Tables de matières:

Introduction générale

Contexte du travail

Aujourd'hui les utilisateurs d'internet ont du mal à trouver les données et les informations qu'ils cherchent à cause de la surcharge d'information du web, pour résoudre ce problème les systèmes de recommandations sont apparus. Ils ont comme buts : minimiser le temps de l'utilisateur passé à la recherche et suggérer des ressources pertinentes qu'ils n'auraient pas consultées.

Pour recommander des ressources plusieurs approches sont possibles :

L'approche collaboratif qui se base sur l'opinion des utilisateurs ainsi la similarité entre utilisateurs, l'approche par contenu qui effectue des recommandations en comparant le contenu des ressources avec le gout des utilisateurs, l'approche hybride qui regroupe les deux approche précédentes et l'approche contextuel qui se base sur le contexte d'un utilisateurs(son environnement).

Dans ce mémoire nous nous intéressons à l'approche hybride : par contenu en comparant les informations des patients avec les documents et collaboratif en recommandons le document qui a un taux de votes supérieur et à l'approche contextuel après avoir extraire le contexte d'utilisation d'une application de gestion des enregistrements électronique des patients: sa maladie, son âge, son sexe…

Problématique

Le problème scientifique que nous traitons liée à la recommandation des documents aux médecins exploitant le filtrage hybride en se basant sur les documents votés par les autres médecins et la comparaison entre contexte du patient et le contenu des documents.

Objectifs

L'objectif principal de notre travail consiste à réaliser un système de recommandation contextuel des documents en se basant sur les informations des patients et recommandation hybride en se basant sur l'indexation pour trouver les mots les plus importants dans un document en se basant sur la tfidf

Et calculer la mesure de similarité entre document et contexte du patients via la mesure cosinus et en se basant aussi sur le taux de votes des documents.

Organisation

Ce mémoire contient trois chapitres qui se répartissent comme suit :

Dans le premier chapitre, nous définissons le filtrage d'information et ces techniques en précisant les avantages et les inconvénients des deux approches par contenu et collaboratif

Dans le deuxième chapitre nous présentons les différentes définitions du contexte ainsi que les domaines d'application.

Dans le dernier chapitre, nous commençons par faire la conception de notre système, Ensuite, nous détaillons comment faire la recommandation par contenu en se basant sur l'indexation et l'appariement contexte-documents et collaborative en se basant sur les votes et nous terminons par une description sur la réalisation de l'application.

Chapitre 1 : Les systèmes de recommandation

1 Les systèmes de recommandation

1.1 Introduction

Vu l'accroissement de la quantité d'information et le nombre d'utilisateurs sur internet il est devenu difficile de trouver des données même les outils de recherche d'information classiques ne fournissent pas toujours des résultats pertinents.

Néanmoins, pendant les dix dernières années, les systèmes de recommandation se sont imposés comme moyen efficace pour réduire la complexité dans la recherche d'informations, c'est ce qu'on va voir dans ce chapitre

1.2 Définition des systèmes de recommandation

Les systèmes de recommandation peuvent être définis de plusieurs façons, qui peuvent se rapporter à différents types de données ou approches spécifiques. La définition que nous utiliserons est une définition générale de Robin Burke (Burke, 2002),

Système de recommandation : Système capable de fournir des recommandations personnalisées ou permettant de guider l'utilisateur vers des ressources intéressantes ou utiles au sein d'un espace de données important.

En pratique, la plupart des systèmes de recommandation consistent en des applications Web qui proposent des listes de ressources à des utilisateurs. De telles ressources peuvent correspondre à différents types de données tels que des films (Miller et al., 2003), de la musique (Su et al., 2010), des livres (Mooney et Roy, 2000), des restaurants (Burke, 2007), des news (Das et al., 2007), des blagues, (Miyahara et Pazzani, 2000), des articles scientifiques (Pavlov et al., 2004), des pages Web (Pitkow et Pirolli, 1999), etc.

Les données d'entrée pour un système de recommandation dépendent du type de l'algorithme de filtrage employé. Généralement, elles appartiennent à l'une des catégories suivantes

- **Les *estimations*** :(également appelées les *votes*), expriment l'opinion des utilisateurs sur les articles (exemple : 1 mauvais à 5 excellent).
- **Les *données démographiques*** : se réfèrent à des informations telles que l'âge, le sexe, le pays et l'éducation des utilisateurs. Ce type de données est généralement difficile à obtenir et est normalement collecté explicitement;

- ***Les données de contenu***, qui sont fondées sur une analyse textuelle des documents liés aux éléments évalués par l'utilisateur. Les caractéristiques extraites de cette analyse sont utilisées comme entrées dans l'algorithme de filtrage afin d'en déduire un profil d'utilisateur. [1]

1.3 Objectif des systèmes de recommandation

L'objectif est à la fois de minimiser le temps des utilisateurs passé à la recherche, mais aussi de les suggérer des ressources pertinentes qu'il n'aurait pas spontanément consultées et ainsi accroître sa satisfaction globale.

1.4 Classification des systèmes de recommandation

Il est possible de classer les systèmes de recommandation de différentes manières. La classification la plus fréquente est une classification selon trois approches :

1) les recommandations basées sur le contenu.

2) filtrage collaboratif (Shahabi et al. 2001 ; Adomavicius et Tuzhilin, 2005 ; Huang et Huang, 2009 ; Lousame et Sánchez, 2009).

3) le filtrage hybride qui combine les deux approches précédentes. En plus de ces trois approches, (Burke, 2007) propose de considérer trois autres approches : la recommandation basée sur les données démographiques, la recommandation basée sur l'utilité et la recommandation basée sur la connaissance.

1.5 Historique

Les interfaces de recommandation facilitent l'interaction humain-machine et l'accès à l'information pertinente aux besoins de l'utilisateur. On retrace les premiers travaux portant sur le filtrage d'information à un article de Luhn [Luhn, 1958] sous le nom de « diffusion sélective d'une nouvelle information ». Le terme « filtrage d'information » a été proposé par Denning [Denning, 1982], qui s'est concentré sur le filtrage des courriers électroniques.

En 1987, deux catégories du filtrage d'information étaient proposées par Malone [Malone et coll., 1987]. La première catégorie est le filtrage cognitif nommé actuellement filtrage basé sur le contenu. La deuxième catégorie est le filtrage social qui correspond au filtrage collaboratif.

Cet axe de recherche été très actif dans la dernière décennie et même plus tôt, et plusieurs laboratoires s'y sont consacré. Leurs travaux vont de la catégorisation du

filtrage d'information au développement des techniques d'extraction des informations pour la sélection des messages. Par exemple, la conférence MUC (Message Understanding Conference) [Hirschman, 1991] rapporte une série d'études qui ont eu un impact non négligeable. Cette conférence était parrainée par l'agence DARPA (United States Advanced Research Projects Agency des États-Unis) en 1989.

Après une année, DARPA a continué les travaux sur les techniques de filtrage d'information par le lancement d'un nouveau projet nommé Tapestry qui avait pour objectif la réunion des efforts de recherche des différents participants à MUC [Harman, 1992a].

Plusieurs systèmes de recherche et de filtrage d'information ont été développés. Donc il fallait trouver un moyen pour les évaluer. DARPA collabore avec NIST (National Institut of Standards and Technology) et parrainent un projet international TREC (Text REtrieval Conference) qui concerne l'évaluation de ces systèmes [Harman, 1992b. Suite à une conférence sur l'incertitude en intelligence artificielle, une étude séminale qui compare des algorithmes prédictifs pour le filtrage collaboratif était publiée par Breese, Heckerman, et Kadie dans [Breese, Heckerman, et Kadie, 1998].

Une interface ou un système de recommandation constitue un type de filtrage d'information qui a pour objectif de prévoir et de présenter les éléments d'informations tels que les cours, la musique, les films… qui correspondront aux intérêts et besoins de l'utilisateur. Ceci peut se faire par la comparaison entre les objets eux-mêmes (approche Item-Item) ou par la comparaison entre les choix des utilisateurs (approche Utilisateur-Utilisateur). Une interface ou un système de recommandation nécessite une recherche et un filtrage d'information. [2]

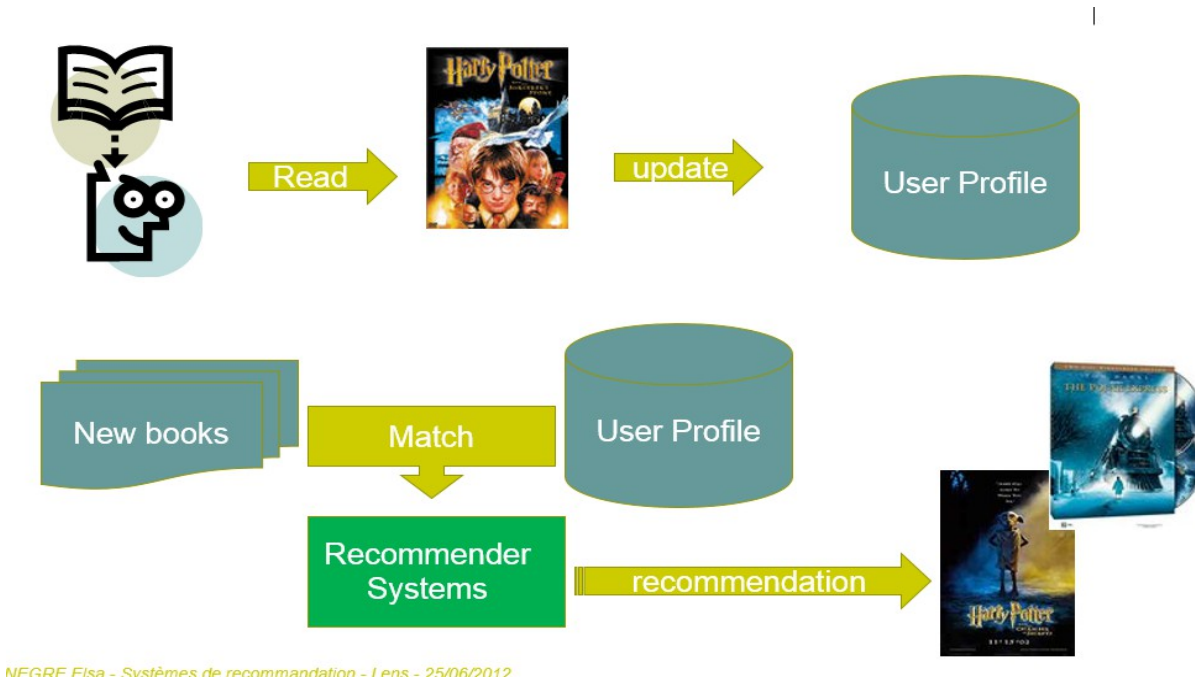
1.6 Recommandation basée sur le contenu

La recommandation basée sur le contenu consiste à analyser le contenu des ressources ou des descriptions de ces ressources afin de déterminer quelles ressources sont susceptibles d'être utiles ou intéressantes pour un utilisateur donné. Ce sous-domaine est fortement similaire au domaine de la recherche d'information. En effet, les mêmes techniques sont utilisées, la différence se trouvant essentiellement dans l'absence de requêtes explicites formulées par l'utilisateur.

Par conséquent, beaucoup de concepts généraux de la recommandation basée sur le contenu proviennent de la recherche d'information.

La plupart des systèmes de recommandation basée sur le contenu identifient les ressources
similaires aux ressources qu'un utilisateur donné a apprécié (figure I.1) (Balabanovi´c et Shoham, 1997 ; Zhang et al., 2002 ; Adomavicius et Tuzhilin, 2005 ; Pazzani et Billsus, 2007). Quand de nouvelles ressources sont introduites dans le système, elles peuvent être recommandées directement sans que cela ne nécessite un temps d'intégration comme cela est le cas dans le cadre des systèmes de recommandation basée sur les usages (recommandation collaborative).
Pour ce faire, le filtrage basé sur le contenu doit remplir deux fonctions fondamentales:

- L'identification des documents pertinents pour l'utilisateur en fonction de son profil.
- La mise à jour du profil utilisateur en fonction des évaluations, implicites ou explicites, de l'utilisateur des documents reçus.



(Figure I.1) recommandation basée sur le contenu

1.6.1 Approche générale

Pour recommander des ressources en se basant sur le contenu, deux éléments doivent être constitués : les profils de ressource et les profils d'utilisateur. La notion de contenu ne se rapporte donc pas uniquement au contenu des ressources, mais également aux attributs descriptifs des utilisateurs.

1.6.2 Profils de ressource

Les profils de ressource consistent en un ensemble d'attributs décrivant les ressources, de façon analogue à l'index utilisé dans le domaine de la recherche d'information. Comme dans le domaine de la recherche d'information, la précision de cette approche est donc hautement dépendante de la nature des ressources : elle est beaucoup plus élevée pour des ressources textuelles que pour des ressources telles que les images, les vidéos ou les ressources audio, dont il est difficile d'extraire des attributs. En général, quand cette approche est employée pour des ressources non textuelles, des méta-données sont utilisées Par conséquent, la plupart des recherches sur la recommandation basée sur le contenu porte sur des données textuelles (Adomavicius et Tuzhilin, 2005 ; Pazzani et Billsus, 2007).

Une étape importante de cette approche est la transformation des données textuelles sans restriction, c'est-à-dire écrites en langage naturel, en une représentation structurée. Une des techniques les plus répandues pour répondre à cette problématique est le stemming (Porter 1997). Le stemming consiste à effectuer une transformation systématique des mots relatifs à un même concept en un même terme qui les représente tous. Ensuite un poids est attribué à chacun de ces termes en fonction de leur importance dans la ressource textuelle. Une façon classique de calculer ce poids est l'utilisation de la formule term-frequency times inverse document-frequency ou tf·idf (Salton et Buckley, 1987). Une limitation de cette technique est qu'elle ne prend pas en compte le contexte des termes. Ainsi, l'application de cette technique à des textes contenant par exemple des tournures négatives ou ironiques peut aboutir à de mauvaises représentations.

C'est pourquoi d'autres méthodes ont été proposées : description de longueur minimale (Lang, 1995), utilisation de stems de plusieurs mots (Jacquemin et al., 1997), etc.

Une fois cette étape finalisée, le système possède soit les listes des mots les plus importants ou les plus informatifs de chaque ressource, soit un ensemble de vecteurs de termes, c'est à-dire un ensemble de poids associé à chaque terme de chaque ressource.

1.6.3 Profils d'utilisateur

Le profil d'un utilisateur définit ses centres d'intérêt. De tels profils consistent en un ensemble d'informations qui peuvent être entrées manuellement par l'utilisateur, ou extraites automatiquement à partir du contenu des ressources qu'il a consultées.

La première possibilité est donc de demander à l'utilisateur de fournir directement ses centres d'intérêts, à l'aide de formulaires, en lui demandant d'entrer une liste de termes, etc. Si un nombre restreint d'informations est demandé, cette approche rendra le système opérationnel rapidement, mais ne pourra pas fournir de recommandations précises. À l'inverse, en demandant un grand nombre d'informations, les recommandations seront plus précises mais le système sera trop contraignant pour l'utilisateur. De plus, les centre 'intérêts des utilisateurs peuvent évoluer au cours du temps, et une telle approche impose une actualisation manuelle régulière, ce qui est également contraignant. Un dernier inconvénient enfin, est que l'utilisateur peut ne pas remplir le formulaire honnêtement, auquel cas, les recommandations qui lui seront fournies ne pourront pas être pertinentes.

La seconde possibilité, l'extraction automatique à partir du contenu des ressources consultées par l'utilisateur, est donc souvent préférable. Une des méthodes les plus simples est de représenter les centres d'intérêt des utilisateurs par des vecteurs de termes à partir des vecteurs de termes représentant les ressources que l'utilisateur a appréciées. Les appréciations peuvent être obtenues de façon explicite en demandant directement aux utilisateurs de les fournir, ou implicitement en utilisant des algorithmes basés sur les usages

Pour calculer les recommandations, il suffit alors de calculer la similarité entre les profils de ressource et les profils d'utilisateur. Cela peut être effectué selon diverses méthodes, comme la mesure de similarité cosinus. C'est dans ce cadre que cette approche est la plus similaire aux approches de la recherche d'information.

Beaucoup d'autres méthodes d'extraction automatique de profils qui se démarquent davantage de la recherche d'information ont été proposées. Dans ce cadre, les recommandations sont calculées selon la probabilité qu'un utilisateur donné appréciera

une ressource. Cela peut être considéré comme un problème de classification où chaque classe représente un niveau d'appréciation (« aime » et « n'aime pas »). Trois des algorithmes de classification célèbres souvent utilisés dans ce contexte sont présentés dans cette section : les arbres de décision, le classificateur naïf de Bayes et les réseaux de neurones.

a) Arbres de décision

Un arbre de décision est obtenu en séparant de façon récursive les ressources en sous-groupes homogènes relativement à des variables déterminées au préalable. Dans le cas de la recommandation de ressources textuelles, ces variables sont en principe des variables booléennes sur la présence ou l'absence de termes. Ensuite, pour chaque sous-groupe, la probabilité que l'utilisateur appréciera une ressource de ce sous-groupe est conservée.

Le problème principal de l'application de cette approche à la recommandation basée sur le contenu est que la précision obtenue est dépendante du nombre de variables manipulées.

Cette approche est simple et performante dans le cadre de recommandations portant sur des ressources ayant un nombre d'attributs limité, mais n'est pas appropriée dès que ces attributs sont en nombre élevé, ce qui est le cas des ressources textuelles sans restriction.

b) Classificateur naïf de Bayes :

Le principe du classificateur naïf de Bayes est de déterminer la classe C pour laquelle la probabilité P(C|q1, ..., qk) qu'une ressource r ayant pour attributs (q1, ..., qk) appartienne à cette classe C soit maximale. Les attributs sont supposés indépendants, et maximiser P(C|q1, ..., qk) revient à maximiser la formule suivante :

$$P(C)\prod_{i=1}^{k} P(q_i|C)$$

formule(I.1)

Les valeurs de P(C) et de P(qi |C) sont estimées à partir d'un corpus d'apprentissage. Pour chaque ressource r, chaque valeur de la formule (1.1) est estimée pour chaque classe (ici chaque niveau d'appréciation). r est alors placée dans la classe pour laquelle cette valeur est la plus élevée.

En dépit du fait que les attributs des ressources sont en réalité inter-dépendants, le Classificateur naïf de Bayes s'avère fournir une grande précision et représente un algorithme simple et ayant un temps de calcul réduit. De plus, contrairement aux arbres de décision, il est applicable aussi bien sur des données ayant un nombre d'attributs limité que sur des données sans restriction.

c) Réseaux de neurones

Dans un réseau de neurones, un neurone est simplement une fonction non linéaire, de variables réelles et bornée. Cette fonction est généralement définie comme suit :

$$f(x_1, ..., x_k; w_1, ..., w_k) = \left\lfloor \sum_{i=1}^{k} w_i x_i \right\rfloor$$

où les variables w1, ...,wk correspondent à des poids à associer aux variables x1, ..., xk, qui sont déterminés à partir d'un corpus d'apprentissage. La fonction tangente hyperbolique est une fonction sigmoïde qui a certaines propriétés particulièrement appropriées pour l'apprentissage de réseaux de neurones (Kalman et Kwasny, 1992). De tels neurones sont associés en réseau selon deux types d'architecture : les réseaux bouclés qui correspondent à des graphes orientés avec circuit et les réseaux non-bouclés qui correspondent à des graphes orientés sans circuit.

Dans le cadre de la recommandation basée sur le contenu, les variables x1, ..., xk correspondent à la fréquence des termes utilisés pour caractériser les ressources (qui peut être normalisée par rapport à la longueur du texte). L'architecture la plus fréquemment adoptée est l'architecture en réseaux non bouclés avec une structure de perceptron multicouche (Hornik, 1993). Plus précisément, cette structure consiste en général en k entrées (les k attributs d'une ressource), une couche d'un certain nombre de neurones cachés, et un certain nombre de neurones de sortie. Chaque neurone de sortie indique un score permettant de déterminer si une ressource appartient à la classe du niveau d'appréciation à laquelle il est associé. Un algorithme répandu pour effectuer l'apprentissage des poids est l'algorithme PLA (Perceptron Learning Algorithm) (Rosenblatt, 1958). Il consiste à initialiser les variables de façon aléatoire et à les ajuster itérativement de façon à minimiser le nombre de ressources disposées dans de

mauvaises classes. En plus de permettre un apprentissage rapide, l'utilisation de réseaux de neurones a l'avantage de permettre un ajustement particulièrement fin grâce à l'utilisation de la fonction sigmoïde. Selon le domaine d'application il peut s'avérer plus ou moins efficace que ses alternatives (Pazzani et Billsus, 1997).

1.6.4 Limitations de la recommandation basée sur le contenu

La principale limitation de la recommandation basée sur le contenu est qu'elle nécessite l'acquisition d'un nombre suffisant d'attributs décrivant les ressources. C'est pourquoi elle est appropriée dans le cadre de ressources textuelles ou quand des descriptions textuelles des ressources ont été entrées manuellement. Dans le cadre de ressources textuelles, une des limitations provient des méthodes de classification de texte utilisées : en effet, deux ressources peuvent être similaires du point de vue de leurs attributs, mais avoir une qualité ou une pertinence incomparable.

Une autre limitation est que ces modèles ne peuvent recommander que des ressources similaires à celles qu'un utilisateur donné a appréciées, ce qui empêche de recommander d'autres ressources que ce même utilisateur pourrait également apprécier. Pour amoindrir ce problème, il est possible de fournir des recommandations aléatoires parmi les recommandations.

Enfin, une dernière limitation est qu'un nouvel utilisateur d'un tel système doit avoir consulté ou fourni des appréciations pour un certain nombre de ressources avant que le système ne puisse lui fournir des recommandations pertinentes. Ce problème est connu sous le nom de démarrage à froid. Une façon de réduire ce problème est de demander un certain nombre d'informations à l'utilisateur au moment de son arrivée (en nombre limité pour ne pas rendre le système trop contraignant) et d'utiliser un profil type correspondant aux informations qu'il a fournies.

1.7 Filtrage collaboratif

L'hypothèse principale sur laquelle repose le filtrage collaboratif est : « de bouche à oreille ». Plus précisément, les gens à la recherche d'information devraient pouvoir bénéficier de ce que d'autres utilisateurs ont déjà trouvé et évalué. Par exemple, les personnes, qui veulent regarder un film ou lire un livre, demandent à leurs amis leurs opinions. Donc, dans le filtrage collaboratif, la sélection des documents à proposer à un utilisateur ne dépend plus des termes constituant le document (filtrage basé sur le

contenu), mais des évaluations faites par les membres de son voisinage. Ainsi, si deux utilisateurs Alice et Bob ont évalué un certain nombre de documents de façon similaire, il y a de fortes chances qu'Alice aime ce que Bob aime, et inversement. Donc les documents que Alice a aimés peuvent être recommandés à Bob et inversement. De la sorte, cette approche résout une difficulté importante rencontrée par l'approche de filtrage basé sur le contenu, à savoir le traitement de documents multimédias (Adomavicius et Tuzhilin, 2005 ; Sarwar *et al.,* 2000a).

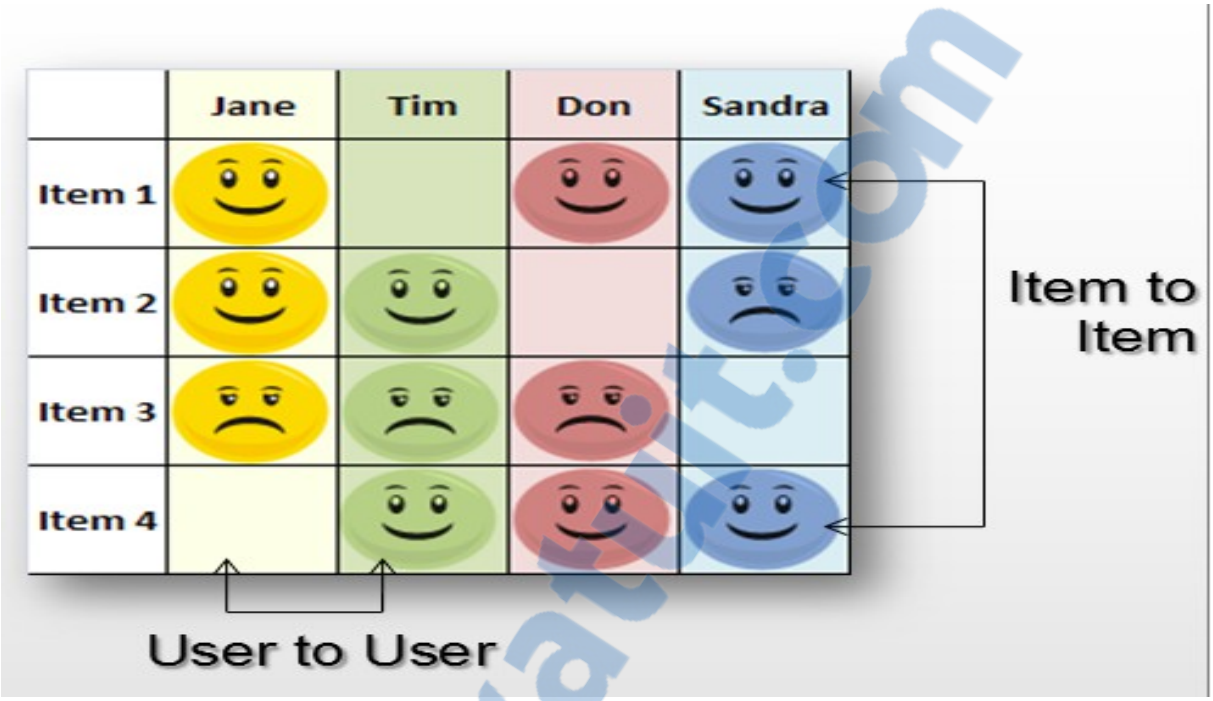
L'algorithme général d'un système de filtrage collaboratif. Typiquement, les étapes de cet algorithme sont les suivantes (Adomavicius et Tuzhilin, 2005 ; Berrut et Denos, 2003):

../ collecter les appréciations de l'utilisateur sur les documents qu'i1consulte;

../ intégrer ces informations dans le profil de l'utilisateur;

../ utiliser ce profi1 pour aider l'utilisateur dans ces prochaines recherches d'information.

Donc, la première étape à prendre en compte est celle de la collecte du taux de satisfaction des utilisateurs pour les documents qu'ils consultent. Ces évaluations sont recueillies en utilisant une des méthodes présentées. Les notes ainsi obtenues sont représentées sous forme d'une matrice. À chaque ligne est associé un document, et à chaque colonne, un utilisateur (Adomavicius et Tuzhilin, 2005)



Figure(I.2) matrice document-utilisateur

A partir de cette matrice on peut recommander des items à des utilisateurs selon deux méthodes :

User to user: Les recommandations sont faites en trouvant des utilisateurs ayant des avis similaires. Jane et Tim aiment tous les deux l'Item 2 et détestent l'Item 3 ; il semble qu'ils ont des avis similaires, ce qui suggère qu'en général Jane et Tim sont du même avis. Donc, l'Item 1 est une bonne recommandation pour Tim.

Item to item: Les recommandations sont faites en trouvant les items qui ont le même intérêt pour plusieurs utilisateurs. Tom et Sandra aiment l'Item 1 et l'Item 4.

Cela suggère que, en général, les utilisateurs qui aiment l'Item 4 aimeront aussi l'Item 1, donc l'Item 1 pourra être recommandé à Tim. [3]

1.7.1 Les limites du filtrage collaboratif

Le filtrage collaboratif présente un certain nombre de limites. En premier lieu, nous pouvons évoquer le problème lié à la taille de la population d'utilisateurs et de l'ensemble de documents à évaluer (le problème de la masse critique). En effet, il faut un certain nombre d'évaluations avant que le système puisse donner des résultats. De plus, cette approche souffre aussi du problème de démarrage à froid. En effet, les

nouveaux utilisateurs commencent par un profil vide. Ainsi, une période d'apprentissage est nécessaire avant que le profil ne reflète concrètement les préférences de l'utilisateur. Pendant cette période, le système ne peut pas filtrer efficacement. Enfin, comme pour le filtrage basé sur le contenu, cette approche n'intègre pas d'autres aspects de pertinence capables d'améliorer le filtrage (Adomavicius et Tuzhilin, 2005 ; Berrut et Denos, 2003 ; Schein *et al., 2002).*

1.8 Filtrage Hybride

L'approche de filtrage hybride repose sur l'idée de tirer profit des avantages des deux approches précédentes, en résolvant les problèmes qui leur sont liés. En fait, ces deux approches paraissent complémentaires. Les chercheurs du domaine estiment que le fait de combiner les deux méthodes pourrait être très bénéfique. D'où l'émergence de plusieurs techniques d'hybridation dont l'objectif consiste à combiner les deux approches (filtrage collaboratif et filtrage basé sur le contenu) de manière efficace (Adomavicius et Tuzhilin, 2005 ; Burke, 2002).

Les récents travaux dans ce domaine visent à développer des algorithmes hybrides de plus en plus efficaces. Burke (2002) décrit sept différents types de méthodes d'hybridation, présentées dans le tableau 2.3. Cependant, selon Adomavicius et Tuzhilin (2005), toutes ces méthodes se basent sur 2 approches principales. La première approche repose sur des méthodes basées sur la mémoire ((*Memory-based»* ou « *Heuristic-based»)* pour générer les recommandations (Balabanovi et Shoham, 1997 ; Basilico et Hofmann, 2004 ; Billsus et Pazzani, 2000 ; Claypool *et al.,* 1999 ; Good *et al.,* 1999 ; Liu et Shih, 2005 ; Melville, Mooney et Nagarajan, 2002 ; Pazzani, 1999 ; Tran et Cohen, 2000 ; Zaier, Godin et Faucher, 2008b). La seconde approche utilise des méthodes basées sur les modèles (<< *Model-based* ») pour déterminer les recommandations (Ansari, Essegaier et Kohli, 2000 ; Basu, Hirsh et Cohen, 1998 ; Condliff, Lewis et Madigan, 1999 ; Huang, Zeng et Chen, 2004 ; Li et Kim, 2003 ; Popescul *et al.,* 2001 ; Rojsattarat et Soonthomphisaj, 2003 ; Schein *et al.,* 2002 ; Soboroff et Nicholas, 1999 Van Setten *et al.,* 2004 ; Vellino et Zeber, 2007 ; Yu, Schwaighofer et Tresp, 2003).

Tableau 2.3 Méthodes d'hybridation (Burke, 2002)

Méthode	**Description**

d'hybridation	
Pondérée	Les résultats pondérés de plusieurs techniques de recommandation sont combinés pour produire une nouvelle recommandation.
Permutation	Le système permute entre les différentes techniques de recommandation selon le résultat de la recommandation.
Mixte	Les recommandations de plusieurs techniques sont présentées en même temps.
Combinaison	Différentes techniques de recommandation sont combinées en un unique algorithme de recommandation.
En cascade	Un système de recommandation raffine les résultats fournis par un autre système.
Augmentation	Le résultat (<< *output* >>) d'une technique de recommandation est utilisé comme données en entrée (<< *input* >>) pour l'autre technique.
Méta-niveau	Le modèle appris par une technique de recommandation est utilisé comme données en entrée (<< *input»)* pour ('autre technique.

1.9 Recommandation basée sur les données démographiques

La recommandation basée sur les données démographiques consiste à répartir les utilisateurs en plusieurs classes en fonction d'informations démographiques leur étant associées, telles que le sexe, l'âge, la profession, la localisation, etc. L'hypothèse sur laquelle repose cette approche est que deux utilisateurs ayant évolué dans un environnement similaire ont des codes esthétiques communs et sont donc plus susceptibles d'avoir des goûts communs que deux utilisateurs ayant évolué dans des environnements différents et ne partageant donc pas les mêmes codes. Un des avantages principaux de cette technique est qu'elle est applicable dès que les informations nécessaires sont obtenues, et permet de fournir des recommandations relativement satisfaisantes dès qu'un utilisateur commence à utiliser le système (Nguyen et al., 2006).

Cependant, pour des raisons de respect de la vie privée, il n'est pas toujours possible d'obtenir de telles informations.

1.10 Recommandation basée sur l'utilité

La recommandation basée sur l'utilité, parfois appelée recommandation basée sur les références, consiste à calculer les recommandations selon une fonction d'utilité pour l'utilisateur (Stolze et R., 2001). Toute la problématique est donc de définir une telle fonction d'utilité. Une façon de procéder est de demander aux utilisateurs de remplir des formulaires. Par exemple, dans le cadre de vente en ligne de micro-ordinateurs, il est possible de demander des renseignements sur l'usage qu'en fera le client. Un certain nombre de méthodes alternatives sont décrites dans (Huang, 2008).

1.11 Recommandation basée sur la connaissance

La recommandation basée sur la connaissance consiste à accumuler des informations relativement élaborées sur un utilisateur pour pouvoir ensuite lui recommander des ressources (Towle et Quinn, 2000). Une analogie avec la vie réelle serait par exemple une recommandation faite par un ami qui nous connaît bien et se serait basé sur des informations précises nous concernant, plutôt que sur nos préférences. Cette approche permet également d'expliciter des liens entre les ressources : par exemple que la cuisine chinoise est proche de la cuisine vietnamienne.

Une forme de recommandation basée sur la connaissance est la recommandation à partir de cas avec des attributs se rapportant à ce genre d'informations (Schmitt et Bergmann, 1999).

1.12 Les problèmes des systèmes de recommandation

Masse critique. Cet aspect illustre la difficulté à gérer le fait qu'il existe peu d'articles effectivement évalués, ou peu d'utilisateurs qui procèdent à ces évaluations. De ce fait, le système de recommandation ne possède que peu ou pas de données pour générer ces prédictions. Ainsi, il faut dépasser un nombre suffisant d'évaluations avant lequel les recommandations, fournies par le système, ne sont pas pertinentes (Adomavicius et Tuzhilin, 2005; Huang, Chen et Zeng, 2004 ; Vellino et Zeber, 2008; Ziegler, 2005).

Démarrage à froid. Souvent, on se retrouve confronté au problème qu'un uti1isateur ne soit comparable avec aucun autre. Ce problème est dû au fait que peu ou pas d'utilisateurs ont évalué un article donné, ou qu'un utilisateur donné a évalué très peu ou pas d'articles. Généralement, ce problème survient quand un nouvel utilisateur ou une

nouvelle ressource est ajouté à la base de recommandation (Adomavicius et Tuzhilin, 2005 ; Condliff, Lewis et Madigan, 1999; Leung, Chan et Chung, 2007; Schein *et al., 2002).*

Principe d'induction. Les systèmes de recommandation se basent sur le principe qu'un utilisateur qui a exhibé un comportement dans le passé tendra à exhiber un comportement semblable dans le futur (Vellino et Zeber, 2008). Cependant, ce principe n'est pas nécessairement valable dans le contexte réel. En effet, un utilisateur peut changer complètement de domaine d'intérêt ou en avoir plusieurs. Pour faire face à ce problème, des techniques de dérive d'intérêt *(Interest drift)* ou de changement de contexte *(Context shifts)* ont vu le jour (Bell, Koren et Volinsky, 2007c; Vellino et Zeber, 2008).

Sécurité et crédibilité. Les systèmes de recommandation ne peuvent pas empêcher les actes de tromperie. II devient ainsi facile de se forger une nouvelle identité et de se livrer au vandalisme, comme fournir de mauvaises informations au système. Il est à noter que dans les systèmes distribués exempts d'autorités centrales, il est plus difficile de contrôler l'identité des utilisateurs et de pénaliser le comportement malveillant. Une étude a montré qu'un pourcent de profils erronés suffit à pénaliser grandement un système de recommandation (Burke *et al.,* 2006). Par conséquent, il est indispensable d'avoir des moyens permettant à chaque utilisateur de décider en quels utilisateurs et en quels contenus avoir confiance (Massa et Avesani, 2004, 2007a, 2007b ; Memmi et Nérot, 2003).

Collecte des préférences. Une des étapes les plus importantes et les plus difficiles des systèmes de recommandation est la collecte des préférences des utilisateurs. En effet, l'obtention des évaluations de la part des utilisateurs sur une ressource donnée qui leur a plu, moins plu, ou pas du tout plus, est une tâche ardue. Ainsi, des techniques de collecte des préférences utilisateur, intrusives ou pas, ont vu le jour (Chan, 1999; Claypool *et al.,* 2001 Miller, 2003).

Complexité computationnelle. L'évaluation des similarités, une partie intégrale du filtrage de collaboration, implique quelques processus comportant de nombreux calculs. Pour un nombre important d'utilisateurs, le calcul de la similarité pour tous les individus du voisinage devient infaisable. Ainsi, une bonne performance peut seulement être

assurée en limitant ces calculs. Pour ce faire, il est important de restreindre suffisamment la taille de la communauté. Ainsi, des mécanismes intelligents de filtrage sont nécessaires ne sacrifiant pas trop d'information importante (Zaier, Godin et Faucher, 2008b ; Ziegler, 2005).

Protection de la vie privée. Un autre problème qui touche les systèmes de recommandation est la protection des informations sensibles constituant le profil utilisateur (information personnelle, intérêts, goûts, habitudes, etc.). Vu la nature de l'information, ces systèmes doivent assurer une telle protection. Ainsi, des moyens, pour préserver l'anonymat des utilisateurs et chiffrer les données transmises, sont nécessaires (Aimeur *et al.,* 2006 ; Aimeur *et al.,* 2008 ; Kobsa, 2007 ; Lam, Frankowski et Riedl, 2006 ; Memmi et Nérot, 2003).

1.13 Conclusion

Les approches des systèmes de recommandation sont particulièrement variées, et peuvent être classées de différentes manières. Dans ce chapitre, nous avons présenté les trois grandes approches: la recommandation basée sur le contenu, la recommandation collaborative et hybride ainsi que d'autres approches.

Dans les dernières années Les systèmes de filtrage d'information ont connu une avancée notable Qu'ils soient basés sur le contenu ou bien collaboratifs.

Le filtrage collaboratif est sans doute le type le plus utilisé il peut s'avérer très utile pour conseiller les utilisateurs sur certains items qu'ils n'ont pas vus, et dont on sait qu'ils, intéressent son groupe. Déterminer un profil thématique pour chaque utilisateur permet par contre d'évaluer si un item est pertinent ou non pour cet utilisateur, en fonction de ses centres d'intérêts identifiés. Or, ces deux aspects du profil paraissent importants. Connaître les goûts thématiques personnels paraît la meilleure méthode pour fournir une aide spécialisée au client, mais associer les clients entre eux permet de faire bénéficier chacun des opinions des autres.

Chapitre 2 : Le contexte

2 Le contexte

2.1 Introduction

La recommandation dépend de plusieurs facteurs tel que la date, le lieu, l'historique de l'interaction, tâche en cours, et d'autres facteurs non explicites mais implicites dans l'interaction et l'environnement de la recherche, appelés: *le contexte.* Dans ce chapitre nous allons présenter les différentes définitions du contexte ainsi que les domaines d'application.

2.2 Définitions

La définition générale du contexte que l'on trouve dans les dictionnaires est la suivante : «Ensemble que forment, par leur liaison naturelle, les différentes parties d'un texte ou d'un discours». On peut également souligner les définitions suivantes :

- «Le contexte d'un événement inclut les circonstances et conditions qui l'entourent» ;
- «En informatique, le contexte est l'ensemble des conditions sous lesquelles un dispositif est en train d'être utilisé, par exemple l'occupation actuelle de l'utilisateur» ;
- «En intelligence artificielle, le contexte est très fortement relié à ses propriétés en communication, linguistique et philosophie. La recherche scientifique est effectuée sur la façon dont ces aspects peuvent être modélisés dans des systèmes informatiques (par exemple basés sur la logique) pour l'utilisation dans le raisonnement automatique».

Clairement définir la notion de contexte pour l'informatique ubiquitaire a intéressé différents chercheurs. En effet, il est difficile de trouver une définition claire et unique, valable dans tous les travaux impliquant cette notion. Par exemple, pour Schilit et Theimer ; [4], le contexte est «la localisation et l'identité des personnes et objets à proximité». Pour Ryan et al [5]. Il s'agit plutôt de «la localisation, l'identité et l'heure». Dans un article plus poussé sur le sujet [6], Dey donne la définition suivante : «toute information pouvant être utilisée pour qualifier la situation des entités [...], typiquement la localisation, l'identité et l'état des personnes, groupes et objets informatiques et physiques». La définition la plus vaste a été donnée par Schilit et al. Dans l'un des premiers articles à aborder le sujet [7]:

« Contexte englobe plus que l'emplacement de l'utilisateur, parce que d'autres choses d'intérêt sont également mobile et changeant. Contexte comprend l'éclairage, le niveau de bruit, la connectivité réseau, les coûts de communication, la bande passante de communication, et même la situation sociale; par exemple, si vous êtes avec votre gestionnaire ou avec un collègue»

Paul Dourish aborde ce sujet pour l'auteur, contexte informatique courant est l'art de créer des systèmes pouvant détecter des aspects de la configuration dans laquelle ils sont utilisés, et agir en fonction [8]. Pour les sociologues, le fait que la plupart des systèmes interactifs n'aient pas cette capacité, La prise en compte du contexte devrait rendre les systèmes plus réactifs aux diverses configurations sociales dans lesquelles ils sont utilisés. [9].

Dans un de ses premiers articles sur l'informatique ubiquitaire [10], Weiser explique que l'idée de l'ubicomp est née de l'observation de la place de l'ordinateur dans les activités de la vie quotidienne. En particulier, des études anthropologiques montrent que la vie professionnelle est régie par des situations partagées et des compétences technologiques non examinées. La notion d'«actions situées» de Suchman est une source de l'idée que les systèmes informatiques devraient répondre aux configurations de leurs usages. Abowd [11] cite également Suchman en disant que le comportement humain est en priorité improvisé, par opposition à un plan prédéfini a priori que la personne exécuterait. L'informatique ubiquitaire basée sur la notion d'actions situées fonctionnerait également selon l'idée de l'improvisation comme comportement, et non d'un script prédéfini que l'usager suivrait.

En 2006 est paru un article de John Canny [12] investiguant le passé, présent et futur du domaine de l'interaction homme-machine. Selon lui, le futur est les technologies sensibles au contexte. L'auteur compare les nouvelles technologies qui lui sont contemporaines aux premières voitures : des «calèches sans chevaux» qui avaient encore des rênes, pour montrer leur inadéquation avec leurs nouveaux usages. Il donne l'exemple du téléphone portable qui a clairement été conçu pour l'interaction vocale et non textuelle. Pour l'auteur, le contexte a trois facettes : le contexte immédiat, le contexte de l'activité (incluant l'historique de l'utilisateur et de quelques autres personnes dans le cas d'une activité collaborative) et le contexte de la situation (permettant de savoir comment les autres acteurs se comportent généralement dans cette

situation). L'auteur souligne également qu'une même technologie peut être un grand succès ou échec selon le contexte dans lequel elle est employée. Par exemple, la reconnaissance vocale pour les ordinateurs de bureau n'a pas marché car l'interaction textuelle est la plus pertinente et pratique dans ce cas. Par contre elle promet d'avoir un plus grand succès dans le domaine des téléphones portables et de l'informatique médicale. Le système doit s'adapter à son utilisation. Enfin, le contexte peut jouer un rôle pour améliorer les algorithmes perceptifs des appareils.

Dans notre cadre d'un environnement intelligent, le contexte est l'ensemble des états des dispositifs et personnes dans l'environnement à un instant. Ces dispositifs et personnes évoluent de manière continue et en partie non prévisible.

Nous allons nous focaliser sur la définition de James L. Crowley et al. [13][14][15] qui s'inspire du théâtre comme source de concepts pour l'observation socialement avisée de l'activité humaine. Dans [16], Crowley soutient que l'activité humaine de tous les jours peut souvent être décrite, comme une pièce de théâtre, sous la forme d'un script. Ce script contient des définitions de rôles et une série de scènes composées elles-mêmes d'une série de situations. La définition d'un rôle contient un espace d'actions possibles, incluant des dialogues, mouvements et expressions émotionnelles. Ainsi, un grand nombre d'activités peut être décrit sous cette forme, notamment un cours à l'école, une réunion de travail, du shopping ou un dîner dans un restaurant. La grande différence entre une pièce de théâtre et la vie, concernant ces activités scénarisables, et qu'un script théâtral suit une séquence fixée de situations alors que la vie réelle est beaucoup moins contrainte et peut plutôt être vue comme un réseau ou graphe de situations, comprenant des boucles et des branches non déterministes. En cela Crowley est en cohérence avec le point de vue de Abowd [17] et Suchman [18] qui considèrent le comportement humain comme étant improvisé. Pour Crowley, l'improvisation réside dans la séquence des transitions empruntées dans le graphe des situations, mais les situations elles-mêmes (et les transitions possibles) peuvent être scénarisées. Crowley [19] propose alors une hiérarchie de définitions de concepts pour l'observation de l'activité humaine, appelée modèle de contexte :

Entité Une personne ou un objet physique ou informatique. Une entité est définie par un ensemble de variables observables corrélées. Par exemple, un vidéoprojecteur peut

être modélisé comme une entité avec pour propriétés, la pièce dans laquelle il se trouve, l'ordinateur auquel il est branché, etc.

Rôle Un filtre permettant de sélectionner, parmi toutes les entités perçues, celles qui sont pertinentes pour ce que l'on veut traiter. L'entité sélectionnée «joue» alors le rôle. Par exemple, Bob peut jouer le rôle de conférencier s'il se tient à côté du tableau et parle.

Relation Un prédicat sur les propriétés d'une ou plusieurs entités jouant des rôles. Par exemple, «à l'intérieur de» est une relation entre deux entités dont une joue le rôle de «bureau» et qui dépend de leurs positions.

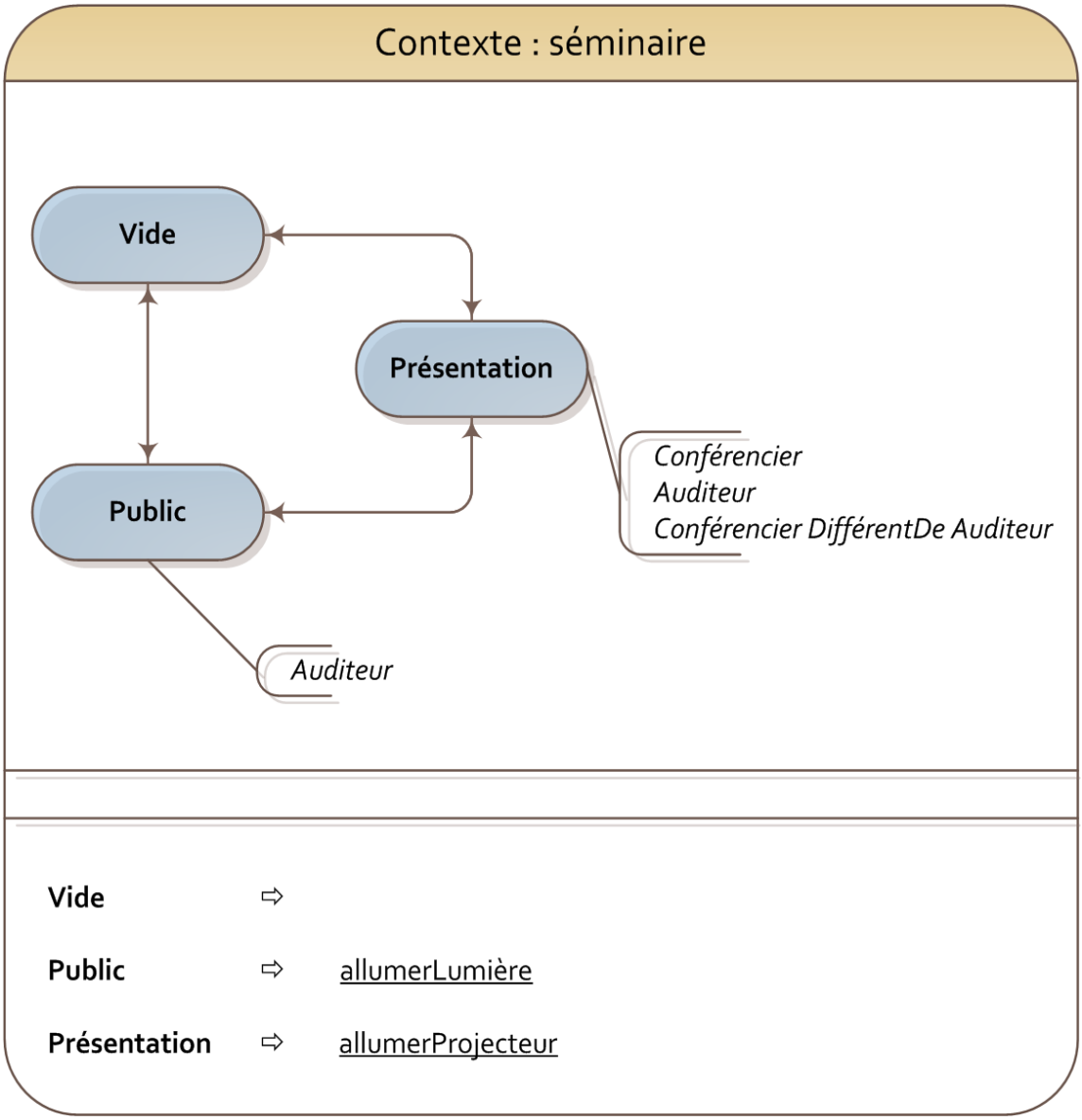
Situation Un ensemble de rôles et de relations. Par exemple, «donner une présentation» est une situation définie par les rôles «conférencier», «auditeur» et les relations «près du tableau» et «différent de».

Contexte contenu Une composition de situations partageant toutes le même ensemble de rôles et de relations pertinents pour la tâche donnée. Par exemple, le contexte «conférence» inclut la situation «présentation».

Définir le modèle de contexte (ou *modèle de situations*) pour une application donnée revient à définir les entités, rôles, relations, situations et arcs entre situations pertinentes pour l'application. Dans [20], Crowley et al. pensent que le contexte est un concept clé pour l'informatique ambiante *(context is key)*. Chaque *situation* est un état particulier de l'environnement, pertinent pour l'application et est définie par une configuration particulière des entités, rôles et relations. Les rôles et les relations sont choisis pour leur pertinence par rapport à la tâche. De même, seules les entités qui peuvent éventuellement jouer des rôles et être pertinentes pour la tâche sont prises en compte. Une situation représente une affectation d'entités à des rôles, complétée par un ensemble de relations entre les entités. Une situation peut être considérée comme «l'état» de l'utilisateur à l'égard de sa tâche. Si les relations entre entités changent, ou si la liaison entre les entités et les rôles change, alors la situation dans le contexte a changé. Pour détecter les changements de situation, une fédération de processus d'observation est nécessaire. Afin de fournir des services, des règles associant des actions à des situations sont définies. Ces actions sont déclenchées lorsque la situation dans laquelle se trouve le système change. Les services correspondants sont alors fournis aux utilisateurs.

Prenons par exemple le contexte d'un séminaire de travail. Les entités pertinentes sont les personnes. Les rôles possibles sont *conférencier* et *auditeur*. La seule relation utile

est *différent De*. Les situations sont Vide, Public et Présentation. Enfin, les actions possibles sont allumerLumière et allumerProjecteur. Le modèle de contexte «Séminaire» est alors représenté par le graphe de la figure II.1. Dans la situation Présentation (qui correspond à la phase où le conférencier fait son exposé), une personne doit jouer le rôle *conférencier* et les autres jouent le rôle *auditeur*. La relation qui doit être satisfaite est que l'entité qui joue le rôle *conférencier* est *différent d*es entités jouant le rôle *auditeur*. La situation Public correspond à la phase où l'audience est présente dans la salle mais la présentation n'a pas encore commencé (il faut donc au moins une entité jouant le rôle*auditeur*). Enfin, la situation Vide représente la situation initiale et finale où personne n'est (encore/plus) présent dans la salle. Lorsque la situation Public est enclenchée, l'action allumerLumière est exécutée ; de même, lorsque le conférencier est en place, la situation Présentation devient active et l'action allumerProjecteur est exécutée. Les affectations d'entités aux rôles ne sont pas bijectives. En effet, une ou plusieurs entités peuvent jouer le même rôle en même temps. De même, une entité peut jouer plusieurs rôles à la fois. Ces affectations peuvent changer dynamiquement. Ainsi, le comportement des humains dans le contexte peut faire passer l'environnement d'une situation à l'autre en respectant les flèches entre situations. L'enchaînement des situations à l'intérieur du contexte est appelé *script*.



(Figure II.1): Un exemple de modèle de contexte : séminaire. **Vide**, **Public** et **Présentation** sont les situations, *conférencier* et *auditeur* - les rôles, *différentDe* - la relation, et allumerLumière et allumerProjecteur sont les actions.

Définition du Context-Awareness

L'informatique du context-aware a été discutée par Schilit et Theimer en 1994 pour être un logiciel qui s'adapte en fonction de son lieu d'utilisation, la collecte des personnes et des objets environnants, ainsi que des changements à ces objets dans le temps. Toutefois, il est généralement convenu que la première enquête de recherche de l'informatique du context-aware était l'Olivetti active Badge (Want, Hopper et al. 1992) en 1992. Depuis lors, il y a eu de nombreuses tentatives pour décrire l'informatique du context-aware. La première définition d'applications du context-aware donné par Schilit et Theimer (1994) a élargi l'idée de context-awareness des applications qui sont simplement informés sur le contexte des applications qui s'adaptent au contexte. context-aware est devenu un peu synonyme d'autres termes: adaptatif (Brown, 1996a), réactif (Cooperstock, Tanikoshi et al. 1995), sensible (Elrod, Hall et al. 1993), situé (Hull, Neaves et al. 1997), contextuelle (Rekimoto, Ayatsuka et al. 1998) et de l'environnement dirigée (Fickas, Kortuem et al., 1997). Définitions précédentes de l'informatique du context-aware se divisent en deux catégories: utilisant le contexte et adaptation au contexte. Voyons d'abord le cas le plus général d'utilisation context-aware. Hull et al. (1997) et Pascoe et al. (Pascoe, 1998; Pascoe, Ryan et al 1998; Ryan, Pascoe et al 1998) définissent l'informatique du context-aware qu'elle est la capacité des appareils informatiques de détecter, interpréter et répondre aux aspects de l'environnement local de l'utilisateur et l'informatique des dispositifs eux-mêmes. Dey et al. ont défini context-awareness pour être l'utilisation de contexte pour automatiser un logiciel , de modifier une interface, et de fournir un maximum de flexibilité d'un service informatique (Dey 1998; Dey, Abowd et al 1998;.. Salber, Dey et al, 1999b).

Les définitions suivantes sont dans la catégorie plus spécifique adaptation au contexte. Beaucoup de chercheurs (Schilit, Adams et al, 1994;. Brown, Bovey et al 1997;. Dey et Abowd 1997; Ward, Jones et al 1997;. Abowd, Dey et al 1998;.. Davies, Mitchell et al 1998; Kortuem, Segall et al., 1998) définissent les applications context-aware pour être des applications qui changent dynamiquement ou adaptent leur comportement en fonction du contexte de l'application et l'utilisateur. Plus précisément, Ryan (1997) leur définit qu'ils soient des applications qui surveillent les données de capteurs environnementaux et permettent aux utilisateurs de choisir parmi une gamme de contextes physiques et logiques en fonction de leurs intérêts ou activités actuelles. Cette

définition est légèrement plus restrictive que la précédente par l'identification de la méthode dans lequel les applications agissent sur contexte. Brown (1998) leur définit des applications context-aware comme des applications qui fournissent automatiquement des informations et / ou prennent des mesures selon le contexte actuel d'utilisateur, tel que détecté par des capteurs. Il prend également un contexte au sens étroit en déclarant que ces actions peuvent prendre la forme de présentation des informations à l'utilisateur, l'exécution d'un programme en fonction du contexte, ou la configuration d'une mise en page graphique en fonction du contexte. Fickas et al. (1997) définissent l'environnement dirigé (synonyme pratique pour context aware) des applications pour être des applications qui surveillent les changements dans l'environnement et adaptent leur fonctionnement conformément aux directives prédéfinies ou définies par l'utilisateur. Dey et Abowd (2000) définissent context-awareness plus généralement avec l'énoncé suivant:

Un système est context aware s'il utilise contexte pour fournir des informations pertinentes et / ou services à l'utilisateur, où la pertinence dépend de la tâche de l'utilisateur.

2.3 L'historique des applications de Context-Aware

Comme mentionné précédemment, le système de Active Badge (Vous et al., 1992) est généralement considéré comme le premier système context-aware. Dans cette application, les utilisateurs portaient Les badges actifs, émetteurs infrarouges qui transmettent un code d'identification unique. Comme les utilisateurs déplacent tout au long de leur bâtiment, une base de données avait été mis à jour dynamiquement des informations sur chaque emplacement actuel d'utilisateur, l'extension du téléphone le plus proche, et la probabilité de trouver quelqu'un à cet endroit (selon l'âge des données disponibles. Lors d'un appel téléphonique a été reçu pour un utilisateur particulier, la réceptionniste a utilisé la base de données pour transférer l'appel à la dernière position connue de cet utilisateur, plutôt que de transférer aveuglément l'appel au bureau d'utilisateur, où il ne peut pas être localisé. Cette application, avec beaucoup des premiers travaux dans le context-aware a été axée sur l'informatique de la location courante, ou comme ils sont connus aujourd'hui, les services basés sur la localisation.

L'autre travail séminal dans le context-aware a été réalisé par le groupe Ubicomp au (Xerox PARC, au début des années 1990). Schilit et al. inventent le terme context-awareness, et construisent une architecture de système qui a soutenu la construction des applications (1994, 1995_, dans le cadre des travaux les très influent de PARCTAB _Want et al., 1995). Dans le travail du PARCTAB, les systèmes fournis présente de l'information aux utilisateurs en fonction de la proximité des services par exemple, des imprimantes et des personnes, tourne les dispositifs ou les reconfigurer en fonction des personnes qui sont à proximité , les informations ou les services basés sur l'endroit où se trouve l'utilisateur , et d'exécuter automatiquement un service d'une manière particulière en fonction de mouvement ou de la proximité des utilisateurs à des locaux ou des dispositifs spécifiques. Depuis lors, il y a eu un travail ayant appliqué context-awareness dans un grand nombre de domaines.

2.4 Les applications de Context-Aware

Il existe trois catégories canoniques d'applications: les guides touristiques, les systèmes de rappel et contrôle de l'environnement.

Les guides touristiques

Comme décrit précédemment, le guide mobile est l'application la plus canonique du context-aware. Lors de l'interaction avec un système de guide mobile, les utilisateurs portent un dispositif informatique portable comme ils voyagent à travers une certaine région comme un musée ou une ville. Comme les utilisateurs participent à différentes expositions ou des lieux touristiques, leurs appareils mobiles présentent des informations pertinentes à ces endroits. Alors que les premiers systèmes fortement axées sur la l'emplacement (Bederson, 1996;. Abowd et al, 1997), les systèmes ultérieurs ont pris en compte les intérêts des utilisateurs et la durée qu'ils ont passé à un lieu touristique ou le temps qu'ils ont eu à visiter, à choisir quelles informations et à afficher (Brown, 1996b;. Davies et al, 1998) quelles sites touristiques à recommander.

Reminders

Une deuxième application context-aware c’est le système de rappel context aware. context-aware reminders présente aux individus les rappels déclenchés par les changements dans le contexte, Un réveil utilise un déclencheur contextuel simple, temps, pour déclencher une alarme, un simple formulaire de rappel. De même, les

services basés sur la localisation peuvent envoyer des rappels lorsque les utilisateurs sont à un endroit particulier ou dans une certaine proximité les uns des autres _Schilit, 1995). Systèmes de rappel plus sophistiqués utilisent une combinaison de différentes formes de contexte d'utilisateur pour déclencher un rappel. En étant plus sophistiqué, ces applications peuvent rappeler aux utilisateurs de façon plus appropriée, offrant le droit de rappel dans la bonne situation de _Dey et Abowd, 2000;. Ludford et al, 2006).

Les contrôles environnementaux

Une troisième application canonique context-aware est un système de contrôle du chauffage et d'éclairage de l'environnement, généralement pour économe en énergie ou économiser les efforts des utilisateurs. Comme beaucoup de gens laissent souvent les lumières allumées inutilement, ou modifier manuellement le chauffage ou le niveau de refroidissement pour rester confortable, de nombreux systèmes ont été développés qui peuvent contrôler ces activités au nom des utilisateurs. Certains sont basés sur des règles simples _Elrod et al., 1993), tandis que d'autres utilisent des mécanismes plus sophistiqués pour savoir comment les utilisateurs utilisent un espace et définit chauffage et d'éclairage en conséquence (Mozer, 1998).

Les applications contemporaines de Context-Aware

Au-delà de ces trois domaines canoniques, les chercheurs et les développeurs ont créé des applications dans une grande variété de domaines, y compris:

- Communications interpersonnelles, y compris la messagerie instantanée (Avrahami et al, 2008;)
- Appels téléphoniques (Schmidt et al, 2000.);
- Santé (Bardram et Nørskov, 2008);
- Systèmes de géo localisation;
- Agriculture (Kjaer, 2008);
- Personnalisation de l'application (Weiss et al., 2008)
- A périphériques qui affichent des informations (Wisneski et al, 1998);

Un plus grand nombre de chercheurs construisent des applications, mais leur objectif n'est pas context-awareness. Ils sont tout simplement en train de construire, une application convaincante utile qui se trouve d'être context aware. Même si cela peut

sembler une petite distinction, context-awareness aura certainement atteint un niveau de maturité approprié quand il est généralement considéré comme un élément d'application, plutôt que comme la mise au point d'une application.

2.5 Conception et implémentation d'applications de Context-Aware

Processus de conception

Le processus de conception pour la construction de la grande majorité des applications context-aware peut se résumer à une idée assez simple: comprendre quel contexte a besoins votre application et quand vous recevez ce contexte, déterminer ce que vous voulez faire avec elle. Malheureusement, cela prend un peu plus de travail que cela pour construire une application canonique context-aware. La première étape dans la construction d'une application est la spécification du contexte: déterminer quels sont les comportements context-aware votre application aura et dans quelles situations ou collections de contexte chaque comportement doit être exécutées et comment. Cette étape est souvent réalisée par l'étude d'une combinaison du domaine et les efforts des utilisateurs prévus de l'application, et en déterminant quels services seraient bénéfiques. La deuxième étape est l'acquisition de contexte: déterminer quels capteurs matériels et / ou logiciels sont nécessaires pour acquérir le contexte identifié dans la première étape. Cela comprend l'installation du logiciel sur la plate-forme appropriée, comprendre exactement ce type d'information que le capteur fournit, à l'aide d'une API (s'il en est pour communiquer avec le capteur, déterminer comment interroger le capteur et comment pour être prévenu lorsque des changements se produisent, et, s'il y a lieu, mémoriser le contexte, le combiner avec un autre contexte, et exécuter l'inférence sur les données de capteur pour atteindre une certaine compréhension de niveau supérieur de la situation. La troisième étape est la livraison du contexte: en précisant comment le contexte doit être livré à partir des capteurs aux applications context-aware éventuellement à distance qui utiliseront le contexte. La quatrième étape est la réception du contexte: avoir l'application précise quel contexte, elle est intéressée par (indiquant peut-être à partir de quel capteurs) et la réception de ce contexte. Cela inclut la conversion du contexte en une forme utilisable par l'application par l'interprétation, et l'analyse du contexte afin de déterminer si ce contexte, lorsqu'il est combiné avec d'autres contextes disponibles, décrit une situation pertinente pour l'application. Enfin, la dernière étape est l'action: analyse de l'ensemble du contexte reçu pour déterminer

quel comportement context-aware pour exécuter, puis l'exécuter. Alors que les applications peuvent être plus complexes que cela, la plupart des applications peuvent être décrites et construites à l'aide de ce processus de conception simple ou plusieurs instances de celui-ci pour les applications ayant des comportements multiples).[21]

Outils pour construction

Sur la base de ce processus de conception, un certain nombre de boîtes à outils ou des architectures logicielles ont été construit pour soutenir la construction d'applications context-aware. Bien que chaque développeur a besoin pour effectuer l'étape de spécification pour son application, le reste du processus de conception peut ou peut ne pas être pertinent en fonction de quelle boîte à outils sous-jacente offre le développeur. Dans le meilleur des cas, la spécification des situations pertinentes et les comportements correspondants serait suffisante pour causer le contexte de s'écouler à partir des capteurs appropriés aux applications, avec l'inférence et l'interprétation produisant pertinente, et alors le comportement du context-aware correcte en cours d'exécution.

En général, il existe trois principales alternatives utilisées pour des applications de construction: pas de soutien, un widget ou un système à base d'objets, et un système basé sur le tableau noir. La grande majorité des systèmes, en particulier ceux construits avant 1998, ont été construits sans soutien architectural ou de soutien hautement personnalisée pour une petite classe d'applications. Pour chacune de ces applications, le processus de conception doit être appliqué et le logiciel mis en œuvre pour soutenir le processus sur une base individuelle. Très peu, si tout code, peut être réutilisé entre les instances d'application, comme la plupart du code est personnalisé pour chaque application.

Avant 1998, les exceptions étaient l'architecture de l'informatique du Context-Aware de Schilit (1995) sur laquelle la plupart, sinon tous, les applications context-aware à Xerox PARC étaient fondées sur et à l'Université de Kent plate-forme Notes Stick-e (Brown, 1996b). L'architecture de l'informatique context-Aware est écarté de devoir réécrire la détection de l'emplacement des modules la livraison à partir de zéro de contexte à chaque fois, et a permis aux développeurs d'applications au lieu de se concentrer sur les détails de ce qui devrait être fait quand un ou plusieurs utilisateurs étaient en un emplacement particulier. La plate-forme Notes Stick-e a fourni de même une

infrastructure commune pour construire des applications context-aware, mais vise à faire de la création d'applications accessibles aux utilisateurs finaux (en rendant le processus similaire à celui de la construction de pages Web).

Depuis 1998, un grand nombre d'infrastructures context-aware ont été construits, en utilisant soit l'approche à base de widgets ou l'approche fondée sur le tableau noir. Ces infrastructures ont été développées principalement pour fabriquer des composants du système context-aware réutilisable, de fournir une infrastructure exécutable persistante qui pourrait prendre en charge plusieurs applications exécutés en même temps et de simplifier la construction d'applications context-aware.

L'approche générale à base de widgets est basée sur le modèle de la création d'interfaces graphiques. Au début des années 1980 ", il n'y avait pas de modèle commun pour la gestion des entrées de l'utilisateur, et pas de support pour la réutilisation des widgets. Au milieu des années 1980 ", nous avons commencé à voir les outils de l'interface graphiques qui soutenaient la réutilisation, y compris l'infrastructure pour la gestion des événements d'entrée et une bibliothèque de composants ou un widget avec des widgets qui pourraient être utilisés sur de multiples applications. Ces outils ont fait fondamentalement plus faciles de construire des interfaces utilisateur graphiques et ont fourni les avantages suivants:

- Ils cachent les spécificités de dispositifs d'interaction physiques du programmeur de l'application de sorte que ces dispositifs peuvent changer avec un impact minimal sur les applications. Si les points d'utilisateur et de clics avec la souris ou avec les doigts sur un touchpad ou utilise les raccourcis clavier ne nécessite pas de modifications à l'application.
- Ils gèrent les détails de l'interaction pour fournir des applications avec des résultats pertinents des actions de l'utilisateur. Dialogue Widget-spécific est gérée par le widget lui-même, ainsi que l'application ne doit souvent mettre en oeuvre un seul rappel pour être informé du résultat d'une séquence d'interaction.
- Ils fournissent les blocs de construction réutilisables de présentation à définir une fois réutilisés, combinés et / ou adaptés pour une utilisation dans de nombreuses applications. Widgets fournissent l'encapsulation de l'apparence et le comportement. Le programmeur n'a pas besoin de connaître le fonctionnement interne d'un widget pour l'utiliser.

D'une manière similaire, dans la fin des années 1990 et le début des 2000, des outils context-aware, comme le Context Toolkit (Dey et al., 2001) et le Cadre de Java Context-Aware Framework (Bardram, 2005) a adopté un modèle similaire et a adopté une approche de widget contexte. Un contexte de widget est un composant de logiciel qui permet aux applications d'accéder aux informations de contexte à partir de leur environnement d'exploitation. De la même manière widgets GUI isolent les applications de certaines préoccupations de présentation, des widgets contextuels isolent les applications de préoccupation de l'acquisition de contexte, en enveloppant des capteurs avec une interface uniforme. Widgets contextuels offrent les avantages suivants:

- Ils fournissent une séparation des préoccupations en cachant la complexité des capteurs réels utilisés par l'application. Que ce soit la présence de personnes détectée en utilisant Badges actifs de présentation, des capteurs de sol, traitement d'images vidéo ou une combinaison de ceux-ci ne devraient pas influer sur la conception de l'application.
- Ils abstraient le contexte d'information pour répondre aux besoins prévus des applications. Un contexte de widget qui permet de suivre l'emplacement d'un utilisateur à l'intérieur d'un immeuble ou d'une ville informe l'application que lorsque l'utilisateur se déplace d'une pièce à l'autre, ou d'un coin de rue à l'autre, et ne signale pas moins d'importants mouvements à l'application. Widgets fournissent des informations abstraites que nous attendons que des applications ont besoin le plus souvent.
- Ils offrent un accès facile aux données de contexte par l'interrogation et les mécanismes de notification disponibles par le biais d'une interface uniforme commune pour l'accès au contexte. Peu importe quel genre de contexte est détecté de l'environnement, et tous les widgets le rendent disponible de la même manière.
- Ils fournissent les blocs de construction réutilisables personnalisables de contexte sensible. Un contexte de widget qui permet de suivre l'emplacement d'un utilisateur peut être utilisé par une grande variété d'applications, de guides touristiques des systèmes de sensibilisation de bureau. En outre, des widgets contextuels peuvent être adaptés combinés de façon similaire aux widgets GUI. Par exemple, Présence widget détecte la présence de personnes dans une chambre. Un contexte de widget de réunion

peut être construit sur d'un widget de présence et d'assumer une réunion commence lorsque deux ou plusieurs personnes sont présentes.

En plus de widgets contextuels, ces infrastructures fournis des composants pour combiner des informations de contexte, de faire des inférences de niveau supérieur à partir du contexte de faible niveau, la collecte ou agrégation de contexte de multiples sources et de découvrir les composants appropriés à utiliser. Comme une collection, non seulement ils fournissent des composants qui peuvent être composés pour créer des applications individuelles, mais servent comme une infrastructure distribuée persistante qui peut servir plusieurs applications en même temps. Cette approche est de plus en plus commercial, avec Microsoft Windows 7 inclusion d'un capteur et plateforme de localisation qui fournit une API pour accéder à l'emplacement et capteurs de lumière, entre autres, pour permettre la construction d'applications context-aware. La principale approche contraste avec l'approche de widget est l'approche basée sur le tableau noir. Tableaux proviennent de la langue de programmation de Linda et tuple-space de début des années 1980 (Gerlernter, 1985). Linda a supporté un petit nombre de fonctions permettant d'interagir avec un système de stockage persistant accessible globalement, l'espace de tuple. Composants exécutables peuvent placer des informations dans le système de stockage, et de lire cette information ou les supprimer. Les systèmes ultérieurs de tableau noir ont ajouté de la possibilité de notifier composants lorsque l'information d'intérêt a été ajoutée au tableau.

Un certain nombre d'infrastructures qui soutiennent la construction d'applications context-aware utilisent cette approche de tableau noir y compris EQUIP (Greenhalgh, 2002), the Event Heap (Johanson et Fox, 2002), et l'Open Agent Architecture (Cheyer et Martin, 2001). Contrairement à l'approche à base de widgets dans lesquels les demandes d'information sont souvent faites directement ou indirectement à un composant particulier par le biais d'un mécanisme de découverte, l'approche tableau noir permet demandes d'informations à venir à et être manipulés par un tuple-space unique et centralisé. En général, systèmes de contexte sur la base tableau noir ont tendance à être mises en œuvre de systèmes traditionnel tableau noir, optimisés pour la performance et augmentée pour soutenir l'encodage XML et le décodage de l'information.

Les deux approches ont leurs avantages et leurs inconvénients. Approches Blackboard sont souvent trop inefficace d'autant plus que la quantité de données dans l'espace de tuple-space se développe et recherche de tuples spécifiques devient plus difficile, peut souvent être trop étroitement intégré avec les applications qu'ils soutiennent pour le rendre facile de modifier ces applications, peut souvent souffrir de problèmes de synchronisation aussi l'ordre des opérations tuple-space n'est pas nécessairement garanti, et généralement ne prend pas en charge les structures de données complexes. Le modèle à base de widgets peut souvent être trop rigide, de perspective du robustesse et de configurabilité , pour gérer la nature dynamique des applications context-aware qui doivent se connecter et se déconnecter de différentes composantes comme un utilisateur se déplace et son contexte change, et peut fournir un modèle plus complexe à traiter à partir du point de vue d'un développeur d'applications .

Cependant, ces deux approches ne s'excluent pas mutuellement. De la perspective du développeur de l'infrastructure, les deux approches ont besoin de sources de l'entrée de contexte, ou l'autre de créer des widgets autour ou à placer dans le contexte d'un espace de tuple. L'approche de widget contexte fournit un bon modèle pour l'utilisation de sources de contexte, quelle que soit l'approche qui est effectivement utilisé pour supporter des applications. D'autre part, le modèle de tableau noir fournit une abstraction plus simple pour ces sources de contexte à fournir effectivement leur contexte pour le consommateur de contexte. Ce modèle peut être utilisé dans l'approche du widget, où widgets peuvent être dit de se connecter à un composant centralisé particulier (comme un système de découverte), qui détermine alors comment acheminer le contexte d'un widget à un consommateur. De perspective du développeur d'une application, la capacité pour les deux abstrait les détails de la technologie de détection sous-jacente via les widgets de contexte et de ne pas se soucier des connexions individuelles à des composants via les tableaux ou les widgets de contexte et un mécanisme de découverte, rend les choses plus faciles. En fait, il y a eu des travaux récents en augmentant le modèle de contexte widget pour inclure une découverte avancée et mécanisme de connexion qui exploite les avantages des deux approches le contexte widget et tableau noir (Dey et Newberger, 2009).

2.6 La construction d'applications Context-Aware

Maintenant que nous avons discuté du processus de conception et différentes approches pour soutenir les applications Context-Aware, nous allons discuter des aspects que les concepteurs doivent tenir compte lors de la construction des applications.

Le contexte est un indicateur de l'intention humaine

Le Saint Graal de context-awareness est de deviner ou comprendre l'intention humaine. Les applications utilisent cette intention humaine de s'adapter de manière appropriée en fournissant des informations ou de prendre certaines mesures. Toutefois, les informations de contexte est seulement un proxy pour cette intention. Dans les guides de musée, un utilisateur debout devant une exposition particulière peut ne pas représenter l'intérêt à l'exposition, mais peut simplement représenter qu'un utilisateur a le dos tourné à l'exposition et est d'avoir une conversation avec un ami là-bas. Contexte supplémentaire est nécessaire pour comprendre l'intention humaine. Cela est vrai pour presque n'importe quelle situation et application. Peu importe combien de contexte vous pouvez acquérir pour une application, vous aurez toujours besoin de plus pour déterminer vraiment l'intention de l'utilisateur. Lors de la conception d'une application, un développeur doit porter d'une manière appropriée les types de situations que la demande devrait détecter et comprendre, sachant que l'application peut faire une adaptation incorrecte lorsque une situation survient que c'est hors de la portée ou de la compréhension de l'application conçue. Les demandes doivent exprimer une certitude dans leurs croyances sur leur information détectée et inférences et si cette certitude n'est pas supérieur à un seuil approprié, demander confirmation de ces informations avant d'agir, comme en témoigne l'agent Microsoft Clippy et le système Lookout (Horvitz, 1999).

Contexte inférences

Systèmes context-aware prennent des données comme entrée, puis déterminent comment adapter ou répondre à ces données. Contexte inférence est l'acte de donner un sens à ces données d'entrée à partir de capteurs et d'autres sources, pour déterminer ou inférer la situation de l'utilisateur. Une fois la situation de l'utilisateur a été déduite, l'application peut prendre une action appropriée. Comme décrit dans la sous-section

précédente, l'entrée détectée n'est pas souvent suffisante pour en déduire la situation de façon appropriée, de sorte que cela soulève d'autres questions telles que la façon de résoudre l'ambiguïté ou l'incertitude dans le contexte, et le rôle des règles et l'apprentissage de la machine. Ces questions seront examinées dans les sous-sections suivantes.

L'ambiguïté du Contexte

Systèmes context-aware disposent de nombreuses sources d'ambiguïté ou d'erreurs: capteurs de contexte peuvent détecter mal, échouent, ou être pas sûr de ce qu'ils ont détecté, les systèmes d'inférence de contexte peuvent d'une manière inexacte parvenir à des conclusions sur une situation ou être pas sûr de leurs conclusions, et les applications peuvent prendre une action incorrecte ou être pas sûr des mesures à prendre. Cependant, la grande majorité des systèmes context-aware font semblant que cette source d'ambigüité n'existe pas, et de se comporter comme s'il n'y avait pas d'erreurs ou d'ambiguïté. Une décision importante pour les concepteurs d'infrastructures et d'applications context-aware est de savoir si le contexte sera modélisé comme étant ambigu ou non. Agissant comme s'il n'y avait pas d'ambiguïté dans un système simplifie la conception et l'utilisation d'un système, mais conduit à des systèmes fragiles face à l'ambiguité. En revanche, en acceptant que ambiguïté signifie que de meilleurs systèmes modélisés représentent le monde réel, mais sont plus difficiles à créer et à utiliser. L'ambiguïté est souvent définie comme un seul nombre représentant la probabilité ou la certitude d'une valeur de contexte particulier.

Une façon de traiter avec le contexte est de combiner de multiples sources disparates d'un même type de source pour améliorer l'exactitude ou la fiabilité du contexte fourni. Ceci est communément connu comme la fusion de capteurs (Wu, 2003). Par exemple, dans la reconnaissance de l'activité, un modèle de Markov caché peut être utilisé avec différents capteurs et les résultats fusionnés peuvent être représentés comme une matrice de confusion sur l'ensemble des activités possibles (Lester et al., 2006). Une autre approche consiste à permettre aux utilisateurs de lever l'ambiguïté manuellement dans le contexte (Dey et Mankoff, 2005). Plutôt que de s'appuyer sur une approche automatisée, cette approche s'appuie sur la connaissance de l'utilisateur de la situation pour aider à résoudre et éliminer toute ambiguïté dans le contexte détecté ou déduit. Un utilisateur peut être présenté, par exemple, par une liste des N meilleures, une liste des

N interprétations les plus probables de contexte, classé selon leur indice de probabilité, et a demandé de choisir l'interprétation "correcte".

Règles vs L'apprentissage automatique

Applications context-aware sont les plus couramment conçues à partir d'un ensemble de règles si alors: si l'application détecte une situation particulière, il doit effectuer une action particulière. Les règles sont faciles à créer que toute la connaissance pour chaque règle est représentée dans un format homogène, et les systèmes à base de règles sont relativement faciles à construire car il a un grand nombre de moteurs de règles existantes qui déterminent quand une règle a été satisfaite. Les règles sont aussi relativement intuitives et sont donc facile à travailler. Cependant, les systèmes à base de règles ont aussi quelques inconvénients: ils sont enclins à des conflits entre les règles en raison de dépendances cachées entre les règles faisant qu'il est difficile de comprendre l'impact de l'ajout ou la suppression d'une règle d'autant plus que le nombre de règles s'accroit, ils sont notoirement difficiles à déboguer lorsque le nombre de règles s'accroit comme il est difficile de comprendre ou de suivre le flux de contrôle entre les règles, et ils sont une source d'inefficacité comme un moteur de règles doit examiner toutes les règles pour déterminer quelle règle s'il y en a , doit être exécutée. Enfin, les règles sont rigides, ce qui signifie qu'ils sont également fragiles. Petites exceptions à une règle signifie qu'une règle ne sera pas congédié ou une règle différente sera tiré, que celui le développeur souhaite.

Une approche alternative commune c’est appliquer l'apprentissage automatique. Plutôt que de créer une série de règles sur la façon dont une application doit adapter son comportement, au contraire, un développeur d'applications peut recueillir des données sur les types de situations que l'utilisateur fera l'expérience et le types d'adaptation désirées. L'apprentissage automatique peut ensuite être appliqué pour apprendre les relations probabilistes entre les situations et les adaptations, plutôt que d'avoir ces relations être codés en dur et déterministe. Ceci nécessite encore que l'application ou de l'infrastructure de support offrent la possibilité d'effectuer le contexte inférence pour mapper les données de capteur aux situations d'utilisation. Sinon, l'apprentissage automatique peut être appliqué à apprendre les relations entre les données des capteurs et l’adaptation, en sautant l'étape intermédiaire de contexte inférence. Les inconvénients de l'application de l'apprentissage automatique sont que ces relations peuvent être

difficiles à apprendre, peuvent nécessiter une grande quantité de données à apprendre, sont difficiles à déboguer, et peuvent ne pas être intuitive pour le développeur de l'application ou de l'utilisateur final.

Confidentialité

Systèmes context-aware ont la capacité et souvent la nécessité de recueillir d'énormes quantités d'informations sur les individus. Un danger inhérent à la collecte de ces informations c'est à la divulgation de renseignements à la mauvaise personne ou au cours de la mauvaise situation. D'autant plus que les systèmes de contexte sont distribués à travers un réseau ou réseaux informatiques, il existe une réelle préoccupation que les informations de contexte pourraient être diffusés de manière inappropriée ou diffusées à un composant qui n'est pas entièrement digne de confiance, et peut diffuser plus d'informations. Les développeurs de l'infrastructure context-aware doivent s'assurer que les données sont partagées uniquement entre les composants qui ont un besoin réel de partager et d'utiliser cette information. De même, les développeurs d'applications context-aware doivent s'assurer que la vie privée de l'utilisateur est maintenue et que les informations ne sont pas utilisées de façon inappropriée, ou d'une manière que l'utilisateur juge inapproprié. Il n'y a pas de réponses faciles sur la façon de soutenir la vie privée.

Evaluation

Applications context-aware, comme la plupart des applications de UbiComp, sont difficiles à évaluer. Parce que les applications context-aware sont bien contexte dépendent, ils peuvent rarement être testés dans un environnement de laboratoire. Les contextes ou situations d'intérêt ne peuvent souvent pas être simulées. Même dans les évaluations sur le terrain, ces situations peuvent être assez rares qu'il peut être difficile de tester et d'évaluer dans d'autre chose d'une manière qualitative. Comment évaluer ces demandes a fait l'objet de beaucoup de recherches en cours (par exemple, (Carter et al, 2008;. O "Neill et al, 2007;.. Oh et al, 2007; Scholtz et Consolvo, 2004)).

Issues de l'utilisateur final

Le but de toute application context-aware c'est de fournir aux utilisateurs un soutien approprié en fonction de leur situation actuelle. Jusqu'à récemment, le développement

d'applications context-aware a limité participation des utilisateurs à l'observation dans les étapes de formation et être utilisateurs actifs après le déploiement. Cependant, il y a deux questions de l'utilisateur final que les développeurs doivent prendre en compte lors de la construction de leurs applications. La première est l'intelligibilité des applications. Comment un utilisateur fait une compréhension d'un comportement d'application context-aware (Bellotti et Edwards, 2001). Comme le contexte se compose principalement d'entrée implicite, il est beaucoup plus difficile pour un utilisateur de comprendre qu'une application a pris des mesures sur la base de l'entrée non explicite, pour comprendre quelle entrée non explicite qui a été et de comprendre ce que l'action a été prise même (dans certains cas). Contrairement aux applications typiques non UbiComp, les applications context-aware n'ont pas toujours un mécanisme pour fournir une rétroaction appropriée aux utilisateurs pour indiquer que certaines mesures ont été prises ou pourquoi. Ceci est assez difficile pour les systèmes basés sur des règles, mais encore plus difficile pour les systèmes basés sur l'apprentissage automatique car généralement ils ne se prêtent pas aux explications générées. Il s'agit d'un domaine de recherche actif (par exemple, (Assad et al, 2007;. Dey et Newberger, 2009; Lim et al, 2009; Tullio et al, 2007)), avec une série d'ateliers sur le thème de L''Informatique Explanation-Aware.

Une deuxième question de l'utilisateur final de préoccupation est le contrôle. Développeurs conçoivent souvent leurs applications basées sur la connaissance d'un domaine particulier. Toutefois, les applications context-aware doivent également être personnalisées à leurs utilisateurs, Plutôt que de développer des variations individuelles d'applications pour chaque utilisateur, les développeurs devraient envisager de mettre le contrôle de la façon dont une application doit se comporter dans les mains des utilisateurs, c'est à dire, permettant aux utilisateurs de créer des règles qui régissent comme un comportement application context-aware. En particulier pour les applications de longue durée quand il peut y avoir beaucoup de changements (à l'environnement, aux situations que l'utilisateur fait de l'expérience, et / ou à des préférences que l'utilisateur a pour le comportement de l'application), les utilisateurs devraient avoir la possibilité de modifier et de personnaliser l’application à leurs besoins. Par exemple.L'ICAP permet aux utilisateurs finaux de créer visuellement des applications context-aware fondé sur des règles. (Dey et al., 2004), et un CAPpella soutient les utilisateurs finaux dans la

création d'applications context-aware simples en utilisant une programmation par l'approche de démonstration (Dey et al., 2006). En abordant à la fois l'intelligibilité et le contrôle, les développeurs peuvent augmenter de manière significative la convivialité des applications context-aware.

2.7 Conclusion

Ce chapitre introduit le concept de la sensibilité au contexte, un aspect fondamental dans l'informatique ubiquitaire. Nous avons présenté un point de vue sur l'historique de la sensibilité au contexte. Nous avons étudié aussi un large éventail d'applications sensibles au contexte, et discuté des approches différentes pour la création d'applications sensibles au contexte La sensibilité au contexte est un élément central de plusieurs systèmes. Bien que le domaine de l'informatique contextualisée soit déjà en pleine expansion, de plus en plus de capteurs prolifèrent partout dans le monde, sa croissance va s'accélérer et son importance ne fera que croître.

Chapitre 3 : La recommandation contextuelle

3 La recommandation contextuelle

3.1 Introduction

Apres avoir vu la recommandation et le contexte dans les chapitres précédents, dans ce dernier chapitre, nous commençons par faire la conception de notre système, Ensuite, nous détaillons comment faire la recommandation par contenu en se basant sur l'indexation et l'appariement contexte-documents et collaborative en se basant sur les votes. Enfin, nous décrivons la réalisation de notre application.

3.2 Conception du système

Nous avons suivi la méthode UP pour mettre en évidence notre démarche nous commencerons dans un premier temps par tracer le diagramme de cas d'utilisations qui nous aidera à définir les diagrammes de séquence pour générer le diagramme des classe représenté.

3.3 Diagramme de cas d'utilisation

L'utilisation de notre indicateur est autorisée seulement par le médecin pour afficher la fiche de suivi d'un patient et consulter les documents recommander par le système.

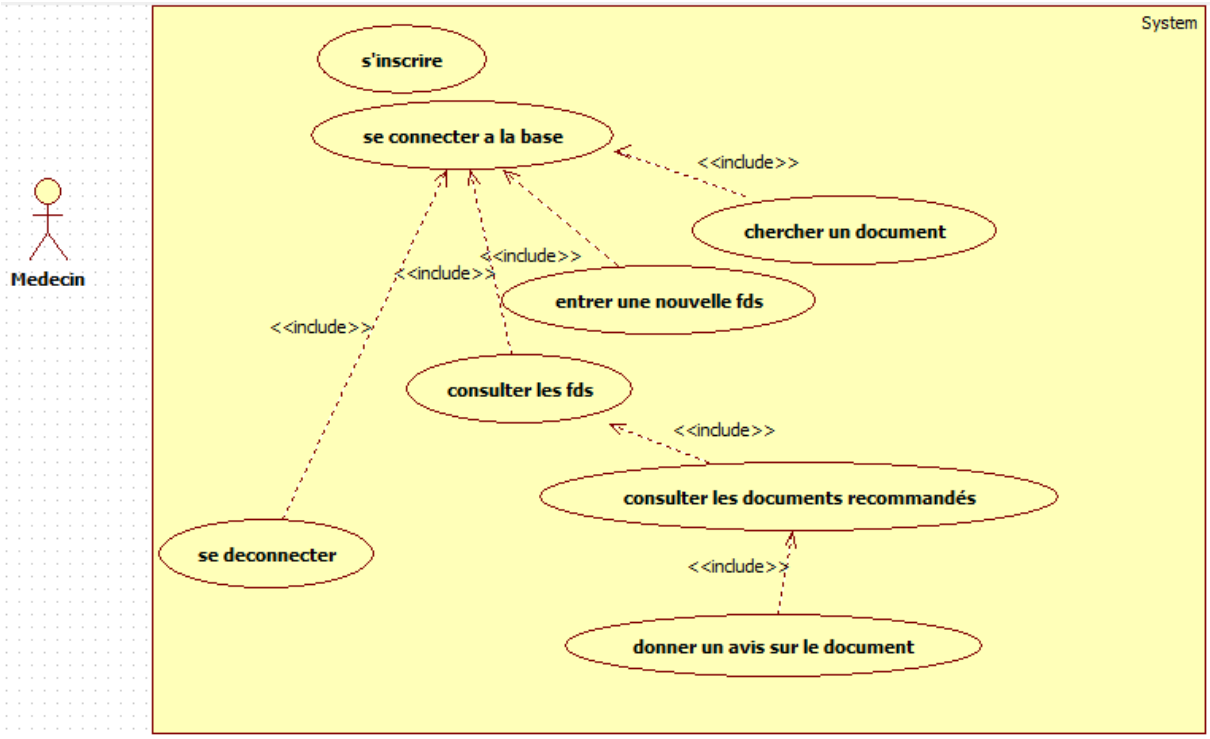


Figure III.1: Diagramme de cas d'utilisation

3.4 Diagramme de séquence

Les diagrammes de séquence décrivant les différents scénarios des actions réalisées par les acteurs du système, seront détaillés dans les figures suivantes. On a deux diagrammes de séquences

S'authentifier et chercher la fiche de suivi

Le médecin fait entrer ces coordonnées pseudo et mot de passe, le système vérifie ces informations et lui rend la réponse, une fois c'est fait le médecin cherche la fiche de suivi d'un patient.

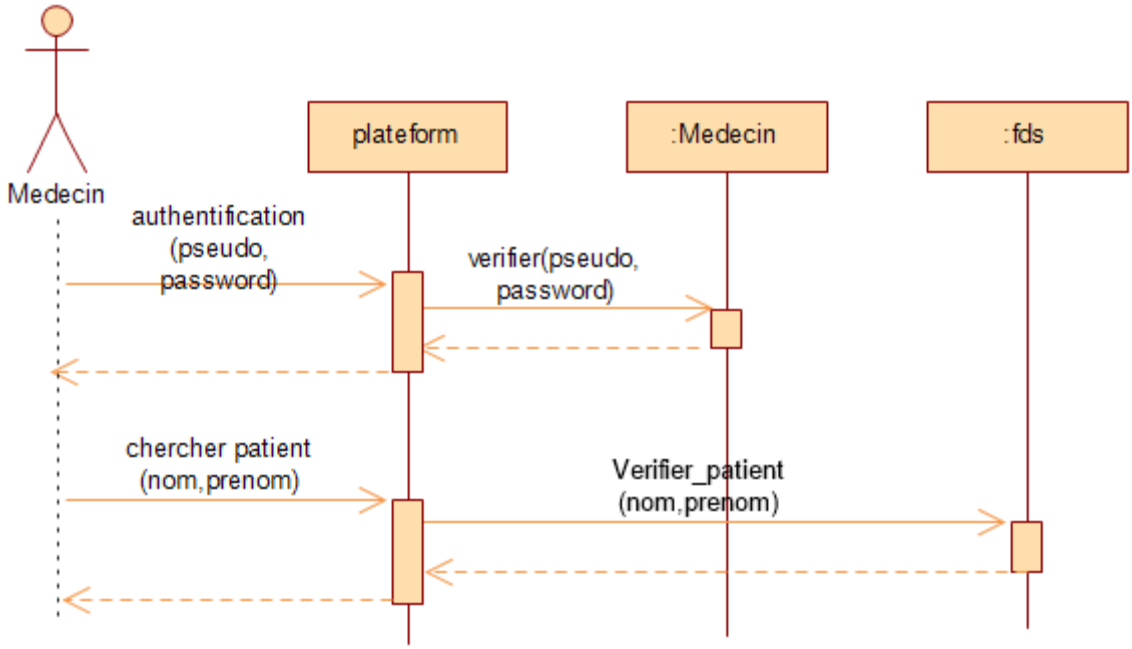
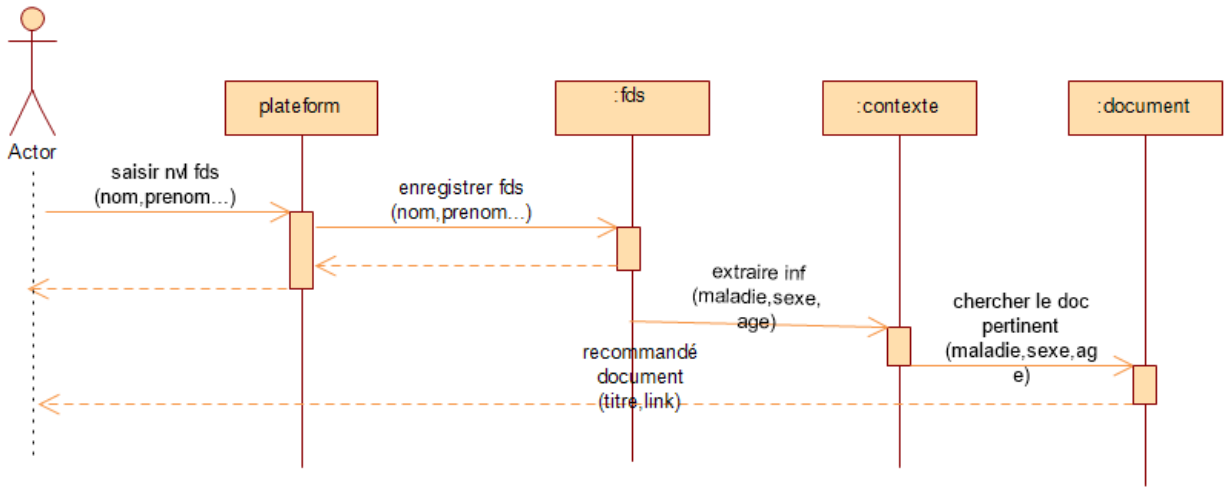


Figure III.2 digramme de sequence authentification et recherche

Extraction du contexte et recommandation

Le médecin commence par entrer un nouveau patient en remplissant la fiche de suivi (nom, prénom, maladie, âge … etc.), le système enregistre cette nouvelle fiche de suivi et extraire le contexte du patient (maladie, Age, sexe), une fois c'est fait il compare le contexte avec les documents (abstract), à la fin le système recommande les documents qu'il trouve pertinent.

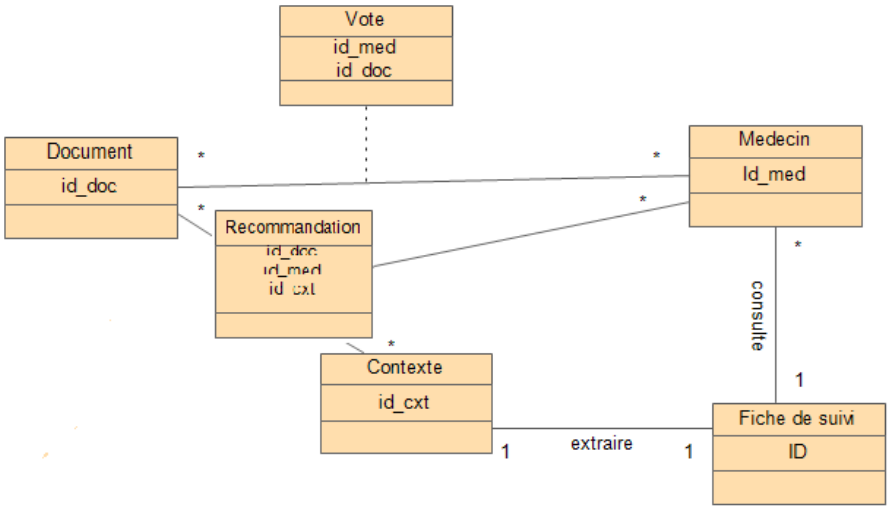


FigureIII.3 Digramme de séquence extraction du contexte et recommandation des documents

3.5 Diagramme de classe

La première démarche consiste à représenter le domaine de l'activité, sous forme d'un schéma conceptuel. Un schéma peut très bien représenter notre travail. Les objets de ce schéma sont :

Médecin, fiche de suivi, document, contexte, vote, recommandation



FigureIII.4 Diagramme de classe

3.6 Indexation et appariement dans la recommandation par contenu

3.7 Processus d'indexation

L'indexation consiste à analyser chaque document de la collection afin de créer un ensemble de mots-clés. Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche. L'indexation permet ainsi de créer une représentation des documents dans le système. Son objectif est de trouver les concepts les plus importants du document (ou de la requête), qui formeront le descripteur du document. L'indexation peut être :

· **Manuelle :** chaque document est analysé par un spécialiste du domaine ou par un documentaliste, elle permet d'assurer une meilleure pertinence dans les réponses apportées par le SRI, mais le temps nécessaire à sa réalisation est très important.

· **Automatique** : le processus d'indexation est entièrement informatisé, elle regroupe un ensemble de traitements automatisés sur un document. On distingue : l'extraction automatique des mots des documents, l'élimination des mots vides, la lemmatisation (radicalisation ou normalisation), le repérage de groupes de mots, la pondération des mots et enfin la création de l'index.

· **Semi-automatique** : le choix final revient au spécialiste ou au documentaliste, qui intervient souvent pour choisir d'autres termes significatifs. Les indexeurs utilisent un thésaurus ou une base terminologique, qui est une liste organisée de descripteurs (mots clés) obéissant à des règles terminologiques propres et reliés entre eux par des relations sémantiques. Le choix et l'intérêt d'une méthode par rapport aux autres dépend d'un certain nombre de paramètres, dont le plus déterminant est le volume des collections.

3.7.1 Analyse lexicale

L'analyse lexicale est le processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est une unité lexicale ou un radical. L'analyse lexicale permet de reconnaître les espaces de séparation des mots, des chiffres, les ponctuations, etc.

3.7.2 Pondération des termes

La pondération des termes permet de mesurer l'importance d'un terme dans un document. Cette importance est souvent calculée à partir de considérations et

interprétations statistiques (ou parfois linguistiques). L'objectif est de trouver les termes qui représentent le mieux le contenu d'un document.

Si on dresse une liste de l'ensemble des mots différents d'un texte quelconque classés par ordre de fréquences décroissantes, on constate que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste. Cette constatation est énoncée formellement par la loi de Zipf: rang*fréquence = constante.

La relation entre la fréquence et le rang des termes permet de sélectionner les termes représentatifs d'un document : on élimine respectivement les termes de fréquences très élevées car ils ne sont pas représentatifs du document (on peut par exemple citer les mots outils), et les termes de fréquences très faibles (ce qui permet d'éliminer les fautes de frappes).

La plupart des techniques de pondérations sont basées sur les facteurs *tf* et *idf*, qui permettent de combiner les pondérations locale et globale d'un terme :

tf (Term Frequency) : cette mesure est proportionnelle à la fréquence du terme dans le document (pondération locale). Elle peut être utilisée telle quelle ou selon plusieurs déclinaisons (*log (tf), présence/absence,. . .*).

· ***idf (Inverse of Document Frequency):*** Ce facteur mesure l'importance d'un terme dans toute la collection (pondération globale). Un terme qui apparaît souvent dans la base documentaire ne doit pas avoir le même impact qu'un terme moins fréquent. Il est généralement exprimé comme suit :

idf=log (N/df), où :

· ***df*** est le nombre de documents contenant le terme et

· ***N*** est le nombre total de documents de la base documentaire.

· ***La mesure tf ∗ idf:*** En combinant les deux techniques précédentes, elle donne une bonne approximation de l'importance du terme dans le document, particulièrement dans les corpus de documents de taille homogène. Cependant, elle ne tient pas compte d'un aspect important du document: sa longueur. En général, les documents les plus longs ont tendance à utiliser les mêmes termes de façon répétée, ou à utiliser plus de termes pour décrire un sujet. Par conséquent, les fréquences des termes dans les documents seront plus élevées, et les similarités à la requête seront également plus grandes. Pour pallier cet inconvénient, il est possible d'intégrer la taille des documents à la formule de pondération : on parle de facteur de normalisation.

3.7.3 Création d'index

Afin de répondre plus rapidement à une requête, des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les suivants : les fichiers inverses (”inverted files”), les tableaux de suffixes (”suffix arrays”) et les fichiers de signatures (”signature files”).

Les fichiers inverses sont actuellement le meilleur choix possible pour la plupart des applications. Les fichiers inverses sont composés de deux éléments principaux:

- Le vocabulaire, qui est l'ensemble de tous les mots différents du texte ;
- Les occurrences (posting) : pour chaque mot, il s'agit de la liste de toutes les positions dans le texte pour lesquelles le mot apparaît.

La figure III.5 montre un exemple de vocabulaire et d'occurrences.

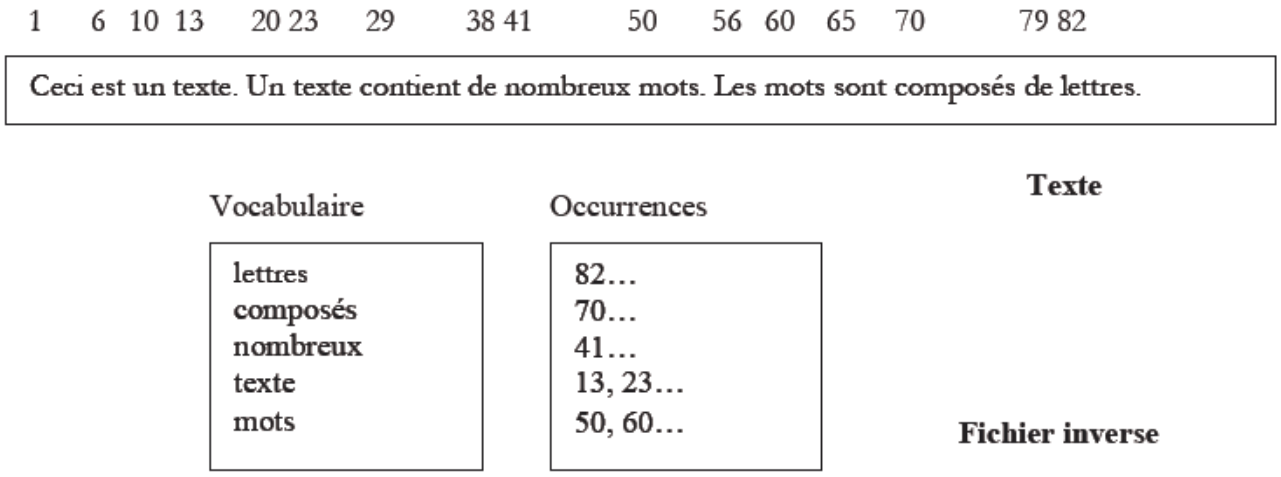


Figure III.5 Un texte simple et le fichier inverse correspondant

3.8 Appariement : Document-requête

La comparaison entre le document et la requête revient à calculer un score, supposé représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction ou d'une probabilité de similarité notée RSV (Q,d) (*Retrieval Status Value*), où Q est une requête et d'un document.

Cette mesure tient compte du poids des termes dans les documents, déterminé en fonction d'analyses statistiques et probabilistes.

La fonction d'appariement est très étroitement liée aux opérations d'indexation et de pondération des termes de la requête et des documents du corpus. D'une façon générale,

l'appariement document-requête et le modèle d'indexation permettent de caractériser et d'identifier un modèle de recherche d'information.

La fonction de similarité permet ensuite d'ordonner les documents renvoyés à l'utilisateur. La qualité de cet ordonnancement est primordiale. Dans notre cas la requête c'est le contexte et la mesure de similarité utilisé est le cosinus

3.9 Recommandation collaborative

Notre système donne la possibilité aux médecins pour voter sur les documents consultés puis il calcule la moyenne des votes et recommande les documents aux autres médecins par rapport au taux de votes et la moyenne puisque tous les médecins sont cardiologues alors on ne calcule pas la mesure de similarité entre eux.

3.10 Implémentation et mise en œuvre

Nous commençons à faire le modèle relationnel à partir de la conception qu'on a faite, alors nous avons obtenu les tables suivantes dans notre base de données :

Medecin (**id_medecin**, username, nom, prénom, password, specialité, addresse) ;

FDS (**id_patient**, nom, prénom, sexe, age, maladie, observation, traitement, examen, id_medecin*) ;

Contexte (**id_contexte**, maladie, sexe, age) ;

Document (**id_document**, titre, auteur, keywords, volume, journal, link, page, issu, date, abstract) ;

Vote (**id_medecin*, id_document*,** note);

Recommandation (**id_medecin*, id_document*, id_contexte***) ;

Remarque : pour la notation, nous avons choisi de mettre en gras les clés primaires et de mettre (*) à la fin de chaque clé étrangère.

3.11 Outils de réalisation

Après avoir préparé notre base de données nous allons voir les outils et les langages utilisés dans la construction de l'application :

PHP: Le PHP est un langage de scripts (donc interprété) s'exécutant sur un serveur. Cela signifie qu'il est exécuté par le serveur avant d'apparaitre en tant que document HTML. Il est donc fondamentalement différent d'un autre langage de script comme le

JavaScript, car pour ce dernier, c'est la machine client (l'ordinateur de l'internaute) qui exécute les instructions JavaScript.
Parmi les nombreux avantages de ce langage, il en est un qui revêt un caractère tout particulier: ses possibilités de dialogues avec différents SGBD (Système de Gestion de Base de Données) dont le très connu MySQL.
Pour faire du PHP, il vous faut un interpréteur PHP (gratuit!), un serveur web Apache, gratuit lui aussi et un éditeur de texte quelconque comme le bloc note, NotePad2 ou Pspad ou enfin le tout-en-un Coda pour Mac OS X (sinon, avec l'éditeurs Mac OS textEdit choisissez l'option Format / Convertir au format Texte pour saisir du code). A cela vous serez rapidement amené à utiliser une base de données sur le serveur de base de données MySQL (encore gratuit!) pour stocker vos informations.
Il existe des packages gratuits avec l'ensemble des composants (Apache - MySQL - PHP) ainsi que divers utilitaires comme PHPMyAdmin. Parmis les différents packages existants, je vous conseille EasyPhp (Apache/MySQL/Php) ou Wamp (pour Window/Apache/MySQL/Php) ou encore XAMPP (Apache/MySQL/Php/Perl) et enfin MAMP (Mac/Apache/MySQL/Php) pour Mac OS X. [21]

Wampserver : est une plate-forme de développement Web sous Windows pour des applications Web dynamiques (SPIP, OwnCloud, WordPress…) à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données. Il offre la possibilité d'installer pratiquement toutes les versions de Apache, PHP et MySQL existantes, de quoi reproduire fidèlement la configuration de votre serveur de production. [22]

HTML 5 : (HyperText Markup Language 5) est la dernière révision majeure d'HTML (format de données conçu pour représenter les pages web). Cette version est en développement en 2013. HTML5 spécifie deux syntaxes d'un modèle abstrait défini en termes de DOM : HTML5 et XHTML5. Le langage comprend également une couche application avec de nombreuses API, ainsi qu'un algorithme afin de pouvoir traiter les documents à la syntaxe non conforme.
Le HTML5 introduit de nouvelles balises et attributs, et en a rendu certains obsolètes. On peut citer, entre autres :

• de nouvelles balises pour mieux structurer la page, comme <header> et <footer>.

• de nouvelles balises multimédia : <audio> et <video>, pour faciliter l'intégration de sons et de vidéos. [23]

CSS3 : Littéralement *Cascading Style Sheets* (feuilles de style cascade), CSS est un langage déclaratif simple pour mettre en forme des pages HTML ou des documents XML. Le langage CSS permet de préciser les caractéristiques visuelles et sonores de présentation d'une page Web : les polices de caractères, les marges et bordures, les couleurs, le positionnement des différents éléments, etc. Le nouveau format de feuilles de style(CSS3) donne un contrôle beaucoup plus fin et poussé sur la typographie, permettant de créer des présentations de texte riches et sophistiquées avec la gestion automatique des colonnes. La gestion de halos et des ombres permet de créer des effets sur les éléments en un clin d'œil alors qu'il fallait auparavant employer des mosaïques d'images complexes à mettre en œuvre et lourdes à afficher sur le navigateur de l'utilisateur.

Les textes peuvent désormais être animés et positionnés en 3D ce qui ouvre un nouveau panel de possibilités pour la réalisation de boutons originaux. [24]

3.12 Réalisation de l'application

La page d'accueil

C'est la première page de notre site elle permet aux médecins de faire l'inscription et l'authentification.

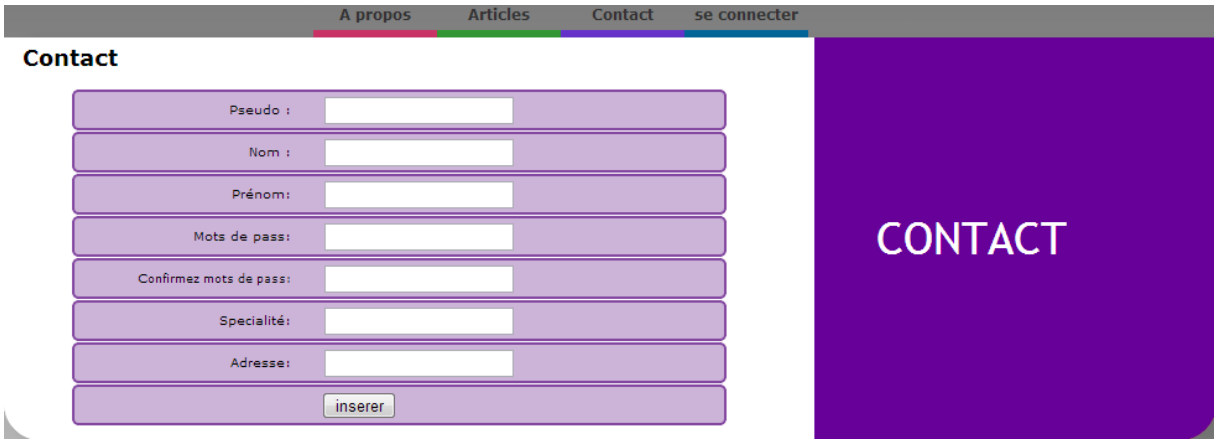


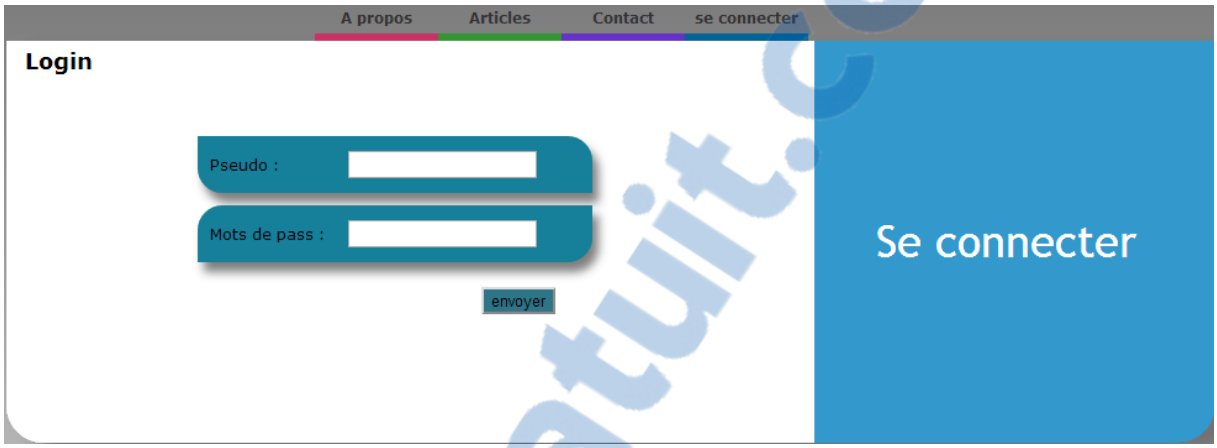
Figure III.6 l'accueil et l'inscription

Inscription et connexion:

Pour utiliser notre application, la première étape consiste en création d'un compte. Le médecin est demandé de saisir quelques informations personnelles comme le nom, prénom,

spécialité, mot de passe ainsi que le pseudonyme. Toutes ces informations vont générer son profil

Puis il peut se connecter via son pseudonyme et son mot de passe



FigureIII.7 la connexion

Fiche de suivi :

Une fois connecté le médecin peut entrer de nouvelles fiches de suivis de ces patients.

FigureIII.8 entrer une nouvelle fiche de suivie

Recherche :

Le médecin a le droit de chercher des patients par nom ou prénom, il trouve aussi la liste de tous les patients qui existent.

FigureIII.9 chercher un patient

Affichage et évaluation :

Avant de commencer l'évaluation nous avons collecté les informations de 106 documents de la requête « cardiology » dans la base documentaire PUBMED voir figure**III.10**. Le résultat est stocké dans un fichier Excel puis importé dans notre base de données, ce fichier nous l'avons remplie par les informations (titre, auteur, lien, résumé, mots clés…) des documents à partir du site http://www.ncbi.nlm.nih.gov/pubmed/

FigureIII.10 le site de Pubmed

L'application affiche au médecin la fiche de suivi du patient recherché, une fois cette page exécutée notre système va extraire le contexte d'utilisation de notre application de gestion des enregistrements électronique des patients et compare ces informations (maladie, sexe, âge) avec les documents de la base de données et affiche les documents pertinents, une fois le médecin click sur le lien du document, il sera dirigé directement vers le site qui contient ces documents ensuite il a le droit de voter sur ce document une seule fois.

ID	NOM	PRENOM	SEXE	AGE	MALADIE	OBSERVATION	TRAITEMENT
11	rah ali	youssef	homme	22	CVD		heptanol

Titre	Lien	Vote
Initiation and maintenance of cardiovascular medications following cardiovascular risk assessment in a large primary care cohort: PREDICT CVD-16.	Lien	
CVD risk among men participating in the National Health and Nutrition Examination Survey (NHANES) from 2001 to 2010: differences by sexual minority status.	Lien	
Performance of Framingham cardiovascular disease (CVD) predictions in the Rotterdam Study taking into account competing risks and disentangling CVD into coronary heart disease (CHD) and stroke.	Lien	
Effects of cerebrovascular disease and amyloid beta burden on cognition in subjects with subcortical vascular cognitive impairment	Lien	
Circulating and dietary omega-3 and omega-6 polyunsaturated fatty acids and incidence of CVD in the Multi-Ethnic Study of Atherosclerosis.	Lien	
Heptanol decreases the incidence of ischemia-induced ventricular arrhythmias through altering electrophysiological properties and connexin 43 in rat hearts.	Lien	

FigureIII.11 recommandations des documents et le vote

3.13 Conclusion

Au terme de ce chapitre, nous venons de conclure la dernière phase du projet, à savoir l'implémentation de notre système de recommandation hybride.

Cette implémentation offre, via un ensemble de programme et de bases de données la possibilité de trouver les documents pertinents et proche du contexte

4 Conclusion et perspectives

Conclusion générale

Ce projet de fin d'étude représente le fruit de tant d'années d'études, il nous a permis d'avoir de nouvelles connaissances dans différents domaines.
Dans ce mémoire nous avons abordé le problème de la recommandation contextuel des documents médicaux exactement dans le domaine de la cardiologie.

Comme nous avons vu l'objectif principal de notre système est de recommandé des documents scientifiques approprier au contexte d'utilisation d'une application de gestion des enregistrements électronique des patients. Cette recommandation est effectuée par la comparaison du contexte avec les documents du corpus via l'indexation et la mesure de similarité pour trouver les documents proches du contexte, on prend en compte aussi les avis et les votes des autres médecins de la même spécialité.

En réalisant ce PFE, nous avons essayé de développer une application médicale qui permet le filtrage collaboratif en se basant sur le taux des votes et le filtrage par contenu concrétiser par la mesure de similarité. L'application permet aussi de donner un avis sur un document consulter.

Perspective

Néanmoins, tout projet de grande envergure, nécessite des efforts considérables et des améliorations continues, à cette occasion on note l'un des champs d'études dans le domaine qui restent à explorer :

L'usage des concepts du web sémantique dans le filtrage d'information par contenu après l'extraction du contexte d'utilisation d'une application de gestion des enregistrements électronique des patients pour élargir notre contexte et trouver plus de documents pertinents

Tout ça en utilisant le WORDNET qui offre un élargissement de la sémantique du contexte par les synonymes des mots ainsi le degré d'analogie entre les mots.

Aussi utiliser des solutions distribuées plus performantes pour optimiser le temps d'exécution de l'application pour élargir la base de données qui contient beaucoup de documents. Et enfin utiliser la bibliothèque JCAF pour l'extraction du contexte à partir du texte des annotations des médecins.

5 Bibliographie

[1] [Vers des systèmes de recommandation robustes pour la navigation Web : inspiration de la modélisation statistique du langage

http://tel.archives-ouvertes.fr/docs/00/58/13/31/PDF/these.pdf

[2] MODÈLE MULTI-AGENTS POUR LE FILTRAGE COLLABORATIF DE L'INFORMATION
http://www.archipel.uqam.ca/2670/1/D1888.pdf

[3] Les systemes de recommandation
http://www.irit.fr/recherches/ADRIA/Documents/Fargier/documentsBR4CP/E_Negre_Lens_juin_12.pps

[4] Bill N. Schilit and Marvin M. Theimer. Disseminating active map information to mobile hosts.**Network, IEEE**, 8(5):22-32, Sep/Oct 1994.

[5] N. S. Ryan and J. Pascoe and D. R. Morse. Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant. In V. Gaffney and M. van Leusen and S. Exxon, editor, **Computer Applications in Archaeology 1997**, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum

[6] Anind K. Dey and Gregory D. Abowd and Daniel Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. **Human-Computer Interaction**, 16(2):97-166, 2001.

[7] Bill Schilit and Norman Adams and Roy Want. Context-aware computing applications. In **In Proceedings of the Workshop on Mobile Computing Systems and Applications**, pages 85-90. IEEE Computer Society, 1994.

[8] Paul Dourish. What we talk about when we talk about context. **Personal Ubiquitous Comput.**, 8(1):19-30, 2004.

[9] Suchman,, Lucy A. **Plans and situated actions: the problem of human-machine communication**. Cambridge University Press, New York, NY, USA, 1987.

[10] Mark Weiser. Some computer science issues in ubiquitous computing. **Commun. ACM**, 36(7):75-84, July 1993.

[11] Gregory D. Abowd and Elizabeth D. Mynatt and Tom Rodden. The Human Experience. **IEEE Pervasive Computing**, 1(1):48-57, 2002.

[12] John Canny. The Future of Human-Computer Interaction.

ACM Queue, 4(6):25-32, July/August 2006.

[13] James L. Crowley and Joëlle Coutaz and Gaëtan Rey and Patrick Reignier. Perceptual Components for Context Aware Computing.
In **UBICOMP 2002, International Conference on Ubiquitous Computing**, pages 117-134, Goteborg, Sweden, sep 2002. Springer Verlag.

[14] Joëlle Coutaz and James L. Crowley and Simon Dobson and David Garlan. Context is key. **Commun. ACM**, 48(3):49-53, March 2005.

[15] James L. Crowley. Social Perception. **ACM Queue**, 4(6):35-43, July/August 2006.

[16] James L. Crowley. Social Perception. **ACM Queue**, 4(6):35-43, July/August 2006.

[17] Gregory D. Abowd and Elizabeth D. Mynatt and Tom Rodden.
The Human Experience. **IEEE Pervasive Computing**, 1(1):48-57, 2002.

[18] Suchman,, Lucy A. **Plans and situated actions: the problem of human-machine communication**. Cambridge University Press, New York, NY, USA, 1987.

[19] James L. Crowley and Joëlle Coutaz and Gaëtan Rey and Patrick Reignier. Perceptual Components for Context Aware Computing.
In **UBICOMP 2002, International Conference on Ubiquitous Computing**, pages 117-134, Goteborg, Sweden, sep 2002. Springer Verlag.

[20] James L. Crowley and Joëlle Coutaz and Gaëtan Rey and Patrick Reignier. Perceptual Components for Context Aware Computing.
In **UBICOMP 2002, International Conference on Ubiquitous Computing**, pages 117-134, Goteborg, Sweden, sep 2002. Springer Verlag.

[21] Context-Aware Computing Anind K. Dey

[22] www.easy-micro.org/definition-du-php consulté le 10/5/ 2014

[23]www.wampserver.com consulté le 10/5/2014

[24] www.definitions-webmarketing.com/definition-du.php consulté le 10/5/ 2014

[25]www.jsand.net/definition-css.wju consulté le 10/5/ 2014

Tables des figures

Résume

Les informations dans le domaine médical se multiplient de jour en jour. Il est devenu indispensable de les stocker et les partager sur internet mais cela a causé d'autres

problèmes. Il est devenu difficile de trouver des informations vu la quantité qui n'arrête pas de progresser. Dans le cadre de cette thèse de master, nous nous intéressons à élaborer un système de recommandation contextuelle en se basant sur le filtrage hybride. Notre objectif est de recommander des documents jugé pertinent à des médecins par rapport au contexte d'utilisation d'une application de gestion des enregistrements électronique des patients. Le principe est d'extraire le contexte de cette utilisation: maladie, âge, sexe…et chercher dans le contenu des documents et prendre en compte le taux de vote des documents.

Mots clés : recommandation contextuelle, contexte, document

Abstract

The amount of information in the medical field is growing day by day. It became necessary to store and share it on the internet, but this caused other problems. It has become difficult to find information given its quantity which does not stop to progress. As a part of this master thesis, we focus on developing a system of contextual recommendation based on the hybrid filter. Our goal is to recommend documents deemed relevant to doctors regarding the context of using a management application for electronic patient records. The principle is to extract the context of this usage: illness, Age ... and search in the contents of documents and taking into account the rate of vote documents.

Key words: contextual recommendation, context, document

ملخص

تتزايد المعلومات في المجال الطبي يوما بعد يوم، لهذا أصبح من الضروري تخزين وتبادل المعلومات على شبكة الانترنت ولكنها تسبب مشاكل أخرى، فقد أصبح من الصعب العثور على المعلومات في الأنترنت التي لا تتوقف عن الزيادة، كجزء من أطروحة هذه الماستر، نحن نركز على تطوير نظام للتوصية السياقية. هدفنا هو أن نوصي الوثائق التي تعتبر ذات الصلة إلى الأطباء فيما يتعلق بسياق استخدام تطبيق إدارة سجلات المرضى الالكترونية، والمبدأ هو استخراج هذا السياق: المرض، العمر، الجنس ... والبحث في محتويات الوثائق والأخذ بعين الاعتبار معدل التصويت على الوثائق.

الكلمات المفتاحية التوصية السياقية ، الوثائق السياق،: