées. Les questions ont été étudiées pour faire appel à différents processus cognitifs. Certaines questions demandent des renseignements très précis sur le contenu d'une des deux pages, alors que d'autres demandent une réponse portant sur le contenu général des pages (voir Figure 2).

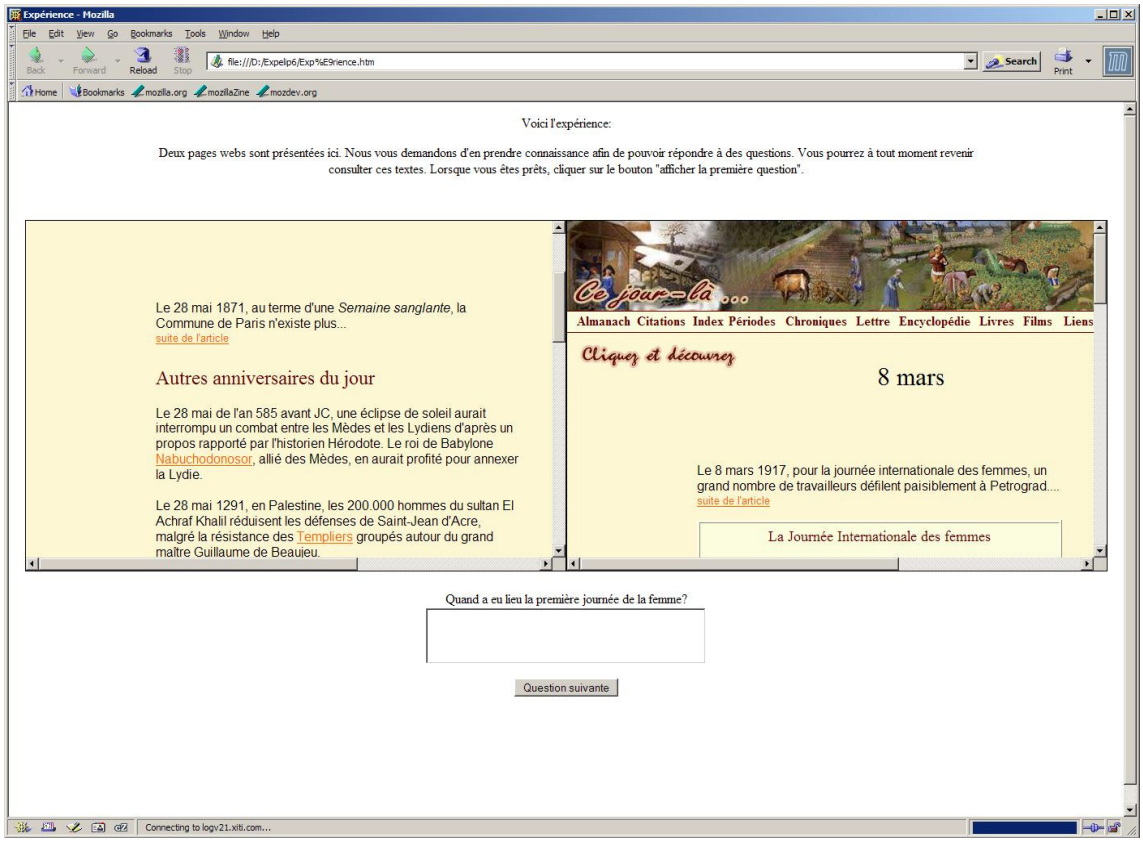

Figure 2. Interface de consultation lors de la réalisation de l'expérience

Quatre questions portant tantôt sur le fond, tantôt sur la forme des deux pages Internet sont ainsi posées. Nous avons qualifié cet environnement *hypermédia* de *clos* car il n'est pas possible d'utiliser d'autres ressources que celles proposées dans l'interface.

Le logiciel de traces MozSpy, décrit ci-dessous, implémentant les recommandations formulées dans le Chapitre 5 a été utilisé.

Pour cette expérience, nous avons utilisé les arbres de décisions (une méthode de classification supervisée) implémentés par Thanh ha Dang [Dang, 2007Dang, 2007Dang, 2007Dang, 2007Dang, 2007Dang, 2007Dang, 2007Dang, 2007Dang, 2007Dang, 2007]. Nous avons ainsi pu comparer l'efficacité de différents paramètres dans la construction de l'arbre de décision (voir paragraphe 1). Une validation croisée à 10 blocs a été réalisée pour estimer l'erreur de généralisation de classification de l'algorithme.

De plus, nous avons implémenté l'Algorithme 3 décrit dans le Chapitre 10 permettant l'extraction semi-automatique du parcours minimal à effectuer pendant l'exercice.

Nous avons sélectionné une trentaine de personnes pour effectuer ce test : 13 novices et 17 experts. Les novices sont des personnes utilisant l'Internet et/ou l'informatique en général une fois par semaine. Les experts au contraire, utilisent une interface informatique et/ou l'Internet quasi-quotidiennement.

Déroulement

Nous avons qualifié cette expérience de *consultation dirigée* car l'activité à réaliser sur l'interface est contrainte. L'utilisateur n'est pas amené à choisir un document parmi plusieurs, car seuls deux documents sont visibles tout au long de l'expérience. De plus cet exercice est de nature séquentielle, les utilisateurs sont priés de répondre à quatre questions posées dans certain ordre. D'autre part, la durée de l'expérience étant relativement courte les utilisateurs n'ont pas été distraits ni interrompus.

Résultats

Nombre de sessions : 30 (13 novices et 17 experts)

	Minimum	Maximum	Moyen	Ecart type
Durée	00 :02 :03	00 :08 :07	00 :04 :21	00 :01 :50
NB d'évènements différents	8	19	9,94	2,92
NB d'évènements total	390	2524	1134,17	666,98
NB de cibles différentes	7	27	12,28	5,5
NB de cibles total	16	65	29,33	14,06
NB de contextes différents	4	9	5,56	1,58
NB de contextes total	5	53	19,33	11,62

Remarques

Des analyses effectuées sur ces données sont présentées dans le Chapitre 11.1.

Cette expérience a été réalisée afin de valider le bon fonctionnement du logiciel de traces. Elle a été particulièrement utile pour concevoir le tableau de la Figure 17. En effet, les protocoles utilisés pour Internet offrent des possibilités de présentation de l'information très souples, et permettent la visualisation d'interface complexe comme les MDI (Multiple Document Interface).

Expérimentation 2 : « Etude de l'impact cognitif de l'utilisation de la *commande sémantique* dans la navigation dans un hypermédia »

But

Cette expérience a été effectuée afin d'observer l'impact cognitif de l'utilisation de la commande sémantique dans un hypermédia éducatif. La commande sémantique est un outil permettant la navigation dans un ensemble de documents liés par des relations ontologiques adaptées. Cette expérience a été effectué sur un manuel électronique de « Science et Vie de la Terre » de 4ème de collège élaboré par les éditions Bordas. Trois types de relations sémantiques bijectives ont été implémentés pour cette base de ressources pédagogiques : les relations temporelles (cause / conséquence), les relations catégorielles (est une sorte de / est composé de) et granulaires (parmi … il y a / compose). Ainsi, la navigation dans l'hypermédia représente à la fois des changements de documents et des « pas » conceptuels [Vignaux, 2002].

Méthodes & Outils

Les concepts de 2 chapitres de géologie du manuel de « Sciences de la Vie et de la Terre » de 4ème des éditions Bordas ont été organisés en fonction des critères définis précédemment. Cinq conditions (5 types d'interfaces) ont été testées et une classe d'une trentaine d'élèves de 5ème a participé aux tests (les élèves n'ayant ainsi aucune connaissance a priori du concept à apprendre). La première condition C1 etait la condition témoin (manuel avec navigation uniquement arborescente). Les documents pouvaient être des images (C2 et C4) ou des textes (C3 et C5) et les liens propres à notre navigation étaient représentés par les liens textuels (C2 et C3) ou fléchés (C4 et C5).

Pour cette expérience, il était demandé aux élèves d'expliquer la formation de l'argile. Ils disposaient pour cela de 20 minutes et d'une des 5 versions (C1-5) d'un manuel électronique. Quand ils avaient expliqué le phénomène « la formation de l'argile résulte de l'altération des feldspath et du mica », il leur était demandé d'expliquer l'apparition du feldspath et du mica. Ils disposaient pour cette recherche de 20 minutes supplémentaires.

Document / Type de lien	Image	Texte
Texte	C2	C3
Flèche	C4	C5

Figure 3. Récapitulatif des différentes conditions/expériences effectuées.

Le logiciel Cafit, décrit ci-dessous, a été utilisé pour effectuer les analyses des traces et les observations détaillées dans le Chapitre 7.1

Déroulement

De la même façon que pour l'expérimentation 1, l'activité à réaliser sur l'interface etait dirigée car il n'y avait qu'un seul exercice à réaliser. Cet exercice se révéla d'une durée relativement courte, et l'expérimentateur, suivant le parcours de l'utilisateur, a surveillé les éventuelles baisses d'attention ou de concentration trop importantes. Chaque passation s'est déroulée séparément.

Résultats

Nombre de sessions : 17

	Minimum	Maximum	Moyen	Ecart type
Durée	00 :05 :07	00 :47 :33	00 :18 :09	00 :10 :15
NB d'évènements différents	7	13	8,94	1,75
NB d'évènements total	61	814	316,47	216,26
NB de cibles différentes	14	59	29,76	10,91
NB de cibles total	22	130	73,18	30,84
NB de contextes différents	10	37	19,35	7,11
NB de contextes total	17	91	45,41	20,91

Remarques

Les analyses ont été effectuées avec l'outil de visualisation de traces. Les résultats sont présentés dans le Chapitre 7. Nous avons ainsi pu montrer l'intérêt de l'outil de visualisation en guidant la lecture du graphe ainsi créé.

Cette expérience nous a permis d'éprouver l'outil de visualisation. L'expérience ayant eu lieu sur plusieurs interfaces, les traces générées ont été particulièrement variées. Il nous a fallu concevoir des outils d'aide à lecture comme le zoom, l'ordonnancement et la sélection des objets de l'interface représentés.

Expérimentation 3 : « Consultation semi-ouverte d'un hypermédia clos »

But

Cette expérience a été effectuée afin de récolter des traces d'utilisations d'un hypermédia éducatif en conditions réelles d'utilisation. Aucune contrainte de manipulation n'a été demandée au professeur pour réaliser son cours. Nous souhaitions recueillir des traces dans des conditions réelles d'utilisation afin d'étudier les possibilités de personnalisation de l'interface ou de l'interaction du logiciel.

Méthodes & Outils

Le dispositif étudié est le manuel numérique de Science de la Vie et de la Terre de 3ème des éditions Bordas. Il se présente comme un logiciel de navigation internet, ou les boutons et les menus sont personnalisé au style de l'éditeur. Il a été conçu sur la plateforme de développement Mozilla, en langage de programmation XUL.

L'interface principale est composée de deux espaces qui présentent respectivement un arbre de navigation, appelé *treechildren*, représentant la structure hiérarchique du manuel papier classique, ainsi qu'un conteneur de documents, appelé *magicviewer*, pouvant représenter d'autres conteneurs de documents à la façon des MDI (Multiple Document Interface). Les conteneurs possèdent plusieurs propriétés de comportements particuliers. Par exemple, la disposition des documents dans un conteneur est automatiquement agencée pour obtenir la plus grande surface d'affichage possible par document. L'apparition des barres de défilement (scroll) sur les documents est également contrôlée et celles-ci ne sont affichées que lorsque qu'un seul document est représenté en pleine page. Plusieurs liens présents dans les pages forment des questions et renvoient directement au Classeur. L'interface du Classeur se présente comme un éditeur de texte dans lequel, le document contenant la question est également représenté. Ces deux interfaces (manuel et classeur) sont manipulables indépendamment l'une de l'autre, et les agencements à l'écran sont laissés libres à l'utilisateur.

Nous ne pouvons malheureusement pas présenter un aperçu des ces interfaces car les éditions Bordas ont décidé de cesser toutes activités relatives au fonctionnement de ces manuels.

Le logiciel MozSpy a été installé sur une quinzaine de postes informatiques d'un collège. Des manuels électroniques de Science et Vie de la Terre de 4ème ont été installés et des cours en conditions réelles ont pu être dispensés pendant plusieurs séances.

Déroulement

Au début du cours, nous avons rapidement interrogés les élèves pour connaître leur fréquence d'utilisation de dispositifs numériques tels que l'Internet ou des logiciels de traitement de texte. Nous avons également interrogé brièvement le professeur sur les facilités d'apprentissage des élèves de la classe. Il s'est avéré que les élèves ayant globalement les meilleures notes sont également ceux qui ont un ordinateur à la maison, et qui s'en servent régulièrement. Ces élèves se sont placés spontanément au premier rang de la classe.

Nous n'avons pas souhaité assister au cours, afin de procéder à une analyse des fichiers de traces de façon neutre, non dirigée par la volonté de prouver une propriété particulière dans le comportement des élèves ou la manipulation des manuels numériques.

En fin de séance, nous avons récoltés l'ensemble des fichiers de traces de façon à conserver dans les noms des fichiers, les rangs des élèves dans la classe.

Résultats

Nombre de sessions : 13.

	Minimum	Maximum	Moyen	Ecart type
Durée	01:22:24	01:49:05	01:42:54	00:08:29
NB d'évènements différents	11	16	13,84	1,56
NB d'évènements total	615	3010	1516,31	568,8
NB de cibles différentes	37	87	56,08	12,57
NB de cibles total	102	263	173,23	44,87
NB de contextes différents	7	13	9	2,12
NB de contextes total	33	98	63,8	17,6

Développements logiciels

MozSpy

Dans le cadre de cette thèse, nous avons développé ce logiciel afin de procéder à la récolte de traces suivant le modèle décrit dans le Chapitre 5. Le cadre industriel de nos recherches est une entreprise éditrice de logiciel pour contenu numérique pédagogique ayant choisi la plateforme de développement Mozilla pour la conception de leur ressources numériques. Ces outils sont des logiciels fonctionnant sur l'Internet et dont les fonctionnalités sont les suivantes :

- consultation de ressources numériques à l'aide d'une interface ergonomique,
- saisie de texte sur une interface permettant un formatage,
- calculatrice,
- logiciel de traitement d'image.

Nous avons développé un logiciel dédié à la plateforme Mozilla permettant d'enregistrer les actions des utilisateurs suivant le formalisme présenté dans le Chapitre 5. Afin de suivre d'intéressantes propriétés permettant à Mozilla d'être indépendant du système informatique opérant sur la machine client, nous avons développé notre logiciel sous un format Javascript-XPCOM. Ce logiciel, appelé MozSpy, traces toutes les actions portées sur le navigateur Internet et les enregistre sur le disque dur de l'ordinateur.

Cafit

Afin de procéder à des analyses des traces récoltées lors des différentes expérimentations présentées dans les protocoles d'expérimentation, nous avons développé le logiciel *Cafit* (Cognitive Analysis From Interaction Traces)

Nous avons développé ce logiciel en Java. Nous avons choisi ce langage pour sa simplicité, sa puissance, ainsi que la communauté libre développant beaucoup d'outils autour de ce langage. Ce langage possède également l'avantage d'être indépendant du système informatique sur lequel il est développé et peut être utilisé sur n'importe quel système d'exploitation.

D'autre part, plusieurs technologies logicielles existantes dans le domaine du logiciel libre ont été utilisées pour la conception de notre propre logiciel :

- *antlr* : *ANother Tool For Language Recognition*. Cette librairie java est utilisée pour la reconnaissance, l'interprétation, la compilation et la traduction de langages informatique. Ainsi, un fichier en langage structuré XML peut être facilement interprété, ce qui constitue une facilité d'utilisation pour construire un arbre, parcourir un arbre, traduire (ou transformer la structure arborescente), détecter des éventuelles erreurs et en faire le rapport.
- *commons-logging* est une librairie utilisée pour faciliter la diffusion des messages, et notamment les messages d'erreur, dans les applications complexes.
- *concurrent* est une librairie utilisée pour faciliter la synchronisation de l'accès aux données. C'est une surcouche optimisée de l'application native java.
- *derby* est un système de base de données qui peut fonctionner à la fois en local et sur un serveur réseau. Il est entièrement codé en java et s'interface très simplement avec celui-ci
- *icu4j* : *Internationnal components for unicode* est une librairie très utilisé pour supporter le format Unicode, l'internationalisation des logiciels. Il permet la gestion du texte en Unicode et fournit les jeux de conversions de caractères, le parsing et formatage des dates, heures, nombres, messages, … basés sur des règles
- *jdom* est une librairie java utilisée pour manipuler les fichiers XML simplement. Il fournit des fonctions pour accéder, manipuler et construire des fichiers au format XML
- *jena* est une plateforme applicative pour les applications du web sémantique. Elle fournit notamment des fonctionnalités pour manipuler facilement les graphes modélisés sous format RDF.
- *Jmat* est une librairie fournissant les outils mathématiques pour les manipulations matricielles. Elle possède également des outils graphiques permettant de tracer des courbes et des points.
- *Junit* est une bibliothèque pour les tests unitaires dans le langage de programmation java.

- *log4j* est une bibliothèque conçue pour faciliter les diffusions de messages dans les programmes java.
- *xerces*. est une librairie utilisée pour parcourir efficacement les fichiers XML, éventuellement en appliquant les règles de transformation, comptages et statistiques.
- *Weka* est une plateforme java fournissant de nombreux outils d'apprentissage artificiel aussi bien pour des tâches supervisées que non supervisées.
- *Colt* est une librairie optimisé pour le calcul scientifique, et notamment le calcul matriciel pour grandes quantités de données.
- *Jgraph* est une librairie suivant le modèle de conception MVC (Model View Controler) qui permet de tracer des graphes.

Figures

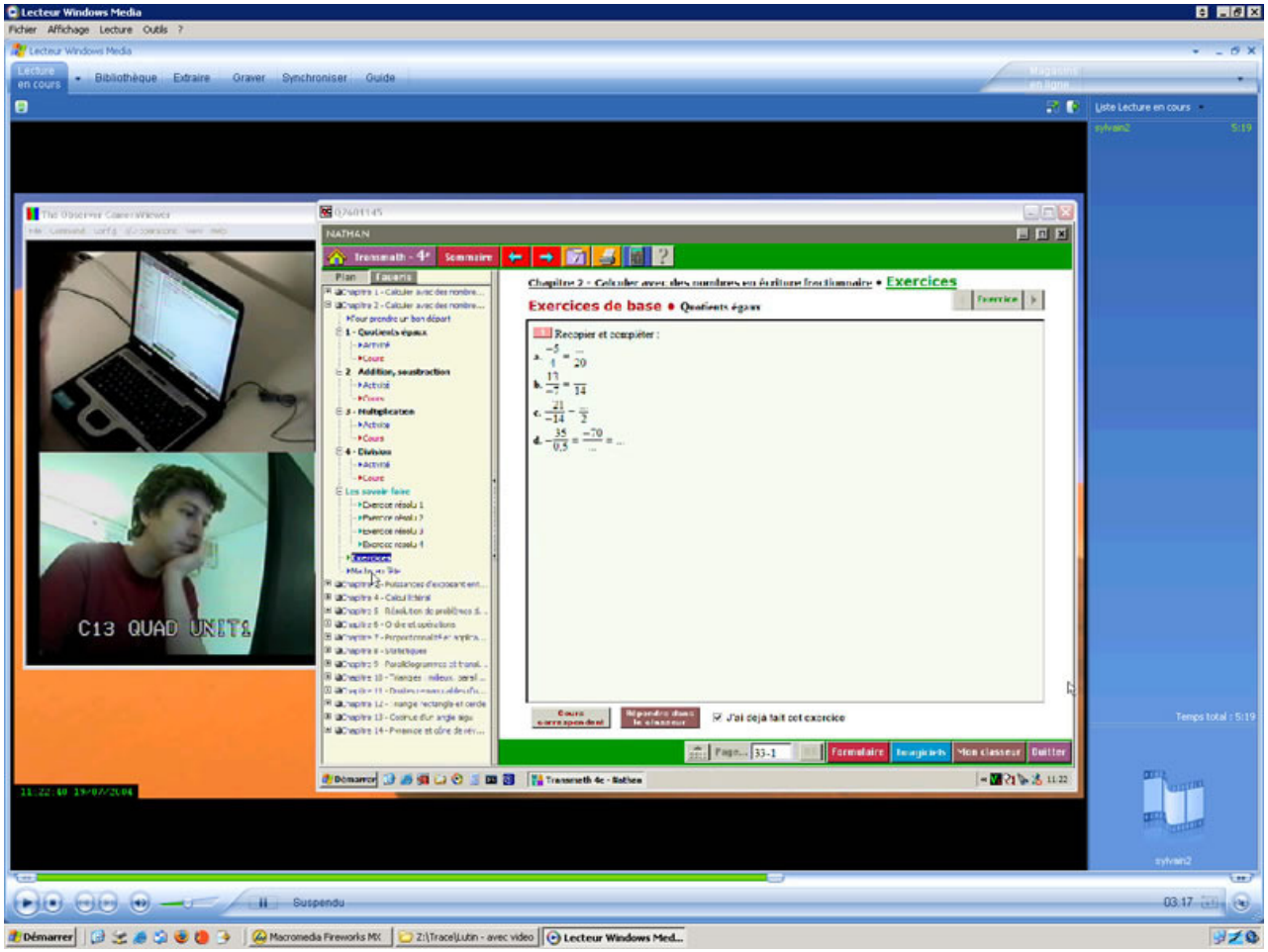

Figure 1. *Extrait d'une vidéo lors d'une manipulation d'un manuel scolaire électronique.*

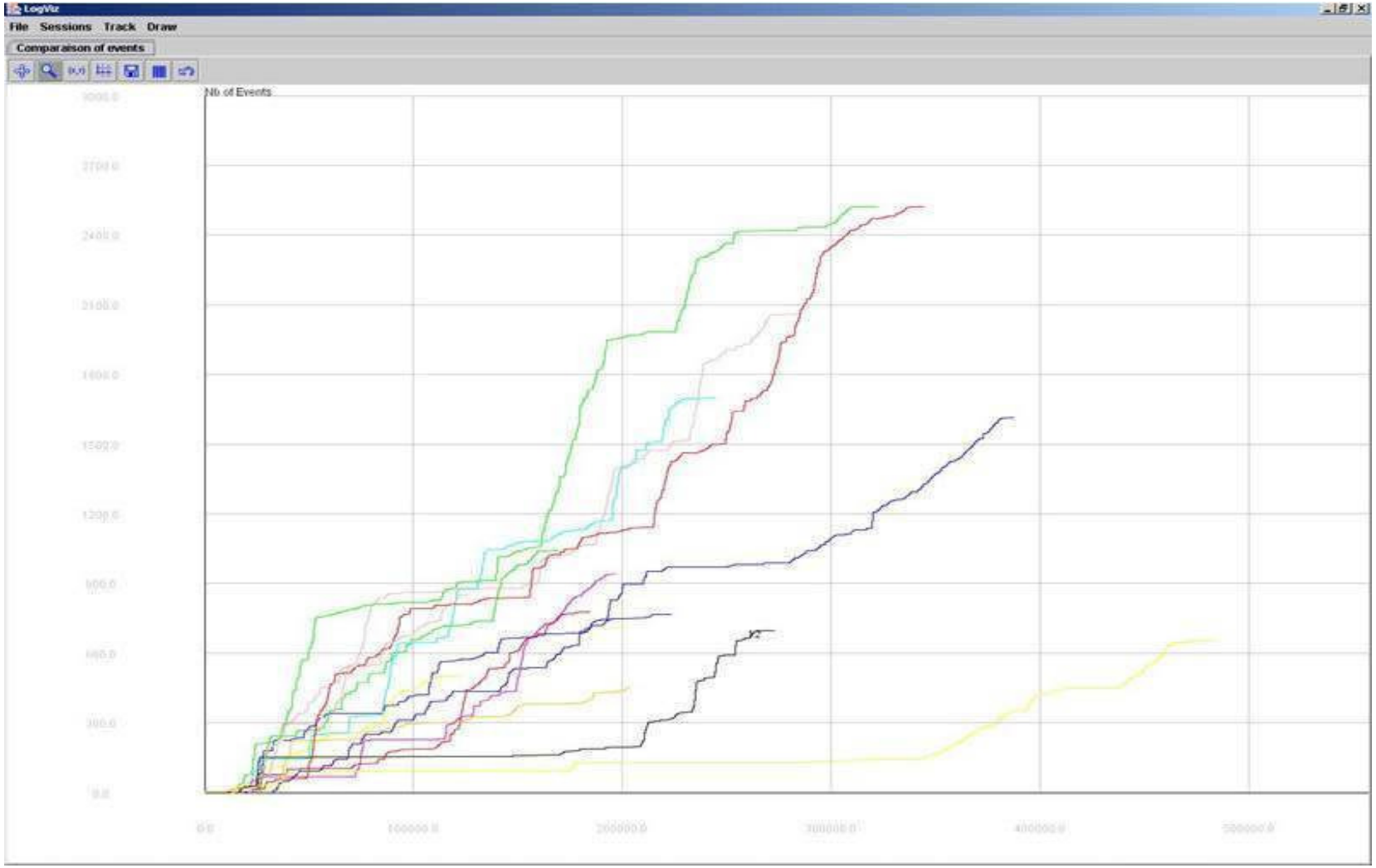

Figure 2. Graphe représentant le nombre d'évènements générés (ordonnée) en fonction du temps (abscisse) de différents utilisateurs (un par couleur)

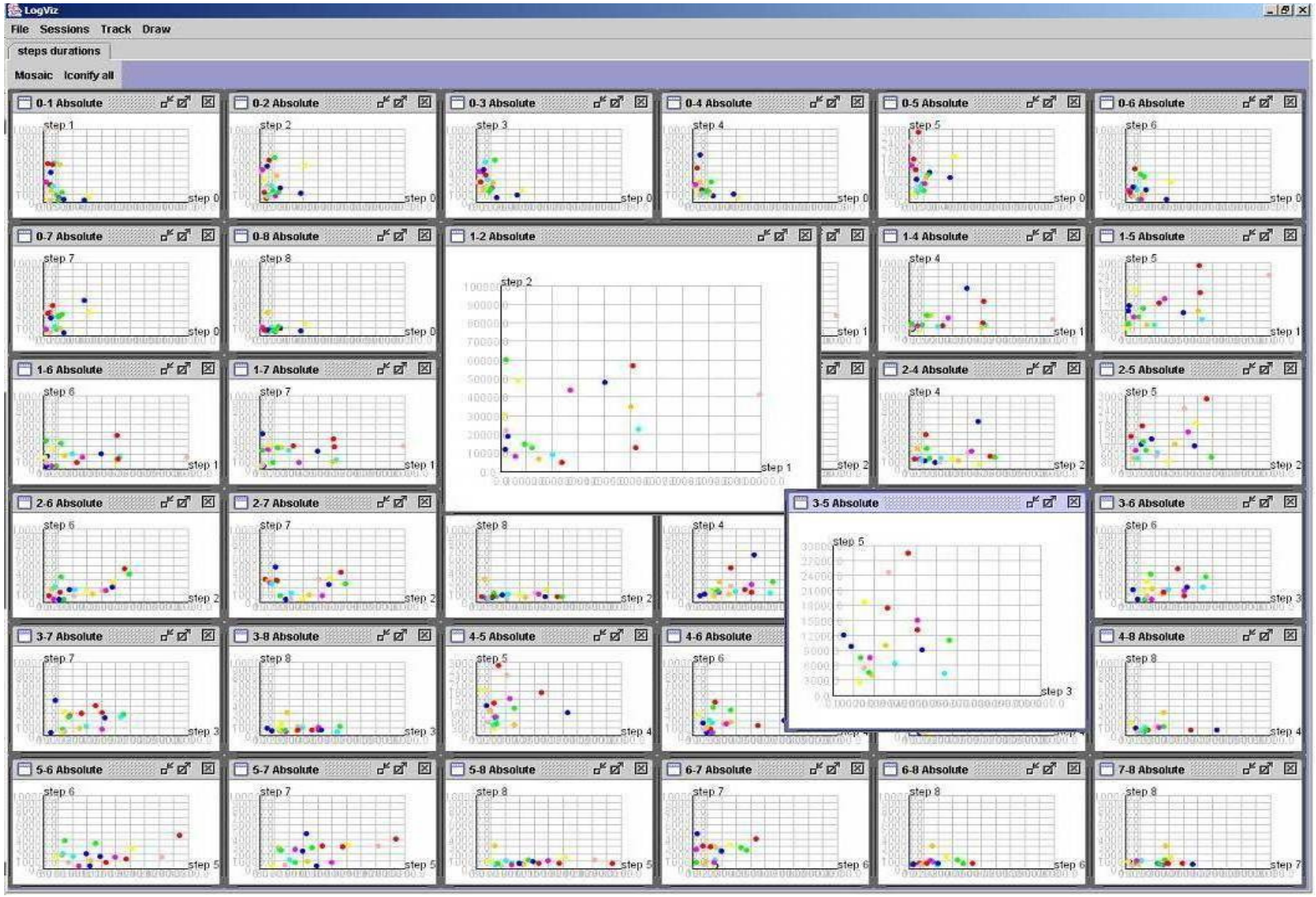

Figure 3. Comparatif des temps d'accomplissement de sous buts par différents utilisateurs