Modélisation et analyse du comportement des utilisateurs exploitant des données vidéo

Sylvain Mongy

Modélisation et analyse du comportement des utilisateurs exploitant des données vidéo.

THÈSE

présentée et soutenue publiquement le 25 Novembre 2008

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille
(spécialité informatique)

par

Sylvain MONGY

Composition du jury

Président :	Sophie TISON, Directrice de Recherche	LIFL, Université de Lille I
Rapporteurs :	Patrick GROS, Directeur de Recherche	INRIA, IRISA
	Philippe JOLY, Maître de Conférence (HDR)	IRIT
Directeur :	Chabane DJERABA, Professeur	LIFL, Université de Lille I

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
Laboratoire d'Informatique Fondamentale de Lille — UPRESA 8022
U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. : +33 (0)3 28 77 85 41 – Télécopie : +33 (0)3 28 77 85 37 – email : direction@lifl.fr

Remerciements

Je tiens à exprimer mes remerciements aux membres du jury de soutenance : le président (nom et titre) pour me faire l'honneur de présider ce jury, les rapporteurs (Patrick Gros, Directeur de Recherche à l'INRIA, IRISA et Philippe Joly, Maître de Conférence (HDR) à l'IRIT) et les examinateurs (Brigitte Trousse, Maître de Conférence à l'INRIA-Sophia Antipolis) pour l'intérêt qu'ils portent à mon travail et pour avoir accepté ces tâches.

J'adresse également mes plus vifs remerciements à Chabane Djeraba (Professeur à l'Université des Sciences et Technologies de Lille) pour son encadrement, tant sur le plan pratique que scientifique et pour m'avoir aidé à comprendre ce qu'est un enseignant-chercheur.

Je tiens aussi à remercier chaleureusement Dan Simovici (Professeur à l'Université de Boston) pour l'ensemble de ses remarques et ses conseils. Ce fut un réel plaisir de travailler à ses côtés au cours de ces deux dernières années.

Outre les personnes mentionnées précédemment je remercie l'ensemble des personnels de l'Université de Lille 1, qui ont fait de ma thèse une expérience agréable et enrichissante, mes collègues de l'équipe FOX-MIIRE, notamment Fatma Bouali (Maître de Conférences à l'Université de Lille 2), avec lesquels j'ai travaillé dans de très bonnes conditions.

Enfin, je remercie tout particulièrement Thierry pour l'ensemble de son oeuvre... et Delphine pour tout le reste.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

1.1 Émergence des comportements vidéo

Le support vidéo est devenu depuis quelques années un vecteur de communication majeur tant au niveau commercial qu'au niveau de la diffusion d'information. Parallèlement aux progrès dans le transfert de données, la vidéo est devenue un support important pour partager l'information. Depuis 2006, les sites de partage de vidéo en ligne connaissent une explosion de leur fréquentation. Les figures (1.1) et (1.2) issues de [Ale] présentent notamment l'évolution du nombre de connexions sur deux acteurs de ce domaine : *YouTube* et *DailyMotion*. On remarque dans les deux cas une progression linéaire du pourcentage de connexions sur ces sites par rapport aux connexions sur l'ensemble du web. Le pourcentage d'accès à un site correspond au nombre d'utilisateurs qui ont visité le site sur une période donnée (une semaine) par rapport au nombre total d'utilisateurs du web.

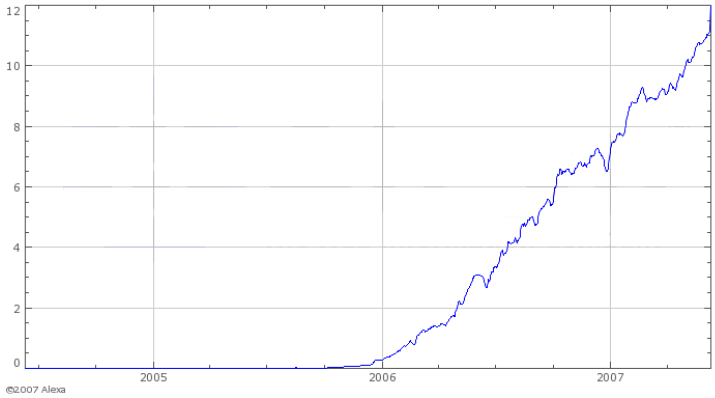


FIG. 1.1 – Pourcentage d'accès au site YouTube par rapport aux accès web [Ale]

Cet usage de plus en plus généralisé de la vidéo ouvre de nouvelles perspectives dans la diffusion de l'information et la compréhension de son usage. L'un des challenges les plus prometteurs est la compréhension des interactions entre les utilisateurs et la vidéo. Comme cela a été fait pour le web, comprendre le comportement des utilisateurs est crucial pour optimiser les outils et rendre l'information plus accessible. Par exemple,

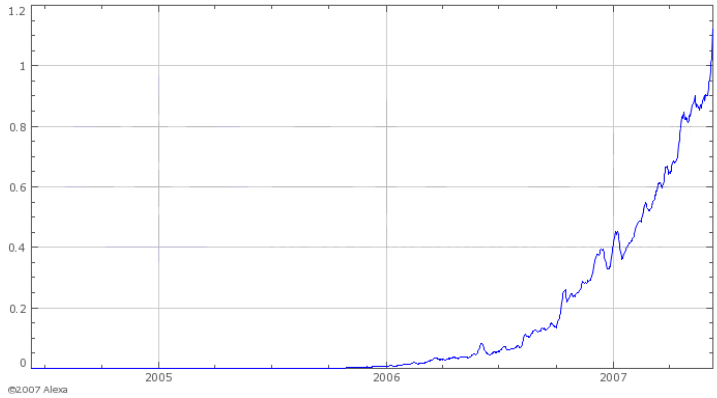


FIG. 1.2 – Pourcentage d'accès au site DailyMotion par rapport aux accès web [Ale]

[Neta] est un défi lancé par *NetFlix* [Netb] destiné à prédire les vidéos qui vont intéresser les utilisateurs en fonction de leur comportement passé. Actuellement, *NetFlix*, l'un des plus grand fournisseurs mondiaux de vidéos, utilise un système de recommandations *Cinematch*. Son travail est de déterminer si les utilisateurs vont apprécier un film donné en fonction de leur degré de satisfaction sur le visionnage d'autres films. Le principe est donc d'offrir aux utilisateurs un système de recommandations basé sur leurs goûts personnels.

Ces comportements des utilisateurs exploitant des données vidéo (que nous noterons comportements vidéo par la suite) sont nouveaux et plus complexes à analyser que d'autres données (texte, image) de par la nature dynamique de la vidéo. Ils nécessitent de nouveaux outils d'analyse qui faciliteront leur compréhension et amèneront à la production de vidéos plus accessibles, dans lesquelles l'utilisateur trouvera plus facilement l'information qu'il recherche.

1.2 Positionnement et originalité

Nous proposons dans notre travail d'analyser le comportement des utilisateurs d'un moteur de recherche vidéo pour améliorer la qualité de l'indexation textuelle. Notre objectif est de comprendre pourquoi et comment chacune des séquences vidéo est visionnée. Par exemple, les utilisateurs recherchant des vidéos concernant le mot-clé « montagne » visionnent successivement les vidéos (18, 73, 29) qui sont retournées dans cet ordre par le moteur de recherche. Si l'on note que dans la majeure partie des cas, la vidéo 29 est visionnée totalement alors que les vidéos 18 et 73 ne le sont que partiellement, on en déduit que, selon l'utilisateur, le concept de « montagne » est mieux exprimé par la vidéo 29 que par les vidéos 18 et 73. En conclusion, la vidéo 29 doit être proposée en premier aux utilisateurs lors des futures recherches sur le concept « montagne ». Le poids de ce mot-clé pour la vidéo 29 s'en trouve augmenté et réduit pour les vidéos 18 et 73.

Pour cela, nous présentons une approche qui combine usage intra-vidéo et usage inter-vidéo pour générer des profils de visite sur un moteur de recherche vidéo (figure 1.3). Nous proposons une nouvelle approche qui définit différents types de données de *log* collectées par le moteur de recherche. Le premier concerne la manière dont un utilisateur visionne les séquences vidéo (lecture, pause, avance rapide, stop...). À ce niveau que nous appelons *intra-vidéo*, nous définissons le *visionnage d'une vidéo* comme unité de comportement. L'autre type de *log* trace les transitions entre les différents visionnages. Cette partie regroupe l'écriture des requêtes, la présentation des résultats et les visionnages successifs des séquences vidéo. À ce niveau plus général que nous appelons *inter-vidéo*, nous introduisons la *session* comme unité de comportement.

Un comportement intra-vidéo est modélisé par un modèle de Markov du premier ordre non caché. Ce modèle est construit en utilisant les différentes actions proposées aux utilisateurs lors des visionnages (lecture, pause, avance rapide, retour rapide, saut, stop). Nous proposons une techniques de regroupement de ces modèles (*K-models*). Cette méthode est une adaptation de la technique classique des *K-moyennes* [Mac66] adaptée à l'utilisation de modèles en lieu et place des moyennes. Nous caractérisons ainsi plusieurs comportements type (lecture totale, lecture partielle, survol...). Grâce à ces comporte-

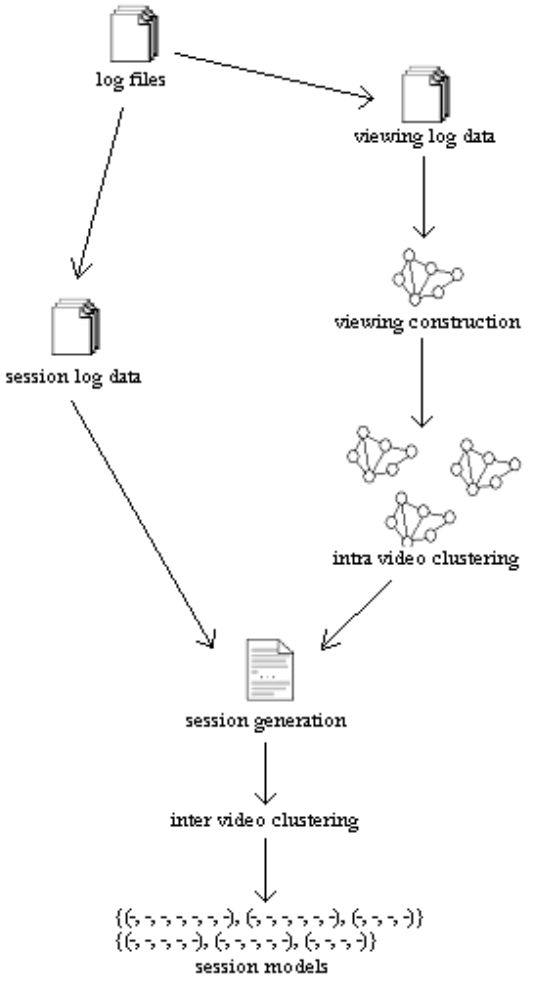


FIG. 1.3 – Description globale du processus de fouille de données vidéo.

ments type, nous sommes à même de savoir quelle fut l'utilité ou l'importance d'une séquence vidéo lors d'une visite (si elle est la plus importante ou seulement le résultat

d'une recherche infructueuse...).

Un comportement inter-vidéo est modélisé par une session. Cette session est une séquence ordonnée des visionnages des séquences vidéo. Chaque visionnage est représenté par l'identifiant de la vidéo et le modèle de comportement qui lui correspond pour cette visite. Pour regrouper ces sessions, nous nous sommes basés sur une technique de regroupement hiérarchique ascendante que nous avons adaptée à ces données particulières. Cette dernière présente la particularité de traiter des classes représentées par plusieurs sous-séquences communes. La majeure partie des travaux sur la modélisation de sessions ne travaille qu'avec une seule sous-séquence représentative. Cela induit une importante perte d'informativité lorsque deux sessions ont plusieurs sous-séquences communes de même longueur et qu'il faut en choisir une et de fait perdre l'information portée par les autres. Notre approche permet de pallier cette limite.

1.3 Structure de la thèse

Après cette introduction, nous discutons au chapitre (2) des travaux se rapprochant le plus de notre champ de recherche dont le but est la compréhension du comportement des utilisateurs exploitant des données vidéo. Nous nous penchons tout d'abord sur la nouveauté de tels comportements et sur les travaux existants dans ce domaine ou en relation proche avec ce domaine. Ensuite, nous présentons un aperçu des travaux dans le domaine du *Web usage mining*. Ce domaine est connexe à celui de notre problématique par l'analyse du comportement et par les techniques employées pour l'analyse de données de type séquentiel, la différence résidant dans la nature des données.

Au chapitre (3), nous détaillons les concepts et les notions théoriques qui seront mis en pratique dans les chapitres suivants. Nous présentons les concepts fondamentaux de la théorie des modèles de Markov, en particulier ceux liés aux modèles dits absorbants. Nous présentons ensuite les techniques d'analyse de séquences, notamment la technique d'extraction de la plus longue sous-séquence commune en programmation dynamique.

Dans le chapitre (4), nous présentons le modèle utilisateur que nous avons défini pour représenter les comportements vidéo. Nous détaillons les deux niveaux de modélisation que nous avons définis dans notre approche.

Les chapitres (5) et (6) montrent la mise en oeuvre notre modèle pour extraire d'un ensemble de données les comportements type des utilisateurs vidéo. Le chapitre (7) positionne notre travail dans un cadre applicatif précis : l'optimisation d'un moteur de recherche vidéo par retour d'expérience.

Enfin, nous concluons au chapitre(8) au travers de remarques et d'impressions globales sur la qualité des résultats obtenus et présentons les directions de recherches futures pouvant découler de notre travail.

Bibliographie

[Ale] Alexa : http ://www.alexa.com/.

[Mac66] J. MacQueen : Some methods for classification and analysis of multivariate observations. *Fifth Berkeley symposium. University of California Press 1*, pages 281–297, 1966.

[Neta] Netflix : http ://www.netflixprize.com/.

[Netb] Netflix : http ://www.netflix.com/.

Chapitre 2

État de l'art

Sommaire

2.1 L'Analyse des comportements vidéo

La recherche dans le domaine de l'optimisation des outils de recherche et de visualisation de données vidéo est un domaine de recherche majeur des années à venir. En effet, l'utilisation du format vidéo numérique est de plus en plus courante et le besoin d'avoir des outils performants pour exploiter les vidéos important. Actuellement, les différentes approches essaient en majorité d'améliorer ces outils de vidéo à la demande (noté VOD) en se basant sur le contenu des vidéos et leur indexation. Ces outils, bien que performants, pourraient encore être améliorés en prenant en compte le comportement des utilisateurs. Quelques approches en relation avec ce sujet précis ont été abordées dans la littérature mais elles ne se placent pas dans le cadre d'une analyse globale des comportements dans le but d'améliorer les outils de recherche de vidéos.

2.1.1 Vidéo à la demande

Actuellement, il existe déjà de nombreux systèmes de vidéo à la demande et de moteurs de recherche dédiés à la vidéo. Ceux-ci essaient de prendre en compte les problèmes classiques liés aux moteurs de recherche en y incluant la spécificité liée à la nature des données vidéo. L'objectif est d'optimiser au maximum les recherches des utilisateurs en leur délivrant ce qu'ils recherchent en un minimum de temps. Différents travaux étudient ce domaine et proposent des outils de recherche et de visualisation [LGS+00, SPAP99, HNK+04, LGD+05]. La limite de ces techniques se situe au niveau de la non-prise en compte des interactions de l'utilisateur. En effet, comme cela a été fait dans d'autres domaines (comme la fouille des usages web), enrichir les techniques de recherche à l'aide du comportement passé des utilisateurs permet d'offrir des résultats plus pertinents aux utilisateurs.

Un premier travail cherchant à mettre l'utilisateur au centre du processus de recherche de documents vidéo est proposé dans [FS97]. Les auteurs y présentent un moteur de recherche vidéo utilisant deux approches en parallèle. La première approche, qu'ils définissent comme approche « pull » consiste à répondre aux requêtes explicitement données par les utilisateurs. Ces requêtes sont comparées aux annotations de la base qui décrivent

les vidéos. La plus-value de ces données d'annotations est qu'elles sont directement écrites par les utilisateurs eux-mêmes. Elles sont partagées et on peut ainsi estimer qu'elles s'affinent au fil du temps pour correspondre de plus en plus aux vidéos qu'elles décrivent. Dans une seconde approche, appelée approche « push » l'objectif est de proposer aux utilisateurs des vidéos correspondant à un profil qui leur est attribué. Ce profil peut soit être défini par l'utilisateur lui-même soit estimé par le programme. Dans ce cas, un profil générique est extrait en fonction de l'expérience passée des utilisateurs et des vidéos qu'ils ont visionnées. Ces profils restent assez basiques : calme, romantique, violent mais sont l'une des premières tentatives de personnalisation d'un moteur de recherche en fonction du comportement des utilisateurs. En combinant ces deux approches, ils offrent ainsi à chaque utilisateur un moteur de recherche personnalisé en fonction de ses goûts.

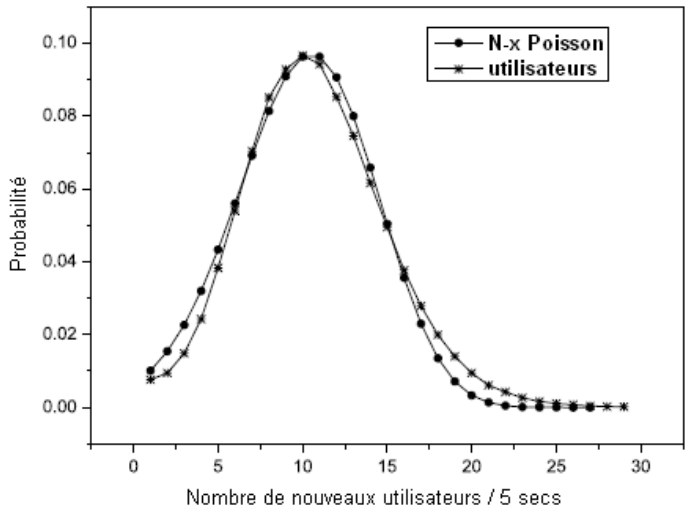


FIG. 2.1 – Comparaison des connexions réelles des utilisateurs à la distribution de Poisson modifiée [YZZZ06]

Dans [YZZZ06], les auteurs analysent les connexions à un site de VOD. Ils se basent sur des données *log* basiques comportant notamment les vidéos auxquelles les utilisateur

ont accédé et le temps passé par les utilisateurs à les visionner. L'objectif de l'article est de proposer des pistes pour améliorer la qualité des systèmes de VOD. Dans un premier temps, les auteurs montrent les fluctuations quotidiennes et hebdomadaires des connexions et les modélisent à l'aide d'une loi de Poisson modifiée (figure 2.1).

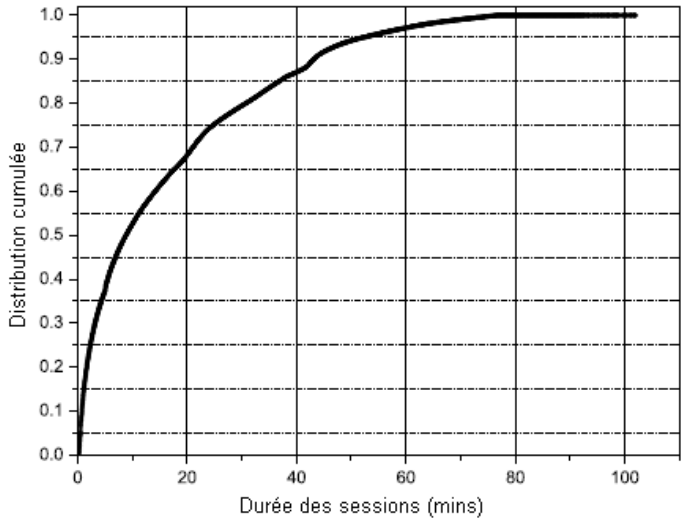


FIG. 2.2 – Distribution de la durée moyenne des sessions de visionnage avant fermeture [YZZZ06]

Ensuite, les auteurs cherchent à analyser la durée moyenne des sessions de visionnage. En comparant ces dernières, ils montrent que plus de la moitié des sessions se terminent en dix minutes ou moins, dont plus d'un tiers en moins de cinq minutes. Cela traduit un comportement « impatient » des utilisateurs qui survolent souvent le début d'une vidéo pour en déterminer l'intérêt (figure 2.2).

Enfin, les auteurs montrent un impact d'un système de recommandations basiques, proposant les vidéos les plus souvent visionnées. Celles-ci sont plus fréquemment visionnées que les autres mais également plus souvent survolées, proportionnellement aux vidéos moins fréquemment visionnées qui sont plus souvent visionnées en intégralité

(figures 2.3 et 2.4). Un tel système montre ici ses limites. En effet, les vidéos les plus populaires ne correspondent pas aux besoins des utilisateurs qui sont différents (accès aux vidéos moins fréquemment visionnées). Leur proposer en priorité les vidéos populaires ne les aidera donc pas à simplifier leurs recherches.

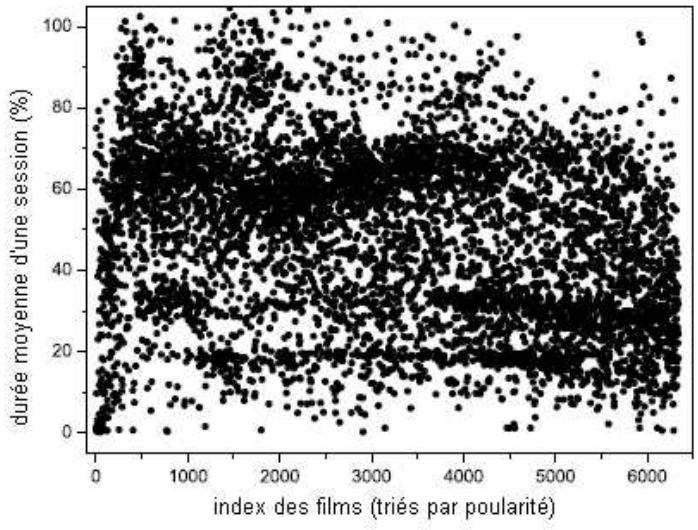


FIG. 2.3 – Durée des sessions de visionnage en fonction de la popularité des vidéos (de la moins à la plus populaire) [YZZZ06]

Cette étude montre l'importance et l'intérêt de proposer des outils d'accès aux données intelligents, prenant en compte le comportement des utilisateurs pour leur offrir une navigation plus agréable entre les vidéos.

Dans [NAMO00], les auteurs proposent un système d'enregistrement automatique de programmes vidéo se basant en partie sur le comportement passé des utilisateurs. Ils formulent ce problème de la manière suivante. Étant donné un ensemble de vidéos enregistrées sur le disque dur d'un utilisateur, un guide des programmes qui vont être diffusés, une fenêtre temporelle définie, et un espace mémoire limité pour le stockage des vidéos, l'agent cherche à déterminer quels programmes il doit enregistrer pour maximiser

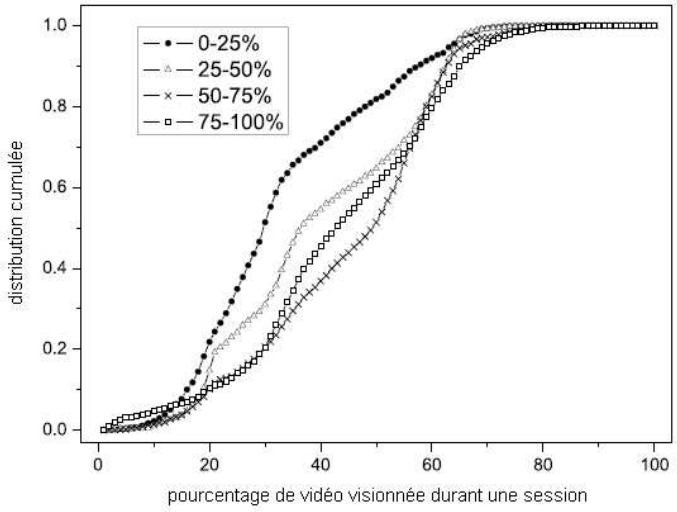


FIG. 2.4 – Distribution cumulée du pourcentage de vidéo visionnée des sessions en fonction de la popularité des vidéos (4 courbes représentant les 4 quarts des vidéos triées selon leur popularité) [YZZZ06]

le temps que passera l'utilisateur à visionner les vidéos enregistrées en ne dépassant pas l'espace mémoire déterminé. Pour cela, il prédit l'intérêt de l'utilisateur sur les différentes vidéos et utilise ensuite la programmation dynamique pour résoudre le problème NP-complet qu'est la maximisation du temps de visionnage sous la contrainte de l'espace limité (ce problème est identique au problème classique du rangement d'objets dans un sac à dos ou « Knapsack Problem »).

Cette prédiction est calculée à partir d'une description textuelle des programmes (mots-clés) et affinée en la comparant à la probabilité que d'autres utilisateurs, dont un même programme a été enregistré sur leur disque dur par le système, l'ont effectivement regardé. D'abord, avec une méthode d'estimation basée sur une variante de la méthode de prédiction bayésienne, l'agent attribue une valeur binaire à chaque mot-clé lié à une vidéo, correspondant à la chance qu'a l'utilisateur de la visionner. Il fait alors la moyenne sur

l'ensemble des mots-clés. Ensuite, il analyse le comportement des autres utilisateurs qui ont eu la même vidéo sur leur disque et qui l'ont regardée ou non. Il combine finalement ces deux probabilités pour obtenir la prédiction globale relative à une vidéo et à un utilisateur donnés. Les auteurs proposent également en perspective de considérer certains utilisateurs bien choisis comme leaders d'opinion au lieu de l'ensemble des utilisateurs pour générer les prédictions.

2.1.2 Résumés automatiques

Une équipe de recherche de *Microsoft* [YMNZ03] propose une approche centrée utilisateur pour la réalisation de résumés vidéo. Sa motivation vient du fait qu'aucune technique basée uniquement sur l'analyse de la vidéo ne permet dans un cas général de construire des résumés contenant l'ensemble des scènes importantes d'une vidéo. Elle propose tout d'abord de collecter les données de *logs* relatives aux interactions des utilisateurs (figure 2.6) à partir d'un outil de recherche vidéo. Celui-ci présente chaque vidéo découpée en plans automatiquement par un algorithme de détection des changements de plan basé sur l'analyse des attributs bas-niveau de la vidéo.

Sur la base des données de *logs* collectées, les auteurs proposent de pondérer l'importance de chacun des plans d'une vidéo. Pour cela ils introduisent la notion de *shotRank*. Ce *shotRank* est inspiré des techniques de la fouille de données web et calcule l'importance d'un plan en fonction des plans qui l'ont précédé lors des visionnages des utilisateurs. En effet, comme le montre la figure (2.5), il existe différents liens qui connectent les plans entre-eux : ils semblent similaires, ils montrent la même personne, ils ont un sujet commun... Le *shotRank* permet donc de calculer l'importance de l'ensemble de ces liens pour un plan donné, et de mesurer la probabilité qu'un utilisateur le visionne pendant sa navigation.

De manière formelle, soit un plan A et $B_1, ...B_n$ n plans connectés à A, la mesure de *shotRank* est donnée par l'équation (2.1). On voit que plus un plan est important, plus il aura d'impact sur l'importance d'un plan auquel il est connecté.

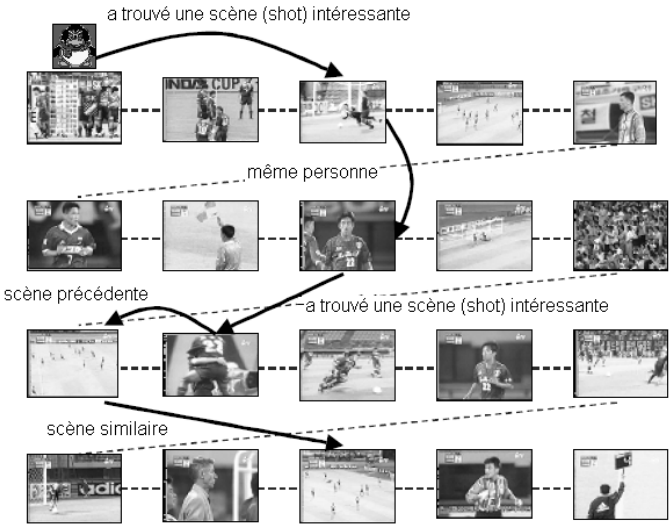


FIG. 2.5 – Interprétation d'un modèle de visionnage [YMNZ03]

$$ShotRank(A) = \frac{1}{n} \times \sum_{i=1}^{i=n} ShotRank_{Bi} \tag{2.1}$$

À partir de la valeur de *shotRank*, les auteurs sélectionnent les plans aux valeurs les plus élevées pour construire le résumé (figure 2.7). Le nombre de plans sélectionnés dépendra du choix de l'utilisateur, définissant le nombre de plans qu'il désire ou la durée totale du résumé. Afin d'éviter de découper arbitrairement une scène, ils analysent également l'enchaînement des plans pour reconstruire les scènes quand plusieurs plans se suivent et ont un même *shotRank* sans temps de pause entre deux plans.

Des expériences, menées sur un ensemble d'utilisateurs qui devaient répondre à un ensemble de questions relatives aux vidéos, ont confirmé que cette pondération des plans estime l'utilité et l'importance de chaque plan de vidéo, et améliore l'exploration des futurs résumés. En effet, ces expériences montrent une baisse intéressante du temps de

```
...
0  11     21002796 //SELECT/STOPPED at SHOT 11
0  18     21003390 //SELECT/STOPPED at SHOT 18
0  19     21003796 //SELECT/STOPPED at SHOT 19
1  19     21004210 //PLAY SHOT 19
0   7     21007312 //SELECT/STOPPED at SHOT 7
3   7     21007312 //JUMP FOR SIMILARITY FROM SHOT 7
0   9     21008484 //SELECT/STOPPED at SHOT 9
... ...
```

FIG. 2.6 – Exemple des *logs* utilisateurs [YMNZ03]

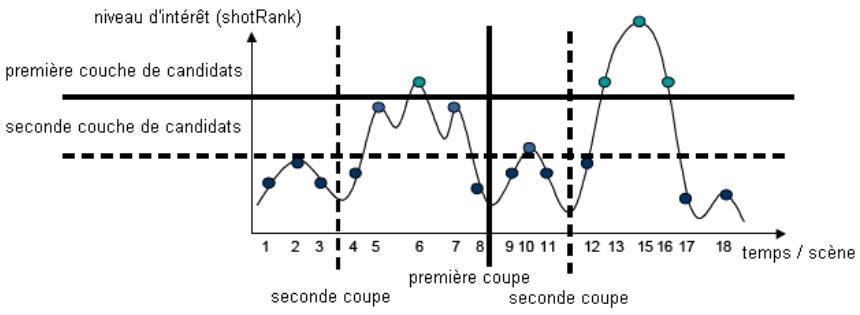


FIG. 2.7 – Sélection des plans de plus grande importance [YMNZ03]

recherche de l'information (40% environ), symbolisant une meilleure synthèse des vidéos dans les résumés.

Un autre projet au but similaire, réalisé par une équipe de recherche d'*IBM* [SMP01], utilise également les données de *log* pour déterminer le comportement des utilisateurs visionnant une vidéo et en extraire un résumé. Dans ce travail, les auteurs estiment nécessaire de comprendre le but du visionnage de chacune des parties d'une vidéo. En effet, un utilisateur peut avoir parcouru plusieurs plans avant de finalement trouver celui ou ceux qui l'intéressent particulièrement. Ce travail part du principe qu'il est possible

d'apprendre le comportement d'un utilisateur en observant uniquement ses interactions avec la vidéo. Par exemple une série d'actions de type [FFW → PLAY → PAUSE] peut être interprétée comme un comportement de recherche d'un plan particulier, sa découverte et sa lecture.

Pour représenter ces comportements, les auteurs proposent de construire un modèle de Markov caché (noté HMM). Dans ce modèle, chaque état représente un type de comportement (curieux, a trouvé quelque chose d'intéressant, recherche quelque chose...) et les observations sont les interactions de l'utilisateur avec la vidéo. En définissant par avance le nombre d'états, ils estiment les paramètres du modèle par apprentissage non-supervisé et entraînent ensuite le modèle avec l'algorithme de *Baum-Welch*. Les labels des types de comportements sont ensuite donnés à chaque état par une analyse détaillée.

Pour collecter les données relatives aux actions des utilisateurs, ils ont développé un lecteur vidéo appelé *MediaMiner* qui enregistre ces données sur un serveur. Ce lecteur propose aux utilisateurs les actions PLAY, PAUSE, SAUT, RALENTI, AVANCE RAPIDE, RETOUR RAPIDE. C'est à partir de ces données que les paramètres des modèles sont estimés. Pour générer des données intéressantes, les utilisateurs doivent répondre à un certain nombre de questions qui doivent les mener à un usage approfondi des vidéos. Ces questions sont par exemple : « Combien y-a-t-il de lignes de code dans le logiciel d'un avion ? » , incitant les utilisateurs à utiliser les différents moyens de navigation.

Ensuite, en partant du principe que plus un grand nombre d'utilisateurs a trouvé une partie d'une vidéo intéressante, plus elle est importante, les auteurs se basent sur l'ensemble des comportements, transformés en HMM, pour extraire les parties les plus visionnées des vidéos et les inclure dans les résumés. Pour conserver une cohérence dans les résumés, il est également à noter que les auteurs prennent en compte le flux audio des vidéos pour ne pas couper les plans en plein milieu d'une phrase.

2.1.3 Performances système

Les travaux suivants permettent de produire des statistiques sur le comportement des utilisateurs et l'analyse de la fréquence des accès aux différentes vidéos. Par exemple,

dans [ASP00], une analyse des données de *log* obtenues à partir des accès utilisateurs à des vidéos sur le web est présentée. Les auteurs examinent des propriétés telles que les variations journalières des requêtes des utilisateurs accédant à des vidéos. Ils proposent de tirer avantage de ces propriétés pour l'amélioration des performances des systèmes multimédia tels que les systèmes de proxy, de cache et les serveurs de vidéo.

À titre d'exemple, leur analyse a montré que, généralement, les utilisateurs visionnent la première portion d'une vidéo pour savoir s'ils sont intéressés ou pas. S'ils sont intéressés, ils poursuivent le visionnage, sinon ils l'interrompent. Cette analyse suggère que mettre en cache les premières minutes d'une vidéo permettrait d'améliorer les performances d'accès au serveur vidéo. En effet, en partant de l'hypothèse qu'une vidéo ouverte sans envoi d'une requête *STOP* signifie que la vidéo est totalement visionnée, il reste 45% des visionnages qui sont arrêtés prématurément. Les auteurs ont observé que la majeure partie de ces fermetures se situe lors du visionnage des premiers 5% d'une vidéo (figure 2.8).

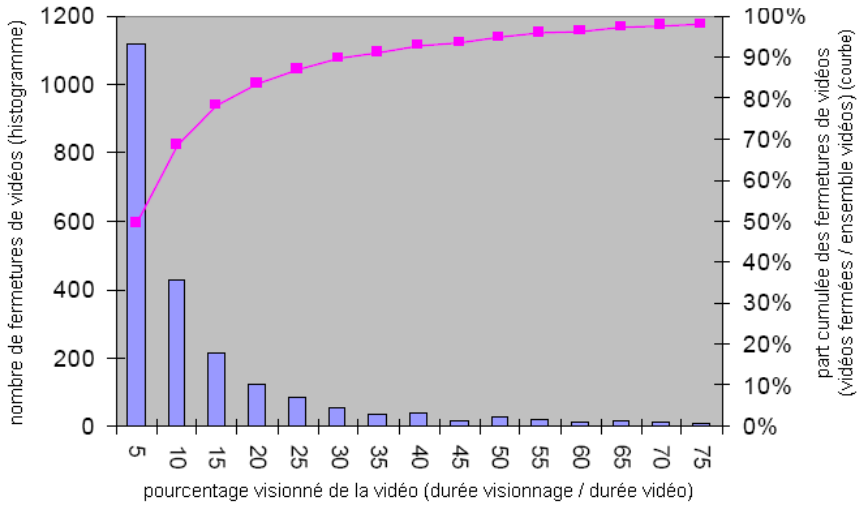


FIG. 2.8 – Degré de lecture partielle des vidéos [ASP00]

2.1.4 Enseignement à distance

Dans [RM02], les auteurs analysent l'usage d'un système d'enseignement multimédia par les étudiants. Cette analyse se base sur les types de comportement des étudiants lorsqu'ils utilisent le système. Ces types de comportement sont assez abstraits et de haut niveau (par exemple, les auteurs proposent deux manière d'appréhender l'apprentissage : de manière très logique et analytique où l'étudiant est autonome, de manière plus accompagnée où l'étudiant a besoin d'être entouré par d'autres étudiants et ses professeurs). Les besoins et les attentes en apprentissage dépendent des caractéristiques du type de personnalité de l'étudiant.

Pour conduire cette analyse, les auteurs ont développé un programme permettant d'extraire les actions effectuées par les étudiants sur le système multimédia et construit des profils utilisateurs permettant de tracer ce que fait chaque étudiant chaque fois qu'il utilise le système. Ces profils utilisateurs comportent les statistiques suivantes : le nombre de sessions de visionnage des vidéos, le nombre total de secondes passées à visionner les vidéos, le nombre de sessions ayant duré plus de 20 minutes, la durée moyenne d'une session, le nombre moyen d'actions par minute lors d'une session, (lecture, pause, saut...). En se basant sur les statistiques collectées sur les divers types de personnalité des étudiants, les auteurs analysent comment le système d'enseignement multimédia impacte sur les résultats des étudiants et comment améliorer l'enseignement à distance.

2.1.5 Qualité de service

L'adéquation des modèles de Markov pour ce champ applicatif a été étudiée et reconnue dans la littérature depuis plusieurs années. Les premiers travaux sur ce sujet peuvent être attribués aux auteurs de [DSSKT94]. Leur objectif est de construire un système de vidéo à la demande avec une qualité de service constante et une bande passante limitée. C'est-à-dire que les utilisateurs doivent pouvoir tout d'abord lire une vidéo sans attente et réaliser une avance rapide ou un retour rapide sur la vidéo sans plus d'attente. Or si la vidéo défile n fois plus vite ainsi, cela requiert n fois plus de ressources au niveau de la bande passante et du serveur de vidéo. Cette caractéristique du système est très

importante car ils ont observé deux propriétés lors des visionnages :

1. les utilisateurs passent la majeure partie du temps de visionnage en lecture,
2. le temps passé en avance ou retour rapide est relativement court.

Ils proposent donc, tout d'abord, de garantir la lecture de la vidéo pour tous les utilisateurs et de réserver une petite partie de la bande passante pour l'avance et le retour rapide. Ensuite, ils introduisent deux politiques dans le cas de manque de ressources pour le parcours rapide : soit l'utilisateur doit attendre que la ressource se libère en continuant la lecture (HD) ou sans lire pour libérer plus de bande passante pour les autres requêtes (RD), soit l'ensemble des parcours rapides en cours sont légèrement réduits en terme de débit pour permettre à la nouvelle demande d'être réalisée sans attente. Pour calculer les ressources nécessaires du serveur, ils modélisent le comportement des utilisateurs par une simple chaîne de Markov à deux états (figure 2.9).

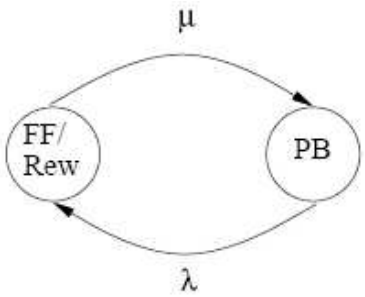


FIG. 2.9 – Modélisation du comportement des utilisateurs (**FF/Rew** : avance et retour rapide ; **PB** : playback ou lecture) [DSSKT94]

Les auteurs montrent finalement des résultats intéressants (figure 2.10) qui confirment que cette allocation dynamique de la bande passante permet de limiter le temps d'attente des utilisateurs. Ils affirment également que cette allocation permet de garantir l'accès aux données à plus d'utilisateur simultanément que l'approche de base qui réserve la bande passante nécessaire à l'avance et au retour rapide pour chaque utilisateur, qu'ils utilisent ces fonctionnalités ou non.

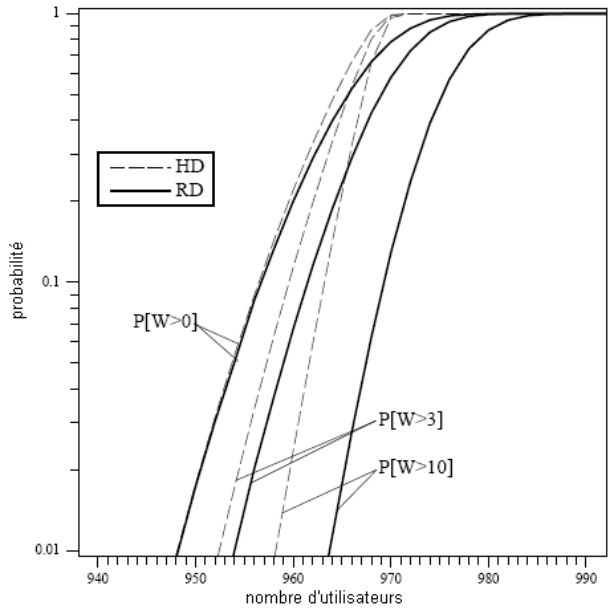


FIG. 2.10 – Probabilité que le temps d'attente d'un utilisateur (W) dépasse t secondes [DSSKT94]

Les auteurs de [SV95] utilisent des modèles de Markov similaires (figure 2.11) dans le développement d'un algorithme de compression de données pour proposer aux utilisateurs un service avance/retour rapide également. Ici, les auteurs proposent une approche qui combine un double encodage de basse résolution et de résolution normale associé à un algorithme de placement des données sur le disque dur pour s'assurer que les utilisateurs puissent parcourir les vidéos rapidement sans temps d'attente. Une analyse du modèle, comparée à des données simulées montre que celui-ci offre une bonne modélisation des utilisateurs, avec des valeurs légèrement supérieures d'environ 5%.

Dans [LLQW96], les auteurs développent un outil d'évaluation des performances de l'architecture d'un système de VOD. Pour décrire l'utilisation des ressources (bande passante et activité du serveur vidéo), ils combinent deux modèles. Le premier concerne

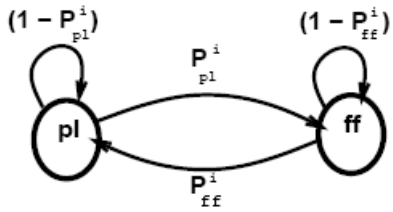


FIG. 2.11 – Modélisation du comportement des utilisateurs [SV95]

l'activité des utilisateurs (leurs interactions avec le système) et le second représente le traitement en paquets des requêtes des utilisateurs. Ces deux modèles peuvent être utilisés pour déterminer les ressources nécessaires pour différents systèmes.

L'activité des utilisateurs est modélisée une fois encore par un modèle de Markov à deux états (figure 2.12). Le premier état (*Normal*) correspond à un utilisateur en train de lire une vidéo. Le second (*Interaction*) correspond à une autre action telle que pause, saut, avance rapide, retour rapide, ralenti. Les auteurs considèrent une probabilité exponentielle de rester dans l'état *Normal* et calculent la probabilité de l'état *Interaction* en faisant la moyenne observée du temps passé dans chacune des autres actions. Ils montrent enfin que la prise en compte de ce modèle utilisateur a une grande influence sur les performances d'un système de VOD.

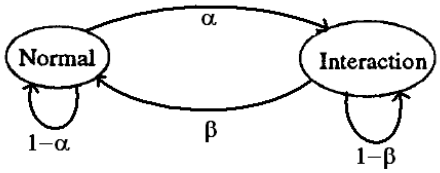


FIG. 2.12 – Modélisation du comportement des utilisateurs [LLQW96]

Dans [BET99], les auteurs tentent de déterminer le comportement des utilisateurs sur un système d'enseignement proposant des vidéos. Ils proposent notamment d'utiliser un

modèle de Markov pour modéliser les interactions des utilisateurs avec l'outil et extraire des valeurs statistiques telles que le temps passé en mode lecture, le temps total de visionnage, le temps moyen d'une pause... Ce modèle se compose d'états représentant les différentes interactions possibles (lecture, pause, avance rapide, fin de visionnage) et de transitions représentant les probabilités de passer d'un état à l'autre à chaque instant du visionnage (figure 2.13). Cette manière de prendre en compte le temps, bien qu'elle implique que l'on ne conserve pas dans le modèle le temps total passé devant la vidéo, est intéressante car elle permet de ne pas ajouter un paramètre supplémentaire au modèle.

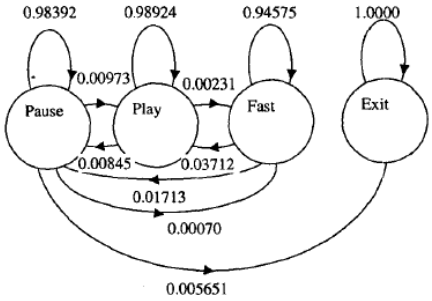


FIG. 2.13 – Chaîne de Markov du comportement [BET99]

De ces modèles, les auteurs extraient la matrice fondamentale qui donne le temps moyen passé dans chacun des états et comparent ces prédictions aux données réelles observées. Les prédictions du modèle de Markov se montrent très proches de la réalité, avec un écart faible entre les valeurs (figure 2.14).

2.1.6 Conclusion

Nous venons de voir que l'analyse du comportement des utilisateurs exploitant des vidéos est un point important dans le domaine de l'exploitation de données vidéo. Celle-ci est appliquée à différentes fins, comme la génération automatique de résumés, l'analyse et la prédiction des connexions sur les serveurs vidéo et d'enseignement à distance et l'amélioration des systèmes de VOD. Ce dernier point est le plus proche de nos travaux

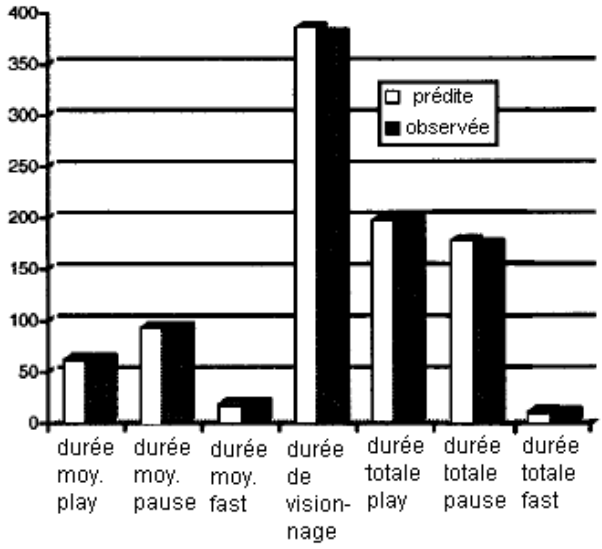


FIG. 2.14 – Comparaison entre les valeurs moyennes observées et estimées [BET99]

qui cherchent à modéliser finement le comportement des utilisateurs. On remarque aussi que, pour ce faire, les modèles de Markov sont utilisés dans beaucoup de travaux existant avec des résultats probants.

Le point non traité dans les travaux précédents est la non-corrélation des comportements généraux des utilisateurs avec leur comportement sur chacune des séquences vidéo. Ces travaux ne prennent pas ou très peu en compte les actions réalisées durant les visionnages lors de l'analyse de la navigation entre les vidéos (la session). Les concepts de navigation et de recherche dans une grande base de données vidéo ne sont pas définis. De plus, il n'existe à notre connaissance aucun jeu de données public ou banc d'essai public relatifs aux données *log* vidéo, ce qui ne simplifie pas la tâche de validation des résultats.

Article	VOD	Résumés	Perf. Syst.	Enseignement	Qualité Serv.	Modèle Markov
[LGS+00]	×					
[SPAP99]	×					
[HNK+04]	×					
[LGD+05]	×					
[FS97]	×					
[YZZZ06]	×					
[NAMO00]	×					
[YMNZ03]		×				
[SMP01]		×				×
[ASP00]			×			
[RM02]				×		
[DSSKT94]					×	×
[SV95]					×	×
[LLQW96]					×	×
[BET99]					×	×

TAB. 2.1 – Synthèse des techniques de fouille des données vidéo

2.2 Un parallèle avec le web usage mining

Pourquoi un parallèle ?

Tout comme pour le multimédia et la vidéo, un pan de la fouille de donnée s'intéresse particulièrement au web. Ces techniques ont deux objectifs principaux qui sont la mise au point de techniques pour l'aide à la conception de sites (notamment le choix des publicités et leur emplacement) et le développement de solutions pour assister les utilisateurs dans leurs navigations.

D'abord, un besoin essentiel se fait ressentir dans le domaine du commerce et de la publication électronique. Comprendre les comportements des utilisateurs et leurs besoins pour placer des publicités ciblées par exemple est devenu un défi essentiel. C'est pourquoi le design des sites web cherche à guider les utilisateurs vers des produits d'appel. De plus, l'analyse du trafic réseau généré est importante dans la détermination du matériel requis pour offrir une qualité certaine de communication.

Ensuite, la recherche est motivée par la volonté d'offrir aux utilisateurs des outils et des services améliorant leur mode de navigation. Les utilisateurs bénéficient d'accès à des moteurs de recherche dotés des dernières technologies, à des sites web personnalisables.

Actuellement, il existe trois axes de recherche importants dans le domaine de la fouille de données web :

- l'extraction des connaissances informationnelles,
- la compréhension de la structure des liens hypertexte,
- *la compréhension du comportement des utilisateurs.*

C'est ce dernier point qui va ici nous intéresser. Le parallèle avec notre domaine de recherche est assez évident. Dans les deux cas, nous nous intéressons à appréhender le comportement des utilisateurs sur des données ciblées, la manière dont ils interagissent avec elles (boutons d'action pour la vidéo, liens hypertexte pour le web), la notion de session correspondant à la séquence des données accédées. De plus, l'un de nos objectifs est commun aux deux domaines : optimiser les techniques actuelles d'accès aux informations, par retour d'expérience sur les moteurs de recherche.

Approche markovienne

Beaucoup de travaux proposent l'utilisation des modèles de Markov d'ordre 1 (la section (3.2) présente la théorie des modèles de Markov en détails) pour prédire les requêtes des utilisateurs sur le web. Le modèle de Markov imaginé par les auteurs de [Bes95] met en place un serveur de pré-chargement des pages dans un cache afin de diminuer la charge sur le serveur et le temps d'attente des utilisateurs. Ce serveur est construit à partir des informations passées relevées lors de l'utilisation du site web. Des probabilités de transition entre pages sont calculées proportionnellement au nombre de fois où ces pages sont accédées dans une fenêtre temporelle donnée. Les expérimentations conduites montrent que la méthode est effective pour la réduction de la charge du serveur et pour la diminution du temps de réponse. Une méthode similaire est proposée par les auteurs de [PM96] dans laquelle un graphe de dépendances est déduit et mis à jour dynamiquement lorsque le serveur traite de nouvelles requêtes. À chaque page est associé un noeud. Un arc entre deux noeuds existe si le noeud cible est visité après le noeud source. Le poids de cet arc est proportionnel au nombre d'accès observés. Cette approche, bien que les expérimentations ont montré une diminution des temps de chargement, semble limitée par le nombre de pages considérées. En effet, l'étude d'un site composé de très nombreuses pages mène à la création d'un graphe de dépendance très complexe et donc très coûteux en mémoire et en temps de traitement.

Dans [CHM+00], les auteurs proposent une méthodologie de visualisation des motifs de navigation dans un site web. Les utilisateurs présentant les même motifs de navigation sont regroupés dans une même classe. Chacune de ces classes est représentée par un modèle de Markov. Les auteurs de [Sar00] proposent un système permettant de démontrer l'utilité des modèles de Markov pour la prédiction des liens et l'analyse des sessions de navigation. Les résultats expérimentaux montrent que ces modèles sont très efficaces dans la prédiction des futures pages accédées par les utilisateurs.

Dans [PP99], les auteurs présentent une étude sur la pertinence des modèles de Markov d'ordre k, $k \geq 1$ à prédire les navigations des utilisateurs. Différents algorithmes pour la reconstruction des chemins sont testés et les résultats montrent que les modèles déduits

des chaînes de longueur importante sont plus précis. Ils proposent ensuite une méthode visant à réduire la taille des modèles et conservant un maximum de précision. Pour cela, ils posent l'hypothèse que des sous-séquences communes à différentes sessions offrent un plus grand pouvoir de prédiction que les sessions elles-mêmes. Les tests montrent qu'il est effectivement possible de réduire ainsi la taille des modèles tout en maintenant une bonne capacité de prédiction.

Approche séquentielle

Comme dans le paragraphe précédent, les auteurs de [WZ02] proposent une approche se basant uniquement sur la recherche de sous-séquences communes pour regrouper les sessions des utilisateurs. Ses auteurs proposent un algorithme d'alignement de séquences pour mesurer les similarités entre les sessions web. Cette mesure prend donc en compte les pages accédées dans un ordre chronologique. Après avoir reconstruit les sessions à partir des requêtes stockées sur un serveur web, ils utilisent l'algorithme *CURE* [GRS98] pour regrouper les sessions. Leurs expérimentations dans un contexte précis (e-learning) montrent l'intérêt de considérer les sessions plutôt que les ensembles de pages. Les classes ainsi extraites ont plus de sens que si elles avaient été basiquement calculées à partir d'une mesure d'intersection d'ensembles, comme le coefficient de *Jaccard* utilisé dans [MDL+00].

Dans [HWV01] les auteurs proposent également un regroupement de sessions web basé sur l'appariement de séquences. Leur approche se base sur une méthode d'alignement de séquences utilisée pour calculer la similarité entre les séquences. Cette technique, appelée distance d'édition, compte le nombre d'actions basiques (insertion, suppression, modification) à effectuer sur une séquence pour la faire correspondre à une autre. À l'aide d'un algorithme de classification hiérarchique, ils arrivent à extraire des classes de sessions intéressantes. L'un des intérêts de cet article est de montrer l'importance de considérer la séquentialité des données d'une manière dynamique lors du regroupement. En effet, les auteurs montrent qu'une mesure d'association simple ne permet pas, au contraire d'un alignement dynamique, de bien discriminer les classes.

Autres approches

Dans [JZM04] les auteurs développent un canevas unifié pour la découverte et l'analyse des motifs de navigation web basé sur une analyse sémantique latente probabiliste (*pLSA*, probabilistic Latent Semantic Analysis). Cette technique caractérise automatiquement les objectifs de navigation des utilisateurs et découvre les relations cachées entre les utilisateurs et les pages web. Ces relations sont mesurées en terme de probabilité. Ainsi, les auteurs sont à même de découvrir différents motifs d'utilisation comme la définition d'une tâche en fonction des pages visitées, l'identification d'un utilisateur-type qui réalise une tâche donnée, la caractérisation de groupes d'utilisateurs réalisant un même ensemble de tâches.

Une autre méthode exploitant un modèle plus simple est proposée par les auteurs de [YJGMD96]. Chaque session de navigation est modélisée par un vecteur contenant le nombre de visites de l'utilisateur sur chaque page du site web. Sur la base de cette représentation, un regroupement est effectué pour trouver les sessions type. Cette représentation est intéressante par sa simplicité mais limitée, comme nous l'avons vu dans la section relative à l'analyse séquentielle, par le fait de ne pas prendre en compte l'ordre dans lequel les pages ont été visitées.

Conclusion

L'analyse du comportement dans le domaine de la fouille de données web est largement étudiée depuis une dizaine d'années. Ces travaux se divisent en deux grandes familles : markovienne et séquentielle.

Les approches markoviennes semblent intéressantes quant à la qualité des résultats obtenus mais ont un inconvénient majeur. En effet, comme nous pouvons le voir dans [CHM+00], l'architecture et les paramètres des modèles extraits ne sont pas liés à la nature même des données et nécessitent une phase d'initialisation très complexe. De plus, les modèles finalement générés ne sont pas directement lisibles. Un non-spécialiste ne sera a priori pas à même de les comprendre.

Les approches séquentielles pallient cet inconvénient. La nature même des modèles

induits est directement liée aux données et ils sont très aisément compréhensibles par un spécialiste qui lit simplement la succession des pages accédées. Elles sont cependant limitées comme nous l'avons écrit précédemment par le choix de la sous-séquence la plus adaptée qui implique une importante perte d'informations.

2.3 Nouveauté de notre approche

Deux points importants distinguent notre approche des travaux actuels. Tout d'abord, il n'existe aucun outil public analysant l'utilisation complète d'une base de données vidéo. Les seuls travaux que nous avons référencés dans le domaine de l'analyse vidéo ne considèrent qu'une vidéo à la fois.

Nous avons développé une technique de regroupement qui correspond à la nature de nos données. En effet, nombre de techniques du *web mining* utilisent des algorithmes basés uniquement sur les distances et l'approche par voisinage produisant des résultats difficiles à analyser. Il n'est pas rare de retrouver deux éléments totalement différents dans une même classe pour peu qu'ils soient connectés par une chaîne de voisins très proches les uns des autres. Dans le cas de séquences, deux séquences totalement diférentes (sans aucun élément en commun) peuvent être groupées dans une même classe si il existe entre-elles une suite de séquences différentes d'un seul élément deux à deux (figure 2.15).

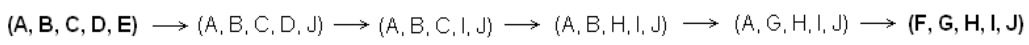


FIG. 2.15 – Deux sessions différentes mais connectées par des voisins proches

Nous proposons d'introduire un modèle de classe qui capitalisera les informations données par tous les éléments d'une même classe, éléments qui devront correspondre à ce modèle. C'est-à-dire que nous représenterons une classe d'éléments par un modèle et que tous les éléments de la classe seront proches de ce modèle.

Bibliographie

[ASP00] S. Acharya, B. Smith et P. Parnes : Characterizing user access to videos on the world wide web. *Multimedia computing and networking. Californie, USA*, 2000.

[Bes95] A. Bestavros : Using speculation du reduce server load and service time on the WWW. *Fourth ACM international conference on information and knowledge management. Baltimore, Maryland, USA*, 1995.

[BET99] P. Branch, G. Egan et B. Tonkin : Modeling interactive behaviour of a video based multimedia system. *IEEE international conference on communications*, 2:978–982, Juin 1999.

[CHM+00] I. Cadez, D. Heckerman, C. Meek, P. Smyth et S. White : Visualization of navigation patterns on a web site using model based clustering. *Sixth ACM International Conference on Knowledge Discovery and Data Mining. Boston, Massachusetts, USA*, pages 280–284, 2000.

[DSSKT94] J. Day-Sircar, J. D. Salehi, J. K. Kurose et D. Towsley : Providing VCR capabilities in large-scale video servers. *ACM international conference on mulimedia. San-Francisco, Californie, USA*, 1994.

[FS97] P. Faudemay et C. Seyrat : Intelligent delivery of personalised video programmes from a video database. *international workshop on database and expert systems applications*, pages 172–177, 1997.

[GRS98] S. Guha, R. Rastogi et K. Shim : An efficient clustering algorithm for large databases. *ACM international conference on management of data. Seattle, Washington, USA*, 1998.

[HNK+04] L. Hollink, G. P. Nguyen, D. Koelma, A. T. Schreiber et M. Worring : User strategies in video retrieval : A case study. *International conference on image and video retrieval. Springer ISSN :0302-9743*, 3115:6–14, 2004.

[HWV01] B. Hay, G. Wets et K. Vanhoof : Clustering navigation patterns on a website using a sequence alignment method. *Seventeenth international joint conference on artificial intelligence. Seattle, Washington, USA*, Août 2001.

[JZM04] X. Jin, Y. Zhou et B. Mobasher : Web usage mining based on probabilistic latent semantic analysis. *ACM international conference on knowledge discovery and data mining. Seattle, Washington, USA*, Août 2004.

[LGD+05] L.Hollink, G.P.Nguyen, D.Koelma, A.Th.Schreiber et M.Worring : Assessing user behaviour in news video retrieval. *IEEE international conference on vision, image and signal processing*, 152, 2005.

[LGS+00] F. C. Li, A. Gupta, E. Sanocki, L. He et Y. Rui : Browsing digital video. *SIGCHI conference on human factors in computing systems. La Haye, Pays-Bas. ACM Press ISBN :1-58113-216-6*, 2000.

[LLQW96] V. O. K. Li, W. Liao, X. Qiu et E. W. M. Wong : Performance model of interactive video-on-demand systems. *IEEE Journal on Selected Areas in Communications*, 14, Août 1996.

[MDL+00] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Y. Sun et J. Wiltshire : Discovery of aggregate usage profiles for web personalization. *KDD workshop on web mining and web usage analysis. Boston, Massachussets, USA*, 2000.

[NAMO00] A. Nakamura, N. Abe, H. Matoba et K. Ochiai : Automatic recording agent for digital video server. *ACM international conference on multimedia*, pages 57–66, 2000.

[PM96] V. Padmanabhan et J. Mogul : Using predictive prefetching to improve WWW latency. *ACM international conference on the special interest group on data communication. Université de Stanford, Californie, USA*, 1996.

[PP99] P. Pirolli et J. Pitkow : Distributions of surfers' paths through the world wide web : empirical characterizations. *World wide web journal*, 2:29–45, 1999.

[RM02] A. I Reuther et D. G. Meyer : The effect of personality type on the usage of a multimedia engineering education system. *Frontiers in education. Seattle, Washington, USA*, 1:T3A7–T3A12, Novembre 2002.

[Sar00] R. Sarukkai : Link prediction and path analysis using markov chains. *Ninth international world wide web wonference. Amsterdam, Pays-Bas*, Mai 2000.

[SMP01] T. Syeda-Mahmood et D. Ponceleon : Learning video browsing behavior and its application in the generation of video previews. *Ninth ACM conference on multimedia. Ottawa, Canada*, Octobre 2001.

[SPAP99] S. Srinivasan, D. B. Ponceleon, A. Amir et D. Petkovic : "what is in that video anyway ?" In bearch of better browsing. *IEEE international conference on multimedia computing and systems*, 1:388–393, 1999.

[SV95] P. J. Shenoy et H. M. Vin : Efficient support for scan operations in video servers. *ACM international conference on mulimedia. San-Francisco, Californie, USA*, 1995.

[WZ02] W. Wang et O. R. Zaïane : Clustering web sessions by sequence alignment. *Thirteenth international workshop on database and expert systems applications. Aix-en-Provence, France*, 2002.

[YJGMD96] T. Yan, M. Jacobsen, H. Garcia-Molina et U. Dayal : From user access patterns to dynamic hypertext linking. *Computer networks & ISDN Systems*, 28:1007–1014, 1996.

[YMNZ03] B. Yu, W. Ma, K. Nahrstedt et H. Zhang : Video summarization based on user log enhanced link analysis. *ACM international conference on mulimedia. Berkeley, Californie, USA*, Novembre 2003.

[YZZZ06] H. Yu, D. Zheng, B. Y. Zhao et W. Zheng : Understanding user behavior in large-scale video-on-demand systems. *EuroSys conference. Louvain, Belgique. ACM ISSN :1-59593-322-0*, 2006.

Chapitre 3

Concepts de notre approche

Sommaire

3.1 Introduction

Dans ce chapitre, nous introduisons différentes notions théoriques et différents outils que nous utilisons pour réaliser l'analyse du comportement des utilisateurs. Nous présentons en particulier trois points : les modèles de Markov, l'extraction de sous-séquences et la dissimilarité de *Kullback-Leibler*.

Tout d'abord, les modèles de Markov nous sont utiles pour représenter les comportements intra-vidéo des utilisateurs, c'est-à-dire les interactions des utilisateurs avec le lecteur observées lors de la lecture d'une vidéo. Comme nous l'avons vu au chapitre précédent, ces modèles sont particulièrement adaptés à ce propos. Nous présentons donc la théorie générale des modèles de Markov avant de nous intéresser aux modèles absorbants et à leurs propriétés que nous exploitons par la suite.

Ensuite, nous présentons une technique d'extraction de la plus longue sous-séquence commune à deux séquences. Cette technique nous est utile pour la modélisation et le regroupement des comportements inter-vidéo. C'est-à-dire l'enchainement des différentes vidéos visionnées par un utilisateur. Nous analysons en détail la technique de programmation dynamique permettant d'extraire cette sous-séquence.

Enfin, nous nous intéressons aux notions de distance et de dissimilarité, essentielles à appréhender pour appliquer les techniques de regroupement. Nous présentons notamment la dissimilarité de *Kullback-Leibler* que nous utilisons pour le regroupement des comportements intra-vidéo dans la section (5.2.3).

3.2 Les modèles de Markov

Les modèles de Markov permettent de modéliser de manière élémentaire, mais robuste, de nombreux phénomènes aléatoires où l'évolution future d'une quantité ne dépend du passé qu'au travers de sa valeur présente.

Par exemple, si on note X_n le nombre de connexions disponibles à un serveur de vidéo à l'instant n, D_{n+1} le nombre de nouvelles demandes de connexion formulées par des utilisateurs à l'instant $n+1$, et q_n le nombre de connexions libérées par des utilisateurs entre l'instant n et $n+1$, alors à l'instant $n+1$, l'état du serveur est $X_{n+1} = (X_n + q_n - D_{n+1})^+$, où x^+ désigne la partie positive de $x \in R$. Dans le cas où la demande est modélisée par des variables aléatoires indépendantes, alors l'évolution future $(X_k, k \geq n+1)$ ne dépend du passé $(X_k, k \in [0..n])$ qu'au travers de l'état présent X_n. Cette propriété, dite propriété de Markov, est la base de la définition des modèles de Markov.

Après avoir présenté les définitions de base des modèles de Markov, nous nous intéresserons particulièrement aux modèles de Markov absorbants. Ce sont ces derniers que nous utilisons pour modéliser les comportements des utilisateurs. Nous verrons les différents théorèmes et propriétés que nous appliquons à notre modèle. D'autres modèles, appelés modèles ergodiques existent mais nous ne les traiterons pas ici. Nous pouvons les retrouver dans la littérature [Bré98, Chu67, Duf97, FS02, MPB98, Yca02].

3.2.1 Définitions

Un modèle de Markov peut être décrit de la manière suivante. Étant donné un ensemble d'états $S = \{s_1, s_2, ...s_r\}$, le processus débute dans l'un de ces états et se propage successivement d'un état à un autre. Chaque mouvement correspond à une étape du processus. Si le processus se trouve dans l'état s_i, alors il peut se trouver dans l'état s_j à l'étape suivante avec une probabilité p_{ij}. Cette probabilité ne dépend pas du chemin effectué depuis le début du processus, mais uniquement de l'état actuel.

L'ensemble des probabilités $p_{ij}, 1 \leq i, j \leq r$, r étant le nombre d'états composant le modèle, est représenté par la matrice de transition du modèle P. Notons également que le processus peut rester dans le même état d'une étape à l'autre avec une probabilité p_{ii}.

Pour définir totalement le modèle, il reste à définir un vecteur $u = (u_1, u_2, ...u_r)$ correspondant aux probabilités de débuter le processus dans l'un ou l'autre des états. La probabilité de débuter dans l'état s_i est u_i. L'état s_i est alors appelé état initial.

Un modèle de Markov respecte finalement les deux contraintes suivantes :
- $\sum_{i=1}^{r} u_i = 1$ (la somme des probabilités des états initiaux est égale à 1),
- $\forall i, 1 \leq i \leq r, \sum_{j=1}^{r} p_{ij} = 1$ (la somme des probabilités des transitions partant d'un état est égale à 1).

Une célèbre image donne une représentation concrète de ces modèles. Ils sont assez analogues à une grenouille cherchant à traverser un lac, et sautant de nénuphar en nénuphar. La grenouille part d'un nénuphar et, à chaque étape, saute sur un autre nénuphar avec une certaine probabilité de transition.

3.2.1.1 La matrice de transition P

Considérons maintenant l'exemple simple suivant. Pour déterminer l'indice de satisfaction d'un utilisateur à la lecture d'une vidéo en fonction de son indice de satisfaction sur la dernière vidéo visionnée, un modèle (très simplifié) de prévision pourrait être représenté par la matrice P suivante. Soient M une mauvaise satisfaction, N un avis neutre sur la vidéo et B une bonne satisfaction :

$$P = \begin{array}{c} \\ M \\ N \\ B \end{array} \begin{array}{c} \begin{array}{ccc} M & N & B \end{array} \\ \begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix} \end{array}$$

Une autre représentation, plus intuitive des modèles de Markov est souvent utilisée. Cette représentation graphique définit les différents états du modèle par des cercles et les transitions d'un état à l'autre sont représentées par des flèches, elles-même labellisées par les probabilités de transition considérée. La figure (3.1) représente le même modèle que la matrice P.

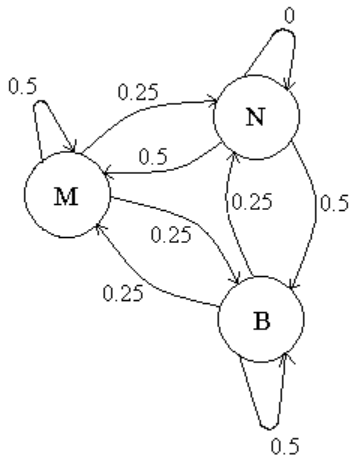


FIG. 3.1 – Modèle de satisfaction des utilisateurs

Nous voyons ici que, selon ce modèle, il n'y a jamais deux vidéos laissant à l'utilisateur un avis neutre à la suite ($p_{NN} = 0$) mais, qu'après un avis neutre, nous avons la même probabilité d'avoir une bonne ou une mauvaise satisfaction ($p_{NM} = p_{NB} = \frac{1}{2}$). Si nous avons une bonne ou une mauvaise satisfaction, il y a une probabilité de $\frac{1}{2}$ d'avoir la même satisfaction lors du visionnage de la vidéo suivante ($p_{MM} = p_{BB} = \frac{1}{2}$). Si elle n'est pas la même, alors nous aurons un avis neutre une fois sur deux pour le visionnage suivant ($p_{MB} = p_{MN} = p_{BM} = p_{BN} = \frac{1}{4}$).

Les valeurs de la première ligne de P représentent donc les probabilités de satisfaction (M, N ou B)suivant une vidéo peu appréciée. Celles de la seconde et de la troisième

représentent quant à elles respectivement un avis neutre et une bonne satisfaction. Une telle matrice carrée est donc appelée matrice des probabilités de transition ou, plus simplement, matrice de transition.

3.2.1.2 Fonctions de prédiction

Considérons maintenant le problème de se projeter plusieurs étapes en avant dans le processus d'analyse, ce qui revient pour notre exemple à déterminer l'indice de satisfaction d'un utilisateur dans plusieurs visionnages. Par exemple, cherchons à déterminer la probabilité d'avoir une bonne satisfaction dans deux visionnages successifs précédés d'un visionnage ayant donné une mauvaise satisfaction. Cette probabilité correspond à l'union disjointe des trois événements suivants :

1. la prochaine est neutre et la suivante bonne
2. la prochaine est mauvaise et la suivante bonne
3. la prochaine est bonne et la suivante bonne

La probabilité du premier événement est le produit des probabilités conditionnelles que la prochaine vidéo soit d'avis neutre sachant que le dernier est mauvais et que la prochaine vidéo soit d'avis bon sachant que la dernière vidéo est d'avis neutre. En se rapportant à la matrice des transitions P, cette probabilité vaut $p_{MN} \times p_{NB}$. En écrivant les deux autres évènements de la même manière, nous obtenons une bonne satisfaction dans deux visionnages sachant que le dernier fut mauvais $p^{(2)}_{MB}$:

$$\begin{aligned} p^{(2)}_{MB} &= p_{MM} \times p_{MB} + p_{MN} \times p_{NB} + p_{MB} \times p_{BB} \\ &= \textstyle\sum_{i \in \{M,N,B\}} p_{Mi} \times p_{iB} \end{aligned}$$

En généralisant cette observation, nous pouvons donner le théorème suivant :

Théorème 3.2.1 *Soit P la matrice de transition d'un modèle de Markov, la ij^{eme} valeur $p^{(n)}_{ij}$ de la matrice P^n est la probabilité que, partant de l'état s_i, le processus se trouve dans l'état s_j après n étapes.*

Considérons maintenant le comportement à long terme d'un modèle de Markov quand on débute dans un état défini par une distribution de probabilités, représentée par un vecteur de probabilités. Un tel vecteur est un vecteur ligne dont toutes les valeurs sont positives ou nulles et dont la somme des composants est égale à 1. Si u est un vecteur de probabilité, alors sa i^{eme} valeur u_i représente la probabilité que le processus débute dans l'état s_i du modèle.

Théorème 3.2.2 *Soient P la matrice de transition d'un modèle de Markov, u le vecteur de distribution initiale du modèle, la probabilité que le modèle soit dans l'état s_i après n étapes est la i^{eme} composante du vecteur*

$$u^{(n)} = u \times P^n$$

Remarque 3.2.3 *Dans le cas où l'on connaît l'état d'entrée d'un modèle, on travaille en prenant un vecteur de distribution initiale dont toutes les valeurs sont nulles, exceptée celle correspondant à l'état d'entrée qui vaudra* 1.

3.2.2 Les modèles de Markov absorbants

3.2.2.1 Définitions

Les modèles de Markov absorbants sont un sous-ensemble des modèles de Markov. Ils ont la particularité de posséder un ou plusieurs états absorbants. Ces états peuvent être considérés comme des puits. Il est possible d'y entrer mais aucune transition ne permet d'en sortir.

Définition 3.2.4 *Un état s_i d'un modèle de Markov est un état absorbant si et seulement si $p_{ii} = 1$ et $p_{ij} = 0, s_j \in S - \{s_i\}$. Un modèle de Markov est un modèle absorbant si et seulement si au moins l'un de ses états est absorbant et s'il est possible d'atteindre un état absorbant depuis tous les états du modèle (en une ou plusieurs étapes).*

Définition 3.2.5 *Dans un modèle de Markov absorbant, un état qui n'est pas absorbant est appelé un état de transition.*

Prenons maintenant l'exemple d'un serveur de VOD (vidéo à la demande) qui cherche à fidéliser ses clients. Ces derniers sont classés en cinq catégories : les utilisateurs *lambda*, les utilisateurs occasionnels (1), les utilisateurs réguliers (2), les utilisateurs intensifs (3), les utilisateurs VIP (*c'est-à-dire* les utilisateurs les plus importants). En fonction des usages, ils évoluent d'une catégorie à l'autre jusqu'à atteindre le statut *lambda* où aucun traitement particulier ne leur sera accordé ou le statut VIP où ils seront des utilisateurs privilégiés. Nous pouvons représenter ce processus par le modèle de Markov de la figure (3.2).

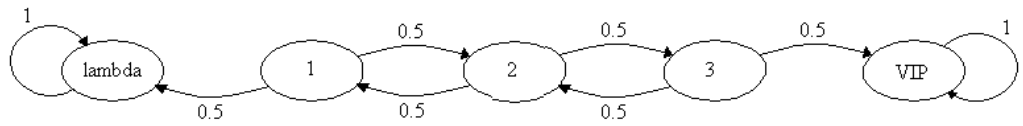


FIG. 3.2 – Modèle de fidélisation des clients de la VOD

Dans ce modèle, les états 1, 2 et 3 sont des états de transition. Les états *lambda* et VIP sont des états absorbants ($p_{lambda\ lambda} = p_{VIP\ VIP} = 1$). Quand un processus atteint un état absorbant, on dit qu'il est absorbé.

3.2.2.2 La forme canonique

Si l'on prend un modèle de Markov absorbant, que l'on réordonne les états de telle sorte que les t états de transitions soient devant les a états absorbants, nous pouvons écrire la matrice de transition sous la forme canonique :

$$P = \begin{array}{c} \\ TR. \\ ABS. \end{array} \begin{array}{c} \begin{array}{cc} TR. & ABS. \end{array} \\ \left(\begin{array}{c|c} Q & R \\ \hline 0 & I \end{array}\right) \end{array}$$

- I est une matrice identité $r \times r$;
- 0 est une matrice nulle $r \times t$;
- R est une matrice non-nulle $t \times r$;

– Q est une matrice carrée $t \times t$.

L'intérêt de cette représentation canonique de la matrice P est de pouvoir facilement calculer la matrice P^n :

$$P^n = \begin{array}{c} \\ TR. \\ ABS. \end{array} \begin{array}{c} \begin{array}{cc} TR. & ABS. \end{array} \\ \left(\begin{array}{c|c} Q^n & * \\ \hline 0 & I \end{array}\right) \end{array}$$

– $*$ est une matrice non-nulle $t \times r$;
– Q^n donne la probabilité d'arriver dans un état de transition à l'état n en partant d'un état de transition ;
– I est toujours une matrice identité $r \times r$;
– 0 est toujours une matrice nulle $r \times t$;

Théorème 3.2.6 *Dans un modèle de Markov absorbant, la probabilité que le processus soit absorbé est de* 1. $lim_{n \to +\infty} Q^n \to 0$.

3.2.2.3 La matrice fondamentale

Théorème 3.2.7 *Soit un modèle de Markov absorbant écrit sous forme canonique, la matrice* $(I - Q)$ *admet une matrice inverse* N *et* $N = I + Q + Q^2 + ...$ *La* ij^{eme} *valeur* n_{ij} *de la matrice* N *est le nombre moyen attendu de passages par l'état* s_j *d'un processus ayant débuté dans l'état* s_i.

Définition 3.2.8 *Soit un modèle de Markov absorbant* P, *la matrice* $N = (I - Q^{-1})$ *s'appelle la matrice fondamentale de* M.

La démonstration de ce théorème peuvent être retrouvées dans [GS97].

Si nous reprenons l'exemple précédent, nous pouvons écrire le modèle sous la forme canonique suivante :

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ lambda \\ VIP \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & lambda & VIP \end{array} \\ \left(\begin{array}{ccc|cc} 0 & 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}$$

Nous voyons ici que la matrice Q vaut :

$$Q = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{pmatrix}$$

et

$$I - Q = \begin{pmatrix} 1 & -0.5 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & -0.5 & 1 \end{pmatrix}$$

Nous pouvons alors en déduire la matrice N :

$$N = (I - Q)^{-1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1.5 & 1 & 0.5 \\ 1 & 2 & 1 \\ 0.5 & 1 & 1.5 \end{pmatrix} \end{array}$$

En observant la seconde ligne de la matrice N, nous pouvons conclure que si l'utilisateur est classé dans la catégorie 2 (utilisateur régulier), alors les nombres moyens d'étapes passées dans les états 1, 2 et 3 avant d'atteindre le statut *lambda* ou VIP (ie. avant que le processus soit absorbé) sont 1, 2 et 1.

3.2.2.4 Temps avant absorption

Considérons maintenant le problème suivant : étant donné un processus débutant dans l'état s_i, quel est le nombre moyen d'étapes qu'il traversera avant d'être absorbé ? Pour répondre à cela, voyons le théorème suivant.

Théorème 3.2.9 *Soient t_i le nombre moyen d'étapes parcourues avant absorption par un processus ayant débuté dans l'état s_i et t un vecteur colonne tel que sa i^{eme} valeur est égale à t_i, alors*

$$t = N \times c$$

où c est un vecteur colonne dont toutes les valeurs sont égales à 1.

Dans le cas de la fidélisation des clients, nous obtenons :

$$t = N \times c = \begin{pmatrix} 1.5 & 1 & 0.5 \\ 1 & 2 & 1 \\ 0.5 & 1 & 1.5 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}$$

Nous pouvons observer que le temps moyen avant absorption, débutant dans l'état 2 est de quatre étapes. Notons également que $t_1 = t_3 = t_2 - 1$, ce qui est logique, étant donné que les états 1 et 3 sont séparés de l'état 2 par une seule étape et sont plus proches des états absorbants. Enfin, le modèle étant symétrique, le temps avant absorption est le même pour 1 et 3.

3.2.2.5 Probabilité d'absorption

Théorème 3.2.10 *Soient b_{ij} la probabilité qu'un processus soit absorbé par l'état s_j sachant qu'il a débuté dans l'état s_i, B une matrice $t \times r$ dont les valeurs sont égales à b_{ij}, alors*

$$B = N \times R$$

où N est la matrice fondamentale et R est une partie extraite de la forme canonique du modèle.

En reprenant l'exemple précédent, nous pouvons extraire de la forme canonique :

$$R = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} lambda & VIP \end{matrix} \\ \begin{pmatrix} 0.5 & 0 \\ 0 & 0 \\ 0 & 0.5 \end{pmatrix} \end{matrix}$$

Alors, nous pouvons calculer :

$$B = N \times R = \begin{pmatrix} 1.5 & 1 & 0.5 \\ 1 & 2 & 1 \\ 0.5 & 1 & 1.5 \end{pmatrix} \times \begin{pmatrix} 0.5 & 0 \\ 0 & 0 \\ 0 & 0.5 \end{pmatrix} = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} lambda & VIP \end{matrix} \\ \begin{pmatrix} 0.75 & 0.25 \\ 0.5 & 0.5 \\ 0.25 & 0.75 \end{pmatrix} \end{matrix}$$

Ici, la première ligne de la matrice B montre que, en débutant dans l'état s_1, il y a une probabilité de $\frac{3}{4}$ qu'un processus soit absorbé par l'état *lambda* et de $\frac{1}{4}$ par l'état VIP.

3.3 Plus longue sous-séquence commune (LCS)

L'extraction de la plus longue sous-séquence commune à deux séquences est un problème récurrent dans le domaine de la fouille de données. Que ce soit en bio-informatique pour comparer des brins d'ADN ou en fouille de données web pour comparer des sessions, l'extraction de cette sous-séquence est souvent utilisée comme technique de comparaison. Notre objectif n'est pas d'optimiser ces techniques mais de présenter l'outil que nous utilisons. Nous nous intéressons donc ici à la technique de programmation dynamique qui permet d'extraire la plus longue sous-séquence commune à deux séquences [CLRS01].

3.3.1 Définition

Une séquence w est une plus longue sous-séquence commune de deux séquences x et y (notée $LCS(x, y)$) si elle est une sous-séquence de x, une sous-séquence de y et que sa longueur est maximale. Une sous-séquence d'une séquence x est obtenue en supprimant zéro, un ou plusieurs éléments de x. De manière plus formelle, soient $w = (w_1, w_2, ...w_n)$ une séquence de longueur n et $x = (x_1, x_2, ...x_m)$ une séquence de longueur m, w est sous-séquence de x si et seulement si :

$$\begin{gathered} \exists\ (k_1, k2, ...k_n) \in N^n\ tel\ que \\ \forall\ i, j < n,\ i < j \Rightarrow k_i < k_j\ tel\ que \\ w_j = x_{k_j} \end{gathered} \tag{3.1}$$

Notons ici que la LCS de deux séquences n'est pas unique. En effet, il peut exister plusieurs sous-séquences différentes de longueur maximale. Elles sont dans ce cas toutes plus longues sous-séquences communes.

3.3.2 Exemple

Soient les séquences :

$$x = (a, b, e, c, f, g, h, d)$$

$$y = (e, f, g, a, b, c, d, h)$$

Nous pouvons extraire plusieurs sous-séquences de longueur 4, chacune étant une plus longue sous-séquence commune.

$$LCS(x, y) = (a, b, c, d)$$

$$LCS(x, y) = (a, b, c, h)$$

$$LCS(x, y) = (e, f, g, h)$$

$$LCS(x, y) = (e, f, g, d)$$

3.3.3 Algorithme de calcul

Nous considérons ici deux séquences x et y de longueur respective n et m dont nous cherchons à extraire la/les plus longues sous-séquences communes. Nous notons :

- $ens_{lcs}(x, y)$, l'ensemble de la/les plus longues sous-séquences communes de x et y ;
- $long_{lcs}(x, y)$, la longueur de la/les plus longues sous-séquences communes de x et y ;
- $LCS(x, y)$, une plus longue sous-séquence commune de x et y.

Une première méthode pour déterminer les plus longues sous-séquences communes de x et y consiste à extraire toutes les sous-séquences de x, à tester si elles sont également des sous-séquences de y et à conserver les sous-séquences de longueur maximale.
Une telle approche systématique implique de nombreux calculs. En effet, la séquence x possède 2^n sous-séquences. Pour de grandes valeurs de n, cette approche n'est pas praticable et requiert un temps de traitement très important, même pour des valeurs de n peu élevées, la complexité de cette technique étant exponentielle en fonction de la longueur des séquences.

Définition 3.3.1 *Soit la séquence* $s = (s_1, s_2, ..., s_m)$ *de longueur* m*, la séquence* $p = (p_1, p_2, ..., p_n)$ *de longueur* n *est un préfixe de* s *si et seulement si* $n \leq m$ *et* $\forall i, 1 \leq i \leq n, s_i = p_i$*.*

Pour être à même d'extraire les plus longues sous-séquences communes de x et y, nous utilisons une approche par programmation dynamique, d'une complexité de $O(mn)$, praticable en temps de traitement et en espace mémoire pour des valeurs de m et de n raisonnables (ce qui est le cas dans le cadre de notre travail). Le principe est ici de calculer les longueurs des plus longues sous-séquences communes sur les préfixes de x et y en considérant des préfixes de plus en plus longs. Une fois l'ensemble des préfixes testé, une méthode de recherche arrière (trace back) permet d'extraire les plus longues sous-séquences communes.

Pour cela, nous définissons une table T à deux dimensions de taille $(m+1) \times (n+1)$ de la manière suivante :

$$T[i, 0] = 0, 0 \leq i \leq m\,;$$

$$T[0, j] = 0, 0 \leq j \leq n\,;$$

$$T[i, j] = long_{lcs}((x_1, x_2, ..., x_i), (y_1, y_2, ..., y_j)), 1 \leq i \leq m, 1 \leq j \leq n.$$

Le calcul de $long_{lcs}(x, y) = T[m, n]$ repose sur une simple observation conduisant à la formule de récurrence suivante :

$$T[i, j] = \begin{cases} T[i-1, j-1] + 1 & si\ x_i = y_j \\ \max(T[i-1, j], T[i, j-1]) & sinon \end{cases} \tag{3.2}$$

Cette formule s'explique par la démonstration suivante :

Démonstration 3.3.2 *Soient*

– $z \in ens_{lcs}((x_1, x_2, ..., x_{i-1}), (y_1, y_2, ..., y_{j-1}))$,

- $z' \in ens_{lcs}((x_1, x_2, ..., x_{i-1}), (y_1, y_2, ..., y_j))$,
- $z'' \in ens_{lcs}((x_1, x_2, ..., x_i), (y_1, y_2, ..., y_{j-1}))$.

z est une plus longue sous-séquence commune de $(x_1, x_2, ..., x_{i-1})$ et $(y_1, y_2, ..., y_{j-1})$. z' est une plus longue sous-séquence commune de $(x_1, x_2, ..., x_{i-1})$ et $(y_1, y_2, ..., y_j)$. z'' est une plus longue sous-séquence commune de $(x_1, x_2, ..., x_i)$ et $(y_1, y_2, ..., y_{j-1})$.

Soit $|z|$ la longueur de la séquence z,
- $|z| = T[i-1, j-1]$,
- $|z'| = T[i-1, j]$,
- $|z''| = T[i, j-1]$.

Si $x_i = y_j$,
- $T[i, j] = 1 + T[i-1, j-1]$ *et*
- zx_i *est une plus longue sous-séquence commune de $(x_1, x_2, ..., x_i)$ et $(y_1, y_2, ..., y_j)$.*

Si $x_i \neq y_j$, Considérons les deux cas suivants :
- *Si $T[i-1, j] \geq T[i, j-1]$, alors $T[i, j] = T[i-1, j]$ et z' qui est une plus longue sous-séquence commune de $(x_1, x_2, ..., x_{i-1})$ et $(y_1, y_2, ..., y_j)$ de longueur $T[i-1, j]$ est aussi une plus longue sous-séquence commune de $(x_1, x_2, ..., x_i)$ et $(y_1, y_2, ..., y_j)$ de longueur $T[i, j]$.*

- *Si $T[i-1, j] < T[i, j-1]$, alors $T[i, j] = T[i, j-1]$ et z'' qui est une plus longue sous-séquence commune de $(x_1, x_2, ..., x_i)$ et $(y_1, y_2, ..., y_{j-1})$ de longueur $T[i, j-1]$ est aussi une plus longue sous-séquence commune de $(x_1, x_2, ..., x_i)$ et $(y_1, y_2, ..., y_j)$ de longueur $T[i, j]$.*

Cette formule est utilisée par l'algorithme `Longest-Common-Subsequence` de la table (3.1) pour calculer toutes les valeurs de T. Ce calcul est de complexité $O(mn)$.

```
fonction Longest-Common-Subsequence(x,m,y,n)
   for i = 0..m do
      T[i,0] := 0
   od
   for j = 0..n do
      T[0,j] := 0
   od
   for i = 1..m do
      for j = 1..n do
         if xi = yj then
            T[i,j] := T[i-1,j-1]+1
         else
            T[i,j] := max(T[i,j-1],T[i-1,j])
         end if
      od
   od
   return T
end fonction
```

TAB. 3.1 – Fonction `Longest-Common-Subsequence`

Pour extraire la/les plus longues sous-séquences communes de x et y, nous utilisons l'algorithme `Trace-Back` de la table (3.2).

3.3.4 Exemple

Soient les séquences :

- $x = (A, G, C, G, A)$ et
- $y = (C, A, G, A, T, A, G, A, G)$,

nous pouvons construire la table T suivante :

```
fonction Trace-Back(x,m,y,n,T)
   i := m
   j := n
   k := T[m,n]
   while i > 0 and j > 0 do
      if T[i,j] = T[i-1,j-1]+1 and xi = yj then
         w[k] = xi
         i = i-1
         j = j-1
         k = k-1
      elseif T[i-1,j] > T[i,j-1] then
         i = i-1
      else
         j = j-1
      end if
   od
   return w
end fonction
```

TAB. 3.2 – Fonction `Trace-Back`

		0	1	2	3	4	5	6	7	8	9
			C	A	G	A	T	A	G	A	G
0		0	0	0	0	0	0	0	0	0	0
1	A	0	0	1	1	1	1	1	1	1	1
2	G	0	0	1	2	2	2	2	2	2	2
3	C	0	1	1	2	2	2	2	2	2	2
4	G	0	1	1	2	2	2	2	3	3	3
5	A	0	1	2	2	3	3	3	3	4	4

La trace arrière sur la table T donne les chemins suivants :

Finalement, nous obtenons :

– $LCS(x, y) = (A, G, G, A)$ et

– $long_{lcs}(x, y) = 4$.

		0	1	2	3	4	5	6	7	8	9
			C	A	G	A	T	A	G	A	G
0		0	0	0	0	0	0	0	0	0	0
1	A	0	0	1	1	1	1	1	1	1	1
2	G	0	0	1	2	2	2	2	2	2	2
3	C	0	1	1	2	2	2	2	2	2	2
4	G	0	1	1	2	2	2	2	3	3	3
5	A	0	1	2	2	3	3	3	3	4	4

Les quatre alignements correspondants sont :

- $\begin{pmatrix} - & A & G & C & - & - & - & G & A & - \\ C & A & G & - & A & T & A & G & A & G \end{pmatrix}$
- $\begin{pmatrix} - & A & G & - & C & - & - & G & A & - \\ C & A & G & A & - & T & A & G & A & G \end{pmatrix}$
- $\begin{pmatrix} - & A & G & - & - & C & - & G & A & - \\ C & A & G & A & T & - & A & G & A & G \end{pmatrix}$
- $\begin{pmatrix} - & A & G & - & - & - & C & G & A & - \\ C & A & G & A & T & A & - & G & A & G \end{pmatrix}$

Bibliographie

[Bré98] P. Brémaud : Markov chains. Gibbs fields, Monte Carlo simulation, and queues. *Springer texts in applied mathematics. Springer*, 1998.

[Chu67] K. Chung : Markov chains with stationary transition probabilities. *Deuxième édition. Springer-Verlag, Berlin-Heidelberg-New York*, 1967.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest et C. Stein : Introduction to algorithms. *Deuxième édition, MIT press*, 2001.

[Duf97] M. Duflo : Random iterative models. *Applications of Mathematics. Springer, Berlin*, 34, 1997.

[FS02] E. Feinberg et A. Schwartz : Handbook of Markov decision processes. Methods and application. *International series in operation research & management science. Kluwer Academic Publishers. Boston, Massachussets, USA*, 2002.

[GS97] C. M. Grinstead et J. L. Snell : Introduction to probability. *American mathematical society. ISBN :978-0821807491*, 1997.

[MPB98] L. Malziak, P. Priouret et P. Baldi : Martingales et chaînes de Markov. *Hermann*, 1998.

[Yca02] B. Ycart : Modèles et algorithmes markoviens. *Mathématiques & Applications. Springer, Berlin*, 39, 2002.

Chapitre 4

Modélisation des comportements

Sommaire

4.1 Une modélisation à plusieurs niveaux

Dans ce chapitre, nous allons présenter ce que sont les comportements vidéo et la manière dont nous les modélisons. En ayant les objectifs de proposer des modèles de comportements synthétiques contenant l'ensemble des informations utiles à une analyse (automatique ou manuelle), aucun modèle usuel (séquences, motifs fréquents, vecteurs, modèles de Markov) n'était directement applicable à notre propos. En effet, la complexité de ces comportements réside principalement dans leur lecture qui peut être faite à différents niveaux. Nous en avons défini deux :

- le niveau inter-vidéo
- le niveau intra-vidéo

4.1.1 Pourquoi plusieurs niveaux ?

Nous avons donc défini un comportement basé sur plusieurs sous-représentations distinctes : les niveaux inter-vidéo et intra-vidéo. Le niveau inter-vidéo regroupe l'ensemble des accès réalisés par un utilisateur pour naviguer d'une vidéo à l'autre. Cela correspond simplement à la séquence des identifiants des vidéos visionnées. À ces données, peuvent s'ajouter des informations relatives aux requêtes et aux résultats des recherches si l'on se place dans le cadre d'une recherche via un moteur de recherche spécifique.

Le niveau intra-vidéo correspond à l'ensemble des actions réalisées par un utilisateur pendant le visionnage d'une séquence vidéo. Ces données représentent précisément les interactions entre un utilisateur et une vidéo et regroupent les actions d'un outil de visionnage (Lecture, Pause, Saut...).

Après avoir défini ces différents niveaux d'analyse et de représentation, deux possibilités étaient envisageables pour représenter l'intégralité du comportement des utilisateurs : regrouper l'ensemble de ces données dans un unique modèle de représentation et les analyser communément ou diviser l'analyse selon les niveaux préalablement définis et regrouper les données après les avoir analysées.

Nous avons choisi la deuxième solution pour les deux raisons majeures suivantes. Tout d'abord, il nous semblait très complexe de pouvoir analyser l'ensemble des données

simultanément. En effet, chacun des niveaux est complexe et requiert des techniques d'analyse très différentes. Pouvoir extraire des comportements type ou des archétypes de navigation de cet ensemble hétérogène de données ne nous a pas paru praticable tant en terme de complexité algorithmique qu'en terme de qualité des résultats extraits. Une approche en plusieurs étapes nous a permis de diviser les analyses et de les réaliser les unes à la suite des autres, chacune s'appuyant sur les résultats de la précédente. Ainsi, l'analyse inter-vidéo s'appuie en partie sur les résultats de l'analyse intra-vidéo. De manière intuitive, il est finalement assez évident de ne pas « additionner des carottes et des navets » et donc de ne pas mélanger des données issues de différents niveaux d'analyse, données totalement hétérogènes et incomparables entre elles.

Ensuite, pouvoir obtenir des analyses intermédiaires à différents niveaux est un autre atout de notre approche. En effet, l'objectif final reste l'extraction de comportements type globaux sur l'exploitation d'une base de données vidéo. Cependant, nous nous sommes aperçu que réaliser des analyses intermédiaires, notamment une analyse uniquement intra-vidéo était très intéressant et nous permettait d'extraire des données pertinentes sur l'exploitation des vidéos. Nous avons dans cet objectif développé un outil statistique d'analyse de l'exploitation des vidéos se basant sur l'analyse des comportements intra-vidéo (Annexe A.4).

Dans la suite de ce chapitre, nous détaillons la modélisation des niveaux intra-vidéo et inter-vidéo.

4.2 Modélisation intra-vidéo

4.2.1 Objectifs

Comme nous l'avons écrit précédemment, l'objectif de la modélisation intra-vidéo est double. Elle doit à la fois s'inscrire dans le processus d'analyse globale du comportement et proposer des résultats intéressants en tant que tels. Dans le cadre de l'analyse globale, nous devons être capables de synthétiser les informations fournies par chacun des comportements pour ensuite affiner l'analyse inter-vidéo. Concernant l'analyse intra-vidéo précisément, nous désirons pouvoir représenter les comportements de manière synthétique et la plus complète possible. L'enjeu est en fait de pouvoir conserver le maximum d'informations pertinentes sur le comportement, en ne conservant pas les données que nous n'exploitons pas tout en proposant un modèle le plus synthétique possible. De plus ce modèle doit répondre aux besoins suivants :

- le modèle doit être exploitable en terme algorithmique. C'est à dire que nous devons pouvoir, par exemple, extraire des comportements type d'un ensemble de visionnages à l'aide d'outils informatiques en un temps raisonnable.
- Les modèles - notamment les modèles représentant les comportements type - doivent être facilement compréhensibles par un expert vidéo, n'ayant pas de notions algorithmiques ou informatiques particulières. En effet, l'une de nos principales idées est de proposer des outils directement utilisables par les professionnels de l'audiovisuel, sans limites liées à la complexité des représentations.

4.2.2 Le modèle de Markov

Nous avons donc choisi de baser notre modélisation sur la représentation des comportements vidéo par les modèles de Markov du premier ordre [Bré98]. Cette représentation est intéressante car son caractère stochastique sied bien au côté imprévisible du comportement d'un utilisateur et permet de synthétiser un ensemble d'actions dans un modèle de taille réduite. De plus, un modèle de Markov du premier ordre est un modèle aisément compréhensible par un non-spécialiste. Ces modèles sont définis comme des graphes fi-

nis dans lesquels les sommets représentent les actions des utilisateurs et les transitions entre ces sommets sont labellisées par les probabilités de transitions entre les actions, probabilités estimées à partir des comportements des utilisateurs.

4.2.2.1 Du visionnage au modèle de Markov

Un visionnage est initialement représenté par la séquence des actions réalisées par l'utilisateur lors de la lecture de la vidéo (lecture, pause, saut) et la durée de chacune de ces actions. Par exemple, un utilisateur a pu regarder le début d'une vidéo pendant dix secondes, puis faire une pause d'une durée de cinq secondes, puis une avance rapide de quinze secondes avant de lire à nouveau vingt secondes de la vidéo. Sous forme séquentielle initiale, ce comportement sera représenté par :

((PLAY, 10), (PAUSE, 5), (FFW, 15), (PLAY, 20)).

Traiter les données sous cette forme n'est pas la meilleure approche. En effet, les techniques de comparaisons de séquences sont coûteuses quelle que soit l'approche choisie (alignement de séquences [WZ02, HWV01, CSS00], extraction de la plus longue sous-séquence commune [BG01]). C'est pour cela que nous allons modéliser les données à l'aide de modèles de Markov qui sont, comme nous l'avons écrit précédemment, mieux adaptés à notre problématique pour ce niveau de modélisation. Dans notre cas, les séquences sont relativement simples. En effet, nous ne considérons que six actions différentes dans notre outil de visionnage :

- Lecture (PLAY)
- Pause (PAUSE)
- Stop (STOP)
- avance rapide (FFW)
- retour rapide (RWD)
- Saut (JUMP)

Ce nombre limité de possibilités nous permet de représenter les comportements par des modèles de Markov avec un nombre limité d'états, ceci correspondant à nos besoins initiaux qui sont une complexité limitée et une bonne lisibilité.

Il est important de noter qu'il n'y a pas équivalence entre une séquence et le modèle de Markov la représentant. En effet, ce dernier induit une généralisation des données. Une séquence donnera toujours le même modèle de Markov mais un modèle de Markov peut donner une infinité de comportements présentant des similarités. En pratique, cette généralisation n'a pas été pénalisante car, comme nous le verrons pas la suite, nous cherchons à découvrir des modèles de comportements qui sont, par définition, des généralisations des comportements.

Partant de ces principes, la figure (4.1) représente les comportements suivants :

1. ((PLAY, 10), (PAUSE, 5), (PLAY, 20), STOP)
2. ((PLAY, 200), STOP)
3. ((PLAY, 10), STOP)
4. ((PLAY, 10), (PAUSE, 2), (FFW, 10), (JUMP, 1), (PLAY, 2), (JUMP, 1), (PLAY, 2), (RWD, 4), (PLAY, 20), STOP)

En observant ces modèles de Markov, nous pouvons noter deux spécificités :

– leur extrême simplicité,
– les modèles 2 et 3 sont totalement identiques.

Il semble ici évident que cette représentation est trop simpliste et ne permet pas une analyse pertinente des comportements. Ceci est dû à la non prise en compte du temps passé dans chacun des états. De ce fait, alors que le comportement du modèle 2 correspond à un temps de lecture de deux cents secondes et celui du modèle 3 de dix secondes, les deux représentations sont identiques, alors que les comportements sont très différents (un rapport entre les temps de lecture de un pour vingt). Pour compléter la modélisation des comportements, il nous faut maintenant intégrer la notion temporelle au modèle.

4.2.2.2 Gestion du temps

Le choix le plus naturel et le plus largement utilisé dans la littérature pour représenter une telle notion est de « paramétrer » chacun des états par une fonction de prédiction qui estime le temps passé dans cet état. Pour notre modèle, nous avons estimé que ce choix était trop complexe. Notre choix s'est donc porté sur un autre type de modélisation

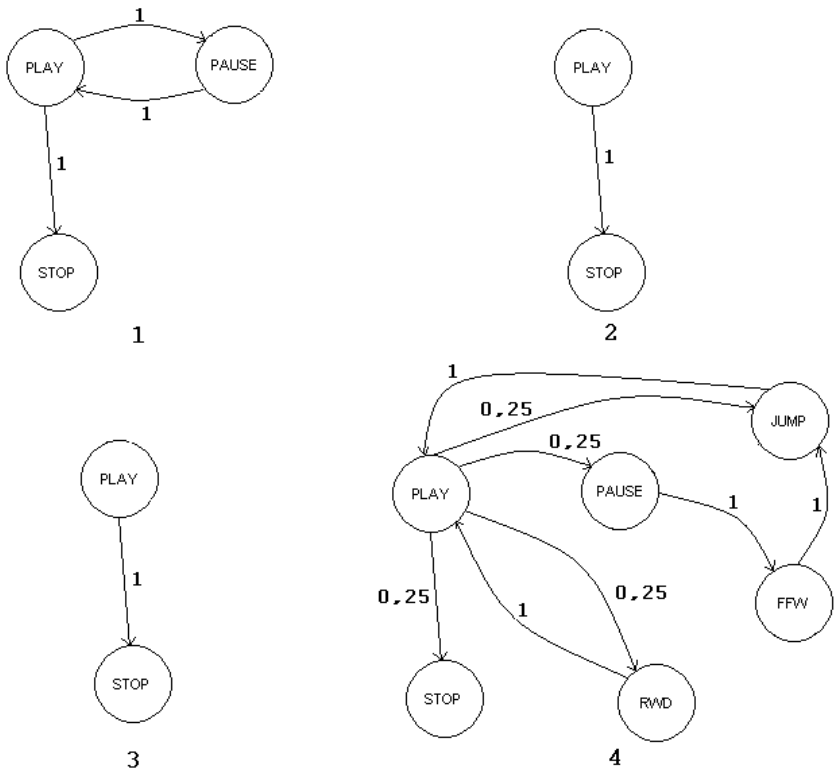


FIG. 4.1 – Modélisation élémentaire

introduit par [BET99]. Ici, les auteurs proposent de « discrétiser » et de considérer une transition d'état à état à chaque seconde du visionnage de la vidéo, même dans le cas où l'utilisateur n'a pas interagi avec l'outil de visualisation. Concrètement, si un utilisateur passe 10 secondes à lire une séquence vidéo, nous enregistrons cela comme une série de 10 transitions PLAY → PLAY dans notre modèle. Ce choix nous permet donc de prendre en compte le temps passé dans chacun des états par l'utilisateur sans ajouter de paramètre ou d'état supplémentaire au modèle. Notons enfin que nous avons découpé le temps en définissant la seconde comme unité de base. Plusieurs séries de tests nous ont montré que définir une unité plus petite comme la milliseconde n'apportait pas de précision supplémentaire. Cela est dû au fait qu'un utilisateur quelconque ne réagit pas à l'échelle de la milliseconde mais est plus proche de l'échelle de la seconde. La figure (4.2) reprend les modèles précédents en ajoutant la notion de temps passé dans chaque état :

1. ((PLAY, 10), (PAUSE, 5), (PLAY, 20), STOP)
2. ((PLAY, 200), STOP)
3. ((PLAY, 10), STOP)
4. ((PLAY, 10), (PAUSE, 2), (FFW, 10), (JUMP, 1), (PLAY, 2), (JUMP, 1), (PLAY, 2), (RWD, 4), (PLAY, 20), STOP)

Nous voyons donc ici toute l'information portée par le temps. Nous pouvons notamment noter que, maintenant, les modèles 2 et 3 sont différents, tout comme le sont les comportements qu'ils représentent.

Notre classe de modèles de Markov de représentation des comportements est donc finalement déterminée par les paramètres suivants pour chaque modèle :

V_i les sommets, $i \in N$, N étant l'ensemble des actions proposées lors d'un visionnage. Nous avons défini $N = \{PLAY, PAUSE, JUMP, FFW, RWD, STOP\}$,

A_{ij} la probabilité de passer d'un état V_i à un état V_j la seconde suivante, $i, j \in N$,

π_i la probabilité de démarrer dans l'état i. Dans notre cas, ce paramètre n'est pas significatif. En effet, quels que soit l'utilisateur considéré et la vidéo visionnée, son

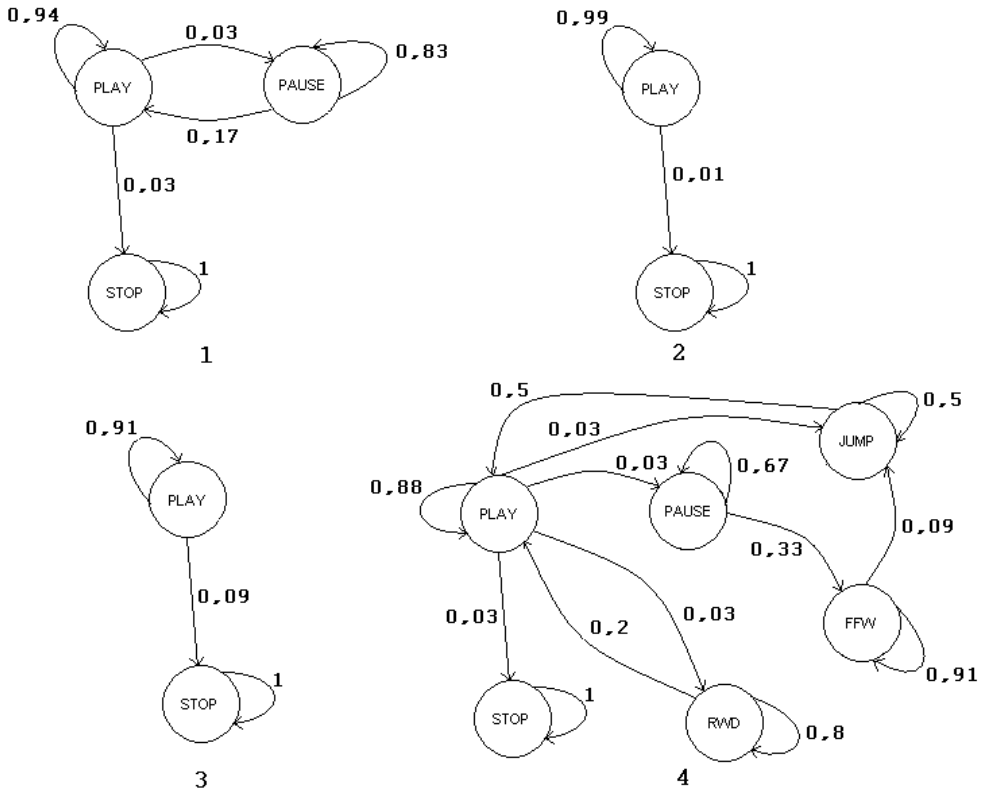


FIG. 4.2 – Modélisation avec prise en compte du temps

ouverture se fait toujours en $PLAY$. Nous avons donc $\pi_{PLAY} = 1$ et $\pi_x = 0, x \in N - \{PLAY\}$.

4.2.2.3 Algorithme de conversion

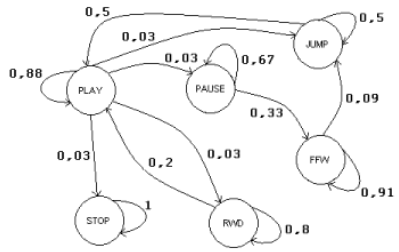
Nous allons maintenant analyser comment nous avons généré ces modèles à partir des séquences des actions des utilisateurs. Le principe est assez simple : pour chaque action enregistrée dans la séquence, nous ajoutons le temps passé à la variable *trans* liée à la transition $(Action, Action)$ et à la variable *act* allouée à l'état $(Action)$. Par exemple, si un utilisateur passe dix secondes en lecture, nous ajoutons 10 à la variable *trans* de la transition $(PLAY, PLAY)$ et à la variable *act* de l'état $(PLAY)$. Pour chaque transition dans la séquence, nous ajoutons 1 à la variable *trans* de la transition $(ActionDepart, ActionArrivee)$ et à la variable *act* de l'état $(ActionDepart)$. Par exemple quand un utilisateur effectue une pause alors qu'il lisait une vidéo, nous ajoutons 1 à la variable *trans* de $(PLAY, PAUSE)$ et à la variable *act* de $(PLAY)$. Ensuite, nous divisons toutes les valeurs *trans* de $(Action1, Action2)$ par les valeurs *act* de $(Action1)$ respectives pour obtenir les probabilités de transitions finales.

De manière formelle, soit $N = \{PLAY, PAUSE, JUMP, FFD, RWD, STOP\}$ l'ensemble des actions proposées. Soit $S = ((s_1, c_1), ...(s_i, c_i), ...(s_l, c_l))$ une suite d'actions réalisées par un utilisateur lors d'un visionnage telles que $\forall i, 1 \leq i \leq l, s_i \in N$ et c_i le nombre de secondes passées dans l'état s_i. Nous définissons le modèle $M = (m)_{ij}$, où i et j appartiennent à l'ensemble des actions N réalisables par un utilisateur, relatif à S de la manière suivante :

1. pour tout $i, j \in N$, $m_{ij} \rightarrow 0$ (initialisation)
2. pour $k = 1..(l-1)$, $m_{s_k s_{k+1}} \rightarrow m_{s_k s_{k+1}} + 1$
3. pour tout $i, j \in N$ tels que $m_{ij} \neq 0$, $m_{ij} \rightarrow \frac{m_{ij}}{\sum_{k \in N} m_{ik}}$
4. pour tout $i \in N$ tel que $\sum_{k \in N} m_{ik} = 0$, $m_{ii} \rightarrow 1$

Si nous reprenons l'exemple du modèle 4 précédent :

– ((PLAY, 10), (PAUSE, 2), (FFW, 10), (JUMP, 1), (PLAY, 2), (JUMP, 1), (PLAY, 2), (RWD, 4), (PLAY, 20), STOP)



	PLAY	PAUSE	JUMP	FWD	RWD	STOP
PLAY	0,88	0,03	0,03	0	0,03	0,03
PAUSE	0	0,67	0	0,33	0	0
JUMP	0,5	0	0,5	0	0	0
FWD	0	0	0,09	0,91	0	0
RWD	0,2	0	0	0	0,8	0
STOP	0	0	0	0	0	1

FIG. 4.3 – Le modèle final et sa matrice stochastique

Nous pouvons obtenir le modèle de (4.3) qui représente le comportement sous forme graphique et sous la forme de sa matrice de transition M. Il est ici très intéressant de noter la compacité de cette représentation qui réduit un visionnage à une matrice stochastique de taille 6×6.

4.3 Modélisation inter-vidéo

4.3.1 Objectifs

Après avoir étudié et modélisé le comportement utilisateur intra-vidéo, nous allons maintenant nous intéresser au comportement inter-vidéo et à sa modélisation. Ce niveau inter-vidéo regroupe l'ensemble des actions réalisées par un utilisateur pour naviguer d'une vidéo à l'autre. Cela correspond à la séquence des identifiants des vidéos visionnées. Pour compléter notre modélisation des utilisateurs, il est nécessaire de représenter leur comportement global sur l'ensemble d'une session. Quel que soit le moyen d'accès aux données (moteur de recherche, accès direct aux vidéos via une liste, recherche directe dans une base de données), nous définissons une session comme étant l'ensemble des visionnages de séquences vidéo effectués par un utilisateur lors d'une visite. En pratique, cette définition est similaire à la définition qui en est le plus souvent faite dans le domaine du web mining. Une session utilisateur correspondant à l'ensemble des données (pages ou vidéos) accédées successivement dans un laps de temps court et/ou après une unique identification de l'utilisateur sur le site ou l'outil de recherche si celle-ci existe.

Notre principal objectif est ici de représenter chaque session individuellement et surtout de définir un modèle capable de représenter synthétiquement et de manière la plus complète des groupes de sessions. Notre philosophie générale reste la même que pour les comportements intra-vidéo, c'est à dire que nous voulons définir des modèles à la fois exploitables et aisément compréhensibles (4.2.1).

4.3.2 Représentation séquentielle des sessions

La première étape de la modélisation a été de définir comment conserver les informations relatives aux sessions individuellement. Nous n'étudions pas ici le lien entre les comportements intra-vidéo et inter-vidéo, mais nous nous limitons au niveau inter-vidéo.

Les spécificités relatives à l'exploitation du niveau intra-vidéo sont développées à la fin de ce chapitre (4.3.4).

Les deux représentations les plus répandues dans la littérature pour représenter de

telles données sont les suivantes :

la séquence. La séquence est, comme son nom l'indique, la représentation de données de manière séquentielle. Dans le cas de données temporelles ou ayant toute autre relation de précédence, on utilise la séquence pour représenter une suite d'évènements. On utilise notamment cette représentation en bio-informatique, en data mining et en web mining. Dans notre cas, une session pourra être représentée par la séquence des différents visionnages de l'utilisateur.

L'itemset. Un itemset est un ensemble d'éléments de même type. Contrairement à la séquence, l'itemset ne prend pas en compte l'ordre d'apparition des évènements. On utilise par exemple souvent les itemsets en data mining pour représenter le panier d'achat d'un consommateur. Dans notre cas, une session pourra être représentée par l'ensemble des différents visionnages de l'utilisateur.

Nous avons finalement opté pour une représentation séquentielle des sessions. En effet, l'un de nos objectifs applicatifs est de pouvoir analyser la qualité d'indexation de vidéos accessibles via un moteur de recherche et d'en optimiser les résultats. Dans ce but, considérons un ensemble d'utilisateurs ayant recherché des vidéos représentant des paysages de montagne selon les mots-clés « paysage » et « montagne ». La liste des vidéos correspondantes est retournée selon un ordre défini par les données d'indexation basées sur le contenu. Il est possible que la vidéo correspondant le mieux aux notions de paysage et de montagne ressenties par les utilisateurs ne soit pas la première vidéo retournée.

À l'aide de cette représentation séquentielle et de l'analyse intra-vidéo, nous sommes à même de découvrir la vidéo la plus adéquate et de la proposer en priorité aux futurs utilisateurs (Chapitre 7).

4.3.3 Un modèle à séquences multiples

4.3.3.1 Approche classique

La plupart des techniques de modélisation séquentielle cherchent, pour représenter de manière concise un ensemble de séquences, à en extraire un motif ou une sous-séquence

représentative. En pratique, cela revient généralement à en extraire la plus longue sous-séquence commune (notée LCS) que nous avons présentée en détails dans la section (3.3). L'utilisation de cette sous-séquence est très intéressante car elle permet tout d'abord de comparer la proximité entre deux séquences. En effet, étant données deux séquences s_1 et s_2 et une fonctions $length(s)$ calculant la longueur d'une séquence, la fonction d_{seqlcs} est définie telle que

$$d_{seqlcs}(s_1, s_2) = 1 - \frac{length(LCS(s_1, s_2))}{max(length(s_1), length(s_2))} \tag{4.1}$$

De nombreuses techniques de regroupement utilisent une distance similaire et permettent l'extraction de classes de séquences intéressantes. Leur principe est simple, on fusionne de manière hiérarchique les séquences les plus proches selon d_{seq} et on les représente alors par leur LCS. Étant données les trois séquences suivantes :

- $(B, A, X, V, G, K, T, H, C)$
- $(X, B, A, V, E, G, K, T, Y, C)$
- (A, K, B, C, G, T, U)

l'arbre des fusions successives est donné par la figure (4.4).

Finalement, le groupe de ces trois séquences sera représenté par la session (A, K, C) après le regroupement hiérarchique.

Nous pouvons de suite mettre en avant l'une des limites de cette technique. En effet, lors de la dernière étape de fusion, la séquence (A, K, C) a été extraite des séquences (B, A, V, G, K, T, C) et (A, K, B, C, G, T, U) comme étant une plus longue sous-séquence commune. C'est en effet le cas mais cette sous-séquence n'est pas unique. La séquence (B, G, T) est également une plus longue sous-séquence commune des deux précédentes. Le choix de conserver l'une ou l'autre des séquences est aléatoire et induit une forte perte d'information sur les éléments composant les groupes de séquences.

Un autre cas qui peut se présenter est celui de deux séquences composées de deux sous-séquences de longueur importante qui s'entrecroisent. Seule la plus longue est conservée. L'information portée par l'autre sous-séquence est perdue à cette étape de fusion alors qu'elle pourrait être intéressante dans une étape ultérieure. Soient les séquences $s_1 =$

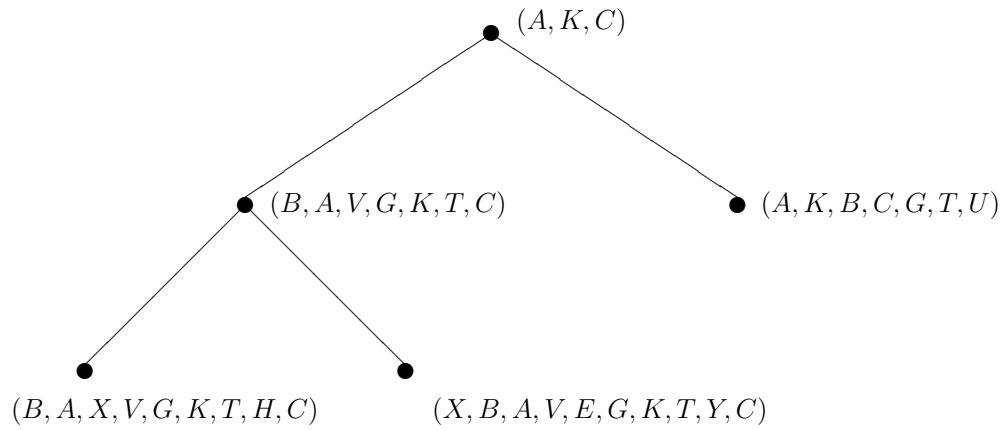


FIG. 4.4 – Arbre de regroupement

$(A, D, G, V, X, L, P, M, Q, S, W)$ et $s_2 = (A, P, D, M, G, Q, V, S, X, W, L)$ les deux sous-séquences $ss_1 = (A, D, G, V, X, L)$ et $ss_2 = (P, M, Q, S, W)$ appartiennent aux séquences initiales. Elles sont toutes deux de longueur importante : $length(ss_1) = \frac{6}{11} \times length(s_1)$ et $length(ss_2) = \frac{5}{11} \times length(s_1)$. Pourtant seulement ss_1 est conservée lors d'une fusion entre les deux séquences.

4.3.3.2 Notre modèle

Pour parer les limites de la représentation par une sous-séquence des classes de séquences tout en conservant les avantages, nous proposons un modèle également basé sur l'extraction de sous-séquences communes. Cependant, notre modèle ne représente plus les classes par une unique séquence mais par un ensemble de séquences extraites de ses éléments. Concrètement, un ensemble de séquences est représenté par une, deux, trois ou quatre sous-séquences extraites. Si l'on reprend l'étude des trois séquences précédentes :

– $(B, A, X, V, G, K, T, H, C)$
– $(X, B, A, V, E, G, K, T, H, Y, C)$
– (A, K, B, C, G, T, U)

et en conservant une approche hiérarchique par fusions successives, nous obtenons l'arbre de la figure (4.5).

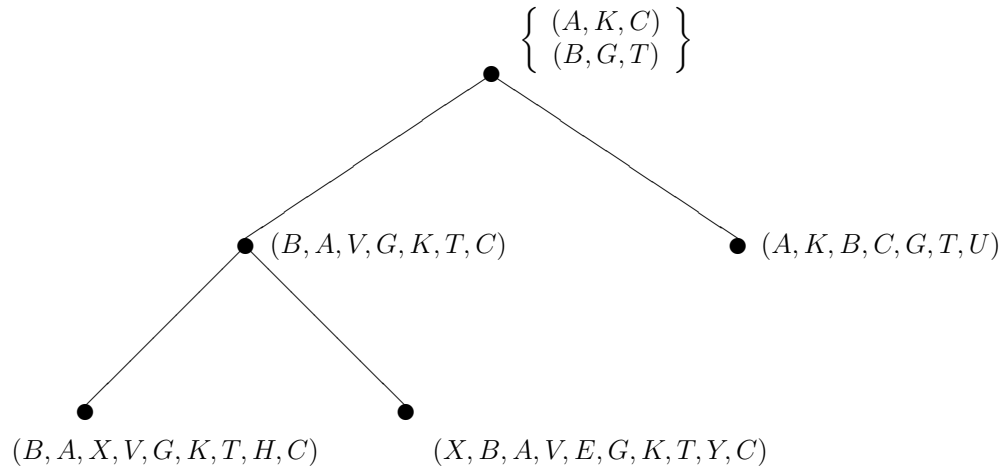


FIG. 4.5 – Arbre de regroupement

Dans les cas où plusieurs sous-séquences sont représentatives (elles sont toutes plus longue sous-séquence commune ou toutes d'une longueur intéressante relativement à la taille des séquences initiales), nous les conservons car elles sont porteuses d'une information nécessaire pour représenter finement les classes de séquences.

4.3.3.3 Paramétrage du modèle

Pour exploiter ces modèles et obtenir des résultats intéressants à analyser, il est nécessaire qu'ils correspondent aux besoins des analystes. Pour certaines applications, il

faut extraire des données très précises mais, dans d'autres cas, on cherche en revanche des modèles génériques. Nous proposons donc trois paramètres pour permettre de répondre à ces besoins :

- la taille minimale des sous-séquences représentatives (noté l),
- le nombre minimal de sous-séquences pour avoir un modèle intéressant (noté r),
- le pourcentage minimal de représentativité de chaque sous-séquence (noté *pct*).

Les deux premiers paramètres sont assez explicites. Ils permettent de définir la taille des modèles et la longueur de leurs sous-séquences. Un modèle avec un nombre élevé de séquences représentatives de longueur importante sera très proche des éléments qu'il représente mais n'en englobe pas beaucoup alors qu'un modèle avec moins de sous-séquences d'une longueur inférieure est plus général et moins discriminant. En pratique et pour ne pas avoir à sectionner les sous-séquences plus longues que l, nous conservons toujours les plus longues sous-séquences extraites. Alors, si nous avons une unique sous-séquence de longueur $3 \times l$ pour modéliser un groupe de séquences et un paramètre $r = 3$, le modèle est valide (nous pouvons découper la sous-séquence représentative en r sous-séquences de longueur minimale l). En fonction de l'application et des besoins de précision / généralisation, les paramètres sont adaptés.

Le dernier paramètre - pourcentage minimal de représentativité de chaque sous-séquence *pct* - permet d'ajouter plus de flexibilité aux modèles. Nous verrons également qu'il apporte une réelle plus-value lors de la génération des modèles type par regroupement au chapitre (6). Étant données n séquences que nous devons représenter par un unique modèle, la sous-séquence s correspond au modèle si et seulement si elle respecte les paramètres l et r et si elle est une sous-séquence de $pct \times n$ des n séquences. Par exemple, si nous considérons à nouveau les trois séquences :

- $(B, A, X, V, G, K, T, H, C)$
- $(X, B, A, V, E, G, K, T, H, Y, C)$
- (A, K, B, C, G, T, U),

pour une valeur $\frac{1}{3} \leq p < \frac{2}{3}$ et une longueur minimale quelconque, chacune des séquences peut appartenir au modèle. Pour une valeur $\frac{2}{3} \leq p < 1$, toute sous-séquence commune à deux des trois séquences peut être conservée, c'est à dire par exemple (B, A, V, G, K, T, C)

ou (B, G, T). Si $p = 1$, seule les sous-séquences communes à toutes les séquences sont candidates. Dans l'exemple, cela correspond aux séquences (B, G, T) et (A, K, C).

Une valeur élevée de p a donc tendance à produire des modèles très représentatifs alors que, pour une valeur faible, les modèles sont plus génériques.

4.3.4 Importance de l'intra-vidéo dans l'inter-vidéo

Maintenant que nous avons défini individuellement ce que sont les comportements intra-vidéo et inter-vidéo, nous allons voir comment lier ces deux niveaux de représentation et comment le modèle intra-vidéo permet d'affiner le modèle inter-vidéo.

Comme il est présenté ci-dessus et précédemment dans la section (3.3), la comparaison entre les séquences s'appuie sur une comparaison binaire des éléments qui les composent. Soit nous retrouvons le même élément, soit il est différent. Cette représentation ne convient pas ici à notre besoin. En effet, deux utilisateurs peuvent accéder à la même vidéo, l'un par erreur et la fermant très rapidement et l'autre la visionnant intégralement. Lorsque nous devrons comparer les comportements de ces deux utilisateurs, il sera nécessaire de faire cette distinction. Dans le cas contraire, deux comportements différents (mêmes vidéos accédées mais visionnées de manières totalement différentes) pourraient être considérés comme très proches.

Pour ce faire, nous proposons de pondérer chaque comparaison d'éléments en intégrant le comportement intra-vidéo. Chacun des éléments d'une séquence n'est plus uniquement un identifiant vidéo mais un couple $(identifiant,\ vecteur\ de\ comportement)$. Ce vecteur de comportement correspond à un vecteur de probabilité que la vidéo ait été visionnée selon différents comportements type. Le chapitre (5.2) présente comment ces comportements sont extraits des données.

Soient les séquences s_1 et s_2 de couple (i, v), la distance entre séquences (d_{seqlcs}) définie précédemment (équation 4.1) devient :

$$d_{seq}(s_1, s_2) = 1 - \frac{WLCS(s_1, s_2)}{max(length(s_1), length(s_2))} \tag{4.2}$$

où $WLCS(s_1, s_2)$ est une fonction dérivée du calcul de sous-séquence commune (3.3) pondéré par la proximité des éléments. En pratique, on considère lors de l'extraction d'une LCS, que deux éléments sont égaux ou non et on incrémente la taille de la longueur de la sous-séquence résultat à chaque élément découvert. Dans le cas de la recherche pondérée, chaque élément est plus ou moins proche d'un autre. C'est-à-dire que deux éléments ayant le même identifiant ne seront pas forcément égaux mais plus ou moins proches en fonction de la différence entre les comportements qui leur sont liés. Cela se traduit par les deux cas suivants :

- Si deux éléments (i_1, v_1) et (i_2, v_2) sont tels que $i_1 \neq i_2$ alors leur proximité est nulle et, comme dans le cas de la LCS, ils ne peuvent intervenir dans le calcul du résultat final.
- Si $i_1 = i_2$ alors leur proximité n'est pas nulle mais est égale à la proximité entre v_1 et v_2. Cette proximité est calculée par la fonction $sim(v_1, v_2)$. Si les vecteurs v_1 et v_2 sont de longueur k, $sim(v_1, v_2)$ est calculée selon la formule suivante :
$$sim(v_1, v_2) = 1 - \frac{\sum_{i=1}^{k} |v_{1_i} - v_{2_i}|}{k}$$

Si l'on reprend la formule de récurrence de base de la fonction LCS (3.2), celle-ci devient dans le cas de notre fonction $WLCS$:

$$T[i,j] = \begin{cases} T[i-1, j-1] + sim(v_i, v_j) & si\ x_i = y_i \\ \max\left(T[i-1, j], T[i, j-1]\right) & sinon \end{cases} \tag{4.3}$$

Ainsi, chaque élément apparaissant dans une LCS intervient en fonction de la proximité entre les différents comportements dont il est issu. Nous pouvons également noter qu'à chacun des éléments de cette LCS est associé un vecteur de comportement qui est calculé en faisant la moyenne par valeur des vecteurs de comportements des éléments des séquences source.

Considérons par exemple les trois séquences s_1, s_2 et s_3 et les trois comportements v_1, v_2 et v_3 tels que :

$$s_1 = (A_{v_1}, B_{v_1}, C_{v_1}, D_{v_1}, E_{v_3})$$
$$s_2 = (A_{v_3}, B_{v_3}, C_{v_2}, D_{v_3}, E_{v_1})$$
$$s_3 = (A_{v_1}, C_{v_2}, D_{v_2}, E_{v_3}, G_{v_3})$$

$sim.$	v_1	v_2	v_3
v_1	1	0.6	0.1
v_2	0.6	1	0.3
v_3	0.1	0.3	1

En comparant s_1 et s_2, nous obtenons une LCS de longueur 5. En tenant compte des similarités entre les comportements, nous obtenons :

$$dist_{WLCS}(s_1, s_2) = 1 - \frac{4 \times sim(v_1, v_3) + sim(v_2, v_1)}{5} = 0.8$$

Les tableaux 4.1 et 4.2 récapitulent l'ensemble des comparaisons simples et pondérées :

d_{seqlcs}	s_1	s_2	s_3
s_1	0	0	0.2
s_2	0	0	0.2
s_3	0.2	0.2	0

TAB. 4.1 – Distances non-pondérées

d_{seq}	s_1	s_2	s_3
s_1	0	0.8	0.375
s_2	0.8	0	0.65
s_3	0.375	0.65	0

TAB. 4.2 – Distances pondérées

L'intérêt de pondérer l'importance des éléments en fonction de la similarité des comportements qui leur sont liés est ici évidente. On voit en effet que dans le premier cas (4.1), les séquences s_1 et s_2 sont peu distantes alors que leurs comportements respectifs le sont. Avec les pondérations (4.2), ces deux séquences sont bien reconnues comme fortement différentes ($d_{seq}(s_1, s_2) = 0.8$). De plus, les séquences s_1 et s_3, plus proches en terme de comportements sont reconnues comme plus proches bien qu'ayant une LCS plus courte ($d_{seq}(s_1, s_3) = 0.375$).

Nous avons vu comment intégrer les deux niveaux de représentation inter-vidéo et intra-vidéo dans une seule modélisation du comportement des utilisateurs, à la fois complète et concise. Dans le chapitre suivant, nous présentons comment nous proposons d'exploiter ces modèles.

4.4 Conclusion

Dans ce chapitre, nous avons présenté un modèle de représentation du comportement des utilisateurs exploitant des données vidéo. Ce modèle a la particularité d'être multi-niveaux :

- niveau inter-vidéo - l'enchainement entre les différents visionnages,
- niveau intra-vidéo - les interactions entre le lecteur et l'utilisateur.

Ce découpage permet une analyse fine de chacun des niveaux pour en extraire un modèle exploitable en tant que tel. Le regroupement de ces deux niveaux en un unique modèle permet une analyse complète et détaillée des utilisateurs, comme nous allons le voir dans les prochains chapitres.

Un autre avantage important est qu'en scindant l'analyse en plusieurs étapes, elle devient plus précise. Il aurait été illusoire d'obtenir une analyse cohérente en traitant à la fois les données de bas niveau (intra-vidéo) et les données relatives aux sessions (inter-vidéo).

Bibliographie

[BET99] P. Branch, G. Egan et B. Tonkin : Modeling interactive behaviour of a video based multimedia system. *IEEE international conference on communications*, 2:978–982, Juin 1999.

[BG01] A. Banerjee et J. Ghosh : Clickstream clustering using weighted longest common subsequences. *Web mining workshop at the 1st SIAM conference on data mining. Chicago, Illinois, USA*, Avril 2001.

[Bré98] P. Brémaud : Markov chains. Gibbs fields, Monte Carlo simulation, and queues. *Springer texts in applied mathematics. Springer*, 1998.

[CSS00] K. Charter, J. Schaeffer et D. Szafron : Sequence alignment using FastLSA. *International conference on mathematics and engineering techniques in medicine and biological sciences*, 2000.

[HWV01] B. Hay, G. Wets et K. Vanhoof : Clustering navigation patterns on a website using a sequence alignment method. *Seventeenth international joint conference on artificial intelligence. Seattle, Washington, USA*, Août 2001.

[WZ02] W. Wang et O. R. Zaïane : Clustering web sessions by sequence alignment. *Thirteenth international workshop on database and expert systems applications. Aix-en-Provence, France*, 2002.

Chapitre 5

Analyse des comportements intra-vidéo

Sommaire

5.1 Introduction

Dans ce chapitre, nous présentons comment, à partir de l'observation d'utilisateurs recherchant des vidéos, nous pouvons modéliser leur comportement et extraire, de l'ensemble des usages, les comportements type représentant l'ensemble des utilisateurs. Ceci est très intéressant pour comprendre comment les vidéos sont exploitées. En effet, à partir des différents comportements extraits, un spécialiste de la vidéo pourra analyser l'impact des vidéos sur les utilisateurs. Imaginons trois comportements extraits et grossièrement analysés comme étant relatifs à :

- la vidéo est totalement visionnée
- la vidéo est très partiellement visionnée
- la vidéo est fermée très rapidement

Si cet expert du domaine audiovisuel remarque qu'une vidéo est visionnée avec le premier comportement dans 80% des cas, c'est certainement qu'elle correspond à l'attente et aux besoins des utilisateurs. En revanche si une autre correspond à 40% du second comportement et 50% du troisième, c'est que cette vidéo n'est pas intéressante. L'annexe (A.4) présente une application qui permet d'extraire de telles affirmations des données.

Passer par une étape de regroupement est indispensable pour analyser un ensemble de données complexes volumineux. L'étape de modélisation a permis de les rendre moins complexes, l'étape de regroupement permet de n'avoir à traiter qu'un ensemble limité de données.

Nous verrons, dans la suite de ce chapitre, deux techniques de regroupement que nous avons définies sur la base de la modélisation présentée dans la section (4.2) et présenterons une analyse des résultats obtenus.

5.2 Regroupement des comportements intra-vidéo avec *K-models*

5.2.1 La notion de comportement type

Dans cette section, nous allons voir une première méthode que nous avons développée pour extraire de l'ensemble des visionnages différents comportements type. Comme nous l'avons écrit précédemment, extraire ces comportements est très intéressant pour comprendre comment les vidéos sont exploitées. En effet, à partir des différents comportements extraits, un spécialiste de la vidéo pourra analyser l'impact des vidéos sur les utilisateurs.

Notre objectif est donc ici de pouvoir extraire des comportements représentatifs des données. Nous retrouvons ici la problématique des techniques de regroupement, c'est-à-dire que nous devons découvrir des comportements représentatifs de classes de visionnages, ces classes devant être les plus compactes possible (les visionnages d'une même classe doivent être les plus similaires possible) et les plus distantes possible les unes des autres (les comportements représentatifs des classes doivent être les plus différents possible). Nous espérons, après cette étape de regroupement obtenir des comportements type représentatifs des différentes façons de visionner une vidéo (lecture totale, lecture d'une partie, survol, fermeture prématurée...).

Il est donc nécessaire dans un premier temps d'extraire des modèles selon le schéma défini dans la section (4.2) et ensuite de les analyser pour les traduire en comportements réels.

5.2.2 Regroupement par les *K-models*

Nous allons ici introduire la technique des *K-models*. Cette technique correspond à une adaptation de la méthode des *K-moyennes* [Mac66] pour utiliser des modèles en lieu et place des moyennes. Nous voulons extraire k classes d'un ensemble de visionnages par partitionnement de l'espace. Chaque classe est représentée par l'un des modèles de Markov décrits précédemment dans lesquels chaque état correspond à une action réalisable

($PLAY$, $PAUSE$...) et chaque transition à la probabilité de réaliser l'action cible quand l'utilisateur se trouve dans l'action source pendant la seconde de visionnage suivante. La principale différence avec l'algorithme des *K-moyennes* réside dans l'utilisation des probabilités en lieu et place des distances pour associer les visionnages aux classes. Le principe est de calculer la probabilité qu'un visionnage ait été généré par l'un ou l'autre des modèles représentatifs des classes. Nous l'assignons ensuite à la classe ayant la plus grande probabilité de l'avoir généré et recalculons les modèles en fonction des visionnages qui ont été attribués à la classe qu'il représente. La figure (5.1) présente un aperçu général de l'algorithme. On y voit qu'un visionnage v est attribué à la classe dont le modèle M_i en est le plus proche ($P(v|M_i)$). Une fois tous les visionnages attribués, les modèles sont recalculés en fonction des classes et l'opération est répétée itérativement jusqu'à ce que les modèles soient stables d'une itération à l'autre ou qu'on ait atteint le maximum d'itérations défini.

Un tel algorithme se découpe en trois phases : initialisation, allocation, maximisation.

5.2.2.1 Initialisation

La phase d'initialisation, tout comme pour les algorithmes de la famille des *K-moyennes*, apparaît ardue. En effet, il est indispensable de définir les bons modèles pour débuter le regroupement. Un choix de modèles trop proches les uns des autres ou trop différents des classes réelles peut conduire à des résultats erronés. Notre premier choix fut de débuter le regroupement avec des modèles prédéfinis. Après avoir mené différentes expériences, nous avons défini quatre modèles comme étant les modèles les plus souvent extraits et les plus représentatifs des données. Ces modèles peuvent se traduire par les comportements suivants :

- visionnage total
- visionnage d'une scène précise
- aperçu général / survol de la vidéo
- fermeture rapide

Cependant, cette approche très statique est assez limitée et rien n'indique que ce qui était vrai pour un jeu de données le sera pour un autre ou pour d'autres ensembles de

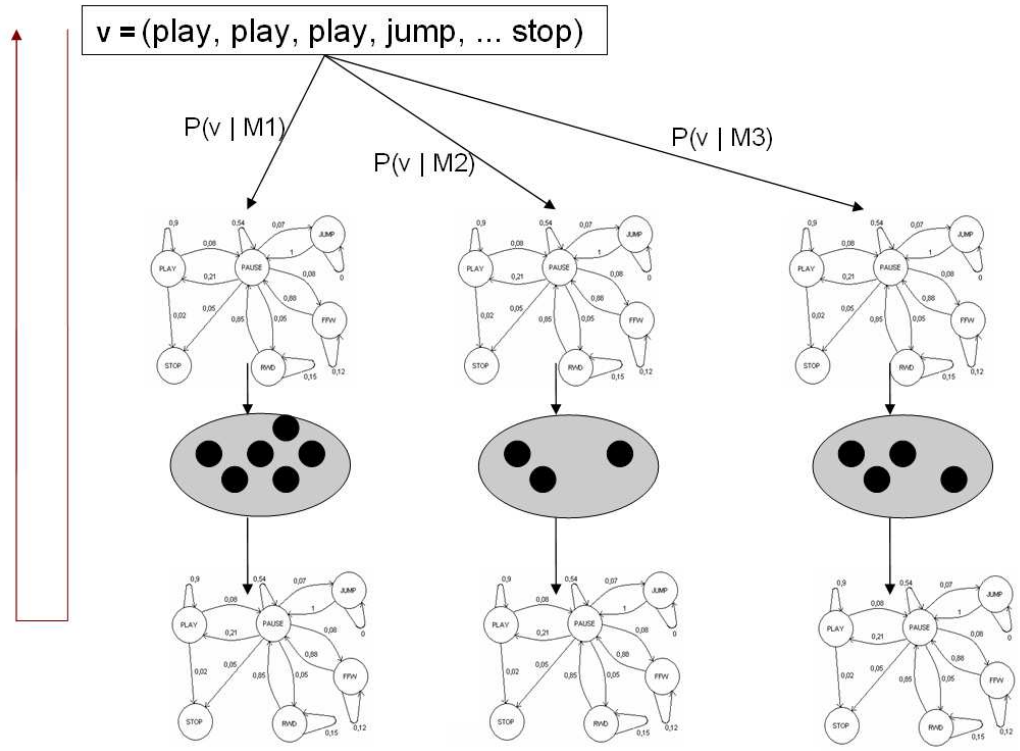


FIG. 5.1 – Aperçu de la technique des *K-models*

vidéos. Nous avons donc également testé une approche similaire à la technique initiale des *K-moyennes* en sélectionnant de manière aléatoire les modèles à partir des visionnages. Soit un visionnage $v = (v_1, v_2, ...v_l)$ composé de l actions, une action étant définie chaque seconde du visionnage et soit le modèle $M = (m)_{ij}$ où i et j appartiennent à l'ensemble des actions N réalisables par un utilisateur, nous définissons M comme le modèle relatif à v de la manière suivante :

1. pour tout $i, j \in N$, $m_{ij} \rightarrow 0$ (initialisation)
2. pour $k = 1..(l-1)$, $m_{v_k v_{k+1}} \rightarrow m_{v_k v_{k+1}} + 1$

(%)	play	pause	jump	fwd	rwd	stop
play	95	2	0	0	2	1
pause	0	75	25	0	0	0
jump	50	0	50	0	0	0
fwd	0	0	0	100	0	0
rwd	33	0	0	0	67	0
stop	0	0	0	0	0	100

TAB. 5.1 – Modèle initial généré avec v

3. pour tout $i, j \in N$ tels que $m_{ij} \neq 0$, $m_{ij} \rightarrow \frac{m_{ij}}{\sum_{k \in N} m_{ik}}$
4. pour tout $i \in N$ tel que $\sum_{k \in N} m_{ik} = 0$, $m_{ii} \rightarrow 1$

Par exemple, le visionnage v suivant :

$$((PLAY, 10), (PAUSE, 3), (JUMP, 1), (PLAY, 30), (RWD, 2), (PLAY, 20), STOP)$$

sera transformé selon le modèle de la table (5.1). Par souci d'espace, nous présentons le visionnage v sous forme d'une liste de couples $(ACTION, s)$ signifiant que $ACTION$ dure s secondes et est donc répété s fois.

Avec ce choix aléatoire des modèles initiaux, nous proposons une technique dynamique, potentiellement adaptable à tout type de données. Il reste un bémol relatif au côté aléatoire du choix. Certaines exécutions de *K-models* donnent parfois des résultats peu intéressants du fait d'une sélection initiale inadaptée. En pratique cependant, il est possible de réitérer l'exécution et il est très rare de ne pas produire des modèles cohérents avec peu d'exécutions.

5.2.2.2 Allocation

La phase d'allocation consiste en l'attribution des visionnages à l'une ou l'autre des classes en fonction des modèles de comportement qui les définissent. Pour chaque visionnage $v = (v_1, v_2...v_l)$ de longueur l tel que $\forall i, v_i \in N$, pour chaque modèle de classe $M = (m)_{ij}$ tel que $i, j \in N$, nous calculons la probabilité $P(v|M)$ que v ait été généré

par M selon la formule (5.1).

$$P(v|M) = \prod_{i=1}^{i=l-1} m_{v_i v_{i+1}} \tag{5.1}$$

Nous pouvons ici remarquer que, par la présence de valeurs nulles dans les modèles, nombre de probabilités de génération sont égales à 0. Cela ne correspond pas à la réalité. En effet, la valeur de transition 0 ne signifie pas une impossibilité de génération mais uniquement que cette transition n'a pas été observée. En pratique, une valeur nulle est remplacée par une petite valeur pour exprimer une probabilité faible car non observée mais cependant possible. Nous avons fixé cette valeur à 0,01 pour des raisons d'implémentation que nous expliquons dans la section (5.2.3.4).

Une fois ces calculs effectués pour chaque modèle M, le visionnage v est attribué au modèle offrant la valeur maximale de génération par la formule (5.2).

$$attribModel(v) = argmax_{i=1}^{i=k}(P(v|M_i)) \tag{5.2}$$

Une fois l'ensemble des visionnages traité, nous obtenons à la fin de cette phase d'allocation un ensemble de classes $C_i, i = 1..k$, représentées par les modèles M_i et composées chacune de n_i visionnages $\{v_{i1}, v_{i2}, ...v_{in_i}\}$.

5.2.2.3 Maximisation

Ayant maintenant les classes C_i et leurs visionnages associés, il est nécessaire de recalculer les modèles pour qu'ils correspondent aux visionnages. Dans le cas des *K-moyennes*, cette mise à jour est le calcul de la moyenne des points d'une classe. Pour les *K-models*, nous allons mettre à jour les modèles en calculant les probabilités de transition entre les états à partir de l'ensemble des transitions observées dans les visionnages. Soit la classe C définie par le modèle $M = (m)_{ij}$ tel que $i, j \in N$ et par l'ensemble des visionnages $\{v_1, v_2, ...v_l\}$ tel que $\forall i, v_i = (v_{i1}, v_{i2}...v_{il_i})$, le modèle M est recalculé de la manière suivante :

(%)	play	pause	jump	fwd	rwd	stop
play	96	1,5	0,5	0	1,5	0,5
pause	8	77	8	0	0	7
jump	50	0	50	0	0	0
fwd	0	0	0	100	0	0
rwd	23	0	0	0	77	0
stop	0	0	0	0	0	100

TAB. 5.2 – Modèle recalculé en fonction des affectations

1. pout tout $i, j \in N$, $m_{ij} \rightarrow 0$
2. pour tout $v_i \in C$ et pour tout $k = 1..(l_i - 1)$, $m_{v_{ik}v_{ik+1}} \rightarrow m_{v_{ik}v_{ik+1}} + 1$
3. pour tout $i, j \in N$ tels que $m_{ij} \neq 0$, $m_{ij} \rightarrow \frac{m_{ij}}{\sum_{k \in N} m_{ik}}$
4. pour tout $i \in N$ tel que $\sum_{k \in N} m_{ik} = 0$, $m_{ii} \rightarrow 1$

Par exemple, soient les trois visionnages suivants :

$$((PLAY, 14), (PAUSE, 3), (PLAY, 22), (RWD, 6), (PLAY, 28), STOP)$$
$$((PLAY, 10), (PAUSE, 3), (JUMP, 1), (PLAY, 30), (RWD, 2), (PLAY, 20), STOP)$$
$$((PLAY, 26), (JUMP, 1), (PLAY, 39), (RWD, 4), (PLAY, 11), (PAUSE, 4), STOP)$$

Le modèle recalculé à partir de ces trois points est présenté dans la table (5.2)

Si l'on considère que le modèle précédent de cette classe était celui de la table (5.1), on note une modification relativement importante sur la ligne PAUSE et un affinage léger des valeurs pour les lignes PLAY et RWD. Ainsi, en itérant le processus plusieurs fois, les modèles vont progressivement se transformer selon les modèles de comportement type.

En principe, l'algorithme s'arrête quand, d'une itération à l'autre, les modèles de chacune des classes sont similaires. En pratique, il arrive que les modèles soient légèrement modifiés à l'infini. En effet, un visionnage peut être identiquement proche de deux modèles et, dans ce cas, peut changer de classe d'une itération à l'autre. Pour éviter cela, nous avons défini un maximum d'itérations possibles fixé à 1000 pour garantir l'arrêt de l'algorithme. La valeur de 1000 a été validée par des expérimentations. Elle garantit un temps de traitement correct et une bonne qualité des résultats.

5.2.3 Représentation matricielle et divergence de *Kullback-Leibler*

5.2.3.1 Motivations

Dans la section précédente, nous avons présenté une technique de regroupement qui permet de regrouper en différents comportements type les visionnages des utilisateurs. Cette technique se base sur le calcul de la probabilité de génération d'un visionnage par un modèle de classe, le visionnage étant représenté par une séquence et la classe par un modèle de Markov.

Dans cette section, nous allons faire le lien entre la représentation matricielle des visionnages, le calcul de leur divergence et la probabilité de génération. L'objectif est double. Dans un premier temps, nous allons montrer que la probabilité de génération varie inversement à la divergence entre un visionnage (représenté sous forme matricielle) et un modèle, calculée à l'aide de la divergence de *Kullback-Leibler*. Ce lien nous permettra de mesurer la qualité du regroupement à l'aide d'indices adaptés.

Ensuite, cette représentation matricielle nous permettra d'introduire un autre modèle de représentation, extrait de cette matrice, que nous présenterons dans la section 5.3.

5.2.3.2 La divergence de *Kullback-Leibler*

C'est en 1951 qu'apparaît pour la première fois la divergence de *Kullback-Leibler* [KL51]. Dans cet article, les auteurs proposent une approche statistique de l'information. L'idée majeure de ces travaux est de ne plus définir l'information de façon absolue mais au contraire de façon relative en prenant en compte deux densités de probabilités. Récemment cette fonction a été utilisée dans différentes applications pour comparer des modèles ou mesurer la vraisemblance d'une donnée en fonction d'un modèle [RAC04, Do03, Kri93].

Dans notre cas, nous nous appuyons sur cette divergence pour comparer les visionnages et les modèles en calculant la vraisemblance d'une instance par rapport à un modèle de Markov. Ce choix a été motivé par le lien que nous avons montré entre la divergence entre un visionnage et un modèle et la probabilité de génération d'un visionnage par un modèle (section 5.2.3.3).

Définition 5.2.1 *Soient deux vecteurs u et v à n dimensions aux valeurs positives ou nulles, la divergence de Kullback-Leibler est $d_{KL}(u,v) = \sum_{i=1}^{i=n} u_i \times \log \frac{u_i}{v_i}$.*
Nous pouvons noter qu'elle n'est définie normalement que si $\forall\ i, v_i \neq 0$. En pratique une valeur nulle est remplacée par une valeur ϵ petite pour effectuer le calcul.

5.2.3.3 Propriétés de la divergence de *Kullback-Leibler*

Rappelons que les modèles de comportements sont des matrices stochastiques de taille 6×6 notées $M = (m)_{ij}$, telles que :

- $\forall i, j, 1 \leq i, j \leq 6, m_{ij} \geq 0$ et
- $\forall i, 1 \leq i \leq 6, \sum_{j=1}^{6} m_{ij} = 1$

Soit $v = (v_1, \ldots, v_l)$ un visionnage, soient $(c)_{ij}$ les nombres de transitions entre les actions observées dans le visionnage v tel que i et j appartiennent à l'ensemble des actions réalisables N, la probabilité que la séquence v ait été générée par un modèle M est $p(v|M) = \prod_{i=1}^{6} \prod_{j=1}^{6} m_{ij}^{c_{ij}}$. Cela implique :

$$\log p(v|M) = \sum_{i=1}^{6} \sum_{j=1}^{6} c_{ij} \log m_{ij}$$

Pour comparer un modèle à un visionnage, nous devons transformer ce visionnage en une matrice stochastique de même taille. Soit la matrice $(n)_i$ telle que $n_i = \sum_{j=1}^{6} c_{ij}, \forall i, 1 \leq i \leq 6$ représentant la fréquence d'apparition de l'action a_i dans le visionnage v, nous pouvons construire la matrice $(F)_{ij}$ telle que $\forall i, j, 1 \leq i, j \leq 6, f_{ij} = \frac{c_{ij}}{n_i}$ qui est la transformation du visionnage v en une matrice stochastique représentant les probabilités de transitions entre les actions de v.

Nous pouvons maintenant écrire :

$$\log p(v|M) = \sum_{i=1}^{6} n_i \sum_{j=1}^{6} f_{ij} \log m_{ij}, \forall i, j, 1 \leq i, j \leq 6.$$

Nous allons maintenant montrer un lien entre la divergence de *Kullback-Leibler* entre un visionnage et un modèle et la probabilité qu'un visionnage ait été généré par un modèle.

Soit F la matrice des probabilités de transitions extraite du visionnage v et soit $M = (m)_{ij}$ un modèle de comportement (rappelons que M et F sont des matrices sto-

chastiques). Notons f_i et m_i les lignes de ces matrices, ces dernières étant des probabilités de distribution.

Nous définissons la divergence $div_{KL}(F, M)$ entre les matrices F et M de la manière suivante :

$$div_{KL}(F, M) = \sum_{i=1}^{6} n_i \times d_{KL}(f_i, m_i)$$

Théorème 5.2.2 *La quantité $div_{KL}(F, M) + \log p(v|M)$ est constante pour un visionnage v donné. Elle ne varie pas en fonction de M.*

Preuve. En partant de la définition de $div_{KL}(F, M)$ nous pouvons écrire :

$$\begin{aligned}
div_{KL}(F, M) &= \sum_{i=1}^{6} n_i \; d_{KL}(f_i, m_i) \\
&= \sum_{i=1}^{6}\sum_{j=1}^{6} n_i f_{ij} \log \frac{f_{ij}}{m_{ij}} \\
&= \sum_{i=1}^{6}\sum_{j=1}^{6} n_i f_{ij} \log f_{ij} - \sum_{i=1}^{6}\sum_{j=1}^{6} n_i f_{ij} \log m_{ij} \\
&= \sum_{i=1}^{6}\sum_{j=1}^{6} n_i f_{ij} \log f_{ij} - \sum_{i=1}^{6} n_i \sum_{j=1}^{6} f_{ij} \log m_{ij} \\
&= \sum_{i=1}^{6}\sum_{j=1}^{6} n_i f_{ij} \log f_{ij} - \log p(v|M) \\
&= \sum_{i=1}^{6}\sum_{j=1}^{6} c_{ij} \log \frac{c_{ij}}{n_i} - \log p(v|M),
\end{aligned}$$

ce qui justifie le théorème.

Corollaire 5.2.3 *Soient F la matrice des transitions d'un visionnage v et M un modèle, $div_{KL}(F, M)$ varie de manière opposée à $p(v|M)$ et nous pouvons écrire :*

$$p(v|M_1) \geq p(v|M_2) \Leftrightarrow div_{KL}(F, M_1) \leq div_{KL}(F, M_2)$$

Preuve.

$$\begin{aligned}
p(v|M_1) \geq p(v|M_2) &\Leftrightarrow \log p(v|M_1) \geq \log p(v|M_2) \\
&\Leftrightarrow -\log p(v|M_1) \leq -\log p(v|M_2) \\
&\Leftrightarrow \sum_{i=1}^{6}\sum_{j=1}^{6} c_{ij} \log \frac{c_{ij}}{n_i} - \log p(v|M_1) \leq \\
&\quad \sum_{i=1}^{6}\sum_{j=1}^{6} c_{ij} \log \frac{c_{ij}}{n_i} - \log p(v|M_2) \\
&\Leftrightarrow div_{KL}(F, M_1) \leq div_{KL}(F, M_2)
\end{aligned}$$

5.2.3.4 Implémentation

Selon sa définition, la divergence div_{KL} ne peut pas traiter des matrices M avec des valeurs nulles (divisions par zéro). Pour ce faire, nous proposons d'utiliser une approximation. Chaque valeur nulle est remplacée par une petite valeur ϵ. Pour conserver le caractère stochastique des matrices, nous devons également multiplier chaque valeur non-nulle par une quantité correspondante.

Supposons par exemple que le vecteur $v = (v_1, \ldots, v_n)$ est un vecteur stochastique avec z valeurs nulles. Remplacer ces valeur par ϵ implique de multiplier chacune des $n - z$ valeurs non-nulles par $\alpha = 1 - z\epsilon$. Par exemple, si les valeurs nulles de ce vecteur se trouvent être les z dernières valeurs, nous devons transformer v en $(\alpha v_1, \ldots, \alpha v_{z-k}, \epsilon, \ldots, \epsilon)$.

Nous devons également choisir $\epsilon < \frac{1}{z}$ pour conserver une valeur de α inférieure à 1.

Pour appliquer ce traitement à une matrice multi-lignes, nous devons choisir $\epsilon < \frac{1}{z_{\max}}$, tel que z_{max} est le plus grand nombre de valeurs nulles présentes sur une ligne de la matrice. Ensuite, chaque ligne i de la matrice est multipliée par $1 - z_i\epsilon$ où z_i est le nombre de valeurs nulles de la ligne i et nous remplaçons finalement ses valeurs nulles par ϵ.

Nous avons choisi de définir $\epsilon = 0.01 < \frac{1}{36}$, 36 étant le nombre de valeurs de nos modèles, dans tous les cas. Cette valeur est suffisamment petite car elle pourrait fonctionner

avec une matrice n'ayant que des valeurs nulles. Nous avons également testé des valeur de ϵ plus petites. Dans notre cas, ce choix n'était pas sensible et n'influait pas sur les résultats. Nous avons donc décidé de retenir la valeur $\epsilon = 0.01$.

5.2.4 Résultats

5.2.4.1 Scénario expérimental

Pour mettre en avant les avantages de notre approche par les *K-models*, nous avons défini un questionnaire et un protocole de tests pour collecter les données relatives aux actions réalisées par les utilisateurs. Nous avons soumis ce questionnaire à une dizaine d'utilisateurs, familiarisés avec l'outil de recherche mais ne travaillant pas sur le moteur de recherche pour qu'ils parcourent les vidéos de la manière la plus naturelle possible.

Le choix de passer par un questionnaire pour collecter des données relatives à l'utilisation des vidéos a été nécessaire pour mener à bien nos expérimentations. En effet, nous avions d'abord laissé les utilisateurs naviguer librement dans les vidéos. Les données ainsi collectées n'étaient pas intéressantes car les utilisateurs n'étaient pas réellement motivés par l'utilisation de l'outil et ne l'exploitaient pas. La mise en place d'un questionnaire a permis de motiver les utilisateurs à mieux utiliser l'outil et de collecter des données relatives à des types d'utilisation différents.

Le questionnaire regroupe les questions suivantes :

1. Classez les bandes-annonces vidéo par catégories (comédie, drame...)
2. De quelle vidéo est extraite l'image suivante ?
3. Quel film vous apparaît comme le plus drôle ?
4. Quel film vous apparaît comme le plus spectaculaire ?
5. Vous allez au cinéma ce soir, quel film allez-vous choisir ?

La première question correspond davantage à un usage fait par un professionnel ou un administrateur qui devra classer les vidéos. Elle n'a pas été réalisée via le module de recherche mais simplement via une liste des bandes-annonces non classées.

Les quatre autres questions nécessitent cette fois l'usage du moteur de recherche, les trois dernières sont assez communes et souvent réalisées par des utilisateurs s'informant sur les films actuellement à l'affiche. La seconde est plus particulière et correspond plus à un usage professionnel.

Chaque utilisateur disposait d'une soixantaine de minutes pour répondre à ces questions. Nous ne nous intéressons pas aux résultats obtenus. L'objectif est de collecter des données relatives à des usages réellement observés dans notre cadre expérimental, chaque question et chaque utilisateur permettant de récupérer des manières de visionner les vidéos différentes et de fait des comportements uniques et sensés.

La taille du panel est de 10 utilisateurs. Le fichier de *logs* ainsi collecté regroupe environ 200 visionnages, chacun étant composé d'une dizaine d'actions en moyenne pour un ensemble de données donc composé de 2000 actions environ. Il est suffisant pour mettre en avant la qualité du regroupement obtenu. Le moteur de recherche et la nature et la collecte des données sont présentés dans les annexes (A.1) et (A.2).

5.2.4.2 Qualité du regroupement

Par ces tests, nous voulons montrer qu'une analyse sur des données réelles mène à l'extraction de modèles de comportements intéressants. En particulier, nous allons voir que le regroupement donne des classes compactes et bien différenciées. Nous verrons également l'influence du choix du nombre de classes (paramètre k) sur la qualité des classes obtenues.

La figure (5.2) présente, pour des valeurs de k variant de 3 à 10, le rapport (noté P_{rel}) entre les probabilités moyennes de génération internes et externes. La probabilité interne moyenne (notée P_{int}) correspond à la probabilité moyenne de génération des éléments par le modèle de leur classe pour l'ensemble des classes. Soient k le nombre de classes, M_i le modèle de la i^{eme} classe c_i, elle est donnée par la formule suivante :

$$P_{int} = \frac{\sum_{j=k}^{j=1} \frac{\sum_{v_i \in c_j} P(v_i|M_j)}{card(c_j)}}{k}$$

La probabilité externe moyenne (notée P_{ext}) correspond, pour chaque classe, à la

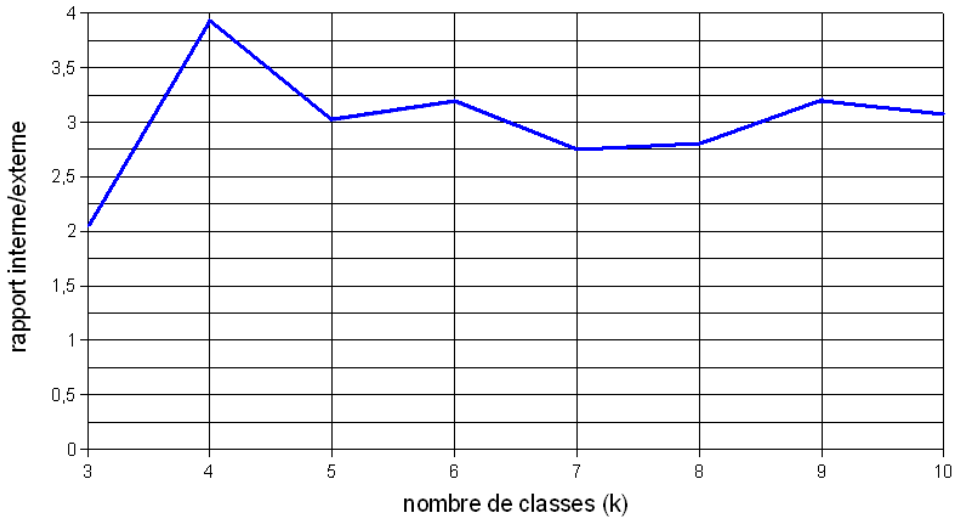


FIG. 5.2 – Rapport P_{int}/P_{ext} en fonction de k

probabilité moyenne de génération des éléments qui ne lui sont pas attribuée. Soit U l'ensemble des visionnages, elle est donnée par la formule suivante :

$$P_{ext} = \frac{\sum_{j=k}^{j=1} \frac{\sum_{v_i \notin c_j} P(v_i|M_j)}{card(U)-card(c_j)}}{k}$$

Finalement le rapport entre les probabilités est donné par la formule :

$$P_{rel} = \frac{P_{int}}{P_{ext}}$$

Plus ce dernier est grand, plus le regroupement est bon. Intuitivement, une valeur plus grande indique que les visionnages ont P_{rel} fois plus de chances d'avoir été généré par leur modèle. Nous voyons sur la figure que le maximum est défini pour $k = 4$. Un regroupement en quatre classes semble donc le plus intéressant pour cet ensemble de données.

5.2.4.3 Qualité du regroupement avec *Ray-Turi*

Pour évaluer la qualité du regroupement de manière plus usuelle, nous proposons d'utiliser l'indice de *Ray-Turi* [RT99]. Cet indice se base sur la mise en relation de deux quantités : la dissimilarité intra-classe et la dissimilarité inter-classe.

Dans notre cas, nous allons utiliser la divergence définie dans la section précédente au lieu d'une mesure de dissimilarité. Ce choix se justifie par le fait que la notion de symétrie de la dissimilarité n'a de sens que si les éléments comparés sont de même nature, ce qui n'est pas notre cas (visionnage et modèle de Markov). De plus, l'indice de *Ray-Turi*, contrairement à d'autres indices de qualité de regroupement, compare toujours un élément à un modèle de classe et non pas ni les éléments entre-eux, ni les modèles entre-eux.

Il est intéressant d'utiliser cette divergence et non pas une autre fonction car, comme nous l'avons montré dans le corollaire (5.2.3), les notions de probabilité de génération et de divergence entre les éléments sont liées. Nous nous assurons ainsi d'estimer la qualité du regroupement avec une fonction connexe au calcul de probabilité utilisé pour le regroupement.

Soient k le nombre de classes, M_i le modèle de la i^{eme} classe c_i, $\forall i, 1 \leq i \leq k$, la divergence intra-classe est donnée par la formule suivante :

$$IntraDiv = \frac{\sum_{j=1}^{k} \frac{\sum_{x_i \in c_j} div_{KL}(x_i, M_j)}{card(c_j)}}{k} \tag{5.3}$$

La divergence inter-classe, notée *InterDiv*, évalue la séparation entre les différentes classes. Pour chaque classe, elle calcule la divergence entre son modèle associé et les visionnages appartenant à toutes les autres classes découvertes et conserve la moyenne sur le nombre total de classes. Soit U l'ensemble des visionnages, la divergence inter-classe est représentée par la formule suivante :

$$InterDiv = \frac{\sum_{j=1}^{k} \frac{\sum_{x_i \notin c_j} div_{KL}(x_i, M_j)}{card(U) - card(c_j)}}{k}, \tag{5.4}$$

Un bon regroupement étant déterminé par des classes les plus compactes possible et les plus distantes possible les unes des autres, cela implique de chercher à maximiser *InterDiv* tout en minimisant *IntraDiv*. Ces deux objectifs sont mis en relation par l'indice de *Ray-Turi* r :

$$r = \frac{InterDiv}{IntraDiv} \tag{5.5}$$

La figure (5.3) présente l'évolution de la valeur de r en fonction de k. Il apparaît ici que $k = 4$, optimise l'indice de *Ray-Turi*. Cela renforce les résultats des mesures précédentes, indiquant 4 comme le nombre de classes le plus adapté pour cet ensemble de données.

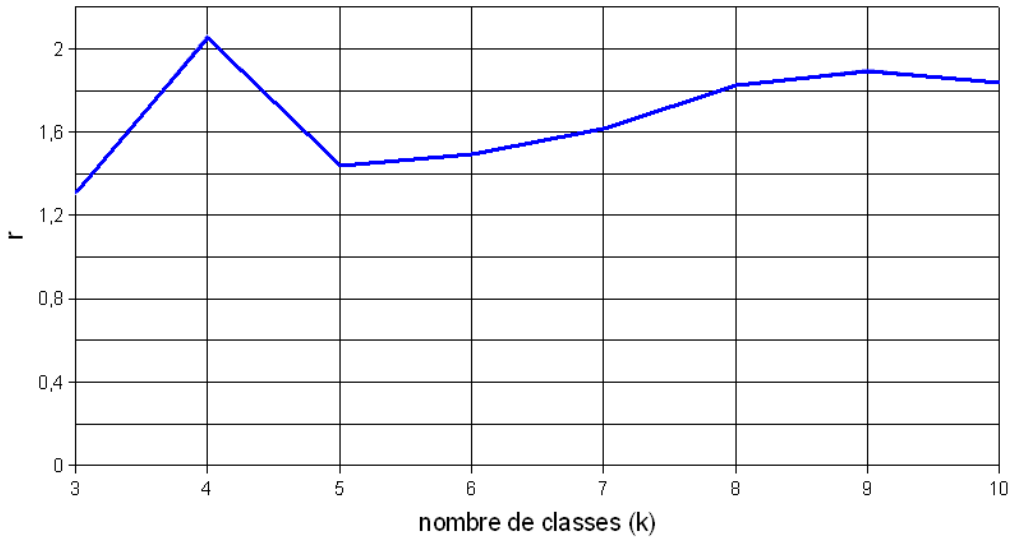


FIG. 5.3 – Indice de *Ray-Turi* en fonction de k

5.2.4.4 Analyse des comportements générés

L'analyse des modèles de Markov obtenus n'est pas une tâche automatique et nécessite l'intervention d'un expert. Le principe est que, à partir des modèles extraits, l'expert puisse en tirer des affirmations comportementales simples. Comme nous l'avons vu dans la section (4.2), ces modèles restent le moins complexe possible pour que cette analyse soit relativement aisée. La table (5.3) présente les quatre modèles qui ont été générés par la méthode des *K-models*. Rappelons ici que nous retrouvons en ligne les états de départ et les états d'arrivée en colonne. Les valeurs numériques correspondant aux probabilités de transitions d'un état de départ vers un état d'arrivée.

À partir de la table (5.3), nous allons maintenant montrer comment un expert en production ou en marketing vidéo pourrait analyser les modèles.

- Le premier modèle correspond à une lecture pratiquement complète de la vidéo avec peu d'interactions avec le lecteur vidéo. Cette conclusion vient en particulier de la forte probabilité de la transition [PLAY → PLAY].
- Le second correspond à un survol rapide de la vidéo. Les transitions [PLAY → PLAY] [PLAY → JUMP → PLAY] apparaissent avec une probabilité importante.
- Le troisième correspond à une lecture très partielle (transition [PLAY → PLAY] avec une probabilité de 22%) avec une forte probabilité de fermeture (transition [PLAY → STOP] avec une probabilité de 70%)
- Le dernier correspond à une fermeture rapide de la vidéo, celle-ci ne correspondant pas aux besoins d'un utilisateur, la transition [PLAY → PLAY] étant très faible et les probabilités d'arrêter la lecture de la vidéo fortes.

(%)	play	pause	jump	ffw	rwd	stop
play	74	3	23	0	0	0
pause	50	0	0	0	50	0
jump	23	0	0	0	22	55
ffw	0	0	0	100	0	0
rwd	0	0	0	0	16	84
stop	0	0	0	0	0	100

(%)	play	pause	jump	ffw	rwd	stop
play	30	33	25	0	0	12
pause	35	0	13	17	0	35
jump	44	13	43	0	0	0
ffw	100	0	0	0	0	0
rwd	0	0	0	0	100	0
stop	0	0	0	0	0	100

(%)	play	pause	jump	ffw	rwd	stop
play	22	2	6	0	0	70
pause	4	0	0	0	0	96
jump	63	0	8	0	1	28
ffw	0	0	0	100	0	0
rwd	0	0	25	0	75	0
stop	0	0	0	0	0	100

(%)	play	pause	jump	ffw	rwd	stop
play	4	54	6	0	0	36
pause	99	0	0	0	0	1
jump	65	0	21	0	0	14
ffw	0	0	0	100	0	0
rwd	0	0	0	0	100	0
stop	0	0	0	0	0	100

TAB. 5.3 – Modèles générés par *K-models*

5.3 Travail sur les vecteurs t

Dans la section précédente, nous avons présenté une technique de regroupement des visionnages qui permet d'extraire des comportements type vis-à-vis de données vidéo. Nous avons montré que l'utilisation des modèles de Markov permet d'extraire des résultats significatifs et directement exploitables. Cependant, nous avons estimé ces modèles encore complexes et avons essayé de les simplifier en se basant sur les propriétés des modèles de Markov absorbants présentées dans la section (3.2), notamment l'extraction du vecteur (vecteur t) de temps avant absorption, qui correspond pour les visionnages au temps moyen observé avant de stopper la lecture en fonction de l'action en cours.

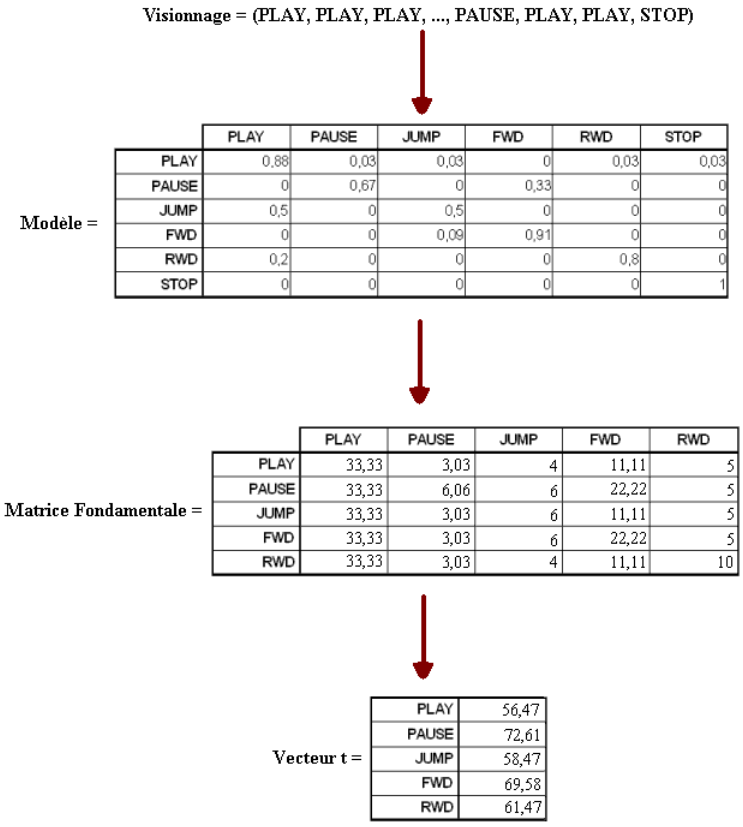
5.3.1 Extraction des vecteurs et regroupement

La première étape de cette approche consiste en la transformation des comportements observés lors des visionnages en vecteurs t, représentant le temps moyen observé dans chaque état avant l'arrêt du visionnage. Dans la section (3.2), nous avons présenté ce que sont ces vecteurs et comment les extraire d'un modèle de Markov absorbant. Dans notre cas, les comportements transformés en modèles de Markov sont des modèles absorbants. L'état $STOP$ est absorbant et les autres états ($PLAY$, $PAUSE$, $JUMP$, FWD, RWD) sont des états de transition.

En se basant sur la théorie des modèles de Markov, nous extrayons donc des visionnages en vecteurs t de dimension 5 (le nombre d'états de transition). Dans un premier temps chaque visionnage observé est transformé en matrice 6×6. Ensuite, on extrait de cette matrice la matrice fondamentale à partir de laquelle sera calculé le vecteur t. La figure (5.4) illustre le processus de transformation.

Maintenant que nous avons défini un nouveau format de données pour les visionnages intra-vidéo, plus simple et plus concis, nous avons essayé d'en extraire des comportements type sous cette forme. Pour ce faire, nous avons utilisé l'algorithme des *K-moyennes* [Mac66]. Pour comparer les vecteurs t entre-eux, nous calculons la moyenne des différences entre chacune des composantes.

Cette étape ne présente pas de particularité par rapport à une exploitation classique



Modèle	PLAY	PAUSE	JUMP	FWD	RWD	STOP
PLAY	0,88	0,03	0,03	0	0,03	0,03
PAUSE	0	0,67	0	0,33	0	0
JUMP	0,5	0	0,5	0	0	0
FWD	0	0	0,09	0,91	0	0
RWD	0,2	0	0	0	0,8	0
STOP	0	0	0	0	0	1

Matrice Fondamentale	PLAY	PAUSE	JUMP	FWD	RWD
PLAY	33,33	3,03	4	11,11	5
PAUSE	33,33	6,06	6	22,22	5
JUMP	33,33	3,03	6	11,11	5
FWD	33,33	3,03	6	22,22	5
RWD	33,33	3,03	4	11,11	10

Vecteur t	
PLAY	56,47
PAUSE	72,61
JUMP	58,47
FWD	69,58
RWD	61,47

FIG. 5.4 – Transformation des visionnages

de la technique de regroupement des *K-moyennes*. L'originalité réside dans la modélisation de nos modèles, directement extraits des modèles de Markov.

5.3.2 Résultats

5.3.2.1 Qualité du regroupement

Comme précédemment, nous avons utilisé le même jeu de données présenté pour la validation des techniques de regroupement par *K-models*. Nous présentons ici l'évaluation du regroupement avec l'indice de *Ray-Turi* présentée dans la table 5.4. Nous rappelons que nous cherchons à maximiser l'indice de *Ray-Turi*.

k	Ray-Turi
3	8.181
4	9.704
5	5.515
6	3.885
7	4.234
8	3.193
9	2.859
10	2.353

TAB. 5.4 – Analyse du regroupement avec *Ray-Turi*

Comme le montre la table 5.4 et la courbes de la figures 5.5, on note un regroupement optimal pour quatre classes.

5.3.2.2 Analyse des modèles

Nous analysons ici les quatre modèles qui ont été produits par le regroupement. La table 5.5 présente les quatre vecteurs extraits. Chaque ligne représente un vecteur défini par le temps moyen avant absorption (ie. avant l'arrêt de la lecture) en fonction de l'état dans lequel se trouve l'utilisateur (en secondes).

Rappelons que les modèles extraits des techniques de regroupement précédentes correspondaient aux comportements suivants :

– lecture complète,
– lecture partielle,

FIG. 5.5 – Indice de *Ray-Turi* en fonction de k

temps (sec)	$PLAY$	$PAUSE$	$JUMP$	FWD	RWD
Vecteur 1	26.4	16.1	11.6	0	0
Vecteur 2	5.8	2.4	0.6	0.1	0
Vecteur 3	84.5	87	86	85.5	0
Vecteur 4	56.7	58.1	0	0	0

TAB. 5.5 – Vecteurs obtenus par le regroupement

– survol de la vidéo,
– fermeture rapide.

En essayant de reporter les nouveaux résultats à ces modèles, nous observons que le vecteur 3 correspond bien au comportement de lecture complète avec des temps avant l'arrêt de la lecture élevés (plus de 80 secondes). Rappelons que les expériences ont été menées sur des vidéos de type bande-annonce cinématographique dont la durée moyenne correspond à ce temps. Le vecteur 2 correspond au comportement de fermeture rapide avec peu de temps avant absorption. Le vecteur 4 quant à lui correspond bien à une lecture partielle et non à un survol de la vidéo. Ce constat peut être fait en observant

que la durée de lecture est plus faible que pour le vecteur 3 et qu'il n'y a pas d'utilisation de la commande $JUMP$. Enfin, le vecteur 1 correspond finalement au survol rapide de la vidéo. Il est intéressant de noter que le temps avant arrêt du visionnage est moins important si l'utilisateur a réalisé l'action $JUMP$ que l'action $PLAY$. Cela montre qu'il lit certainement le début de la vidéo avant de réaliser un $JUMP$ pour lire une partie située plus loin dans la vidéo.

En conclusion, cette dernière approche de regroupement produit également des résultats intéressants et des modèles qui correspondent aux conclusions issues des précédents regroupements. La perte d'information sur les modèles due à leur réduction à des vecteurs à 5 dimensions ne semble pas impacter sur la nature des classes obtenues. De plus, l'analyse de ces valeurs permet une analyse plus directe des résultats. Ces dernières sont moins précises mais donnent des informations plus globales, notamment le temps passé en $PLAY$ avant absorption qui semble être un bon indicateur de l'intérêt que les utilisateurs ont porté aux vidéos.

Bibliographie

[DB79] D.L. Davies et D.W. Bouldin : A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 1, 1979.

[Do03] M. N. Do : Fast approximation of Kullback–Leibler distance for dependence trees and hidden Markov models. *IEEE signal processing letters*, 10, Avril 2003.

[KL51] S. Kullback et R. A. Leibler : On information and sufficiency. *The annals of mathematical statistics*, 22, Mars 1951.

[Kri93] V. Krishnamurthy : On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure. *IEEE Transactions on signal processing*, 41, 1993.

[Mac66] J. MacQueen : Some methods for classification and analysis of multivariate observations. *Fifth Berkeley symposium. University of California Press 1*, pages 281–297, 1966.

[RAC04] Z. Rached, F. Alajaji et L. Lorne Campbell : The Kullback-Leibler divergence rate between Markov sources. *IEEE transactions on information theory. ISSN :0018-9448*, 50:917–921, Mai 2004.

[RT99] S. Ray et R. H. Turi : Determination of number of clusters in K-means clustering and application in colour image segmentation. *Fourth international conference on advances in pattern recognition and digital techniques. Calcutta, Inde*, Decembre 1999.

Chapitre 6

Analyse des comportements inter-vidéo

Sommaire

6.1 Introduction

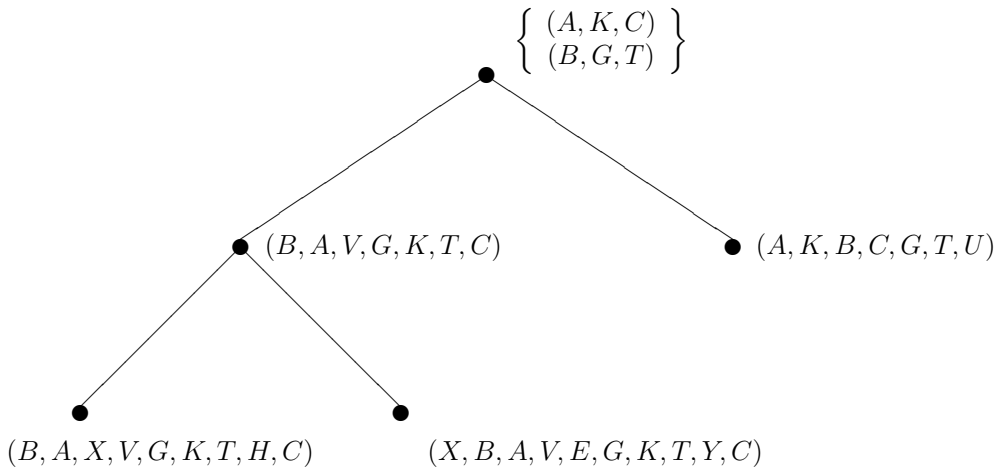


FIG. 6.1 – Arbre de regroupement

Dans le chapitre précédent (5), nous avons vu différentes méthodes pour regrouper et extraire les comportements type intra-vidéo. Pour compléter notre analyse de l'exploitation des données vidéo, nous proposons maintenant de nous attacher à analyser les comportements au niveau inter-vidéo. La section (4.3) a présenté ce qu'était le niveau inter-vidéo et comment modéliser un comportement et un groupe de comportements. Le principe est de modéliser le comportement observé lors d'une session par la séquence des vidéos accédées lors d'une session. Chaque vidéo est représentée par son identifiant et la manière dont elle a été visionnée (ie. le comportement intra-vidéo observé). C'est par ce biais que nous faisons le lien entre les deux niveaux de modélisation et que nous garantissons un modèle final complet. Un ensemble de sessions est quant à lui représenté par un ensemble de sous-séquences. L'intérêt de cette représentation par séquences multiples est double. Contrairement aux approches extrayant une unique sous-séquence, elle per-

met de ne pas avoir à choisir aléatoirement entre deux sous-séquences de même longueur pour représenter un ensemble. De plus, l'analyse intra-vidéo permet de différencier des groupes de sessions composés des mêmes séquences de vidéo mais visionnées de manières différentes.

En reprenant les données présentées lors de la modélisation, soient les trois sessions où chaque lettre représente une vidéo (nous ne considérons pas ici les comportements intra-vidéo) :

- $(B, A, X, V, G, K, T, H, C)$
- $(X, B, A, V, E, G, K, T, Y, C)$
- (A, K, B, C, G, T, U)

la figure (6.1) présente le modèle à séquences multiples les représentant.

Dans la suite du chapitre, nous présentons notre technique de regroupement ascendante hiérarchique, basée sur le principe de la modélisation par séquences multiples qui permet d'extraire les sessions-type. Ensuite, nous verrons sur un ensemble de données simulées les avantages de notre approche et l'intérêt à la fois de l'importance de la prise en compte du niveau intra-vidéo et de la représentation par des séquences multiples des modèles de sessions.

6.2 Regroupement par sous-séquences multiples

6.2.1 Introduction

Nous allons maintenant présenter notre approche pour extraire d'un ensemble de sessions les archétypes des visites. Après avoir présenté les principes de notre technique, nous verrons comment nous devons initialiser les données, puis nous présenterons l'algorithme proprement dit avant de finir en illustrant les avantages de cette approche sur des données expérimentales.

L'algorithme de regroupement lui-même est un algorithme hiérarchique ascendant. Il débute en considérant des petits groupes de sessions comme classes et fusionne hiérarchiquement les classes les plus proches. Le regroupement prend fin lorsque le niveau d'homogénéité requis est atteint. Nous avons choisi d'utiliser une telle approche car elle permet la lecture des résultats du regroupement à différents niveaux. En conservant la hiérarchie des plus hauts niveaux de regroupement, nous pouvons directement observer les sous-groupes qui ont été regroupés.

Son originalité réside dans la représentation des classes. Les techniques de regroupement de séquences usuelles ne traitent qu'une unique séquence pour représenter une classe. Nous avons développé un outil capable de comparer et de fusionner des classes représentées non plus par une unique séquence mais par un ensemble de séquences.

La figure (6.2) présente un aperçu du regroupement. Chaque point représente une session, les ensembles de points sont les groupes initiaux et les rectangles les ensembles de sous-séquences extraits puis fusionnés hiérarchiquement. L'algorithme est exécuté de la manière suivante :

1. création des groupes de sessions initiaux
2. extraction des sous-séquences représentatives des groupes initiaux
3. sélection des deux groupes les plus proches
4. fusion de ces groupes et retour en (3) si les paramètres de taille sont respectés

Nous rappelons ci-dessous les trois paramètres des modèles, définis dans la section (4.3.3.3) qui entrent en ligne de compte pour obtenir le meilleur regroupement :

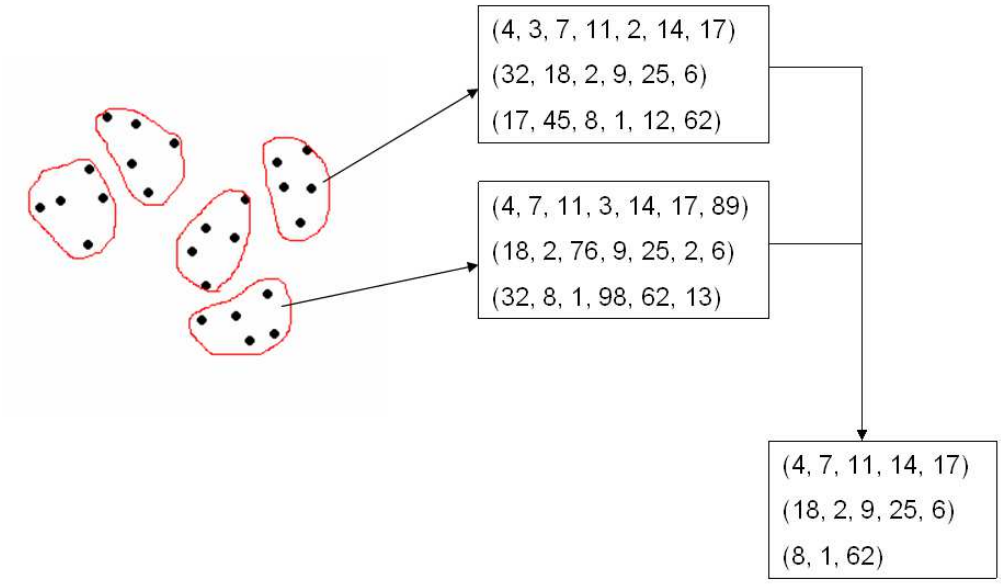


FIG. 6.2 – Aperçu du regroupement hiérarchique

- la longueur minimale des sous-séquences représentatives (notée l), c'est-à-dire les sous-séquences conservées pour représenter les classes
- le nombre minimal de sous-séquences représentatives pour chaque classe (noté r)
- le pourcentage de représentativité d'une sous-séquence (noté pct).

Ce dernier paramètre permet de choisir le niveau d'homogénéité des classes. Si il est fixé à 100%, cela signifie que chaque sous-séquence doit appartenir à toutes les sessions de la classe et, en général, s'il est fixé à x%, chaque sous-séquence doit correspondre à x% des sessions. Une valeur élevée entrainera donc l'extraction de classes très compactes et une valeur plus faible des classes plus larges mais également plus hétérogènes.

6.2.2 Initialisation

Le principe de l'algorithme étant la fusion successive de classes représentées par des ensembles de sous-séquences, il est nécessaire pour débuter la phase hiérarchique et introduire le paramètre *pct* de définir des classes initiales, elles-même étant des ensembles de sous-séquences. Pour cela, nous proposons de découper l'espace initial des sessions en petits groupes de sessions et d'en extraire les sous-séquences communes en fonction de la valeur des paramètres l, r et *pct*.

Une autre approche aurait pu être de débuter directement avec les sessions et de réaliser un regroupement hiérarchique classique. Cependant, cela n'aurait pas permis de prendre en compte le paramètre *pct* sans de nombreux calculs supplémentaires.

6.2.2.1 Création des groupes

Pour créer les classes initiales, nous proposons une approche consistant à considérer des groupes de sessions. Le principe est de diviser l'ensemble des données en petits groupes de sessions proches. En parcourant les données, nous construisons un groupe en considérant une session et en la regroupant avec les z sessions les plus proches d'elle (nous avons défini $z = 4$ dans nos expérimentations). Nous avons utilisé cette heuristique car, bien qu'elle ne garantisse pas un découpage optimal des données, elle permet de calculer les groupes en un temps raisonnable. Pour calculer la proximité entre les sessions, nous nous basons sur l'extraction de la plus longue sous-séquence commune pondérée par le vecteur de comportement présentée dans la section (4.3.4) par la formule (4.2).

Notons également ici le choix de créer des classes de $z+1 = 5$ éléments. Ce choix est dû à un besoin de considérer assez de sessions pour extraire des sous-séquences intéressantes tout en restant assez faible pour ne pas engager des calculs trop coûteux, comme nous le verrons dans la section suivante (6.2.2.2).

6.2.2.2 Extraction des sous-séquences

Une fois les ensembles initiaux définis, il reste à en extraire les sous-séquences communes. Pour cela, le principe est de considérer des sous-ensembles et d'en extraire les

sous-séquences. En fonction de la valeur de *pct*, nous allons traiter des groupes de une à cinq sessions. Si l'on considère une session à la fois, cela revient à définir $pct = 20\%$, si l'on considère toutes les sessions, $pct = 100\%$. Avec les valeurs intermédiaires 2, 3 et 4, *pct* correspond respectivement à 40%, 60% et 80%. Nous pouvons voir ici les raisons du choix de générer des groupes de $z + 1 = 5$ sessions. Cela permet de trouver un équilibre entre le besoin de définir un paramètre *pct* de manière significative et de ne pas surcharger les calculs. Avec ceci, nous considérerons au maximum C_5^3 ou C_5^2 sous-groupes, soit dix sous-groupes pour les valeurs de *pct* égales à 40% et 60%.

Ensuite, pour chacun des sous-groupes, nous allons extraire l'ensemble des plus-longues sous-séquences communes avec la fonction $WLCS$ et conserver les n sous-séquences les plus longues satisfaisant le paramètre l.

Ces ensembles de sous-séquences sont finalement les classes initiales à partir desquelles sera exécuté l'algorithme de regroupement hiérarchique.

6.2.3 Algorithme

6.2.3.1 Une approche hiérarchique

Le principe de notre algorithme est, à partir des ensembles de sous-séquences précédemment construits, de réaliser des fusions successives entre les ensembles les plus proches. Chaque étape va regrouper deux ensembles en une seule nouvelle classe. L'algorithme génère ainsi une arborescence de regroupement. Les feuilles de cette arborescence sont les ensembles initiaux et, plus on monte dans la hiérarchie, plus le regroupement est important.

Ce regroupement hiérarchique se poursuit jusqu'au moment où l'algorithme ne peut plus générer de classes correspondantes aux paramètres initiaux, c'est-à-dire avec un nombre de séquences r d'une longueur minimale l.

Dans la suite de cette section, nous allons voir l'implémentation et les spécificités de l'algorithme. Nous verrons tout d'abord les choix de son implémentation avant de nous pencher sur le coeur de la méthode : les fonctions de comparaison et de fusion des classes. Ce sont ces dernières qui distinguent notre approche.

6.2.3.2 Implémentation

Le principe est donc de comparer deux à deux les classes à chaque niveau de regroupement et de fusionner la paire la plus proche. En pratique, cela implique une forte redondance des calculs et une forte augmentation du temps de traitement. En effet, il est plus judicieux de conserver les distances calculées et de les réutiliser à chaque étape. Étant donné qu'une seule classe est créée, la très grande majorité des paires calculées conserve les même valeurs de distance.

En allant plus loin, on peut également observer qu'il est intéressant de conserver une liste ordonnée selon les distances entre les classes de ces paires. Ainsi, il suffit à chaque étape de prendre la première paire pour obtenir les deux classes à fusionner.

Soient $lc = c_1, c_2, ...c_z$ la liste des z classes initiales, soit p une pile de couples de classe telle que tout élément de p est de la forme $(c_a, c_b, comparer(c_a, c_b), fusionner(c_a, c_b)$ où c_a et c_b sont deux classes distantes de $comparer(c_a, c_b)$ et dont la fusion produit la classe $fusionner(c_a, c_b)$, la pile p est construite selon la fonction de la table (6.1) où la fonction $isValid(c)$ vérifie que la classe c vérifie les paramètres r et l, les fonctions *comparer* et $fusionner$ retournent respectivement la distance entre deux classes et la classe fusion de deux classes. Nous verrons par la suite comment ces fonctions sont implémentées (sections (6.2.3.4) et (6.2.3.3)).

Une fois cette pile créée, elle sera mise à jour à chaque fusion. Pour cela, l'élément de la pile avec la plus petite distance est récupéré, tous les éléments contenant l'une des deux classes de l'élément sont supprimés, tous les couples entre les classes candidates et la nouvelle classe sont calculés et ajoutés à la pile si nécessaire. La table (6.2) présente cette fonction où la fonction $p.retirerEltContient(c)$ enlève de la pile p toutes les paires de classes construites à partir de c.

Sur la base de ces deux fonctions, nous construisons la fonction globale qui réalise le regroupement complet (table 6.3). Elle retourne la pile p qui correspond à l'ensemble final des modèles de classes extraites, respectant les paramètres r et l.

```
fonction creerPile
   pile p
   for i = 1..z do
      for j = i+1..z do
         c := fusionner(classe(i), classe(j))
         if isValid(c) then
           p.inserer(classe(i), classe(j), comparer(classe(i), classe(j)),
c)
         end if
      od
   od
   return p
end fonction
```

TAB. 6.1 – Fonction creerPile

```
fonction oneStep(p)
   (c1, c2, d, m) = p.retirerEltMiniDist()
   p.retirerEltContient(c1)
   p.retirerEltContient(c2)
   lc.retirerElt(c1)
   lc.retirerElt(c2)
   for c in lc do
      new = fusionner(c, m)
      if isValid(new) then
         p.inserer(c, m, comparer(c, m), new)
      end if
   od
   lc.ajouterElt(m)
end fonction
```

TAB. 6.2 – Fonction oneStep

6.2.3.3 Fusion des classes

Le principe de la fusion de deux classes est de produire, à partir de deux ensembles de sous-séquences, un ensemble de sous-séquences communes le plus large possible. Pour cela, l'algorithme va fusionner deux à deux les séquences issues des deux classes en

```
fonction regrouper(lc)
   p = creerPile
   while p.notEmpty() do
      oneStep()
   od
   return p
end fonction
```

TAB. 6.3 – Fonction regrouper

essayant de maximiser leur taille. Soient c_1 et c_2 deux classes composées de séquences et c la classe obtenue par la fusion de c_1 et c_2, la table (6.4) présente la fonction qui réalise cette fusion où la fonction $dist_{WLCS}$ calcule la distance entre deux séquences par l'extraction de la sous-séquence la plus longue, pondérée par les comportements (formule 4.2) et la fonction $WLCS$ (formule 4.3) retourne la plus longue sous-séquence commune pondérée.

Nous pouvons noter qu'en fusionnant les séquences deux à deux de cette manière, nous assurons la validité du paramètre initial *pct* tout au long de l'exécution. En effet soient s_1 et s_2 deux séquences correspondant chacune à *pct* des données, $WLCS(s_1, s_2)$ correspond à la fois à *pct* des données par s_1 et à *pct* des données par s_2. Si les deux séquences recouvrent le même ensemble de données, $WLCS(s_1, s_2)$ correspond à *pct* des données. Dans le cas général, le recouvrement de $WLCS(s_1, s_2)$ est compris entre *pct* et $2 \times pct$.

Notons également que, lors de la fusion de deux séquences, leurs vecteurs de comportement sont également fusionnés pour chacun des éléments extraits. Soient $v_1 = (v_{11}..v_{1K})$ et $v_2 = (v_{21}..v_{2K})$ deux comportements à fusionner, le vecteur $v_n = (v_{n1}..v_{nK})$, résultat de la fusion, est donné par la formule (6.1).

$$\forall i, 1 \leq i \leq K, v_{ni} = \frac{(v_{1i} + v_{2i})}{2} \tag{6.1}$$

```
fonction fusionner(c1, c2)
   while c1.notEmpty() and c2.notEmpty() do
      min := 1
      for s in c1 do
         for t in c2 do
            tmp := distWLCS(s, t)
            if tmp < min then
               min := tmp
               res := WLCS(s, t)
               s1 := s
               s2 := t
            end if
         od
      od
      remove(c1, s1)
      remove(c2, s2)
      insert(c, res)
   od
   return c
end fonction
```

TAB. 6.4 – Fonction fusionner

6.2.3.4 Comparaison des classes

La comparaison et l'évaluation des distances entre les classes sont très similaires à la technique de fusion. De la même manière, cette fonction va comparer les séquences 2 à 2 en cherchant à chaque étape à conserver la distance la plus faible pour faire la moyenne des distances entre les appariements de séquences. La table (6.5) présente cette fonction.

6.2.4 Complexité de notre approche

L'étude de la complexité de l'algorithme en fonction du nombre de sessions à regrouper peut se décomposer en deux phases :

- la création des groupes initiaux et la création de la pile des classes à fusionner,
- les étapes de fusion successives.

```
fonction comparer(c1, c2)
   res := 0
   cpt := 0
   while c1.notEmpty() and c2.notEmpty() do
      min := 1
      for s in c1 do
         for t in c2 do
            tmp := distWLCS(s, t)
            if tmp < min then
               min := tmp
               s1 := s
               s2 := t
            end if
         od
      od
      remove(c1, s1)
      remove(c2, s2)
      cpt := cpt + 1
      res := res + min
   od
   return res / cpt
end fonction
```

TAB. 6.5 – Fonction comparer

La création des groupes initiaux consiste en la comparaison des sessions deux à deux. Pour la première passe, nous comparons une session aux $n-1$ autres sessions, n étant le nombre total de sessions à regrouper. La seconde passe réalisera $n-6$ comparaisons et la k^{ieme} $n-1-(k\times 5)$. La complexité est ici en $O(n \times log(n))$. De plus, chaque groupe candidat est, une fois créé, inséré dans la pile p de paires de groupes, triée selon la distance entre les paires. Soit m le nombre de paires de groupes candidates, l'insertion d'une paire a un coût de $O(log(m))$. Un groupe candidat est un groupe dont le modèle de représentation respecte les paramètres définis pour le regroupement.

La complexité totale de cette phase initiale est donc comparable à $O(n\times log(n)+m\times log(m))$. La valeur de m est comparable, dans le pire des cas, à $(\frac{n}{5})^2$ si tous les groupes

sont fusionnables. Dans le cas général, étant donné que la pile ne conserve que les paires de groupes qui satisfont les paramètres après fusion, la valeur de m est largement inférieure à n^2.

Ensuite, une étape de fusion consiste en la fusion de la paire de groupes du sommet de la pile puis le calcul des paires de groupes possibles avec ce nouveau groupe et leur insertion dans la pile. Pour cela, nous parcourons la pile et extrayons chaque paire qui contenait l'un des deux groupes fusionnés et nous le remplaçons par une nouvelle paire avec le groupe fusionné pour une complexité en $O(m \times log(m))$. Notons enfin que, étant donnée la taille relativement limitée des sessions traitées, la fonction de comparaison à un coût limité.

La figure (6.3) présente l'évolution du temps moyen de traitement observé lors du regroupement en fonction du nombre de séquences considérées.

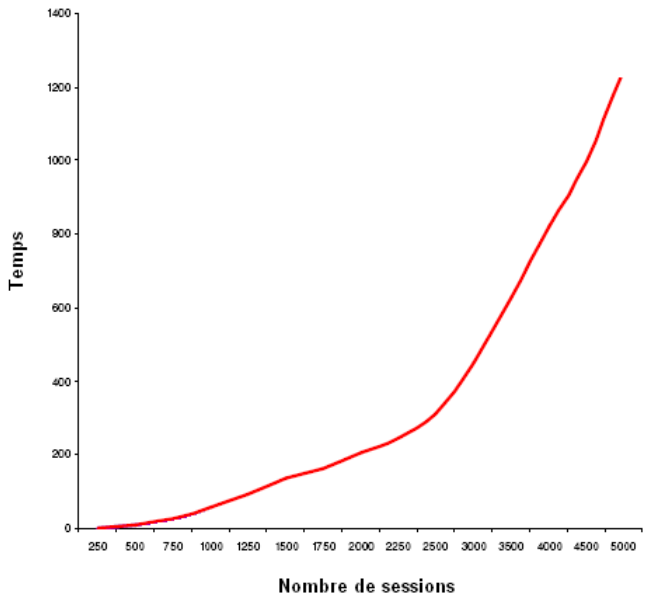


FIG. 6.3 – Temps de traitement du regroupement.

6.3 Résultats expérimentaux

Cette section met en avant les deux avantages de notre approche comparée aux techniques usuelles. Tout d'abord, nous allons voir comment l'analyse du comportement intra-vidéo permet de différencier des groupes de sessions composées des mêmes séquences de vidéo mais visionnées de manières différentes. Ensuite, nous allons montrer l'avantage de décrire les classes par un groupe de sessions comparé à une simple extraction de sous-séquence.

6.3.1 Création des jeux de données

En raison de la nouveauté des données exploitées ici, nous avons réalisé nos expérimentations sur des jeux de données de test.

La création des jeux de données de test se présente en deux phases. Premièrement, nous avons utilisé des modèles de comportement intra-vidéo. Nous avons défini quatre comportements type (lecture totale, survol rapide, lecture d'un passage précis, fermeture rapide). De nouvelles expérimentations sur des données réelles nous permettront de valider totalement ces modèles. À partir de ces modèles, nous avons créé aléatoirement un vecteur de comportement pour chaque visionnage des sessions vidéo.

Ensuite, pour générer les sessions vidéo, nous avons défini pour chaque classe une source de génération avec un ensemble de séquences d'identifiants vidéo. Ces séquences sont semblables aux résultats de différentes requêtes qui auraient retournées une liste de vidéos. Comme nous le verrons dans la section (6.3.3), un ensemble de séquences d'identifiants correspond à différentes requêtes hypothétiques en relation avec un thème donné. Pour chaque classe, nous avons généré les sessions en fusionnant aléatoirement plusieurs de ces séquences. Nous avons enfin ajouté dans ces sessions entre 5% et 20% de bruit en introduisant des identifiants de vidéo n'ayant pas de lien avec le contenu de la classe dans les séquences. Nous avons finalement généré différents jeux de données composés de deux mille sessions. Chacune est composée de cinq à vingt visionnages correspondant à des comportements de dix à vingt actions de base (lecture, pause...). Les fichiers de tests sont donc composés d'environ cent mille à huit cent mille actions de

base. La figure 6.4 présente quelques sessions générées pour le deuxième scénario ci-après.

```
session 1: (18, 73, 29, 41, 23, 17, 25, 12, 19, 49, 87, 10, 129, 2, 73, 32, 6, 91, 301, 302, 65 ,303, 304)
session 2: (18, 73, 59, 29, 41, 301, 302, 303, 304, 62, 17, 25, 12, 19, 87, 10, 129, 2, 73, 92, 32 ,91, 31)
session 3: (129, 2, 13, 73, 32, 91, 301, 301, 303, 304, 18, 73, 29, 41, 16, 17, 25, 12, 19, 87, 11, 10)
session 4: (301, 302, 303, 69, 304, 129, 2, 73, 32, 60 ,91 ,17, 25, 12, 19, 87, 10, 18, 73, 29, 41)
session 5: (129, 2, 73, 34, 32, 91, 18, 73, 29, 33, 41, 301, 302, 303, 304, 17, 25, 12, 11, 19, 87, 10)
session 6: (1, 17, 25, 12, 19, 87, 10, 13, 301, 302, 303, 304, 18, 73, 29, 41, 129, 2, 73, 32, 91, 40)
session 7: (129, 2, 73, 32, 91, 17, 25, 12, 19, 13, 87, 10, 18, 73, 29, 14, 41, 301, 302, 303, 304, 28)
session 8: (18, 73, 29, 41, 129, 2, 73, 32, 91, 17, 25, 12, 19, 87, 10, 301, 302, 303, 304)
```

FIG. 6.4 – Exemple des sessions analysées.

La suite présente deux exécutions correspondant à des scénarios prédéfinis qui mettent en avant les avantages de notre technique. Considérons dans la suite que les utilisateurs exploitent et réalisent des recherches dans une base de données de vidéo en rapport avec la nature contenant des vidéos de montagnes et de volcans.

6.3.2 Exploitation du comportement intra-vidéo

Ce scénario est basé sur l'affirmation suivante. La base vidéo n'est pas correctement indexée et de nombreuses vidéos traitant des volcans sont indexées comme des vidéos de montagne.

Nous avons généré deux classes. La première correspond à une recherche sur les volcans et retourne uniquement une seule vidéo complètement visionnée. Elle correspond à la séquence (1, 2, 3, 4, 5, 6, **10**) dans laquelle seulement la vidéo **10** est visionnée (représentée en gras dans la table 6.6). La seconde est le résultat d'une recherche sur les montagnes et toutes les vidéos sont visionnées complètement (table 6.6). Elle correspond à la séquence (**1**, **2**, **3**, **4**, **5**, **6**) dans laquelle toutes les vidéos sont visionnées totalement. Avec une approche classique, ne prenant pas en compte le comportement intra-vidéo, les deux classes ne sont pas découvertes et le regroupement retourne une seule classe {(1, 2, 3, 4, 5, 6)}, composée d'une unique sous-séquence.

Avec notre approche à deux niveaux, le regroupement est à même de découvrir que les vidéos ont été exploitées différemment et les deux classes sont effectivement découvertes. Pour toute valeur inférieure à 6 de la longueur minimale des séquences représentatives (l), les classes {(1, 2, 3, 4, 5, 6)} et {(1, 2, 3, 4, 5, 6, 10)} sont découvertes.

	vidéos fréquemment accédées	approche intra et inter-vidéo	approche inter-vidéo
rech. sur les volcans	(1, 2, 3, 4, 5, 6, **10**)	(1, 2, 3, 4, 5, 6, **10**)	(1, 2, 3, 4, 5, 6)
rech. sur les montagnes	(**1**, **2**, **3**, **4**, **5**, **6**)	(**1**, **2**, **3**, **4**, **5**, **6**)	

TAB. 6.6 – Description des classes du premier jeu de données

6.3.3 Modélisation des classes par plusieurs séquences

Cette expérimentation met en avant l'avantage de modéliser les classes avec un ensemble de séquences au lieu d'une seule séquence. Nous avons généré des données correspondant à quatre classes. Trois d'entre elles correspondent à des recherches sur des stations de ski et sont composées d'une séquence spécifique et de trois autres partagées : Montagne, correspondant à la séquence (18, 73, 29, 41) ; Snowboard (17, 25, 12, 19, 87, 10) ; Ski (129, 2, 73, 32, 91). La dernière classe est également composée d'une recherche sur la montagne (18, 73, 29, 41) et d'une autre sur le trekking (2, 3, 4, 8). Pour ce jeu de données, toutes les vidéos ont été visionnées complètement (table 6.7).

En positionnant le nombre de séquences représentatives (n) à 2 ou 3, les sessions correspondant à l'une des trois premières classes sont fusionnées pour former une classe « sessions traitant des stations de ski ». Avec une valeur de 4, cette classe est divisée et chaque classe source est découverte. Pour ces valeurs, la classe correspondant au trekking est correctement analysée et n'est pas fusionnée avec les autres données. Mais si nous positionnons le nombre de séquences à 1, nous plaçant ainsi dans le cadre d'une simple extraction de sous-séquence, toutes ces données sont fusionnées dans une unique classe et la différence entre les stations de ski et le trekking n'est pas détectée lors du regroupement. On voit finalement l'intérêt d'une approche multi-séquences. Elle permet de retrouver des éléments correspondant à plusieurs recherches réalisées les unes à la suite des autres dans un ordre variable et de les reconstituer dans une même classe. Sur notre exemple, on peut ainsi reformer une classe générique correspondant au travail sur les stations de ski ($n = 2$ ou $n = 3$).

	Resultats		
vidéos fréquemment accédées	minSize = 2 or 3	minSize = 4	minsize = 1
(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91) (301, 302, 303, 304)		{(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91) (301, 302, 303, 304)}	
(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91) (401, 402, 403, 404)	{(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91)}	{(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91) (401, 402, 403, 404)}	{(18, 73, 29, 41)}
(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91) (501, 502, 503, 504)		{(18, 73, 29, 41) (17, 25, 12, 19, 87, 10) (129, 2, 73, 32, 91) (501, 502, 503, 504)}	
(18, 73, 29, 41) (2, 3, 4, 8)	{(18, 73, 29, 41) (2, 3, 4, 8)}	{(18, 73, 29, 41) (2, 3, 4, 8)}	

TAB. 6.7 – Regroupement du second jeu de données

6.4 Conclusion

Cette section a présenté notre approche pour extraire les usages d'un ensemble de sessions ou de visites type. Nous avons montré que nous sommes à même d'extraire des données plus complexes que les approches séquentielles classiques (section 6.3.3). Notamment, dans les approches classiques, si deux séquences de même longueur sont candidates, le choix d'en sélectionner une est aléatoire et induit une forte perte d'informativité.

De plus, la prise en compte du comportement intra-vidéo dans l'analyse par le biais de l'extraction pondérée (fonction $dist_{WLCS}$) permet une meilleure discrimination des

séquences et empêche des fusions inadaptées (section 6.3.2). Cette approche est comparable à l'analyse des sessions web où l'on prend en compte le temps passé sur chaque page pour estimer son importance dans la session mais de manière plus fine, le temps n'étant pas nécessairement suffisant ou adapté pour le traitement des comportements vidéo.

La section suivante va maintenant présenter une application aux différentes informations intra et inter-vidéo que nous avons extraites. Elle montrera notamment une technique simple de détection d'erreurs d'indexation en exploitant parallèlement aux données extraites une technique d'indexation de données avancée.

Chapitre 7

Application à un moteur de recherche vidéo

Sommaire

7.1 Introduction

Nous avons vu dans les chapitres précédents comment extraire les comportements des utilisateurs. Dans ce chapitre, nous allons mettre en pratique les résultats obtenus dans un outil de recherche vidéo. Beaucoup de moteurs de recherche effectuent leurs recherches sur des annotations - manuelles ou automatiques - extraites des données recherchées, les vidéos dans notre cas. Or, quelle que soit la méthode d'annotation, des erreurs existent et certaines données sont erronées.

Par exemple, en attribuant un mot-clé erroné à une vidéo, elle peut être retournée comme pertinente lors d'une recherche alors qu'elle ne l'est pas. Nous avons déjà abordé ce type de problème dans la section (6.3.2). Si les recherches sur les mots-clés *rue* et *monument* retournent dix vidéos et qu'en moyenne, les utilisateurs en visionnent neuf en grande partie et en ferment une prématurément, notre approche signalera une erreur potentielle à l'expert métier qui administre les données. Il pourra alors manuellement vérifier l'annotation et la corriger le cas échéant.

Nous proposons également ici une optimisation des résultats par retour d'expérience sur les utilisateurs. Nous allons proposer aux utilisateurs des vidéos qui ne sont pas retournées par une recherche mais qui sont généralement regardées lors des visites précédentes qui ont exploité les vidéos retournées. Cette approche est similaire à ce que l'on peut trouver sur les sites de commerces en ligne : « Les internautes qui ont acheté ceci ont également acheté... ».

En se basant sur les travaux de [UBD06], nous proposons donc de mettre en valeur notre approche afin d'apporter une plus-value à un moteur de recherche vidéo en détectant les erreurs d'indexation et en proposant des vidéos alternatives aux utilisateurs.

7.2 Intégration dans l'outil d'indexation

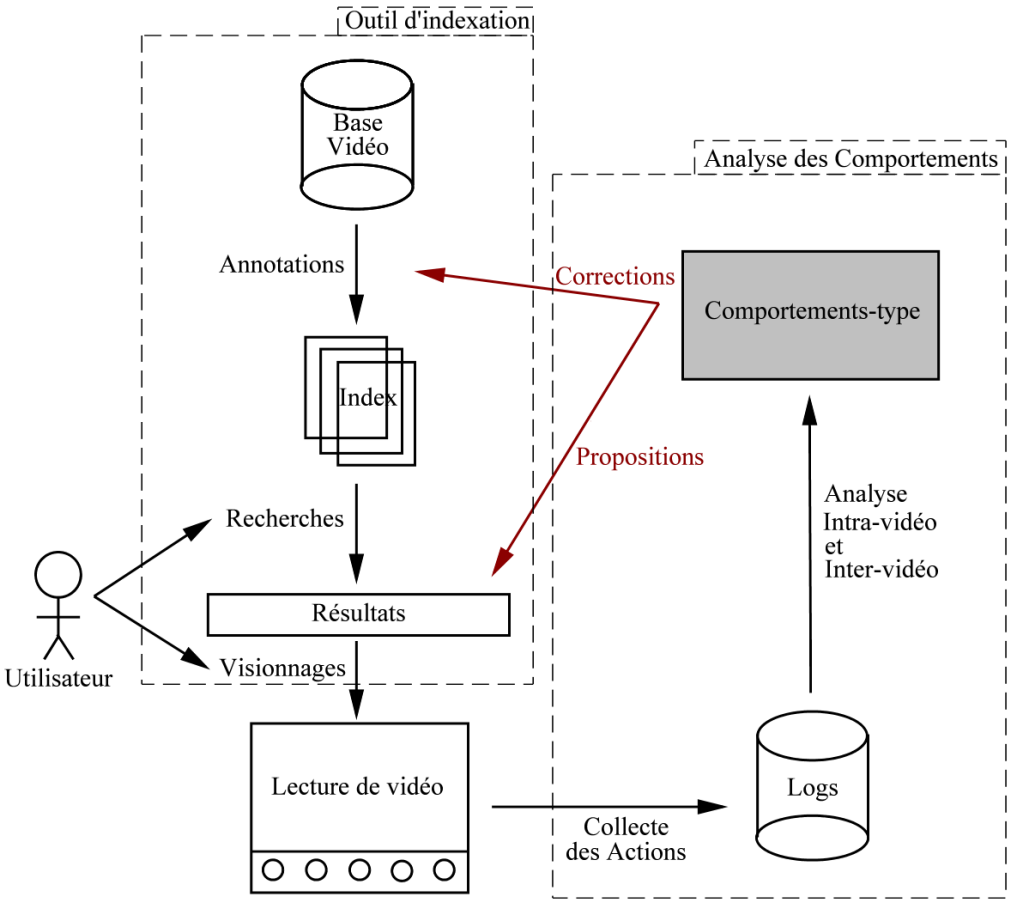


FIG. 7.1 – Indexation et analyse comportementale

Notre module d'analyse et d'extraction des comportements prend place dans un outil d'indexation et de recherche de documents vidéo. Nous ne nous attarderons pas ici sur les techniques d'indexation employées, posons simplement que la base de données vidéo

est indexée selon des catégories de mots-clés sur lesquelles sont effectuées les requêtes des utilisateurs. Les détails de l'outil d'indexation que nous utilisons sont présentés dans [UBD06, UBDS07].

La figure (7.1) présente le lien entre l'outil d'indexation et celui d'analyse des comportements. Dans un premier temps, les utilisateurs réalisent des recherches et visionnent certaines des vidéos qui leur sont retournées. Les actions qu'ils réalisent sur les vidéos ainsi que les requêtes sont collectées. Sur la base de ces données, l'outil d'analyse extrait les comportements type à partir desquels nous allons découvrir les erreurs d'indexation et pondérer l'importance des vidéos.

Ce système est incrémental. Plus les utilisateurs utiliseront l'outil, plus les erreurs d'indexation seront découvertes et plus les propositions alternatives seront pertinentes.

Il est à noter que notre modèle de comportement ne prend pas en compte les données d'indexation (requêtes, mots-clés, catégories). Il était indispensable de ne pas se baser sur ces données, notre objectif étant de les améliorer et de les corriger. L'indépendance entre le modèle utilisateur et les données d'indexation est importante pour obtenir des résultats pertinents. En effet, si les données d'indexation sont en partie erronées, s'appuyer sur ces dernières entraînerait des résultats biaisés et l'impossibilité de déterminer les erreurs dans l'indexation des vidéos

Dans la suite de ce chapitre, nous présentons comment extraire des comportements et des requêtes les propositions alternatives et les erreurs d'indexation.

7.3 Propositions alternatives

Le principe de proposer aux utilisateurs des données qui ne sont pas forcément retournées par les recherches n'est pas nouveau. En effet, depuis plusieurs années sur le web, de telles applications sont utilisées sur les sites commerciaux. L'idée générale est que certaines données sont souvent exploitées ensemble, bien que n'ayant rien en commun au vu des données d'indexation.

Dans notre cas, nous nous appuyons sur l'analyse des comportements inter-vidéo pour intégrer un tel module dans le moteur de recherche vidéo. Comme le présente la figure (7.2), le principe est que, pour chaque requête envoyée, on teste les appariements possibles avec d'autres vidéos. Ces appariements sont directement liés aux comportements type inter-vidéo et sont de deux catégories :

- Les propositions directes. Ce sont des vidéos apparaissant dans une même sous-séquence que certaines vidéos retournées par le moteur de recherche. Ce n'est a priori pas la proposition la plus intéressante car le plus souvent, une sous-séquence correspond à une recherche. Cependant s'il y a une réindexation ou un changement dans les données, cette proposition peut permettre de retrouver d'anciennes vidéos.
- Les propositions indirectes. Ce sont des vidéos apparaissant dans les sous-séquences d'une même classe telle que les vidéos de la recherche se trouvent dans une autre sous-séquence de la même classe. Ces propositions sont plus intéressantes car, si l'on considère encore qu'une sous-séquence correspond généralement à une recherche, elles permettent d'anticiper sur les autres recherches que pourrait faire l'utilisateur ensuite.

Cette approche, parallèle à la recherche sur la base des données d'indexation, pourrait toutefois entraîner le retour d'un trop grand nombre de vidéos à l'utilisateur. Nous n'avons pu tester cela dans des conditions réelles pour le valider. Si tel était le cas, nous pourrions envisager de ne conserver qu'un sous-ensemble des propositions, sélectionnées en fonction de leur popularité. Une étude de l'impact des propositions serait nécessaire pour en définir les modalités.

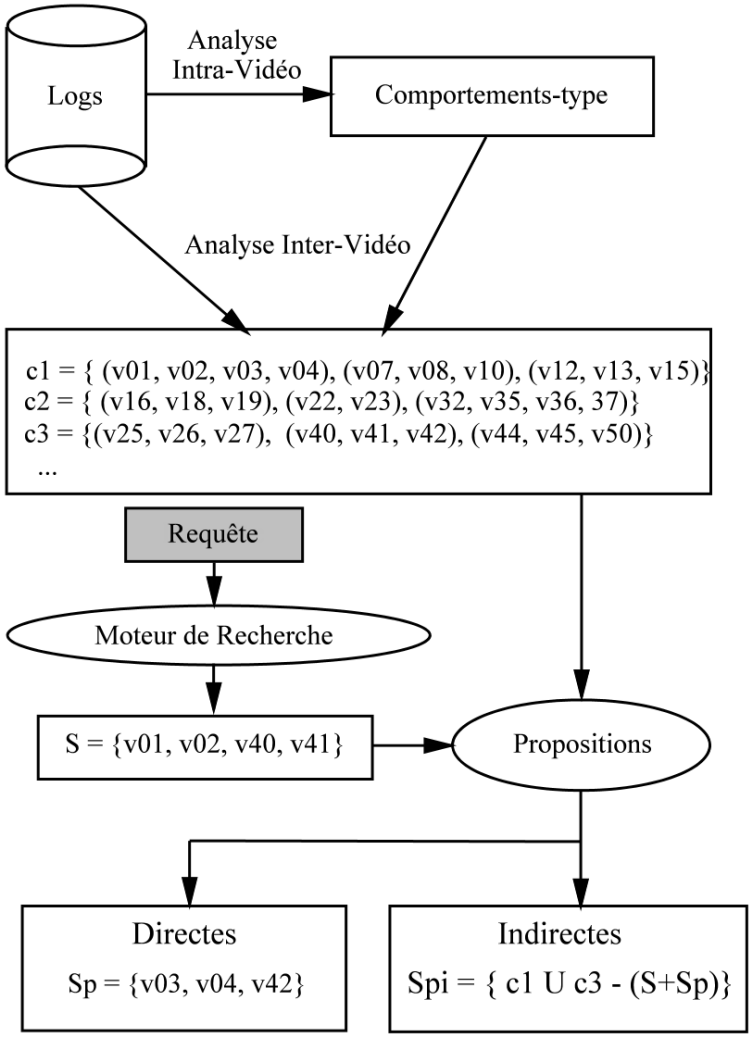


FIG. 7.2 – Algorithme de propositions alternatives

7.4 Détection des erreurs d'indexation

Cette section présente la technique de détection des erreurs d'indexation. Le but est de proposer à l'expert une liste d'erreurs potentielles automatiquement détectées. La figure (7.3) schématise les étapes de l'algorithme. À partir des comportements intra-vidéo extraits, l'expert définit les modèles selon trois niveaux de détails :

- la vidéo a été totalement exploitée. Elle correspond aux besoins des utilisateurs (notée +).
- la vidéo a été partiellement exploitée (survol rapide, lecture partielle). Elle correspond relativement aux besoins des utilisateurs (notée =).
- la vidéo n'a pas été exploitée (fermeture prématurée). Elle ne correspond pas aux besoins des utilisateurs ou elle a été mal indexée (notée −).

Ensuite, l'algorithme regroupe l'ensemble des comportements observés sur chaque mot-clé. Ainsi, on retrouve pour chaque mot-clé une liste de couples $(id_{video}, id_{comportement})$ où $id_{comportement}$ correspond à l'un des comportements décrits ci-dessus et est extrait sur la moyenne des comportements observés sur une vidéo pour un mot-clé. En analysant un mot-clé, si l'on découvre une vidéo mal exploitée (ici $(v3, -)$), deux possibilités se présentent :

- on retrouve le même type de comportements négatifs pour tous les mots-clés qui ont amené le visionnage de la vidéo. Le problème est alors lié à la vidéo. Cette dernière peut soit être de mauvaise qualité, être totalement mal indexée ou ne présenter aucun réel intérêt, quelle que soit la recherche liée.
- La vidéo est bien exploitée pour les autres mots-clés (+ ou =). Le problème est lié au mot-clé analysé qui ne doit pas correspondre à l'idée que s'en font les utilisateurs.

Dans ces deux cas, l'algorithme signale à l'expert un problème potentiel. Celui-ci corrigera alors les données d'indexation ou supprimera la vidéo s'il considère le problème potentiel comme un problème réel.

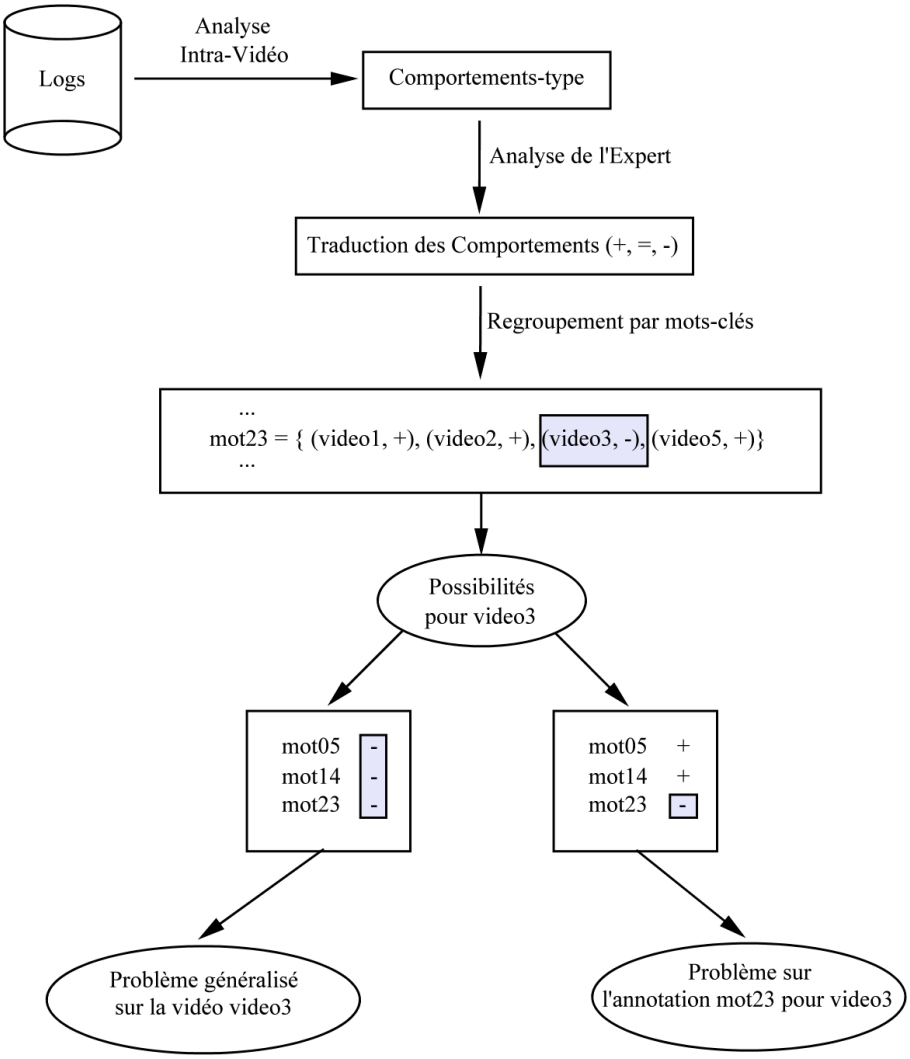


FIG. 7.3 – Algorithme de détection des erreurs

7.5 Expérimentations

Dans notre cadre expérimental, nous n'avons pu mesurer l'impact des propositions alternatives. Faute d'un outil exploité par de nombreux utilisateurs, nous n'avons pas pu mesurer son influence. Nous allons toutefois, à l'aide de données simulées mettre en avant l'intérêt de la découverte des erreurs d'indexation et la découverte de vidéos qui ne sont pas exploitées.

Pour illustrer cela, nous proposons donc de nous placer dans le contexte d'un moteur de recherche de vidéo et d'en simuler les données relatives aux utilisateurs. Nous considérons ainsi une base de cinquante vidéos, chacune indexée par cinq mots-clés. Sur ces cinquante vidéos :

- trente-quatre sont correctement indexées. Dans ce cas, nous générons à partir d'un modèle prédéfini des comportements correspondant à une lecture complète ou presque complète de ces vidéos pour tous les mots-clés.
- onze sont de mauvaise qualité et/ou sans intérêt pour les utilisateurs. Nous générons pour celle-ci à partir d'un autre modèle des visionnages très courts relatifs à une fermeture rapide de ces vidéos pour tous les mots-clés.
- sept ont un mot-clé qui ne leur correspond pas. Cette fois, nous générons des visionnages complets pour les mots-clés corrects et des fermetures rapides pour le mot-clé erroné.

Nous générons ainsi un total de dix mille visionnages. À partir de ces derniers, nous extrayons les comportements type et les traduisons comme un expert le ferait en lecture complète (+), aperçu (=) et fermeture rapide (−). Chaque visionnage se voit attribué le symbole qui lui correspond le plus. Ensuite, nous regroupons pour chaque vidéo, la liste des couples *(mot-clé, symbole de comportement)*. Finalement, nous retournons la liste des vidéos potentiellement à problèmes, c'est-à-dire qui comportent en majorité des visionnages de type − pour un ou plusieurs mots-clés. La figure (7.4) présente une partie détaillée des résultats que nous avons obtenus. On y retrouve notamment deux vidéos qui semblent de mauvaise qualité ou sans intérêt, une vidéo avec un mot-clé qui ne lui correspondrait pas et une dernière vidéo exploitée uniquement en partie.

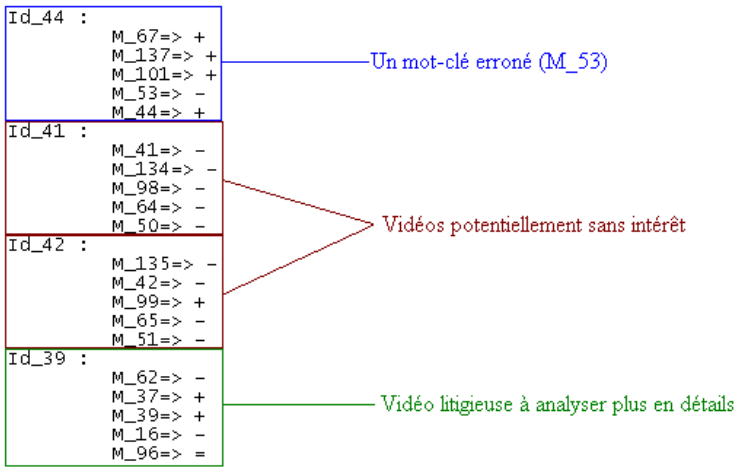


FIG. 7.4 – Analyse des comportement en fonction des mots-clés

La table suivante (7.1) présente le résumé global de nos expérimentations, elle met en rapport le nombre de vidéos en fonction de leur nature (bonne, erreur mot-clé, mauvaise qualité) pour les valeurs réelles que nous avons définies et les valeurs que nous avons obtenues après analyse avec notre technique.

nature de la vidéo	réel	observé
bien exploitée	34	38
un mot-clé erroné	7	5
mauvaise qualité	11	8
indéterminée	×	1

TAB. 7.1 – Analyse des vidéos en fonction de leur nature

À partir de ces valeurs, nous pouvons construire le tableau de contingence correspondant (table 7.2). Dans celui-ci, nous considérons simplement bonnes vidéos et vidéos à problème (avec un mot-clé ou sans intérêt).

Ce tableau permet d'évaluer la Précision (7.1), le Rappel (7.2) et la mesure F_1 (7.3)

Vrais Positifs (VP)	bonnes vidéos détectées comme bonnes	34
Faux Positifs (FP)	vidéos à problème détectées comme bonnes	4
Vrais Négatifs (VN)	vidéos à problème détectées comme à problème	14
Faux Négatifs (FN)	bonnes vidéos détectées comme à problème	0

TAB. 7.2 – Tableau de contingence

de notre technique :

$$\text{Précision} = \frac{\text{VP}}{\text{VP} + \text{FP}} = 0.895 \tag{7.1}$$

$$\text{Rappel} = \frac{\text{VP}}{\text{VP} + \text{FN}} = 1 \tag{7.2}$$

$$F_1 = \frac{2 \times \text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} = 0.945 \tag{7.3}$$

Nous observons donc dans le cas de notre expérimentation sur des données simulées des très bonnes valeurs de Rappel et de Précision. Notre approche permet de découvrir la majeure partie des erreurs relatives à l'indexation sans jamais considérer une bonne vidéo comme une vidéo à problème, simplifiant ainsi le travail final d'un administrateur qui utilisera ces suggestions pour modifier l'indexation des vidéos.

Bibliographie

[UBD06] T. Urruty, F. Belkouch et C. Djeraba : Efficient indexing for high dimensional data : applications to a video search tool. *Twelfth international conference on knowledge discovery and data mining. Philadelphie, Pennsylvanie, USA*, 20-23 Août 2006.

[UBDS07] T. Urruty, F. Belkouch, C. Djeraba et D. A. Simovici : RPyR : nouvelle structure d'indexation avec classification par projections aléatoires. *Vingt-troisièmes journées informatique des organisation et systèmes d'information et de décision. Perros-Guirec, France*, 22-25 Mai 2007.

Chapitre 8

Conclusion

Sommaire

8.1 Contribution

Notre travail tout au long de ces trois années a consisté en la modélisation et l'analyse des comportements des utilisateurs exploitant des données vidéo. Sa principale originalité réside en l'analyse à deux niveaux des données. Cette approche combine usage intra-vidéo et usage inter-vidéo pour générer des profils de visite sur un moteur de recherche vidéo. Nous proposons une nouvelle approche qui définit différents types de données de *log* collectées par le moteur de recherche. Le premier concerne la manière dont un utilisateur visionne les séquences vidéo (lecture, pause, avance rapide, stop...). À ce niveau que nous appelons *intra-vidéo*, nous définissons le *visionnage d'une vidéo* comme unité de comportement. L'autre type de *log* correspond aux transitions entre les différents visionnages. Cette partie regroupe l'écriture des requêtes, la présentation des résultats et les visionnages successifs des séquences vidéo. À ce niveau plus général que nous appelons *inter-vidéo*, nous introduisons la *session* comme unité de comportement.

Un comportement intra-vidéo est modélisé par un modèle de Markov du premier ordre non caché. Ce modèle est construit en utilisant les différentes actions proposées aux utilisateurs lors des visionnages (lecture, pause, avance rapide, retour rapide, saut, stop). Nous proposons une technique de regroupement de ces modèles (*K-models*). Cette méthode est une adaptation de la technique classique des *K-moyennes* adaptée à l'utilisation de modèles en lieu et place des moyennes. Nous caractérisons ainsi plusieurs comportements type (lecture totale, lecture partielle, survol...). Grâce à ces comportements type, nous sommes à même de savoir quelle fut l'utilité ou l'importance d'une séquence vidéo lors d'une visite (si elle est la plus importante ou seulement le résultat d'une recherche infructueuse...)

Un comportement inter-vidéo est modélisé par une session. Cette session est une séquence ordonnée des visionnages des séquences vidéo. Chaque visionnage est représenté par l'identifiant de la vidéo et le modèle de comportement qui lui correspond pour cette visite. Pour regrouper ces sessions, nous nous sommes basés sur une technique de regroupement hiérarchique ascendante que nous avons adaptée à ces données particulières. Cette dernière présente la particularité de traiter des classes représentées par plusieurs

sous-séquences communes. La majeure partie des travaux sur la modélisation de sessions ne travaille qu'avec une seule sous-séquence représentative. Cela induit une importante perte d'informativité lorsque deux sessions ont plusieurs sous-séquences communes de même longueur et qu'il faut en choisir une et de fait perdre l'information portée par les autres. Notre approche permet de pallier cette limite.

Ces travaux doivent permettre de mieux comprendre les utilisateurs et d'améliorer les outils d'accès aux bases de données vidéo, par exemple la découverte des erreurs d'indexation ou la proposition aux utilisateurs de données en fonction de leur profil de visite comme nous l'avons introduit dans notre travail.

8.2 Perspectives

8.2.1 Enrichissement du modèle de comportement grâce au suivi du regard

8.2.1.1 Le suivi du regard

Les systèmes de suivi de regard permettent de connaître précisément la position du regard d'un individu. Ces zones particulières où notre regard se pose sont très pertinentes en terme de comportement. Ainsi par la connaissance de la position du regard, on peut savoir à chaque instant où se focalise l'attention visuelle d'un individu. L'intérêt de disposer de telles informations est multiple :

- pouvoir évaluer la qualité ergonomique d'interfaces homme-machine,
- comprendre les stratégies d'inspection de conducteurs automobiles ou de pilotes aéronautiques,
- aider à estimer l'efficacité d'un marketing publicitaire.

Pour ce faire, l'identification des composantes du mouvement oculaire (fixations et saccades) constitue une étape capitale dans le processus d'analyse du comportement visuel. Elles fournissent en effet les éléments de bases qui seront par la suite exploités dans différentes applications : dans un cadre analytique, pour faciliter l'interprétation psychologique et cognitive du comportement visuel ou bien dans un cadre interactionnel, comme données d'entrée d'une interface homme-machine. Les périodes de fixations correspondent au temps pendant lequel l'oeil est quasiment stationnaire, permettant ainsi un traitement de l'information perçue par le cerveau. Les saccades quant à elles correspondent aux périodes rapides durant lesquelles l'oeil se déplace d'une fixation à une autre, et durant lesquelles aucun traitement cognitif n'est possible. C'est donc durant les périodes de fixations que le cerveau donne une sémantique à l'image regardée. C'est pour cela que l'étude des fixations et des saccades est essentielle pour la compréhension et l'interprétation du mouvement oculaire.

Nous n'aborderons pas ici plus en détails les techniques de suivi du regard, nous avons décrit notre approche dans [LMI+06].

8.2.1.2 Enrichissement du modèle de comportement

Initialement, le suivi du regard est utilisé pour découvrir les zones d'intérêt dans des images. Appliquer une telle approche à la vidéo est très complexe et n'est pas notre objectif. En effet, la limite majeure est la difficulté d'annoter un grand nombre de vidéos et d'en repérer les zones d'intérêt, le nombre de frames ou de scènes à traiter étant très important. Si cela est envisageable à notre échelle sur un petit nombre de vidéos, cela est impossible pour une base vidéo d'exploitation.

Le suivi du regard reste toutefois intéressant pour enrichir notre modèle. En observant une personne visionner une vidéo, on remarque très vite que son attention peut être très variable : une vidéo peut être lue totalement par un utilisateur dont le regard se porte sur la vidéo comme par un utilisateur n'étant pas du tout intéressé et dont le regard se porte sur autre chose.

Nous avons défini deux états comme étant potentiellement dépendants de l'attention : les états $PLAY$ et $PAUSE$. En effet, une lecture peut être faite normalement comme elle peut continuer après qu'un utilisateur s'en soit détourné sans la clore. De même pour une $PAUSE$, elle peut être liée à un utilisateur dont l'attention a été détournée, comme à un utilisateur scrutant une partie précise de la vidéo. Les autres états (FWD, RWD, $JUMP$) sont par défaut actifs et n'en sont pas dépendants.

Le suivi du regard apportera certainement une plus-value intéressante à notre modélisation des utilisateurs. Même si notre objectif n'est pas d'en utiliser toutes les possibilités en traçant la totalité des mouvements et en repérant les zones d'intérêt dans la vidéo, enrichir le modèle en discriminant les états $PLAY$ et $PAUSE$ comme actifs (quand l'utilisateur regarde effectivement la vidéo) ou inactifs (quand l'utilisateur regarde une autre partie de l'écran ou se détourne totalement) permettra une analyse plus détaillée. Par exemple, cela permettra de savoir si une vidéo publicitaire, jouée automatiquement est effectivement regardée ou non ou encore si une pause est relative à la recherche d'un détail dans l'image ou à une pause de l'utilisateur.

Dans la suite de nos travaux, nous pensons pouvoir arriver à suivre les regards via des webcams et pouvoir l'appliquer à grande échelle, ce qui n'était pas réalisable aujourd'hui.

8.2.2 Amélioration des moteurs de recherche vidéo

Comme nous l'avons présenté au chapitre (7), l'utilisation des résultats de l'analyse du comportement des utilisateurs est intéressante pour découvrir et corriger les erreurs d'indexation des données de manière automatique ou semi-automatique et pour proposer aux utilisateurs des vidéos alternatives en fonction de leur profil (ie. les vidéos qu'ils ont déjà visionnées). Nous n'avons pas pu cependant, dans le cadre de la thèse, mener des expériences en conditions réelles sur l'impact de ces méthodes sur la qualité des moteurs de recherche.

Nous pensons pouvoir valider une telle approche en la testant sur des données réelles et par la suite optimiser ces techniques pour proposer aux utilisateurs des outils de recherche plus performants. Dans cette perspective, nous envisageons d'utiliser d'autres techniques de regroupement que le regroupement hiérarchique utilisé pour la modélisation inter-vidéo. En effet, si nous sommes amenés à traiter de grands ensembles de données, cette approche pourrait rapidement se montrer trop complexe. D'autres approches, à la complexité moins élevée, devront dans ce cas être appliquées à notre propos.

Nous pensons finalement que l'utilisation de l'expérience des utilisateurs peut apporter une réelle plus-value dans le cadre de la recherche de données vidéo.

Publications

Revue internationale

[Mon07] S. Mongy. A Study on video viewing behavior. *International journal of parallel, emergent and distributed systems. Taylor & Francis.* 22 :163-172. Janvier 2007

Chapitre de livre

[MBD06b] S. Mongy, F. Bouali et C. Djeraba. Analyzing user's behavior on a video database. *Multimedia data mining and knowledge discovery. Edited by V. Petrushin & L. Khan. Springer.* Chapter 23, pages 479-492. Decembre 2006.

[MBD06a] S. Mongy, F. Bouali et C. Djeraba. Video usage mining. *Encyclopedia of multimedia. Edited by Borko Furht. Springer.* pages 928-934. 2006.

Conférence internationale

[MSD07] S. Mongy, D. Simovici et C. Djeraba. On clustering users' behaviors in video sessions. *International conference on data mining. Monte Carlo Resort, Las Vegas, Nevada, USA.* Juin 2007.

[MD07] S. Mongy et C. Djeraba. A study on video viewing behavior. Application to movie trailer miner. *IS&T/SPIE nineteenth annual symposium electronic imaging science and technology. San Jose, Californie, USA.* Janvier 2007.

[LMI+06] S. Lew, S. Mongy, N. Ihaddadene, C. Djeraba et D. Simovici. Eye/gaze tracking in web, image and video documents. *Fourteenth ACM international conference on multimedia. Santa Barbara, Californie, USA.* pages 481–482. 23-27 Octobre 2006.

Atelier international

[MBD05] S. Mongy, F. Bouali et C. Djeraba. Analyzing user's behavior on a video database. *Sixth international workshop on multimedia data mining 'Mining integrated media and complex data'. Chicago, Illinois, USA.* pages 95-100. Août 2005.

Conférence nationale

[Mon06] S. Mongy. Analyse du comportement des utilisateurs exploitant une base de données vidéo. *Sixièmes journées francophones extraction et gestion de connaissances. ENIC Telecom Lille 1. Villeneuve d'Ascq, France. RNTI-E-6 Cépaduès-Éditions.* pages 391-402. 17-20 Janvier 2006

[MD06] S. Mongy et C. Djeraba. Extraction de comportements liés à la lecture de bandes-annonces cinématographiques. *Vingt-deuxièmes journées bases de données avancées. Lille Grand Palais, Lille, France.* 17-20 Octobre 2006

Atelier national

[MD05] S. Mongy et C. Djeraba. Modélisation des comportements sur l'exploitation d'une base de données vidéo. *Atelier fouille de données complexes dans un processus d'extraction des connaissances, cinquièmes journées francophones extraction et gestion de connaissances. Université René Descartes Paris 5, Paris, France.* 18 Janvier 2006.

Bibliographie

[Ale] Alexa : http ://www.alexa.com/.

[ASP00] S. Acharya, B. Smith et P. Parnes : Characterizing user access to videos on the world wide web. *Multimedia computing and networking. Californie, USA*, 2000.

[Bes95] A. Bestavros : Using speculation du reduce server load and service time on the WWW. *Fourth ACM international conference on information and knowledge management. Baltimore, Maryland, USA*, 1995.

[BET99] P. Branch, G. Egan et B. Tonkin : Modeling interactive behaviour of a video based multimedia system. *IEEE international conference on communications*, 2:978–982, Juin 1999.

[BG01] A. Banerjee et J. Ghosh : Clickstream clustering using weighted longest common subsequences. *Web mining workshop at the 1st SIAM conference on data mining. Chicago, Illinois, USA*, Avril 2001.

[BHR00] L. Bergroth, H. Hakonen et T. Raita : A survey of longest common subsequence algorithms. *Seventh international symposium on string processing and information retrieval. ISBN :0-7695-0746-8*, 2000.

[Bré98] P. Brémaud : Markov chains. Gibbs fields, Monte Carlo simulation, and queues. *Springer texts in applied mathematics. Springer*, 1998.

[CHM+00] I. Cadez, D. Heckerman, C. Meek, P. Smyth et S. White : Visualization of navigation patterns on a web site using model based clustering. *Sixth*

ACM International Conference on Knowledge Discovery and Data Mining. Boston, Massachusetts, USA, pages 280–284, 2000.

[Chu67] K. Chung : Markov chains with stationary transition probabilities. *Deuxième édition. Springer-Verlag, Berlin-Heidelberg-New York*, 1967.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest et C. Stein : Introduction to algorithms. *Deuxième édition, MIT press*, 2001.

[CSS00] K. Charter, J. Schaeffer et D. Szafron : Sequence alignment using FastLSA. *International conference on mathematics and engineering techniques in medicine and biological sciences*, 2000.

[DB79] D.L. Davies et D.W. Bouldin : A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 1, 1979.

[Do03] M. N. Do : Fast approximation of Kullback–Leibler distance for dependence trees and hidden Markov models. *IEEE signal processing letters*, 10, Avril 2003.

[DSSKT94] J. Day-Sircar, J. D. Salehi, J. K. Kurose et D. Towsley : Providing VCR capabilities in large-scale video servers. *ACM international conference on mulimedia. San-Francisco, Californie, USA*, 1994.

[Duf97] M. Duflo : Random iterative models. *Applications of Mathematics. Springer, Berlin*, 34, 1997.

[FS97] P. Faudemay et C. Seyrat : Intelligent delivery of personalised video programmes from a video database. *international workshop on database and expert systems applications*, pages 172–177, 1997.

[FS02] E. Feinberg et A. Schwartz : Handbook of Markov decision processes. Methods and application. *International series in operation research & management science. Kluwer Academic Publishers. Boston, Massachussets, USA*, 2002.

[GRS98] S. Guha, R. Rastogi et K. Shim : An efficient clustering algorithm for large databases. *ACM international conference on management of data. Seattle, Washington, USA*, 1998.

[GS97] C. M. Grinstead et J. L. Snell : Introduction to probability. *American mathematical society. ISBN :978-0821807491*, 1997.

[HNK+04] L. Hollink, G. P. Nguyen, D. Koelma, A. T. Schreiber et M. Worring : User strategies in video retrieval : A case study. *International conference on image and video retrieval. Springer ISSN :0302-9743*, 3115:6–14, 2004.

[HSGG99] L. He, E. Sanocki, A. Gupta et J. Grudin : Auto-summarization of audio-video presentations. *Seventh ACM international conference on multimedia. Orlando, Floride, USA*, 1999.

[HSGG00] L. He, E. Sanocki, A. Gupta et J. Grudin : Comparing presentation summaries : slides vs. reading vs. listening. *SIGCHI conference on human factors in computing systems. La Haye, Pays-Bas*, 2000.

[HWV01] B. Hay, G. Wets et K. Vanhoof : Clustering navigation patterns on a website using a sequence alignment method. *Seventeenth international joint conference on artificial intelligence. Seattle, Washington, USA*, Août 2001.

[JZM04] X. Jin, Y. Zhou et B. Mobasher : Web usage mining based on probabilistic latent semantic analysis. *ACM international conference on knowledge discovery and data mining. Seattle, Washington, USA*, Août 2004.

[KL51] S. Kullback et R. A. Leibler : On information and sufficiency. *The annals of mathematical statistics*, 22, Mars 1951.

[KMPZ04] R. Kohavi, L. Mason, R. Parekh et Z. Zheng : Lessons and challenges from mining retail e-commerce data. *Machine learning journal*, 2004.

[Kri93] V. Krishnamurthy : On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure. *IEEE Transactions on signal processing*, 41, 1993.

[LGD+05] L.Hollink, G.P.Nguyen, D.Koelma, A.Th.Schreiber et M.Worring : Assessing user behaviour in news video retrieval.

IEEE international conference on vision, image and signal processing, 152, 2005.

[LGS+00] F. C. Li, A. Gupta, E. Sanocki, L. He et Y. Rui : Browsing digital video. *SIGCHI conference on human factors in computing systems. La Haye, Pays-Bas. ACM Press ISBN :1-58113-216-6*, 2000.

[LLQW96] V. O. K. Li, W. Liao, X. Qiu et E. W. M. Wong : Performance model of interactive video-on-demand systems. *IEEE Journal on Selected Areas in Communications*, 14, Août 1996.

[LMI+06] S. Lew, S. Mongy, N. Ihaddadene, C. Djeraba et D. Simovici : Eye/gaze tracking in web, image and video documents. *Fourteenth annual ACM international conference on multimedia. Santa Barbara, Californie, USA*, pages 481–482, 23-27 Octobre 2006.

[Mac66] J. MacQueen : Some methods for classification and analysis of multivariate observations. *Fifth Berkeley symposium. University of California Press 1*, pages 281–297, 1966.

[MDL+00] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Y. Sun et J. Wiltshire : Discovery of aggregate usage profiles for web personalization. *KDD workshop on web mining and web usage analysis. Boston, Massachussets, USA*, 2000.

[Mon06] S. Mongy : Analyse du comportement des utilisateurs exploitant une base de vidéos. *Sixièmes journées francophones extraction et gestion de connaissances, RNTI-E-6 Cépaduès-Editions. Villeneuve d'Ascq, France*, Janvier 2006.

[MPB98] L. Malziak, P. Priouret et P. Baldi : Martingales et chaînes de Markov. *Hermann*, 1998.

[NAMO00] A. Nakamura, N. Abe, H. Matoba et K. Ochiai : Automatic recording agent for digital video server. *ACM international conference on multimedia*, pages 57–66, 2000.

[Neta] Netflix : http ://www.netflixprize.com/.

[Netb] NETFLIX : http ://www.netflix.com/.

[PM96] V. PADMANABHAN et J. MOGUL : Using predictive prefetching to improve WWW latency. *ACM international conference on the special interest group on data communication. Université de Stanford, Californie, USA*, 1996.

[PP99] P. PIROLLI et J. PITKOW : Distributions of surfers' paths through the world wide web : empirical characterizations. *World wide web journal*, 2:29–45, 1999.

[PPPS03] D. PIERRAKOS, G. PALIOURAS, C. PAPATHEODOROU et C. SPYROPOULOS : Web usage mining as a tool for personalization : A survey. *User modeling and user-adapted interaction journal*, 13:311–372, 2003.

[RAC04] Z. RACHED, F. ALAJAJI et L. Lorne CAMPBELL : The Kullback-Leibler divergence rate between Markov sources. *IEEE transactions on information theory. ISSN :0018-9448*, 50:917–921, Mai 2004.

[RM02] A. I REUTHER et D. G. MEYER : The effect of personality type on the usage of a multimedia engineering education system. *Frontiers in education. Seattle, Washington, USA*, 1:T3A7–T3A12, Novembre 2002.

[RT99] S. RAY et R. H. TURI : Determination of number of clusters in K-means clustering and application in colour image segmentation. *Fourth international conference on advances in pattern recognition and digital techniques. Calcutta, Inde*, Decembre 1999.

[Sar00] R. SARUKKAI : Link prediction and path analysis using markov chains. *Ninth international world wide web wonference. Amsterdam, Pays-Bas*, Mai 2000.

[SD08] D. SIMOVICI et C. DJERABA : Mathematical tools for data mining. Set theory, partial orders, combinatorics. *À Paraître prochainement. Springer*, 2008.

[SMP01] T. SYEDA-MAHMOOD et D. PONCELEON : Learning video browsing behavior and its application in the generation of video previews. *Ninth ACM conference on multimedia. Ottawa, Canada*, Octobre 2001.

[SPAP99] S. Srinivasan, D. B. Ponceleon, A. Amir et D. Petkovic : "what is in that video anyway ?" In bearch of better browsing. *IEEE international conference on multimedia computing and systems*, 1:388–393, 1999.

[SV95] P. J. Shenoy et H. M. Vin : Efficient support for scan operations in video servers. *ACM international conference on mulimedia. San-Francisco, Californie, USA*, 1995.

[SY01] R. Srikant et Y. Yang : Mining web logs to improve website organization. *Tenth international world wide web conference. Hong-Kong*, Mai 2001.

[UBD06] T. Urruty, F. Belkouch et C. Djeraba : Efficient indexing for high dimensional data : applications to a video search tool. *Twelfth international conference on knowledge discovery and data mining. Philadelphie, Pennsylvanie, USA*, 20-23 Août 2006.

[UBDS07] T. Urruty, F. Belkouch, C. Djeraba et D. A. Simovici : RPyR : nouvelle structure d'indexation avec classification par projections aléatoires. *Vingt-troisièmes journées informatique des organisation et systèmes d'information et de décision. Perros-Guirec, France*, 22-25 Mai 2007.

[W3C] W3C : http ://www.w3.org/Style/XSL.

[WSM02] B. Westphal et T. Syeda-Mahmood : On learning video browsing behavior from user interactions. *Eleventh international world wide web conference. Honolulu, Hawaï, USA*, Mai 2002.

[WZ02] W. Wang et O. R. Zaïane : Clustering web sessions by sequence alignment. *Thirteenth international workshop on database and expert systems applications. Aix-en-Provence, France*, 2002.

[Yca02] B. Ycart : Modèles et algorithmes markoviens. *Mathématiques & Applications. Springer, Berlin*, 39, 2002.

[YJGMD96] T. Yan, M. Jacobsen, H. Garcia-Molina et U. Dayal : From user access patterns to dynamic hypertext linking. *Computer networks & ISDN Systems*, 28:1007–1014, 1996.

[YMNZ03] B. Yu, W. Ma, K. Nahrstedt et H. Zhang : Video summarization based on user log enhanced link analysis. *ACM international conference on mulimedia. Berkeley, Californie, USA*, Novembre 2003.

[YZZZ06] H. Yu, D. Zheng, B. Y. Zhao et W. Zheng : Understanding user behavior in large-scale video-on-demand systems. *EuroSys conference. Louvain, Belgique. ACM ISSN :1-59593-322-0*, 2006.

[ZRL96] T. Zhang, R. Ramakrishnan et M. Livny : Birch : an efficient data clustering method for very large databases. *ACM special interest group on management of data conference*, 1996.

Annexe A

Outils et développements

Sommaire

A.1 Présentation de l'outil de visionnage

Pour mener nos expériences, nous avons développé un outil de recherche et de visionnage de documents vidéo. Cet outil est très générique et correspond aux différents outils de recherche que l'on peut habituellement trouver. Il se décompose en un module de recherche (figure A.1), une fiche d'information sur les vidéos (figure A.2) et un lecteur vidéo (figure A.4).

Nous avons choisi de travailler avec des bandes-annonces de cinéma, nous verrons par la suite les raisons de ce choix mais il est à noter que l'outil fonctionne avec tout type de vidéo.

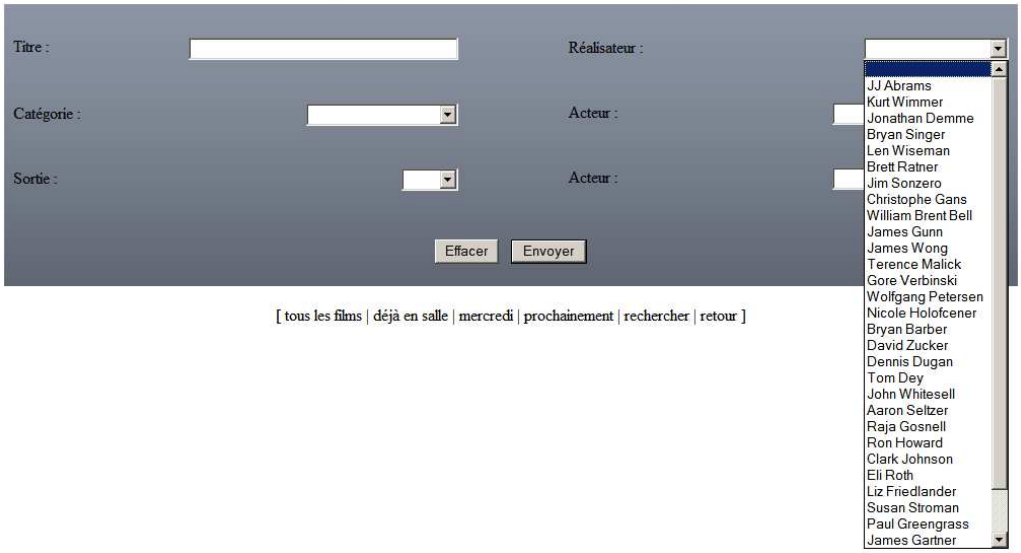


FIG. A.1 – Outil de recherche de vidéos

Chaque vidéo est stockée avec une image la représentant et un fichier XML dans lequel sont enregistrés les attributs qui lui sont propres (figure A.3). L'outil propose, à partir de ces données, d'effectuer des recherches sur les différents attributs et lui retourne une liste de résultats triée selon la proximité avec la requête. L'utilisateur peut finalement

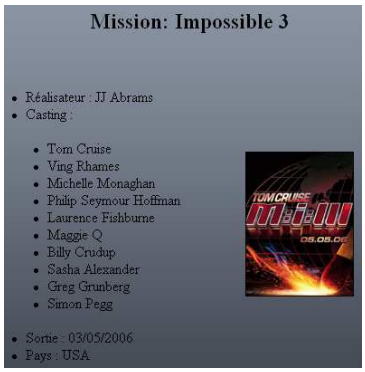


FIG. A.2 – Fiche détaillée

```
<trailer>
      <art>mi3.bmp</art>
      <file>mission_impossible_3.mov</file>
      <title>Mission: Impossible 3</title>
      <director>JJ Abrams</director>
      <casting>
            <actor>Tom Cruise</actor>
            <actor>Ving Rhames</actor>
            <actor>Michelle Monaghan</actor>
            <actor>Philip Seymour Hoffman</actor>
            <actor>Laurence Fishburne</actor>
            <actor>Maggie Q</actor>
            <actor>Billy Crudup</actor>
            <actor>Sasha Alexander</actor>
            <actor>Greg Grunberg</actor>
            <actor>Simon Pegg</actor>
      </casting>
      <country>USA</country>
      <classification>action</classification>
      <onscreen>03/05/2006</onscreen>
      <outscreen>01/01/2010</outscreen>
</trailer>
```

FIG. A.3 – Attributs d'une vidéo

FIG. A.4 – Lecteur de vidéos

visionner les différentes vidéos. Le module de lecture est lui aussi classique et ne propose aucune particularité par rapport aux différents lecteurs couramment utilisés.

C'est donc à partir de ce moteur de recherche que nous collectons les données relatives aux actions des utilisateurs que nous allons présenter dans la section suivante.

A.2 Définition et collecte des données de visionnage

Définition des actions réalisables

Dans cette section, nous allons nous intéresser spécifiquement aux actions et aux données utilisateurs relatives à la lecture des vidéos. Les données relatives aux actions réalisées pendant une session entre la lecture de vidéos sont présentées au chapitre (6).

Notre objectif est donc ici de collecter l'ensemble des actions qu'ont réalisées les utilisateurs lors du visionnage des différentes vidéos. Comme le montre la figure A.4, le lecteur est reprend les actions de base que l'on retrouve dans la grande majorité des lecteurs communs. Ce choix est évident, nous espérons à terme pouvoir exploiter nos résultats à grande échelle et donc, intégrer notre analyse à tous les lecteurs vidéo.

Nous retrouvons donc les actions de base suivantes :

- $PLAY$
- $PAUSE$
- FFW
- RWD. En fonction des lecteurs et des vidéos, ces deux dernières actions sont légèrement différentes. En effet, la vitesse de progression peut varier selon les cas.
- $JUMP$. Cette action correspond à la navigation dans la vidéo directement avec la barre de progression.
- $STOP$. Le $STOP$ est la dernière action réalisée sur la vidéo. Elle clôt le visionnage définitivement.

Définition du langage

Sur la base des actions précédemment définies, toutes les actions des utilisateurs sont collectées dans un fichier de *logs*. Pour créer ce fichier, nous avons défini une grammaire XML adaptée. Chaque action est stockée avec son identifiant de session, son identifiant vidéo, son type, son instant de début, sa longueur et le moment de début et de fin dans la vidéo. Un exemple est proposé dans la figure (A.5) qui montre trois actions réalisées sur une même vidéo dans une session et la DTD correspondante dans la figure (A.6).

```
<log>
    <action scene="Scene1" session="Session2">
            <type>play</type>
            <time>1108948520</time>
            <length>5</length>
            <start>0</start>
            <end>5</end>
    </action>
    <action scene="Scene1" session="Session2">
            <type>rewind</type>
            <time>1108948525</time>
            <length>3</length>
            <start>5</start>
            <end>0</end>
    </action>
    <action scene="Scene1" session="Session2">
            <type>play</type>
            <time>1108948528</time>
            <length>9</length>
            <start>0</start>
            <end>9</end>
    </action>
</log>
```

FIG. A.5 – Données de *logs* de visionnage

```
<?xml version='1.0' encoding='UTF-8'?>

<!ELEMENT log (action)*>
<!ATTLIST log
    xsi:schemaLocation CDATA #IMPLIED
    xmlns CDATA #IMPLIED
    xmlns:xsi CDATA #IMPLIED
  >

<!ELEMENT action (end|start|length|time|type)*>
<!ATTLIST action
    idsession CDATA #IMPLIED
    idsequence CDATA #IMPLIED
  >

<!ELEMENT type (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT end (#PCDATA)>
```

FIG. A.6 – DTD de visionnage

Les identifiants de vidéo et de sessions sont nécessaires pour pouvoir reconstruire les différents visionnages. En effet, notre outil fonctionne sur un modèle client/serveur et plusieurs utilisateurs peuvent visionner des vidéos en parallèle. Cela induit un mélange des données dans le fichier de *log*. Il est donc indispensable de conserver ces informations pour reconstruire chacun des visionnages. Notons que nous ne conservons pas ici l'identifiant de l'utilisateur car nous ne nous intéressons pas à ce niveau d'information. Le niveau le plus global que nous analysons est celui de la session.

Enfin, nous ne stockons pas non plus d'identifiant de visionnage. Cela n'est pas gênant car, chaque visionnage se termine systématiquement par l'action STOP et nous pouvons donc les discriminer, même lorsqu'un utilisateur visionne deux fois la même vidéo dans une seule session.

Reconstruction des visionnages

Pour reconstruire les visionnages et pouvoir ensuite les traiter et les regrouper (section 5.2), il nous faut donc tout d'abord regrouper les actions par visionnage. Nous avons choisi de réaliser cela à l'aide de feuilles de style XSLT [W3C]. Cette technologie a l'avantage d'être adaptée au traitement de grands fichiers XML et de les traiter rapidement. La

figure (A.7) présente la feuille de style que nous avons créée. Elle permet de réaliser effectivement le regroupement des actions par visionnages.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="xml" indent="yes" encoding="ISO-8859-1" />

  <xsl:key name="act_by_scn" match="action" use="@scene" />

  <xsl:template match="log">
    <xsl:element name="mod_scn">
      <xsl:for-each
          select="action[count(. | key('act_by_scn', @scene)[1]) = 1]">
        <xsl:sort select="@scene" />
        <xsl:element name="scene">
          <xsl:attribute name="id">
            <xsl:value-of select="@scene" />
          </xsl:attribute>
          <xsl:for-each select="key('act_by_scn', @scene)">
            <xsl:sort select="@session" />
            <xsl:apply-templates select="." />
          </xsl:for-each>
        </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>

  <xsl:template match="action">
    <xsl:copy>
      <xsl:attribute name="session">
        <xsl:value-of select="@session" />
      </xsl:attribute>
      <xsl:apply-templates select="node()" />
    </xsl:copy>
  </xsl:template>

  <xsl:template match="action/*">
    <xsl:copy-of select="." />
  </xsl:template>

</xsl:transform>
```

FIG. A.7 – Feuille de transformation XSLT

Après transformation, nous obtenons le fichier XML présenté figure (A.8).

Nous retrouvons bien dans ce fichier les données dans le format que nous dési-

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mod_scn>
  <scene id="Scene01">
    <action>
      <type>play</type>
      <time>1107366225</time>
      <length>3</length>
      <start>0</start>
      <end>3</end>
    </action>
    <action>
      <type>ffw</type>
      <time>1107366228</time>
      <length>6</length>
      <start>3</start>
      <end>21</end>
    </action>
    <action>
      <type>play</type>
      <time>1107366234</time>
      <length>15</length>
      <start>21</start>
      <end>36</end>
    </action>
    <action>
      <type>stop</type>
      <time>1107366249</time>
      <length>0</length>
      <start>36</start>
      <end>0</end>
    </action>
  </scene>
  <scene id="Scene01">
    <action session="Session01">
      <type>play</type>
      <time>1107366262</time>
      <length>31</length>
      <start>0</start>
      <end>41</end>
    </action>
    <action session="Session01">
      <type>stop</type>
      <time>1107366293</time>
      <length>0</length>
      <start>41</start>
      <end>0</end>
    </action>
  </scene>
</mod_scn>
```

FIG. A.8 – Actions par visionnages

rons, c'est-à-dire le regroupement des actions chronologiquement par visionnage (balise *<scene>*).

A.3 Construction des sessions

L'outil de base qu'exploitent les utilisateurs est le même que celui présenté en annexe (A.1) dans les figures (A.1), (A.2) et (A.4). Notre objectif est maintenant d'analyser l'enchaînement dans le visionnage des différentes vidéos, c'est-à-dire « quelles vidéos sont accédées et dans quel ordre ? ».

Regroupement des actions

Tout comme pour l'analyse intra-vidéo, la première tâche à réaliser est de classer les données de *log* pour pouvoir les traiter. Ici, il faut donc les regrouper par sessions de manière chronologique pour ensuite reconstruire les visionnages de chacune des sessions. Nous rappelons ici qu'une session correspond à une visite d'un utilisateur sur le moteur de recherche, visite composée de différentes recherches et visionnages réalisées de manière continue, sans pause entre les différents accès.

La base des données tracées restent les actions que réalisent les utilisateurs (ie. $PLAY$, $PAUSE$, $JUMP$...). Comme le montre la figure (A.9), chacune des actions est répertoriée avec un identifiant de sessions (attribut *session* de la balise *action*). Cet identifiant est simplement généré au moment où un utilisateur se connecte et sera conservé tout au long de sa visite. Ainsi, nous pouvons discriminer les différentes sessions. La figure (A.10) présente la DTD correspondant à la collecte des actions.

À partir de ces données, nous regroupons les actions par session et par visionnage. L'objectif étant de construire des séquences de visionnages de vidéos, cette phase de transformation est nécessaire. Comme lors du travail sur les comportements intra-vidéo, nous utilisons une feuille de transformation XSLT [W3C]. À nouveau son avantage est d'être parfaitement adapté au travail sur de grands fichiers XML. La figure (A.11) présente le fichier XML obtenu après transformation.

Nous voyons dans cette figure que les données sont désormais triées pour être analysées en fonction des sessions, chacune étant composée d'une suite chronologique de visionnages.

```
<log>
    <action scene="Scene1" session="Session2">
            <type>play</type>
            <time>1108948520</time>
            <length>5</length>
            <start>0</start>
            <end>5</end>
    </action>
    <action scene="Scene1" session="Session2">
            <type>rewind</type>
            <time>1108948525</time>
            <length>3</length>
            <start>5</start>
            <end>0</end>
    </action>
    <action scene="Scene1" session="Session2">
            <type>play</type>
            <time>1108948528</time>
            <length>9</length>
            <start>0</start>
            <end>9</end>
    </action>
</log>
```

FIG. A.9 – Données de *logs* de visionnage

```
<?xml version='1.0' encoding='UTF-8'?>

<!ELEMENT log (action)*>
<!ATTLIST log
    xsi:schemaLocation CDATA #IMPLIED
    xmlns CDATA #IMPLIED
    xmlns:xsi CDATA #IMPLIED
  >

<!ELEMENT action (end|start|length|time|type)*>
<!ATTLIST action
    idsession CDATA #IMPLIED
    idsequence CDATA #IMPLIED
  >

<!ELEMENT type (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT end (#PCDATA)>
```

FIG. A.10 – DTD de visionnage

Transformation en séquences

Comme nous l'avons introduit précédemment, nous devons extraire des données de *logs* des séquences d'identifiants vidéo pondérés par un comportement propre au visionnage. Pour cela, nous proposons d'adjoindre à chaque identifiant un vecteur de comportements. Ce vecteur représente les probabilités d'appartenance du visionnage à chacune des classes extraites lors de l'extraction des comportements type intra-vidéo. Nous avons choisi de conserver l'ensemble des probabilités relatives à tous les modèles et non pas uniquement le modèle le plus proche du visionnage car ceci permet une comparaison plus fine des données sans trop pénaliser le temps de traitement. En pratique, il y aura k comparaisons au lieu d'une seule, k étant le nombre de comportements intra-vidéo type.

Soient $m_1, m_2, ...m_k$ les modèles de comportements extraits de la phase d'analyse intra-vidéo, soit s la séquence des actions observées pendant le visionnage i, le vecteur de comportement v_i lié au visionnage i est défini selon la formule (A.1)

$$v_i = (P(s|m_1), P(s|m_2), ...P(s|m_k)) \tag{A.1}$$

où $P(s|m_i)$ représente la probabilité que la séquence s ait été générée par m_i.

FIG. A.11 – Données regroupées en sessions

Pour obtenir ces vecteurs à partir des données de *logs* (A.9), nous allons, pour chacun des visionnages, reconstruire la séquence des actions et calculer les probabilités par rapport aux modèles intra-vidéo extraits sur l'ensemble des sessions. Ensuite, nous allons collecter dans un fichier global l'ensemble des sessions chacune étant une séquence de couples (identifiant vidéo, vecteur de comportement). Une fois toutes les données transformées, nous obtenons un fichier similaire à celui présenté figure (A.12) dans lequel nous voyons des sessions reconstruites après une analyse intra-vidéo ayant généré 4 comportements type.

```
session01 [(video01, (0.75, 0.42, 0.26, 0.12)) (video43, (0.51, 0.63, 0.31, 0.11)) ... (video16, (0.81, 0.44, 0.15, 0.08))]
session02 [(video43, (0.12, 0.15, 0.26, 0.72)) (video14, (0.61, 0.33, 0.17, 0.09)) ... (video25, (0.42, 0.69, 0.31, 0.12))]
...
sessionN  [(video31, (0.87, 0.38, 0.19, 0.14)) (video07, (0.91, 0.27, 0.09, 0.06)) ... (video04, (0.14, 0.11, 0.15, 0.84))]
```

FIG. A.12 – Ensemble des sessions reconstruites

A.4 Analyse statistique cinématographique

Dans le cadre de notre analyse des comportements vidéo, nous avons développé un outil de visualisation des résultats. En se basant sur le domaine applicatif d'un moteur de recherche de bandes-annonces cinématographiques, nous proposons à un administrateur de site ou un professionnel de l'audiovisuel d'analyser les comportements en fonction des vidéos concernées.

La figure (A.13) présente la visualisation des données de comportements observées sur le visionnage d'une vidéo par un utilisateur. On y retrouve des données statistiques sur ce visionnage et le pourcentage de proximité avec les comportements type. Un lien permet de visualiser ces comportements type (A.14).

Cette représentation qui concerne ici un unique visionnage est également disponible pour l'ensemble des visionnages sur une bande-annonce donnée. C'est alors la moyenne observée qui est présentée dans l'histogramme.

L'ensemble de ces données est accessible par film par différentes rubriques proposées dans une barre de navigation :

- Ensemble des films de la base
- Films actuellement en salle
- Films prochainement en salle
- Films les plus regardés
- Films les moins regardés

Il est également possible d'accéder aux données observées sur les dix derniers utilisateurs.

Cet outil doit permettre, en se basant sur l'analyse intra-vidéo des comportements, d'observer rapidement et visuellement quelles bandes-annonces sont appréciées des utili-

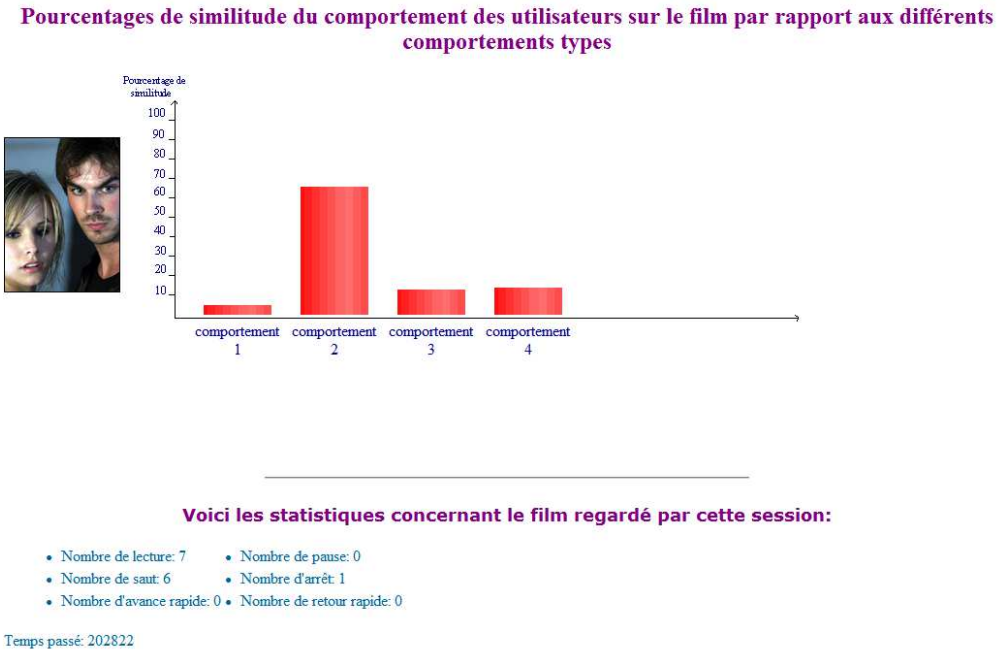


FIG. A.13 – Visualisation d'un visionnage

(%)	Lecture	Pause	Saut	Avance rapide	Retour rapide	Stop
Lecture	10	2	78	0	0	8
Pause	0	0	1	0	0	98
Saut	37	6	25	0	0	29
Avance rapide	0	0	0	100	0	0
Retour rapide	0	0	0	0	100	0
Stop	0	0	0	0	0	100

FIG. A.14 – Visualisation d'un visionnage

sateurs et quelles autres le sont moins.