UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale MIPTIS

LABORATOIRE D'INFORMATIQUE, ÉQUIPE BDTLN

THÈSE présentée par :

Julien ALIGON

soutenue le : 13 décembre 2013

pour obtenir le grade de : Docteur de l'université François - Rabelais de Tours

Discipline/ Spécialité : Informatique

Similarity-Based Recommendation of OLAP Sessions

THÈSE DIRIGÉE PAR :

GIACOMETTI Arnaud	Professeur des Universités, Université François - Rabelais de Tours
MARCEL Patrick	Maître de Conférences HDR, Université François - Rabelais de Tours

RAPPORTEURS :

AUFAURE Marie-Aude	Professeur des Universités, Ecole Centrale Paris
TESTE Olivier	Professeur des Universités, Université Toulouse 2 Le Mirail

JURY :

AUFAURE Marie-Aude	Professeur des Universités, Ecole Centrale Paris
GIACOMETTI Arnaud	Professeur des Universités, Université François - Rabelais de Tours
MARCEL Patrick	Maître de Conférences HDR, Université François - Rabelais de Tours
RAÏSSI Chedy	Chargé de Recherche, INRIA Nancy
RIZZI Stefano	Professeur, Università di Bologna
TESTE Olivier	Professeur des Universités, Université Toulouse 2 Le Mirail

Aknowledgment

I sincerely thank Marie-Aude Aufaure and Olivier Teste for agreeing to be the reviewers of this dissertation. I also offer my gratitude to Chedy Raissi and Stefano Rizzi for their participation in the jury committee of my thesis.

I also express my deepest gratitude to Enrico Gallinucci Matteo Golfarelli, Stefano Rizzi and Elisa Turricchia for their multiple collaborations that have been notably for me the most rewarding experiences.

I especially want to thank my supervisors Arnaud Giacometti and Patrick Marcel for trusting me throughout these three years of PhD. I also thank them for their high scientific and human qualities as well as for multiple advice they were able to bring me and for which this PhD took place in the most ideal conditions.

I also do not forget all the members of the BDTLN team and the computer science department whose cohesion and group mind made these years enjoyable, namely Jean-Yves Antoine, Beatrice Bouchou-Markoff, Thomas Devogele, Nathalie Friburger, Haoyuan Li, Denis Maurel, Nizar Messai, Evelyne Moreau, Veronika Peralta, Yacine Sam, Agata Savary, Arnaud Soulet and Mohamed Taghelit.

I also thank Leila Abdellatif, Christelle Grange, Aurora Leroy, Beatrice Pawlik, Thierry Ressault and François Laurand for their help, always responsive, they were able to give me.

I also wish every success to Mouhamadou Saliou Diallo, Anaïs Lefeuvre and Jakub Waszczuk for the completion of their respective PhD. Moreover, I also thought to the new doctors who are Cheikh Niang and Chedlia Chakroun to whom I wish the best for their future.

This dissertation could not be done without the unfailing support, whether in woes and joys of my friends. I think in particular Tony Godet, friend for almost 22 years to whom I wish a great end of doctorate in medicine, but also Richard Aguenier, Jérôme Ah-Leung, Stéphanie Ah-Leung, Victor Batard, Arnaud Cantin, Aurore Cantin, Florent Delalande, Michael Dubosq-Gérimont, Elisa Fournier, Bertrand Labalme, Arnaud Lefebvre, Paul Quenioux, Mickaël Renault, Sylvain Rivierre, Emilie Soeur and Etienne Thirant.

Finally, I thank, from the bottom of my heart, my parents, Michele and Jean-Yves, and my sister, Caroline, my sister-in-law, Kasia, and my brother, Stephane, for all the support and encouragement I have received all throughout these years and whose events have only strengthened our ties. I extend all my love to my nephew Aleksander and whose birth occurred at the end of writing this dissertation filled me with great joy.

Remerciements

Je remercie très sincèrement Marie-Aude Aufaure ainsi qu'Olivier Teste pour avoir accepté d'être les rapporteurs de ce mémoire. J'adresse également toute ma reconnaissance à Chedy Raïssi et Stefano Rizzi pour leur participation au jury de soutenance de ma thèse.

J'exprime également ma profonde gratitude envers Enrico Gallinucci, Matteo Golfarelli, Stefano Rizzi et Elisa Turricchia pour leurs multiples collaborations, qui auront notamment été pour moi des expériences des plus enrichissantes.

Je tiens à remercier tout particulièrement mes directeurs de thèse, Arnaud Giacometti et Patrick Marcel, pour m'avoir fait confiance tout au long de ces trois années. Je les remercie également pour leur hautes qualités scientifiques et humaines ainsi que pour les multiples conseils qu'ils ont su m'apporter et qui ont fait que cette thèse se déroule dans les conditions les plus idéales.

Je n'oublie également pas l'ensemble des membres de l'équipe BDTLN et du département informatique dont la cohésion et l'esprit de groupe ont rendu agréable ces années de doctorat, à savoir Jean-Yves Antoine, Béatrice Bouchou-Markoff, Thomas Devogele, Nathalie Friburger, Haoyuan Li, Denis Maurel, Nizar Messai, Evelyne Moreau, Veronika Peralta, Yacine Sam, Agata Savary, Arnaud Soulet et Mohamed Taghelit.

Je remercie également Leïla Abdellatif, Christelle Grange, Aurore Leroy, Béatrice Pawlik, Thierry Ressault et François Laurand pour l'aide, toujours réactive, qu'ils ont su m'apporter.

Je souhaite aussi tout le succès possible à Mouhamadou Saliou Diallo, Anaïs Lefeuvre et Jakub Waszczuk pour l'aboutissement de leurs thèses respectives. J'ai d'ailleurs aussi une pensée aux nouveaux docteurs Cheikh Niang et Chedlia Chakroun à qui je souhaite le meilleur pour l'avenir.

Ce mémoire n'aurait pu se faire également sans le soutien indéfectible, que ce soit dans les malheurs et les joies, de mes amis. Je pense en particulier à Tony Godet, ami depuis maintenant près de 22 ans et à qui je souhaite une excellente fin de doctorat en médecine, mais aussi à Richard Aguenier, Jérome Ah-Leung, Stéphanie Ah-Leung, Victor Batard, Arnaud Cantin, Aurore Cantin, Florent Delalande, Michael Dubosq-Gérimont, Elisa Fournier, Bertrand Labalme, Arnaud Lefebvre, Paul Quenioux, Mickaël Renault, Sylvain Rivierre, Emilie Soeur et Etienne Thirant.

Enfin, je remercie du fond du coeur mes parents, Michèle et Jean-Yves, ainsi que ma soeur, Caroline, ma belle-soeur, Kasia, et mon frère Stéphane, pour tout le soutien et les encouragements que j'ai pu recevoir tout au long de ces années et dont les épreuves n'ont

fait que renforcer nos liens. J'adresse également toute mon affection à mon petit neveu Aleksander et dont la naissance survenue à la toute fin de rédaction de cette thèse me remplie d'une immense joie.

French Report

Introduction

Contexte de la Thèse

L'objectif des entrepôts de données est de stocker de grandes quantités de données. Un cube de données permet d'organiser les données d'un entrepôt de données selon différents axes d'analyse et mesures d'agrégation. OLAP est le paradigme principal pour accéder aux données multidimensionnelles dans les entrepôts de données. Pour obtenir une grande expressivité d'interrogation, malgré un petit effort de formulation de la requête, OLAP fournit un ensemble d'opérations (comme drill-down et slice-dice) qui transforment un requête multidimensionnelle en une autre, de sorte que les requêtes OLAP sont généralement formulées sous la forme de séquences appelées sessions *OLAP* [Sapia, 2000]. Lors d'une session OLAP, l'utilisateur analyse les résultats d'une requête et, selon les données spécifiques qu'il voit, applique une seule opération pour déterminer une nouvelle requête qui va lui donner une meilleure compréhension de l'information. Les séquences résultantes de requêtes sont fortement liées à l'utilisateur, le phénomène analysé, et les données actuelles.

Alors qu'il est universellement reconnu que les outils OLAP ont un rôle clé dans le soutien à une exploration souple et efficace de cubes multidimensionnels dans les entrepôts de données, il est aussi communément admis que le grand nombre d'agrégations et les sélections possibles qui peuvent être exploités sur des données peut rendre ardu l'expérience utilisateur. Différentes approches ont été prises dans la littérature pour résoudre ce problème, en particulier, fondée sur les préférences en matière de personnalisation (par exemple, [Golfarelli et al., 2011, Jerbi et al., 2009]) et les techniques de recommandation (par exemple, [Sarawagi, 2000, Giacometti et al., 2009]) qui ont été spécifiquement conçu pour les systèmes OLAP.

Le travail présenté dans cette thèse porte sur la problématique de recommandation de requêtes, dont les questions suivantes sont au cœur de cette thèse :

- Comment recommander des requêtes en tenant compte des aspects spécifiques des sessions OLAP ?
- Comment le faire efficacement ?
- Comment tirer parti des sessions précédentes ?
- Comment évaluer l'efficacité des recommandations ?

Les contributions pour chacune de ces questions sont données dans la section suivante.

Contribution de la Thèse

L'affirmation initiale qui sous-tend l'approche de cette thèse est qu'une session OLAP conduite par un utilisateur expérimenté n'est pas seulement un chemin menant à une seule requête intéressante (celle à la fin de la session). En effet, toute la séquence de requêtes appartenant à une session est une valeur en soi, car il donne à l'utilisateur une vue différente et complémentaire de l'information. Pour cette raison, nous proposons une approche dont le but est de ne pas recommander une requête OLAP simple, mais plutôt une session OLAP.

En cohérence avec les approches de filtrage collaboratif, l'objectif de l'approche dans le choix des sessions à recommander est de réutiliser les connaissances acquises par d'autres utilisateurs lors des sessions précédentes. Ainsi, le système de recommandation est composée de trois phases concernant l'aspect séquentiel des sessions que sont :

- La phase de *Selection* permettant de sélectionner les recommandations éventuelles.
- La phase de *Ranking* permettant de proposer la meilleure recommandation parmi les recommandations préalablement classées.
- La phase de *Tailoring* permettant d'adapter la recommandation obtenue dans la phase précédente par rapport à la session en cours.

Le système est également basé sur des exigences spécifiques permettant de répondre aux lacunes que l'on retrouve dans la plupart des systèmes de recommandation (par exemple le fait que les requêtes sont rarement synthétisés pour l'adapter à la session en cours).

Pour recommander des requêtes de manière efficace, nous proposons de définir un modèle de requête basé sur l'expression de requête écrite dans un langage de requête particulier. En effet, l'efficacité est un problème majeur pour un système de recommandation où, généralement, le contexte de la session en cours doit être comparé aux contextes des sessions passées dans une étape, en ligne, du système. Ainsi, les requêtes n'ont pas besoin d'être exécuté et les clauses de requêtes sont utilisées par le système de recommandation.

Pour tirer parti des sessions précédentes par le système de recommandation, nous proposons de définir des mesures de similarité entre les sessions. Ces mesures de similarité sont utilisés lors des phases de *Selection* et *Ranking* en respectant l'aspect séquentiel des sessions. La comparaison des sessions OLAP doit être fondée sur des besoins spécifiques, qui sont émises dans cette thèse, afin de proposer des mesures de similarité entre sessions adaptées au contexte OLAP. Ces exigences ont conduit à proposer une approche de similarité entre sessions à deux niveaux, comprenant une mesure de similarité entre requêtes et une mesure de similarité pour comparer des séquences de requêtes (sur la base de l'algorithme d'alignement de sous-séquences de Smith-Waterman).

Pour évaluer l'efficacité des recommandations, nous proposons un ensemble de mesures de qualité, définies à partir de critères de qualité exprimés pour la recommandation de sessions. En particulier, les évaluations expérimentales vérifient les sessions proposées par le système de recommandation en utilisant ces mesures de qualité. Préalablement, cette thèse présente un ensemble de générateurs de logs synthétiques permettant d'évaluer expérimentalement les propositions de mesures de similarité entre requêtes et sessions, ainsi que le système de recommandation. Les logs, basés sur des sessions conçues par des étu-

diants à en Aide à la Décision, sont également utilisés pour montrer que le système de recommandation peut recommander des sessions pertinentes à partir de séances d'analyse diverses.

Plan du Mémoire

Cette thèse est organisée comme suit :

Le Chapitre 1 est consacré à l'état de l'art sur le fouille d'usage pour la recommandation de requêtes. Nous commençons par présenter les bases sur les systèmes de recommandation ainsi que leurs méthodes d'évaluation. Ensuite, une étude détaillée des systèmes de recommandation dans les bases de données et des entrepôts de données est proposée. Puisque la séquence de requêtes doit être considérée pour notre système de recommandation, un état de l'art sur les similarités entre sessions et requêtes est donné. Ce chapitre présente également des critères considérés comme pertinents pour un système de recommandation OLAP basé sur des mesures de similarité.

Le Chapitre 2 introduit les définitions nécessaires pour proposer des mesures de similarité entre sessions OLAP. Les modèles de données, requête, session et logs sont formellement définis. En particulier, le modèle de requête est basée sur l'expression de requête. Ensuite, des mesures de similarité entre les requêtes, les sessions et les logs sont présentés.

Le Chapitre 3 est consacré au système de recommandation dont trois phases sont détaillées : *Selection Ranking* et *Tailoring*. La phase de *Selection* compare chaque session passée à la session courante en utilisant la mesure de similarité entre les sessions et sélectionne d'éventuelles recommandations. La phase de *Ranking* classe les recommandations précédemment obtenues en avantageant celles qui ont des requêtes similaires entre les recommandations possibles. La phase de *Tailoring* adapte la recommendation la mieux classée à la session en cours.

Chapitre 4 rapporte les expériences évaluant la pertinence du système de recommandation et les mesures de similarité entre les requêtes et les sessions. Des tests spécifiques menées sur les mesures de similarité sont composés de deux évaluations : une première est subjective et est basée sur des questionnaires utilisateurs indiquant la perception qu'a l'utilisateur concernant les similarités entre requêtes et sessions, la seconde est objective et différents patrons obtenus avec des générateurs de logs synthétiques permettent de valider les capacités des mesures de similarité. Le système de recommandation est évalué par en terme de temps de calcul et de pertinence, sur la base de différents types de sessions d'analyse synthétiques mais aussi réelles conçus par les étudiants de maîtrise. Les mesures de qualité définies à partir de critères exprimés pour la recommandation de sessions permettent d'évaluer la pertinence du système de recommandation.

La conclusion présente les contributions de cette thèse et leurs perspectives en discutant du problème de *Dmarrage Froid* pendant la réalisation de sessions mais aussi un benchmark for les sessions OLAP and l'adaptation du système de recommandation dans des contextes autres que l'OLAP.

Conclusion

Résumé de la contribution

Cette thèse présente une approche pour recommander des sessions OLAP, dans un contexte de filtrage collaboratif, et sur la base de mesures de similarité entre les requêtes et les sessions. Nous donnons les différentes contributions de ce travail dans chacune des sections suivantes. Notez que ce travail est l'objet d'un travail en cours dans [Aligon et al., 2013a].

Besoins pour la Recommandation et les Mesures de Similarités

Après un bref examen des techniques classiques pour la fouille d'usage dans la recommandation de pages web, une étude des systèmes de recommandation dans les bases de données et des entrepôts de données a permis d'identifier plusieurs lacunes. En effet, les aspects séquentiels sont rarement abordées dans ces travaux et aucune approche n'a jamais considérés la possibilité de recommander des sessions . En outre, les requêtes sont rarement synthétisées pour la recommandation et sont souvent choisis parmi les requêtes passées . Cette thèse répond à ces lacunes en proposant un ensemble d'exigences à prendre en compte dans un contexte de recommandation . En particulier :

- Les sessions doivent être recommandée .
- Le système de recommandation devrait être basé sur des mesures de similarité entre les sessions .
- Les expressions de requêtes sont préférées aux tuples, afin de générer des recommandations par une étape en ligne .
- Des critères de qualité doivent être définis pour évaluer les sessions recommandées par le système, que sont : $relevance$, $novelty$, $foresight$, $obviousness$ et $adaptation$.

Etant donné que le système de recommandation se base sur une mesure de similarité, une étude des mesures classiques en recherche d'information a été présentée. Pour étendre ces mesures dans un contexte OLAP, plusieurs besoins sont proposées :

- La similarité entre requêtes doit donner un poids plus important pour les prédicats de sélection plutôt que pour le group-by set ou l'ensemble des mesures
- La similarité entre sessions doit prendre en compte l'ordre entre les requêtes qui composent les sessions à comparer
- Les requêtes récentes sont considérées comme plus pertinentes que les anciennes requêtes

Notez que l'extension des mesures de similarité pour les sessions OLAP est publiée dans [Aligon et al., 2013b].

Definitions de Mesures de Similarités à trois Niveaux

Cette thèse a proposé plusieurs mesures de similarité entre logs OLAP, organisées en trois niveaux. Les mesures de similarité entre les logs dépendent de mesures de similarité entre les sessions qui dépendent de mesures de similarité entre les requêtes. En particulier,

la mesure de similarité entre les requêtes ne dépend que de la définition de la requête, qui est une structure basée sur les fragments composés d'un group-by set, d'un ensemble de prédicats de sélection et un ensemble de mesures. Les mesures de similarité entre les sessions sont des extensions, dans le contexte OLAP, de mesures classiques en recherche d'information que sont le Coefficient de Dice, l'alignement de sous-séquences, le TF-IDF, et la Distance de Levenshtein. Notamment, l'alignement de sous-séquences est à la base du système de recommandation. Les mesures de similarité entre les logs étendent les mesures classiques entre ensembles, que sont l'Exactitude, la Distance de Hausdorff et le Coefficient de Jaccard. Elles sont à la base des définitions de mesures de qualité pour les sessions recommandées et répondent à des critères de qualité.

Proposition d'un système de recommandation de session OLAP, basé sur des Mesures de Similarité

Un système de recommandation de sessions OLAP basé sur une mesure de similarité est proposé. Notez que cette proposition est un travail en cours dans [Aligon et al., 2013a]. Trois phases composent ce système :

- La première phase aligne les sessions du log avec la session en cours, basée sur une version modifiée de l'alignement de sous-séquences. Les requêtes qui suivent les sous-sessions alignés des sessions du logs sont considérées comme des futures possibles à recommander.
- La deuxième phase classe chaque future en identifiant les zones les plus denses de requêtes similaires dans les sessions du log.
- La dernière phase adapte le futur, ayant le meilleur classement, en modifiant ou en ajoutant des fragments dans ses requêtes, en utilisant des motifs extraits à partir du logs et de la session courante, et le recommande. En particulier, cette phase adapte la technique de [Aligon et al., 2011].

Vérification des Mesures de Similarité et du Système de recommandation

Le système de recommandation est évalué en termes de temps de calcul et de pertinence avec des sessions provenant de générations de logs synthétiques ou de logs dont les sessions ont été conçues par des étudiants de Master. Deux générateurs synthétiques, basés sur différents modèles, créent des sessions reproduisant des comportements d'analyse. Les sessions conçues par des étudiants, dont une évaluation est publié dans [Aligon, 2013], permettent d'utiliser la subjectivité des sessions d'analyse pour tester le système de recommandation, même si un pré-filtrage des sessions est quand même nécessaire. Les tests ont montré que la mesure similarité basée sur l'alignement de sous-séquences surpasse les autres. Les tests sur l'efficacité du système de recommandation ont montré que les sessions recommandées répondent aux critères de qualité exprimées pour la recommandation.

Perspectives

Nous présentons dans les sections suivantes les perspectives à court et long termes des travaux présentés dans cette thèse.

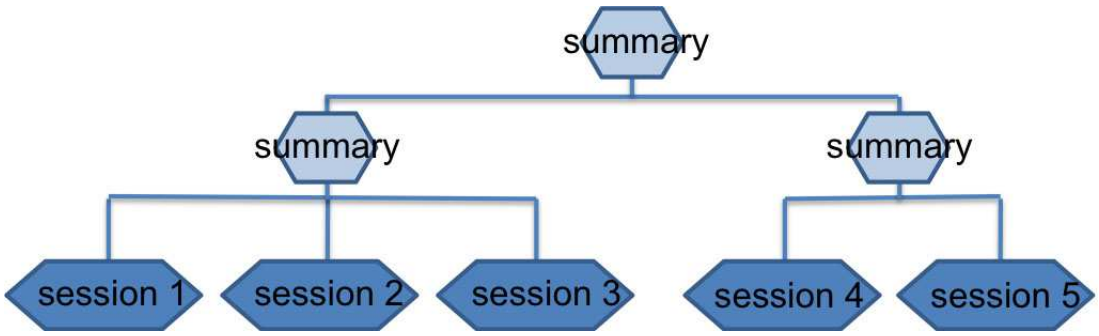


FIGURE 1 – Clustering Hiérarchique Agglomératif de Logs et Résumés

Un Outil pour l'Aide à la Conception de Sessions

Lorsqu'un utilisateur commence juste à concevoir sa session, le système de recommandation ne peut généralement pas proposer de requêtes puisque aucune session en cours ou pas assez de requêtes ne sont disponibles. Bien connu dans le contexte de la recommandation, le problème de *Dmarrage Froid* pourrait être réglé à l'aide des travaux proposés dans [Aligon and Marcel, 2012] et [Aligon et al., 2012] . En effet, l'idée serait que l'utilisateur explore les logs de requêtes afin d'identifier les requêtes particulières qu'il pourrait réutiliser pour initier sa session. Puisque qu'un log est très grand, un utilisateur serait submergé par la quantité d'informations à explorer. Ainsi, seules les requêtes les plus pertinentes d'un log doivent être présentées à un utilisateur et organisées entre elles pour faciliter la navigation.

Pour faire face au problème de la navigation entre requêtes, une solution pourrait consister à identifier les différents groupes de requêtes similaires pour différents niveaux de densité. En règle générale, une classification ascendante hiérarchique pourrait produire des clusters, organisés dans une structure arborescente. Pour naviguer entre les nœuds, des opérateurs sélectionnant des nœuds et permettant de naviguer entre eux pourraient être proposées. Malheureusement, les nœuds de l'arbre peuvent être très difficiles à analyser car de nombreuses sessions peuvent être présentes. Une ligne de pensée, présenté ci-dessous, serait de produire une représentation pertinente et concise d'un groupe de sessions, nommé *rsum* (voir la figure ref fig :hierarcal-clustering2).

Pour faire face au problème de la représentation de groupes de sessions, une solution est proposée dans [Aligon and Marcel, 2012] et [Aligon et al., 2012] qui abordent le problème de résumé des expressions de requête.

Plus précisément, un cadre est introduit dans lequel les opérateurs de résumé sur les requêtes , les sessions et les logs sont définis . L'intuition du résumé de log est que la forme du résumé devrait être de la même que le log d'origine et ce résumé peut perdre des informations. En outre , le séquencement d'une session doit être reflété dans le résumé. Cela est capturée en proposant une relation de spécialisation entre les sessions, une autre relation de spécialisation entre les requêtes, et une relation de couverture entre les requêtes et les sessions. Plus précisément, comme illustré à la figure 2, une session résume une autre session si la première est plus générale que l'autre, à savoir, chacune de ses requêtes couvre une sous-session de la session originale. De plus, une requête couvre une séquence de

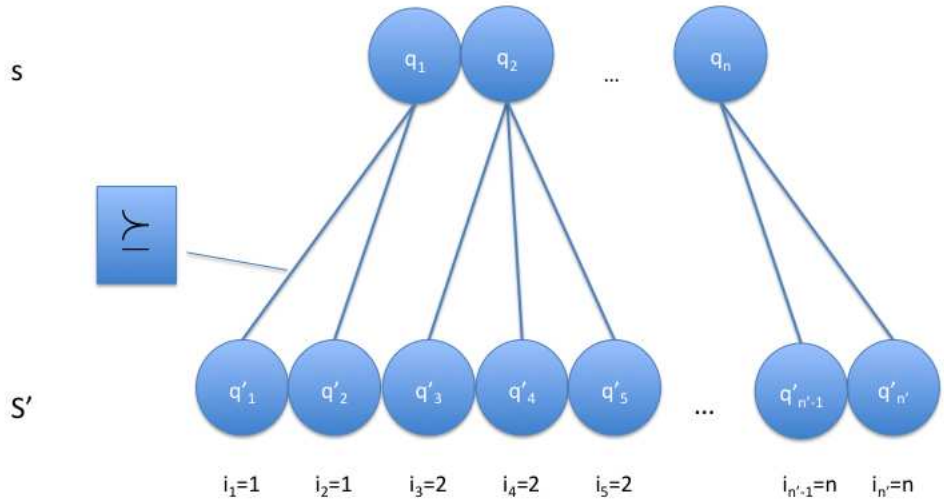


FIGURE 2 – Relation de Spécialisation entre Sessions

requêtes si elle est plus générale que chaque requête de la séquence. Un algorithme permet de calculer automatiquement un résumé de log où des mesures de qualité assurent que le résumé ne perd pas trop d'informations. Le principe général de résumé de log est donné figure 3 et montre les deux étapes de l'algorithme que sont :

- La réduction du nombre de requêtes par session.
- La fusion des sessions, réduisant le nombre de sessions, en généralisant les paires de sessions par une seule session.

En considérant à la fois la structure hiérarchique entre les sessions de log et la technique de résumé pour obtenir une vue d'ensemble de chaque cluster, un utilisateur pourrait être en mesure de sélectionner les requêtes à réutiliser dans sa session en cours. Enfin, lorsque le nombre de requêtes dans la session en cours est suffisamment importante, le système de recommandation prend le relais.

Un Benchmark pour les sessions OLAP

Pour mener et simuler des tests, une perspective est le développement d'une plate-forme pour évaluer la qualité d'une session d'analyse sur un cube. En effet, bien que la qualité des données soit un domaine bien étudié dans le contexte de la gestion des données, la qualité du processus d'interrogation n'a pas encore été étudié. La qualité des données est modélisé en fonction des dimensions de qualité (comme par exemple, l'exactitude.), chacune d'elles groupant des facteurs de qualité (comme par exemple, la correction sé-

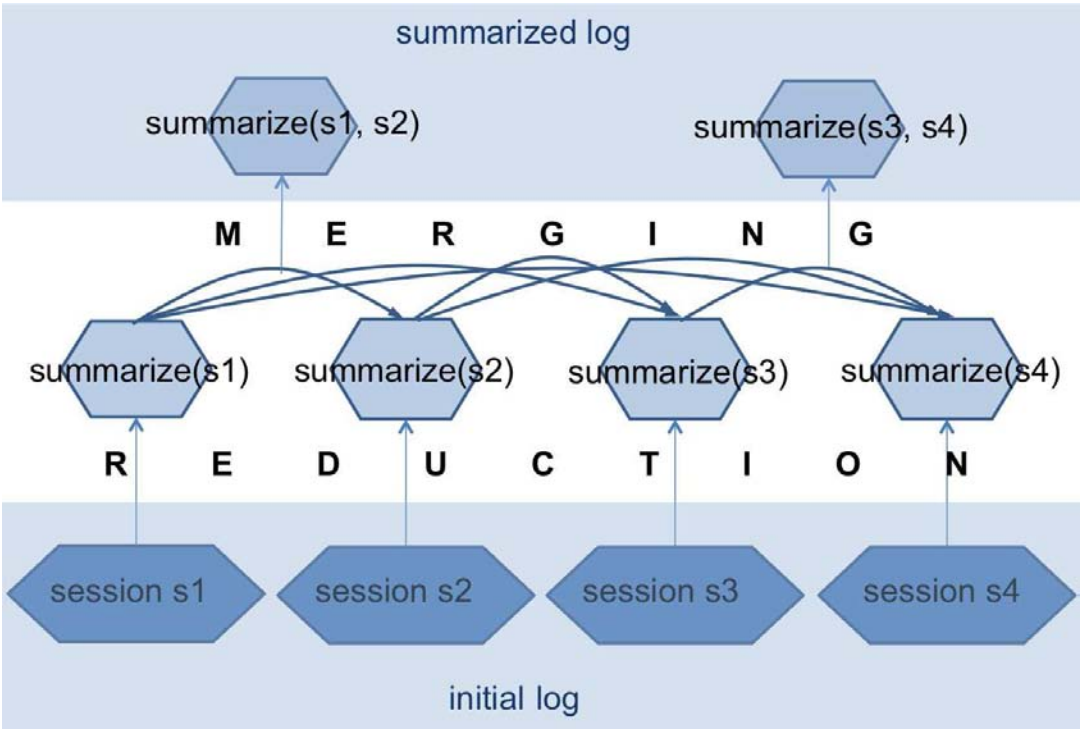


FIGURE 3 – Principe de Résumé de Log

mantiquen, ou la correction syntaxique), chacune d'elle mesurée par diverses métriques [Berti-Equille et al., 2011]. Une approche similaire pourrait être utilisée pour modéliser la qualité de requêtes d'analyse et la qualité de sessions d'analyse. En outre, un tel modèle permettrait le développement d'une plate-forme pour l'évaluation et la validation d'approches centrées sur l'utilisateur dans le contexte de OLAP.

Notamment, dans des domaines tels que la recherche d'information, des benchmarks sont utilisés [(NIST), 2012] pour valider les approches sur les résultats de recherche exploratoire et de navigation. Dans le domaine de l'entreposage de données, les critères actuels sont exclusivement axés sur la performance (voir par exemple [(TPC), 2012, O'Neil et al., 2009, Darmont et al., 2007]), et l'efficacité du processus d'interrogation en termes de réussite de la session d'analyse, n'est même pas abordée. Jusqu'à présent, les approches de personnalisation ou recommandation sont basées sur des logs de requêtes synthétiques [Mishra and Koudas, 2009] ou bien sur le traitement de logs existants, sans aucune garantie d'être réaliste [Chatzopoulou et al., 2009].

Une plate-forme pour l'évaluation des approches centrées sur l'utilisateur devrait permettre de mesurer dans quelle mesure les approches sont efficaces pour que les réponses trouvées soient pertinentes, l'effort dépensé pour effectuer l'analyse soit réduite, etc. Une telle plate-forme pourrait bénéficier des efforts précédents pour développer un benchmark pour valider les approches de personnalisation dans les bases de données [Peralta et al., 2009].

Le développement d'une telle plate-forme suppose d'étendre la définition de la session

en incluant autant d'informations que possible, comme par exemple, les opérations OLAP.

Adaptation du Système de recommandation dans d'autres contextes

Une autre perspective à long terme est d'adapter le système de recommandation dans des contextes autres que l'OLAP. En effet, le système de recommandation peut être transposable dans des contextes où les séquences d'opérations complexes sont possibles et où des analyses sont nécessaires. Des applications du système de recommandation peuvent être trouvées dans les bases de données bien sûr, mais aussi dans la fouille de données où des séquences complexes peuvent être considérées comme des séquences de tâches de fouille de données. Ces séquences complexes doivent être adaptés à la mesure de similarité entre sessions, en prenant en compte les aspects spécifiques de chaque domaine. Dans le Web, le système pourrait également gérer des séquences de recherche sur le Web ou bien des séquences d'analyse sur les réseaux sociaux. En particulier, le système de recommandation pourrait aussi prendre en compte les relations entre les utilisateurs et pourrait inclure une mesure de similarité entre utilisateurs. En effet, le système proposé dans cette thèse utilise uniquement des sessions indépendamment d'un utilisateur, sans regrouper préalablement des sessions similaires dans des profils d'utilisateurs.

French Abstract

L'OLAP (On-Line Analytical Processing) est le paradigme principal pour accéder aux données multidimensionnelles dans les entrepôts de données. Pour obtenir une haute expressivité d'interrogation, malgré un petit effort de formulation de la requête, l'OLAP fournit un ensemble d'opérations (comme drill-down et slice-and-dice) qui transforment une requête multidimensionnelle en une autre, de sorte que les requêtes OLAP sont normalement formulées sous la forme de séquences appelées *Sessions OLAP*. Lors d'une session OLAP, l'utilisateur analyse les résultats d'une requête et, selon les données spécifiques qu'il voit, applique une seule opération afin de créer une nouvelle requête qui lui donnera une meilleure compréhension de l'information. Les séquences de requêtes qui en résultent sont fortement liées à l'utilisateur courant, le phénomène analysé, et les données. Alors qu'il est universellement reconnu que les outils OLAP ont un rôle clé dans l'exploration souple et efficace des cubes multidimensionnels dans les entrepôts de données, il est aussi communément admis que le nombre important d'agrégations et sélections possibles, qui peuvent être exploités sur les données, peut désorienter l'expérience utilisateur.

Cette thèse présente une approche pour recommander des sessions OLAP, dans un contexte de filtrage collaboratif, et fondé sur des mesures de similarité entre les requêtes et les sessions. Après une brève étude des techniques classiques d'extraction de l'expérience utilisateur dans le domaine des recommandations de page web, une étude sur les systèmes de recommandation dans les bases de données et entrepôts de données permet d'identifier plusieurs lacunes. En effet, les aspects séquentiels sont rarement pris en compte dans ces travaux et aucune approche n'a déjà envisagé de recommander des sessions. Par ailleurs, les requêtes sont rarement synthétisées pour la recommandation et sont souvent choisis parmi les requêtes passées. Cette thèse répond à ces inconvénients en proposant un ensemble d'exigences à prendre en compte dans un contexte de recommandation. Puisque le système de recommandation est basée sur une mesure de similarité, une étude des mesures classiques en recherche d'information est également présentée. Par la suite plusieurs mesures de similarité sont étendus dans un contexte OLAP et sont organisées dans une approche à trois niveaux entre les logs OLAP. Les mesures de similarité entre les logs dépendent de mesures de similarité entre les sessions qui dépendent de mesures de similarité entre requêtes. Ensuite, un système de recommandation, basé sur une mesure de similarité entre sessions, est proposé. Trois phases composent ce système. La première phase aligne les sessions du log à la session courante et identifie d'éventuelles recommandations. La deuxième phase classe chaque recommandation en identifiant les zones les plus denses de requêtes similaires dans les sessions du log. La dernière phase adapte la recommandation, ayant le meilleur score de classement à la session courante, en utilisant des motifs extraits

du log et de la session courante, et la recommande. Enfin, le système de recommandation est évalué en termes d'efficacité et de pertinence avec des sessions provenant de générateurs de logs synthétiques ou de logs dont les sessions ont été conçues par les étudiants de Master en Aide à la Décision.

Finalement, plusieurs perspectives de recherche sont présentées. En particulier, une proposition pour palier au problème du $Dmarrage\ \ Froid$, lors de la composition de sessions, est décrite. Une discussion est aussi exprimée sur le besoin d'un benchmark pour les sessions OLAP mais aussi sur l'adaptation du système de recommendation à d'autres contextes que l'OLAP.

Mots clés : Système de recommandation, Log, Session OLAP, Mesures de Similarité OLAP

Abstract

OLAP (On-Line Analytical Processing) is the main paradigm for accessing multidimensional data in data warehouses. To obtain high querying expressiveness despite a small query formulation effort, OLAP provides a set of operations (such as drill-down and slice-and-dice) that transform one multidimensional query into another, so that OLAP queries are normally formulated in the form of sequences called *OLAP sessions*. During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, applies one operation to determine a new query that will give her a better understanding of information. The resulting sequences of queries are strongly related to the issuing user, to the analyzed phenomenon, and to the current data. While it is universally recognized that OLAP tools have a key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses, it is also commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating.

This dissertation presents an approach for recommending OLAP sessions, in a collaborative filtering context, and based on similarity measures between queries and sessions. After briefly reviewing classical techniques for usage mining in Web Page Recommendation, a study of recommender systems in Databases and Data Warehouses allows to identify several shortcomings. Indeed, sequential aspects are rarely addressed in these works and no approach ever considered to recommend sessions. Besides, queries are rarely synthesized for the recommendation and are often chosen among past queries. This dissertation answers these shortcomings by proposing a set of requirements to take into account in a recommendation context. Since the recommender system is based on a similarity measures, a study of classical measures in information retrieval is also presented. Afterward several similarity measures are extended in an OLAP context and are organized in a three-level approaches between OLAP logs. Similarity measures between logs depend on similarity measures between sessions that depend on similarity measures between queries. Then, a recommender system based on similarity measure between sessions is proposed. Three phases compose this system. The first phase aligns the log sessions with the current session and identifies possible recommendations. The second phase ranks each recommendation by identifying densest areas of similar queries in the log sessions. The last phase adapts the recommendation, ranked first to the current session, using patterns extracted from the log and the current session, and recommends it. Also, the recommender system is assessed in terms of efficiency and effectiveness with sessions coming from synthetic log generations or logs whose sessions have been devised by Master's students in Business Intelligence.

Finally, several research perspectives are presented. In particular, a proposal to over-

come the *Cold Start* problem, during session design, is described. A discussion is also expressed the need of a benchmark for OLAP sessions as well as the adaptation of the recommender system in contexts other than OLAP.

Keywords : Recommender System, Log, OLAP Session, OLAP Similarity Measures

Contents

List of Tables

List of Figures

List of Algorithms

Introduction

Thesis Context

The aim of data warehouses is to store large amounts of data. A data cube allows to organize the data of a data warehouse according to different analysis axes and aggregation measures. OLAP is the main paradigm for accessing multidimensional data in data warehouses. To obtain high querying expressiveness despite a small query formulation effort, OLAP provides a set of operations (such as drill-down and slice-and-dice) that transform one multidimensional query into another, so that OLAP queries are normally formulated in the form of sequences called *OLAP sessions* [Sapia, 2000]. During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, applies one operation to determine a new query that will give her a better understanding of information. The resulting sequences of queries are strongly related to the issuing user, to the analyzed phenomenon, and to the current data.

While it is universally recognized that OLAP tools have a key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses, it is also commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating. Different approaches were taken in the literature to address this issue; in particular, in the area of personalization, both preference-based (e.g., [Golfarelli et al., 2011, Jerbi et al., 2009]) and recommendation techniques (e.g., [Sarawagi, 2000, Giacometti et al., 2009]) were specifically devised for OLAP systems.

The work presented in this dissertation focuses on query recommendation problematic whose following questions are at the heart of this thesis:

- How to recommend queries considering the specific aspects of the OLAP sessions?
- How to do it efficiently?
- How to leverage past sessions?
- How to evaluate the effectiveness of the recommendations?

The contributions for each of these questions are given in the next section.

Thesis Contribution

The original claim underlying the approach of this dissertation is that an OLAP session issued by a skilled user is not just a path aimed at leading her to a single, valuable query (the one at the end of the session). Indeed, the whole sequence of queries belonging to a

session is valuable in itself, because it gives the user a different and complemental view of information. For this reason, we propose an approach whose goal is not to recommend single OLAP queries, but rather OLAP sessions.

Consistently with collaborative filtering approaches, the goal of the approach in deciding which sessions to recommend is to reuse knowledge acquired by other users during previous sessions. Thus, the recommender system is composed of three phases respecting the sequential aspect of the sessions that are:

- *Selection* phase allowing to select possible query sequence recommendations
- *Ranking* phase allowing to propose the best recommendation among recommendations beforehand ranked
- *Tailoring* phase allowing to adapt the recommendation obtained in the previous phase with the current session.

The system is also based on specific requirements allowing to answer to the shortcomings that can be found in most recommender systems (for example the fact that queries are rarely synthesized to adapt it to the current session).

To recommend queries efficiently, we propose to define a query model based on the query expression written in a particular query language. Indeed, efficiency is a major problem for a recommender system where, usually, the context of the current session has to be matched with the contexts of past sessions in an online step of the system. Thus, the queries do not need to be executed and only the query clauses are used by the recommender system.

To leverage past sessions by the recommender system, we propose to define similarity measures between sessions. These similarity measures are used during the *Selection* and *Ranking* phases respecting the sequential aspect of the sessions. The comparison of OLAP sessions must be based on specific requirements, that are issued in this dissertation, in order to propose similarity measures between sessions adapted to the OLAP context. These requirements led to propose a two-level approach for session similarity including a similarity measure between queries and a similarity measure for comparing sequences of queries (based on the Smith-Waterman subsequence alignment algorithm).

To evaluate the effectiveness of the recommendations, we propose a set of quality measures, defined from quality criteria expressed for session recommendation. In particular, experimental evaluations assess the sessions proposed by the recommender system with these quality measures. Beforehand, this dissertation presents a set of synthetic log generators allowing to experimentally assess the proposals of similarity measures between queries and sessions, as well as the recommendation system. Logs, based on sessions devised by Master's students in Business Intelligence, are also used to show that the recommendation system can recommend effective sessions from diverse analysis sessions.

Dissertation Content

This dissertation is organized as follows:

Chapter 1 is devoted to the state of the art of usage mining for query recommendation. We begin by presenting the basics of recommender systems as well as their evaluation

methods. Then, a detailed study of the recommender systems in Databases and Data Warehouses is proposed. Since the sequencing of queries has to be considered for our recommender system, a state of the art of session and query similarities is given. This chapter also introduces the criteria that are considered as relevant for similarity measures adapted to an OLAP recommender system.

Chapter 2 introduces the definitions that are required to propose similarity measures for OLAP sessions. The models of data, query, session and log are formally defined. In particular, the query model is based on the query expression. Then, similarity measures between queries, sessions and logs are presented.

Chapter 3 is devoted to the recommender system whose three phases are detailed: *Selection*, *Ranking*, and *Tailoring* phases. *Selection* phase compares each past session with the current session using similarity measure between sessions and select possible recommendations. *Ranking* phase ranks the recommendations obtained previously by promoting those having similar queries between the possible recommendations. *Tailoring* phases adapts the best ranked recommendation to the current session.

Chapter 4 reports the experiments assessing the relevance of the recommendation system and the similarity measures between queries and sessions. Specific tests conducted with similarity measures are composed of two evaluations: a first one is subjective and is based on user questionnaires indicating the user perception regarding query and session similarities; the second one is objective and different templates obtained with synthetic generators allow to validate the capabilities of the measures. The recommender system is assessed by its efficiency and its effectiveness, based on different types of synthetic analysis sessions and real logs devised by Master's students. The quality measures defined from the criteria expressed for session recommendation allow to assess the effectiveness of the recommender system.

The conclusion presents the contributions of this thesis and their perspectives by discussing the problem of *Cold Start* during session design but also a benchmark for OLAP sessions and the adaptation of the recommender system in contexts other than OLAP.

Chapter 1

Usage Mining in Query Recommendation

1.1 Introduction

This chapter is devoted to a state of the art of the usage mining in query recommendation. First of all, an overview of the basics of recommender systems but also their evaluations are given in Section 1.2. Section 1.3 discusses recommendation systems in Databases and Data Warehouses where basic knowledge in databases (see [Abiteboul et al., 1995]), data mining (see [Han, 2005]) and data warehouses (see [Golfarelli and Rizzi, 2009]) are required. Since the recommendation system proposed in this dissertation is also based on similarity measures, a state of the art on session and query similarities is given in Section 1.4. Requirements and observations that we consider relevant for recommendation system and similarity measures, in an OLAP context, are given in Section 1.5.1 and 1.5.2, respectively. Finally, Section 1.6 concludes the chapter.

1.2 Recommender Systems

1.2.1 Recommender Systems Basics

Recommender systems [Adomavicius and Tuzhilin, 2005] are now well established as an information filtering technology, used in a wide range of domains, from E-commerce to web search to social networks. They are traditionally categorized as either content-based (suggestions are based on the user's past actions only) or collaborative (suggestions are based on similarities between users) or hybrid combination thereof. The basic underlying model is the users×items matrix that records (boolean or non boolean) ratings expressing the user's interest in the items. Many methods have been proposed for estimating the missing ratings in this large and sparse matrix, either by using the full collection of ratings or by computing models learned from these ratings.

1.2.1.1 Content-based Approach

Typical content-based recommendation is based on the similarity between item profiles and user profiles, where item profiles consists of scores for selected item features and user profiles are constructed using highly rated item profiles. Rating estimates are found using simple K-nearest neighbors search or machine learning techniques like bayesian classification or neural networks.

Example 1.2.1. *We illustrate the content-based approach with an example of recommendation of movies. We propose a set of movies ((1) Per qualche dollaro in piu (2) Il buono, il brutto, il cattivo (3) C'era una volta il West, (4) Reservoir Dogs, (5) Requiem for a Dream, (6) The Fountain, (7) Django Unchained) rated by four users. The matrix of ratings is the following:*

	(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*
Arnaud		*0.9*	*1.0*				
Julien			*0.1*			*0.8*	
Matteo	*1.0*	*1.0*		*0.7*	*0.6*		*0.8*
Patrick	*1.0*			*0.7*	*0.7*		*0.7*

Supposing that the features for the items are (recent movie, significant budget, popularity), each movie can be modeled with a vector:

- *(1): (0, 0.01, 0.83)*
- *(2): (0.02, 0.01, 0.9)*
- *(3): (0.06, 0.05, 0.8)*
- *(4): (0.57, 0.01, 0.78)*
- *(5): (0.74, 0.05, 0.68)*
- *(6): (0.87, 0.35, 0.51)*
- *(7): (1, 1, 0.81)*

Now, a profile can be defined for each user, by computing for instance the weighted average of movie profile ratings :

- *Arnaud:* $\frac{0.9*(0.02,0.01,0.9)+1.0*(0.06,0.05,0.8)}{2}$ *= (0.04, 0.03, 0.81)*
- *Julien:* $\frac{0.1*(0.04,0.05,0.8)+0.8*(0.52,0.4,0.55)+0.8*(0.87,0.35,0.51)}{3}$ *= (0.372, 0.2, 0.3)*
- *Matteo:* $\frac{1.0*(0,0.01,0.83)+1.0*(0.02,0.01,0.9)+0.7*(0.57,0.01,0.78)+0.6*(0.74,0.05,0.68)+0.8*(1,1,0.81)}{5}$ *= (0.33, 0.17, 0.67)*
- *Patrick:* $\frac{1.0*(0,0.01,0.83)+0.7*(0.57,0.01,0.78)+0.7*(0.74,0.05,0.68)+0.7*(1,1,0.81)}{4}$ *= (0.40, 0.19, 0.60)*

Suppose that we want to recommend a movie for Arnaud, whose profile can be described as preferring old popular movies with a small budget. Applying a similarity measure, for instance the cosine similarity, between his profile and the movie profiles that he did not rate, the movie (1) will be recommended. Indeed, cosine((0.04, 0.03, 0.81), (0, 0.01, 0.83)) = 0.97 which is the highest value.

However, the content-based approach suffers from several problems such as the recommendation depending on the features given for the items and also the cold start problem, i.e. no recommendation is provided if a user has never rated any item.

1.2.1.2 Collaborative Filtering Approach

The classical collaborative approaches globally consider the entire matrix, estimating ratings with techniques like K-nearest neighbors search, using similarities between user profiles (or item profiles) consisting of the ratings given by users to items.

Example 1.2.2. *Supposing that we want to estimate the ranking of the movie (2) for Patrick. Using the vectors of profiles defined for each user in the previous example, we can use a similarity measure, for instance the cosine similarity, between Patrick's profile and the others. The closest is Matteo's profile with cosine((0.40, 0.19, 0.60), (0.33, 0.17, 0.67)) = 0.99. Thus, the rating of the movie (2) can be deduced by the score given by Matteo, weighted by the similarity between the profiles, i.e 0.99.*

The collaborative approach generally suffers from three main problems. The first one, as for the content-based approach, is the cold start problem. The second one is the sparsity of the ratings given by the users. Indeed, due to the large number of items, a user rates a very small number of them. Consequently, each item has few ratings and the approach can lead to never recommend items, even if the ratings are high. The third one is the important computation time required for obtaining scores of recommendation due to the large number of users and items.

1.2.1.3 Hybrid Approach

Hybrid algorithms, that combine two or more recommendation approaches, are assumed to be the most effective in particular situations [Töscher et al., 2009], and less sensitive to some of the common problems in recommender systems such as the cold start problem.

In particular, [Töscher et al., 2009] adopts the *latent factor model*. The idea of this model is to approximate the rating matrix by characterizing both users and items with latent factors. It is supposed in this model that the ratings depend on specific factors for a domain (for instance the degree of movie understanding). Thus the aim of the model is to deduce the factors from the ratings. For this, a solution is to decompose the rating matrix into the product of the user and item feature matrices (for instance using the Singular Value Decomposition (SVD) technique ([Sarwar et al., 2000])). The problem is that the rating matrix is sparse while the decomposition of the matrix cannot be applied in the presences of missing values. A solution is to identify the minimal number of factors allowing to minimize the number of missing values among the ratings. Moreover, the use of additional data (provided by the users such as navigation behaviors, age, sex etc.) can replace the missing values by scores computed with the *matrix factorization* principle (see [Koren et al., 2009]).

1.2.2 Recommender System Evaluation

A comprehensive survey of collaborative recommender systems evaluation is presented in [Herlocker et al., 2004]. Recommender systems are mostly evaluated with accuracy based measures. More precisely, [Gunawardana and Shani, 2009] distinguishes between different recommendation tasks and their evaluation, notably i) rating prediction, that is

typically evaluated with error measures like RMSE or MAE, ii) good item recommendation, where ratings are boolean, typically evaluated with precision, recall and false positive rate measures, and iii) utility optimization, where recommendations are ranked in how they maximize some utility function, and whose evaluation is based on a model of the way user interacts with recommendations to check if the ranking matches how the user would have ordered the items.

It has been recently understood that accuracy alone fails to capture the usefulness of recommendations. Other objective metrics related to suitability are being developed, most notably: coverage, learning rate, novelty and serendipity.

Coverage [Herlocker et al., 2004, Ge et al., 2010] concerns the degree to which recommendations cover the set of available items and the degree to which recommendations can be generated to all potential users. More precisely, *prediction coverage* measures the percentage of the items for which the system is able to generate a recommendation, and *catalogue coverage* measures the percentage of the available items which effectively are ever recommended to a user. Coverage should be measured in combination with accuracy, so recommenders are not tempted to raise coverage by making bogus predictions [Herlocker et al., 2004].

As the performance of recommender systems vary with the amount of learning data available, the learning rate is used to measure the quality (often accuracy) of recommendations as a function of the data available. This measures for instance how much the system is sensitive to the cold-start problem (when only little data is available).

Serendipity and novelty are concerned with the non obviousness of recommendations. Novelty directly measures non obviousness, often by referring to a fixed list of obvious recommendations, while serendipity measures how far novel recommendations may positively surprise users, for instance by reporting the rate of useful novel recommendations.

As to evaluation protocol, [Gunawardana and Shani, 2009] discusses different methods to evaluate a recommendation algorithm. In fact, two types of evaluations are possible:

– the online evaluation.
– the offline evaluation.

The online evaluation relies on users testing the recommendation system. This evaluation allows to analyze the user's behaviors when recommendations are proposed. Thus, various recommendations can be proposed to users in order to know how the system can help her to conduct the analysis or search, i.e. knowing the user's intent (for instance, the specific information that she needs) and the user's context (for instance, the items that she already knew). The drawback of this evaluation is that in most cases, it is very costly to organize a survey and that is why offline evaluations are often preferred.

An offline evaluation is conducted without the help of users testing the recommendation system. Consequently, the evaluation process has to be as close as possible to an online evaluation by imitating the user's behaviors when he is exposed to recommendations. Thus, this evaluation is usually done by using data coming from previous action sequences that have been devised by multiple users. Several ways are possible to test the recommendation of items in an offline evaluation but each of them are based on the simulation of user past actions and hidden items. The three following proposals consider a single training set (i.e the set of initial data that are the sequences of actions) used by the system to propose

recommendations:

- The first is based in the time-stamps of the user actions, when they are available in the data. The solution proposes to consider as past actions all the sequence of actions performed before a given randomly time (the same for each user). Thus, all the actions after this time are hided and the recommendation system attempts to propose the same actions.
- The second is close to the previous one but considers a different randomly time separating the past and hidden actions, for each user. This allows to test the recommendation system according to various lengths of past action sequences.
- The third is not based on a particular time-stamp whose we assume this is not important to take into account. The idea simply considers a sample of users but also a sample of past action sequences and hidden actions sequences, randomly chosen.

The drawback of the three previous proposals is they cannot conduct several tests with different training sets, since all the data are used by the system to propose a recommendation. Thus, another approach, allowing to partition the set of initial data, is the cross-validation principle ([Stone, 1974]). The idea is to divide the data into a fixed number of test sets. One test set is used to validate the recommendation while the others sets are considered as training sets used by the system. This solution is relevant only if the partitions are independent of each other. Indeed, if a user action of the training set depends on another action presents into the validation set, the evaluation may be biased.

1.2.3 Sequence Recommendation in the Web

With the huge amount of data available in the web, the recommendation of web queries is essential to propose to the users a solution to easily navigate large websites. For example, web page recommendation systems are usually implemented into the web server, by proposing the next web pages on which a user could be interested. [Ögüdücü, 2010] gives an overview of these systems, focusing on the web mining techniques and the use of user traces. As discussed in [Borges and Levene, 2000] and [Madria et al., 1999], the web mining can be divided into three sub-domains: *Web Structure Mining*, *Web Content Mining* and *Web Usage Mining*. *Web Structure Mining* and *Web Content Mining* are focused on the data directly available in the web page such as the textual information or the links between the pages. *Web Usage Mining* uses data mining techniques in order to identify patterns coming from user traces (e.g. user profiles, web logs, cookies etc.).

The sequence recommendation in the web is particularly relevant to study against the aim of this dissertation (suggesting queries for continuing navigation). Indeed, this research area has long addressed the problem of sequence recommendation using proven techniques. However, there are important differences in the database and web querying context: in the Web Querying, simple keyword queries are used and there is no declarative query language, the use of a search engine imposes two levels of answers: the search engine result page (SERP) which is a collection of links, and documents that can be consulted.

In particular, several recent approaches are based on clickstream analysis, such as [Fonseca et al., 2005] and [Jones et al., 2006] that extract adjacent query pairs for query expansion or query substitution. Other works ([Baeza-Yates et al., 2004], [Beeferman and Berger, 2000], [Wen et al., 2001]) are also focused on clustering approaches

which group similar queries into different clusters and suggest those that are in the same cluster than a current query.

Data mining techniques, that are used in the *Web Usage Mining*, are association rule extraction, sequence pattern generation and clustering. Each of these techniques are respectively described in Sections 1.2.3.1, 1.2.3.2 and 1.2.3.3.

1.2.3.1 Association Rules

The principle of association rule extraction was introduced by [Agrawal et al., 1993]. Supposing a set of transactions D where a transaction is a set of items (in our case an item could be an URL), an association rule is defined as the implication $X \rightarrow Y$ where X is the *body* of the rule and Y is the *head* of the rule. The intuition of these types of rules is to identify regularities between X and Y. In order to measure the relevance of the rule, i.e. how the rule is strong with respect to the transactions, two common measures are used: the *support* and $confidence$ measures. The support measure computes the number of transactions including $X \cup Y$ on the total number of transactions in D. The confidence measure computes the ratio between the support of $X \cup Y$ and the support of X.

In order to automatically identify the association rules, frequent itemsets are extracted from a set of initial transactions. The Apriori algorithm ([Agrawal et al., 1993]) is a classical approach in this context by mining all the frequent itemsets whose support value is higher than a given threshold.

To recommend web pages using association rules, [Mobasher et al., 2001] proposes to capture the behavior of a current session by using a fixed-size window. Considering a current session c_s as a sequence of n web pages, and a window of size m (with $m \leqslant n$), only the last sequence of m pages of c_s represents the current behavior. Then, the $m+1$ frequent itemsets are mined from the log sessions and the association rules are extracted from these itemsets. Only the rules, whose body matches with the fixed-size window and head is a singleton, are selected. Finally, each head is added to the recommendation set, and is scored with the confidence value of its rule.

An example of the intuition is given below.

Example 1.2.3. *Let* $S = \{s_1, s_2, s_3\}$ *be a set of past sessions with*
$s_1 =< home/index, home/page1 >$,
$s_2 =< home/index, home/page2 >$, $s_3 =< home/index, home/page1, home/page2 >$.

Let $c_s =< home/page3, home/index, home/page1 >$ *be a current session. Let* $w_c =< home/index, home/page1$ *be the current window taking the two last pages viewed in* c_s.

After applying the Apriori algorithm with a minimal support value of $\frac{1}{3}$, *one frequent itemset of size 3 is found:* $\{home/index, home/page1, home/page2\}$.

The association rules related to this frequent itemset are:
– $home/index, home/page1 \rightarrow home/page2$ *(supp:* $\frac{1}{3}$*) (conf:* $\frac{1}{2}$*)*
– $home/index, home/page2 \rightarrow home/page1$ *(supp:* $\frac{1}{3}$*) (conf:* $\frac{1}{2}$*)*
– $home/page1, home/page2 \rightarrow home/index$ *(supp:* $\frac{1}{3}$*) (conf:* 1*)*

Only the rule $home/index, home/page1 \rightarrow home/page2$ *is considered because the body matches with* w_c *and the head is a singleton. Consequently, the page* $home/page2$ *is rec-*

ommended to the user.

Other variants, based on the extraction of association rules, exists. For instance in [Tao et al., 2003], the pages are weighted using a function of time spent at a page, and a weighted association rule algorithm is proposed. In [Liu et al., 1999], the use of different support values for different page is possible. The aim being to give a less important weight for the page frequently found in the URL (typically the top level of the URL hierarchy).

1.2.3.2 Sequence Pattern

Sequence pattern mining was initially proposed by [Agrawal and Srikant, 1995]. The principle is closed to association rule mining, but the main difference is that the order between itemsets is taken into account. Thus, assuming a set of transactions D, a transaction is a sequence of itemsets. The idea is to finds all frequent sequence patterns from the transactions. For instance, a possible sequence pattern could be $\{home/index\} \rightarrow \{home/page1\}$ meaning that when the page $home/index$ is viewed, the page $home/page1$ is generally viewed later. The support measure estimating if the sequence pattern is strong or not among the transactions, is defined as the ratio between the number of transactions containing the frequent sequence and the total number of transactions. GSP ([Srikant and Agrawal, 1996]) is a classical algorithm to extract sequence patterns.

In the web context, sequential patterns are used to capture web pages that are often viewed among a large proportion of past sessions.

Markov-based models are also a solution to identify sequence patterns. Indeed, the set of sequential patterns that are extracted from past sessions, can be viewed as stochastic processes. Overall, the Markov model of the first order are used to model sequences. Considering a sequence s composed with an ordered list of states (for instance a state can be an URL), the principle is that the probability to reach a state only depends of the previous state. Consequently, the probability to generate a sequence is: $P(s) = P(s_1) \prod_{t=2}^{l} P(s_t|s_{t-1})$ where s_t is a state at the position t in the sequence and l is the total number of states in the Markov model. In [Borges and Levene, 2000], a Markov model is used to predict next pages from user sessions. An Hypertext Probabilistic Grammar is proposed to model the sequences of web pages visited by users.The probability that a web page follows another one is based on the number of times a Web page was required, the number of times it was the first page in the sessions and the number of times it was the last. A parameter α of the model is introduced to give an initial probability for the Web pages appearing first in the sequences. It has also been supposed that only the last visited web pages are relevant to predict next pages. Thus, the probability of a sequence is defined as: $P(s) = P(s_1) \times P(s_1, s_2) \times \cdots \times P(s_{k-1}, s_k)$ where $P(s_i, s_j)$ is the probability to reach s_j from s_i. Consequently, the web page sequences following the last visited pages of a user into the Markov model, are predicted if their probabilities are high.

Example 1.2.4. *Let S be the set of transactions (here user sessions), given in the previous example. Figure 1.1 depicts the grammar representing the user sessions with $\alpha = 0.5$. We note that two additional states S and F, corresponding to the initial and final states of the model are included.*

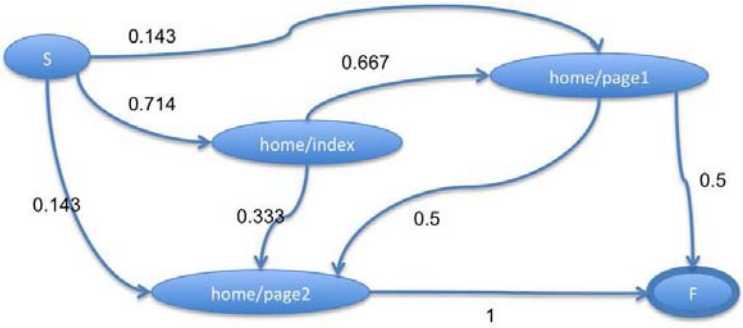


Figure 1.1: Hypertext Probabilistic Grammar from user sessions of Example 1.2.4 with $\alpha = 0.5$

The initial probabilities of each web page are computed as follow, considering that a total of 7 web pages have been visited:

- $P(< home/index >) = \frac{0.5*3}{3} + \frac{0.5*3}{7} = 0.714$ *(home/index appears three times in first position among the sequences)*
- $P(< home/page1 >) = \frac{0.5*2}{7} = 0.143$
- $P(< home/page2 >) = \frac{0.5*2}{7} = 0.143$

*From the grammar, we can now compute probabilities to have particular sequences. For instance the probability to have the sequence <home/index, home/page1> is 0.714 * 0.667 = 0.476.*

1.2.3.3 Clustering

Clustering methods in Web Page Recommendation context can be categorized in two groups: methods that group pages according to their contents ([Han, 2005]) or those grouping by the page frequencies based on the user sessions ([Glover et al., 2002]). In particular, to group similar pages, similarity measures between feature vectors describing web pages are defined, typically Cosine Similarity or Euclidian Distance. For instance, a feature vector can be based on the anchor texts whose links point to the web page to describe ([Glover et al., 2002]). A page is described by a set of terms weighted by their frequencies among the anchor text vectors.

Different clustering approaches exist such as partitioning methods or hierarchical methods. For instance, the *k-means* algorithm ([Steinhaus, 1956], [MacQueen, 1967]) is popular in data mining and its aim is to partition n items in k clusters, at most. The principle is to assign an item to a cluster only if this item is similar to the mean value of the items previously added in the cluster. The *Girvan-Newman* algorithm ([Girvan and Newman, 2002]) is a graph clustering. The aim of this algorithm is to detect clusters where items into the same clusters are very similar between them but very distant with items of other clusters. To do this, the algorithm is based on the betweenness approach, i.e. if an edge is too central (given by betweenness score that is the number of shortest paths taking this edge) between items of different clusters, this edge is removed from the initial graph. Once all edges with high betweenness scores have been removed, a dendogram structure is produced.

Algorithm 1 *Girvan-Newman* Algorithm

Input: G: an undirected weighted graph
Output: A dendogram
1: calculate the betweenness for all edges in the network.
2: **repeat**
3: remove the edge with the highest betweenness;
4: recalculate the betweenness for all edges affected by the removal;
5: **until** no edges remain

Table 1.1: User sessions for Example 1.2.5

SID	User Session
s_1	p_5, p_4, p_3, p_6
s_2	p_5, p_4, p_3
s_3	p_5, p_4, p_6
s_4	p_1, p_5, p_6
s_5	p_1, p_2, p_8
s_6	p_2, p_7, p_9
s_7	p_2, p_8, p_9, p_7
s_8	p_2, p_8, p_7
s_9	p_2, p_7, p_9

The pseudocode is given in Algorithm 1.

The use of web page clusters in a recommender system allows to define profiles that can be obtained by aggregating all the vectors in one. The identification of the profiles is a first step before recommending web pages and are used for preprocessing web user sessions. For example, the current session can be matched with the content of the clusters represented by their centroid (i.e. the profile). Thus, a set of pages is recommended and are ranked if possible. Then, a set of top-k pages is recommended.

Clustering techniques can also be applied for grouping similar user sessions ([Banerjee and Ghosh, 2001], [Ögüdücüand Özsu, 2006], [Wang and Zaïane, 2002]). The user session is compared to each centroid of a cluster. The centroid having the highest similarity value is selected and the recommender system uses the centroid to recommend a top-k pages, previously ranked. The sessions being considered as a sequence of pages, the similarity measures used in a clustering algorithm have to take into account this specificity. The Longest Common Subsequence (LCS) is able to compute a similarity value between two sessions, based on the relative time spent on the longest common subsequence ([Banerjee and Ghosh, 2001]). Another measure is based on the sequence alignment. The aim is to find the longest alignment between the sessions to compare by including gaps or not. A more detailed study of sequence comparison is presented in Section 1.4.

To recommend web pages, association rules combined with Markov model could predict the next web pages from a current session using the clusters (see [Khalil et al., 2006]).

Example 1.2.5. *Consider the set of sessions given in Table 1.1. Applying the Girvan − Newman algorithm, the graph of sessions is computed especially based on betweenness values. Two clusters are identified that are:* $\{s_1, s_2, s_3, s_4\}$ *and* $\{s_5, s_6, s_7, s_8, s_9\}$*. Then, a*

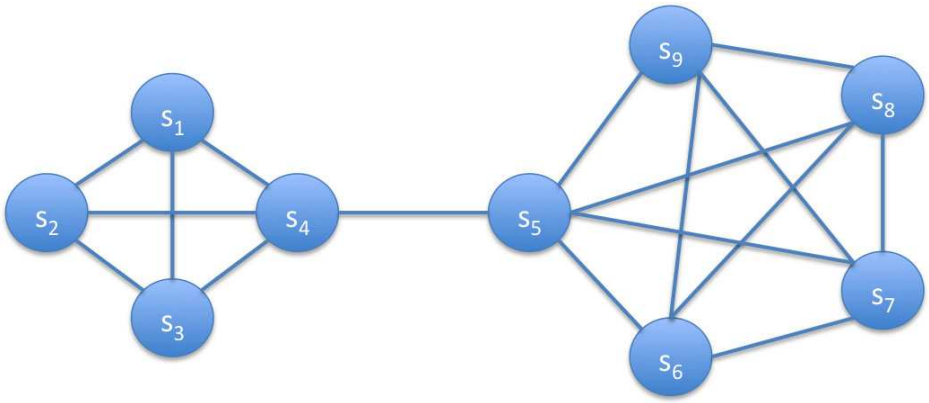


Figure 1.2: Graph of similar sessions from Table 1.1

Markov model could be formed using these clusters to recommend next pages.

However, these approaches are often costly in terms of computation time and are commonly used in an off-line step. But a small number of profiles are generally expected. Thus, these profiles are workable during an online step, where they are matched with a current user session. Consequently, the recommendation system is able to propose items (for instance Web pages) to the current session.

1.3 Query Recommendation in Databases and in Data Warehouses

Recently recommender systems started to gain interest in the database community. In particular, recent papers [Jagadish et al., 2007, Khoussainova et al., 2009, Kersten et al., 2011] pointed out that DBMSs are ubiquitous and are becoming increasingly complex to use, mainly due to the complexity of schemas and the huge size of the instances. Getting to know what is in the database instance, discovering interesting information, is simply becoming out of human reach, users being left with the burden of navigating the database instance by iteratively evaluating queries that may be difficult to express. If this consideration is particularly relevant for relational databases (e.g., scientific databases like the SkyServer [1]), it is obviously even more relevant in a data warehousing context, where one prominent use of such systems is to analyze the warehouse instance with OLAP queries as a basis for decision support.

However, as noted in [Eirinaki et al., 2013] recommending database queries differs from classical recommender system approaches, at least in three main aspects: 1) Queries are expressed with query languages that are declarative and sophisticated, and we call $fragments$ the parts of query expression that are projections, selection predicates and tables. 2) no explicit ratings can be exploited, and 3) recommended queries should be intuitive so that users can understand and refine them if necessary. Therefore, query recommendation in

1. http://cas.sdss.org

databases is closer to a *good item* recommendation task (see Section 1.2.2), where past actions are taken from query logs or the user's current session.

In addition, data warehouses can be seen as databases with the following peculiarities:

1. A data warehouse is a database shared by *multiple users, mostly executives, whose interests are diverse and may vary over time.*
2. A data warehouse is a read-mostly database and its instance has an inflationist evolution (data is added and never or very seldom deleted). It is for instance likely that a user issues periodically *similar sequences of queries* more than once, in the sense that *queries may not be fully identical.*
3. A data warehouse has a particular schema that reflects a known topology, often called the lattice of cuboids, which is systematically used for navigation.
4. A typical analysis session over a data warehouse is a sequence of queries sharing an analytical goal, each one being written based on the past results of the session.

We next examine the state of the art approaches for recommending queries in databases, first in a non data warehouse context and then in a data warehouse context.

1.3.1 Query Recommendation in Databases

This section discusses recommendation in Databases.

In non data warehouse databases, the most representative works for supporting interactive database exploration by means of recommendations are QueRIE, SnipSuggest and YMALDB.

QueRIE [Eirinaki et al., 2013] transfers the users×items matrix of the Collaborative Filtering model in the database context and turns it into a session×tuples (tuple-based approach) or sessions×query fragments (fragment-based approach) matrix to choose among the past queries stored in a query log, the ones to recommend. More precisely, the framework represents a query and a session as vectors of basic elements that are tuples or fragments, respectively. These vectors represents the signature of the queries and are used to compute a similarity (using cosine or Jaccard similarity) between a current session and the former sessions. Then, a vector aggregates all former session vectors to represent the log and each query vector is compared to this vector. Finally, the top-k queries having the best similarity scores are recommended to the user.

Example 1.3.1 illustrates the intuition of both approaches.

Example 1.3.1. *Let D be a database composed with one table $Movies$ whose schema is $Movie[title, year, genre, director]$. Let I_{Movie} be the instance of table $Movie$ represented in Table 1.2:*

Let l be a log composed of two sessions: $s_1 = <q_1>$
with $q_1 = SELECT\ title\ FROM\ Movies\ WHERE\ director =$ "$Sergio\ Leone$"
$s_2 = <q_2, q_3>$
with $q_2 = SELECT\ title\ FROM\ Movies\ WHERE\ genre =$ "$Western$" and
$q_3 = SELECT\ title, year\ FROM\ Movies\ WHERE\ genre =$ "$Western$"

Let s_c be the current session composed of one SQL query q_c:
$<SELECT\ title, genre\ FROM\ Movies\ WHERE\ director =$ "$Quentin\ Tarantino$"$>$

	title	*year*	*genre*	*director*
(1)	*Per qualche dollaro in piu*	1965	*Western*	*Sergio Leone*
(2)	*Il buono, il brutto, il cattivo*	1966	*Western*	*Sergio Leone*
(3)	*C'era una volta il West*	1968	*Western*	*Sergio Leone*
(4)	*Reservoir Dogs*	1992	*Thriller*	*Quentin Tarantino*
(5)	*Requiem for a Dream*	2000	*Drama*	*Darren Aronofsky*
(6)	*The Fountain*	2006	*Drama*	*Darren Aronofsky*
(7)	*Django Unchained*	2012	*Western*	*Quentin Tarantino*
(8)	*Mobius*	2013	*Thriller*	*Eric Rochant*

Table 1.2: Table *Movie* of Example 1.3.1

Using the tuple-based approach, the queries are represented as the following vectors (0 means that the tuple is not in the query answer, 1 otherwise):

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
q_1	1	1	1	0	0	0	0	0
q_2	1	1	1	0	0	0	1	0
q_3	1	1	1	0	0	0	1	0
q_c	0	0	0	1	0	0	1	0

The sessions aggregate their respective query vectors (with the sum of the vectors). The results are the following:

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
s_1	1	1	1	0	0	0	0	0
s_2	2	2	2	0	0	0	2	0
s_c	0	0	0	1	0	0	1	0

To represent the set of former sessions with respect to the current session, the session vectors of s_1 and s_2 are aggregated by s_{pred} and weighted by their respective similarity (here cosine similarity) with the current session vector:

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
s_{pred}	1	1	1	0	0	0	1	0

Finally, each query of the former sessions are compared to s_{pred}. Considering the top-2 queries, q_2 and q_3 are closer to s_{pred} than q_1. Thus, q_2 and q_3 are recommended to the user.

Using the fragment-based approach, the fragment of the queries are the elements of the vectors. In order to be less restrictive, the WHERE clauses are represented with patterns. For instance, director = "Quentin Tarantino" and director = "Sergio Leone" will be represented by the same pattern: director EQU STR. Here are the vectors of the different queries:

	genre	*title*	*year*	*Movies*	*director EQU STR*	*genre EQU str*
q_1	0	1	0	1	1	0
q_2	0	1	0	1	0	1
q_3	0	1	1	1	0	1
q_c	1	1	0	1	1	0

The sessions aggregate their respective query vectors. The results are the following:

	genre	*title*	*year*	*Movies*	*director EQU STR*	*genre EQU str*
s_1	0	1	0	1	1	0
s_2	0	2	1	2	0	2
s_c	1	1	0	1	1	0

To represent the set of former sessions with respect to the current session, the session vectors of s_1 and s_2 are aggregated and weighted by their respective similarity with the current session vector:

	genre	*title*	*year*	*Movies*	*director EQU STR*	*genre EQU str*
s_{pred}	0	1.976	0.555	1.976	0.866	1.110

Finally, each query of the former sessions are compared to s_{pred}. The similarity table is given below:

	q_1	q_2	q_3
s_{pred}	0.876	0.920	0.884

Considering the top-2 queries, q_2 and q_3 are closer to s_{pred} than q_1. Thus, q_2 and q_3 are recommended to the user.

The tuple-based approach appears to be more precise than the fragment-based approach because the instance of the database is used. But this solution can be very costly since the computation time depends on the number of tuples in the database. The fragment-based approach seems to be more efficient, relying only on the query expression.

The YMALDB approach [Drosou and Pitoura, 2013] focuses on exploring query results for locating interesting pieces of information. In this approach, the answer retrieved by a query is analyzed to detect correlations, that are subsequently used, together with correlations existing in the database instance, to synthesize a new query to recommend. More precisely, the framework does not consider the use of former sessions, unlike [Eirinaki et al., 2013], for recommending queries but only the result as that of the current query and the database instance. The approach finds interesting (Attribute, Value) pairs among those present into the result of a current query. To do this, an *interestingness* score is calculated for each possible (Attribute, Value), favoring pairs rare in the database, but well supported in the query result. The (Attribute, Value) pair having the best score will allow to synthesize a new query taking the same clauses as that of the current query but replacing the selection predicates by the pair. Example 1.3.2 illustrates the intuition.

Example 1.3.2. *Consider the database D of Example 1.3.1 and the current query q_c: $SELECT\ title, genre\ FROM\ Movies\ WHERE\ director = \text{“}QuentinTarantino''$*

The result of q_c over D is:

title	genre
Reservoir Dogs	*Thriller*
Django Unchained	*Western*

Thriller and Western values of the attribute genre are equally frequent in the query result. But looking at the database, Thriller value is less frequent than Western. Thus, Thriller is selected to replace the selection predicate of q_c, i.e. (director, “Quentin Tarantino'').

Consequently, the recommended query is:
$SELECT\ title, genre\ FROM\ Movies\ WHERE\ genre =$ "$Thriller$"
With this query, the user can discover the new movie "$Mobius$".

The SnipSuggest approach [Khoussainova et al., 2010] is meant to assist query formulation by recommending fragments of database queries (SQL clauses) to a user currently writing a query. It is based on a probabilistic modeling of past sessions, in the sense that it extracts correlations existing in the query log to recommend the most likely fragment for the user's current query. More precisely, a recommendation is given for a particular clause currently written by a user (typically, recommendations of table name are given during the writing of the $FROM$ clause, selection predicates or joins for $WHERE$ clause etc.). Beforehand, the set of all fragments is extracted from the past queries. Each possible sub-set is added into a directed graph of fragment sets where the n+1 fragments of a child node are a superset of the n fragments of their parents. A confidence score is given for each edge. Thus, each parent-child relation can be considered as an association rule. Indeed, the body is the parent node and the head is the (n+1)th fragment of the child node (that differs between the child and the parent nodes). Regarding the query being currently written, all of its fragments are extracted. Each rule, whose body exactly contains these fragments, is selected in order to consider the head as a possible recommendation. Because the number of possible recommendations can be high, the top-k recommendations are considered. Several possibilities exist to obtain the top-k recommendations but the main is to select the k heads of rules maximizing the confidence scores (either by a sum or recalculating the confidence from the k heads). Example 1.3.3 gives the intuition.

Example 1.3.3. *Let l be a log composed of 20 former queries:*
(10X) $SELECT\ title\ FROM\ Movies\ WHERE\ genre =$ "$Western$",
(2X) $SELECT\ title\ FROM\ Movies\ WHERE\ director =$ "$Quentin\ Tarantino$" $AND\ genre =$ "$Western$",
(8X) $SELECT\ title\ FROM\ Movies\ WHERE\ director =$ "$QuentinTarantino$" $AND\ genre =$ "$Thriller$" $AND\ year > 1995$>

Let q_c be the partial query being currently written:
$SELECT\ title\ FROM\ Movies\ WHERE\ director =$ "$Quentin\ Tarantino$"

Consider the following association rules, matching with q_c and extracted from the former queries:

- $SELECT\ title, FROM\ Movies, WHERE\ director =$ "$Quentin\ Tarantino$" $\rightarrow$ $WHERE\ genre =$ "$Western$"
  $(supp : \frac{1}{10})(conf : \frac{1}{5})$
- $SELECT\ title, FROM\ Movies, WHERE\ director =$ "$Quentin\ Tarantino$" $\rightarrow$ $WHERE\ genre =$ "$Thriller$"
  $(supp : \frac{2}{5})(conf : \frac{4}{5})$
- $SELECT\ title, FROM\ Movies, WHERE\ director =$ "$Quentin\ Tarantino$" $\rightarrow$ $WHERE\ year > 1995$
  $(supp : \frac{2}{5})(conf : \frac{4}{5})$

Suppose a user, unfamiliar with director "$Quentin\ Tarantino$", *wishes to specify his WHERE clause. Top-2 recommendations are:*

- *"Thriller", "year* $>$ *1995" by maximizing the sum of the confidence scores*
- *"Western","year* $>$ *1995" by maximizing the confidence scores from rules whose head includes "Western" or "year* $>$ *1995"*

1.3.2 Query Recommendation in Data Warehouses

This section discusses about recommendation in Data Warehouses.

In data warehouses, several works [Sarawagi, 1999, Sarawagi, 2000, Sathe and Sarawagi, 2001, Sapia, 2000] were interested either in suggesting interesting/surprising tuples by mining the warehouse instance [Sarawagi, 1999, Sarawagi, 2000, Sathe and Sarawagi, 2001] or prefetching tuples using a probabilistic modeling of a query log [Sapia, 2000, Aufaure et al., 2013].

More precisely, [Sarawagi, 1999], [Sarawagi, 2000] and [Sathe and Sarawagi, 2001] are based on the analysis of the query answer. Considering that an answer can be represented as a cross-table, the two approaches investigate interesting differences between pairs of cells into the cross-table (for instance an important difference). The use of these particular differences are relevant in a context of recommendation. Indeed, these cells are typically the phenomenon that a user will explore by devising OLAP sessions. Three operators, $Diff$ ([Sarawagi, 1999]), $Inform$ ([Sarawagi, 2000]) and *Relax* ([Sathe and Sarawagi, 2001]), allow to mine query results and the cube by automatically navigating into the group-by set in order to explain interesting differences between cells. Starting from an initial query result, the $Diff$ operator identifies important differences between values of cell pair, that can be qualified as interesting, and applies drill-down operation over the group-by set, until obtaining a new interesting cell pairs. These OLAP operations allow to explain the initial difference between values of the cell pair, by showing the pairs of cells that contribute the most to the difference. The *Relax* operator proceeds with the same principle than the $Diff$ operator but applying a sequence of selections and Roll-up operations. The $Inform$ operator finds surprising parts of the cube (using the Maximum Entropy principle) and avoids to propose parts a cube already viewed by a user. Example 1.3.4 illustrates the intuition.

Example 1.3.4. *We introduce the running example of this dissertation. IPUMS is a public database storing census microdata for social and economic research [Minnesota Population Center, 2008]. Its* CENSUS *multidimensional schema has five hierarchies, namely* RACE, TIME, SEX, OCCUPATION, *and* RESIDENCE, *and measures (aggregated either by Sum, Max, Min or Avg)* Income, PropInsr *(property insurance cost),* PerWt *(person weight),* CostGas, CostWtr, *and* CostElect. *It is* City$rollsupto$State *(the complete roll-up orders are shown in Figure 1.3);*

A possible query, over this schema, is :
"What is the evolution of the average income for each sex between 2000 *and* 2003*".*
The associated result is depicted in Figure 1.4a. As surrounded in red, we can identify a significant increase of the average income between 2002 and 2003 for female.

By applying the Diff operator, years 2002 *and* 2003 *and sex Female will be selected to refine the analysis. Then, a sequence of drill-down operations is applied and the cell pairs explaining the increase of the average income are considered. For example, a possible*

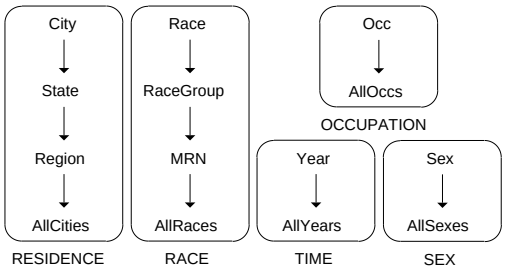


Figure 1.3: Roll-up orders for the five hierarchies in the CENSUS schema (MRN stands for MajorRacesNumber)

AvgIncome	Year			
Sex	**2000**	**2001**	**2002**	**2003**
Female	30 000	31 000	32 000	40 000
Male	31 000	32 000	33 000	34 000

Filters: AllCities, AllRaces, AllOccs

(a) Result of the query "*Evolution of the average income for each sex*"

AvgIncome	Year	
	2002	**2003**
	30 000	45 000

Filters: AllCities, AllRaces, Occ="Computer Engineer", SEX= "Female"

(b) A possible result after applying the $Diff$ operator

Figure 1.4: Application of the $Diff$ Operator for Example 1.3.4

result is given in Figure 1.4b where the average income for occupation Computer Engineer increases significantly.

In [Sapia, 2000], the aim of the approach is to equip the OLAP cache manager with a probabilistic model of former queries. Thus, from a current query, the most likely next queries can be prefetched in order to improve the response time. Two different Markov models are used to define the probabilities between queries. In the first model, the different states represent queries as patterns, i.e. a set of fragments that are measures, group-by set and selection predicate levels. This model allows to compute the probabilities to obtain a new pattern from a previous one. The second Markov model specifically focuses selection predicate values related to a particular hierarchy. This model allows to know the probabilities to change a value by another one. Example 1.3.5 illustrates the intuition.

Example 1.3.5. *Consider the* CENSUS *schema presented in Example 1.3.4, with the two following queries:*

– q_1*: "What is the average income for each sex in* 2002?*"*

– q_2*: "What is the average income for each sex in* 2003?*"*

A pattern, named p_1*, for both queries includes the following fragments:*

– *the measure:* AvgIncome

– *the group-by set levels:* AllCities, Allraces, Year, AllOccs, Sex

– *selection predicate levels:* Year

Consider a log composed of 10 queries and whose three patterns p_1*,* p_2 *and* p_3 *are extracted.*

Pattern p_2 *includes the following fragments:*

– *the measure:* MaxCostGas

– *the group-by set levels:* Region, Allraces, Year, AllOccs, AllSex

– *selection predicate levels:* Region

Pattern p_3 *includes the following fragments:*

– *the measure:* AvgCostWtr

– *the group-by set levels:* State, Allraces, AllYear, Occs, AllSex

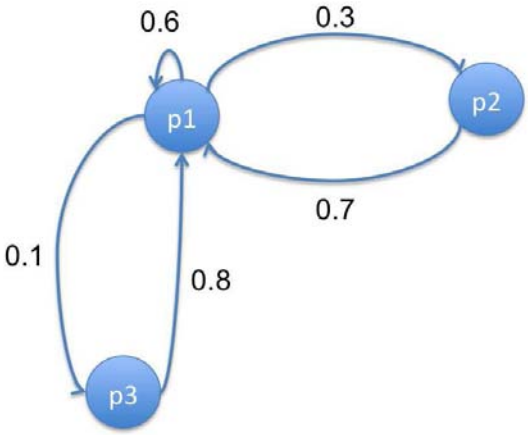
– *selection predicate levels:* State

A possible Markov model between patterns is depicted in Figure 1.5a. For instance, if we want to predict the next pattern of q_2*, the most likely is* p_1*.*

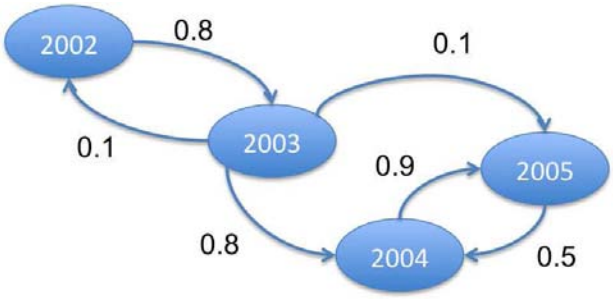
Consider the values of level Year*, the Markov model for selection values is depicted in Figure 1.5b. For instance, the most likely value after* 2003 *is* 2004*.*

Thus, if we want to prefetch the result of the following query q_2*, it is expected the query "What is the average income for each sex in* 2004?*" is asked.*

In [Aufaure et al., 2013], queries are recommended using a probabilistic model of former sessions, inspired by the work [Sapia, 2000]. In particular, former queries are grouped using a density-based clustering. This clustering uses a similarity measure especially based on the query structure that is a set of fragments (measure set, group-by set and selection set). A Markov model organizes the query clusters into series of states. A transition matrix contains scores of transition between each query of a state with each query of the other state. Considering a current query, this one is matched with the closest state of the Markov model. Precisely, the average similarity is computed between the current query



(a) Markov model for the log



(b) Markov model between selection values for the level Year

Figure 1.5: Markov Models of Example 1.3.5

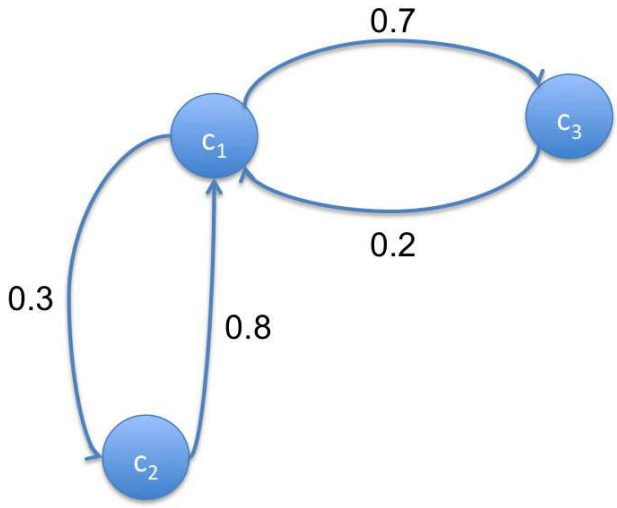


Figure 1.6: Markov Models of Example 1.3.6

and each query of the state. Then, the most probable state, using the transition matrix, is considered. Finally, the most similar query to the current query, among those present in the predicted state, is recommended.

Example 1.3.6 illustrates the intuition.

Example 1.3.6. *Consider a log composed of 5 queries. A query is formed with a set of measures, a group-by set and a set of selection predicates. The former queries are:*
$q_1 =< \{\mathsf{AvgIncome}\}, \langle \mathsf{AllCities}, \mathsf{AllRaces}, \mathsf{Year}, \mathsf{AllOccs}, \mathsf{Sex}\rangle, \{(\mathsf{Year} = 2002)\} >$
$q_2 =< \{\mathsf{AvgIncome}\}, \langle \mathsf{AllCities}, \mathsf{AllRaces}, \mathsf{Year}, \mathsf{AllOccs}, \mathsf{Sex}\rangle, \{(\mathsf{Year} = 2003)\} >$
$q_3 =< \{\mathsf{AvgCostWtr}\}, \langle \mathsf{Region}, \mathsf{AllRaces}, \mathsf{AllYear}, \mathsf{AllOccs}, \mathsf{Sex}\rangle, \{(\mathsf{Sex} = Female)\} >$
$q_4 =< \{\mathsf{AvgCostWtr}\}, \langle \mathsf{Region}, \mathsf{AllRaces}, \mathsf{AllYear}, \mathsf{AllOccs}, \mathsf{Sex}\rangle, \{(\mathsf{Sex} = Male)\} >$
$q_5 =< \{\mathsf{AvgIncome}\}, \langle \mathsf{Region}, \mathsf{AllRaces}, \mathsf{Year}, \mathsf{Occs}, \mathsf{Sex}\rangle, \{(\mathsf{Region} = North)\} >$

Possible clusters of similar queries are $c_1 = \{q_1, q_2\}$, $c_2 = \{q_3, q_4\}$ *and* $c_3 = \{q_5\}$. *A possible Markov model between these clusters is depicted in Figure 1.6.*

Consider the following current query
$q_c =< \{\mathsf{AvgIncome}\}, \langle \mathsf{AllCities}, \mathsf{AllRaces}, \mathsf{Year}, \mathsf{AllOccs}, \mathsf{Sex}\rangle, \{(\mathsf{Year} = 2004)\} >$

The closest state to q_c *is* c_1. *Looking at the Markov model, the most probable next state is* c_3. *Finally,* q_5 *is recommended.*

[Jerbi et al., 2009] discusses three types of possible recommendations that are
- query fragments for assisting a user to devise his query,
- analysis context to anticipate the user analysis,
- alternative analysis context that could be interesting to investigate further the current analysis (for example to enhance a current result).

An analysis context is a graph that represents fragments and values for an intermediate

result of an analysis sequence. Focusing on the anticipation of user analysis, the recommendation system is based on a current analysis and a user profile. Firstly, the user profile is matched with the current analysis context. Then, the preferences that depends of the current context are integrated into it by modifying the related nodes of the graph. Finally, the modified nodes are ranked and then recommended.

[Negre, 2009], [Giacometti et al., 2009] and [Giacometti et al., 2011] considered leveraging the specificities of data warehouses and OLAP queries to recommend queries in a collaborative or hybrid fashion. In particular, [Negre, 2009] and [Giacometti et al., 2009] proposes a generic framework to recommend a set of queries, organized in three customizable steps. These steps are respectively based on distance measures between references (a crossing of levels of a group-by set), queries and sessions. Using these distances, the log sessions are compared with the current session. Then, the set of final queries belonging to the closest log sessions are recommended and ranked according to their distance with the final query of the current session. In [Giacometti et al., 2011], a graph of queries is recommended, based on the application of the $Diff$ ([Sarawagi, 1999]) and $Relax$ ([Sathe and Sarawagi, 2001] operators on the query log, to detect differences in the past query results that could explain a difference in the current query answer. However, the techniques proposed relied on a query model that is either a partially evaluated query (considering query references) or a query answer, resulting in a poor scalability. In addition, recommendation was either too much prescriptive (only one query, [Giacometti et al., 2009]) or too little prescriptive (a graph of queries, [Giacometti et al., 2011]).

The existing approaches for query recommendation in databases are summarized in table 1.3, where we indicate:

- the category of the recommender system, content-based (CB), collaborative (CF) or hybrid (H),
- the source of information, i.e., whether the approach is log-driven, result-driven or both, and whether the approach is session-based or not, and if it is session based, whether sequential aspects are considered,
- the query model used, i.e., whether the approach leverages query expression or query answer,
- the technique used, i.e., whether the approach is similarity-based, stochastic or preference-based,
- the form and source of the recommendation.

In the context of query recommendation in Data Warehouses, the works cited above rarely address sequential aspects which are never as a basis to recommend query expressions. In particular, no approach ever considered recommending a sequence of queries. Recommended objects are rarely synthesized queries. Indeed recommendation is often a query chosen among past queries stored in some query log, or tuples retrieved from the database instance.

1.4 Similarity Measures for Sessions

This Section reviews the literature for similarity functions that could possibly be used to compare OLAP sessions. Since OLAP sessions are sequences of queries, we first review the

Table 1.3: Query recommendation approaches in databases

Reference	Cat.	Input			Tech.	Output	
		Source	Session?	Model		Form	Source
QueRIE [Eirinaki et al., 2013]	H	log	yes not seq.	expr. answer	sim.	queries	log
SnipSuggest [Khoussainova et al., 2010]	CF	log	yes seq.	expr.	stoch.	fragment	log
YMALDB [Drosou and Pitoura, 2013]	CB	answer	no	answer	stoch.	queries tuples	synth. DB
Icube [Sarawagi and Sathe, 2000]	CB	answer	yes not seq.	answer	stoch.	tuples	DB
Promise [Sapia, 2000]	CF	log	yes seq.	expr.	stoch.	query	synth.
[Aufaure et al., 2013]	CF	log	yes seq.	expr.	stoch.	query	log
[Jerbi et al., 2009]	CB	answer	no seq.	answer	pref.	fragment query	synth.
[Giacometti et al., 2009] [Negre, 2009]	H	log	yes seq.	answer	sim.	query	log
[Giacometti et al., 2011]	H	log answer	yes not seq.	answer	stoch.	queries	log

approaches for comparing sequences (Section 1.4.1) and then those for comparing database queries (Section 1.4.2).

1.4.1 Sequence Comparison Approaches

Comparing sequences has attracted a lot of attention especially in the context of string processing, with applications like information retrieval, spell-checkers, bioinformatics, and record linkage [Cohen et al., 2003, Moreau et al., 2008]. The existing approaches are inspired by different principles.

In *token-based approaches* sequences are treated as bags of elements, and classical set similarity functions like Jaccard and Hausdorff, and all their variants, can be used or adapted. Of course, these approaches are not sensible to the order of sequence elements. When the sequences to be compared are taken from a *corpus*, the popular *term frequency-inverse document frequency* (tf-idf) weight can be adopted, which weighs each element of a sequence using (positively) their frequency in the sequence and (negatively) their frequency in the corpus. A cosine is then used to measure the similarity between two vectors of weights.

Some approaches compare two sequences by comparing their subsequences. A basic approach here is to use the size of the longest common subsequence (LCS). [2] An approach often used in statistical natural language processing relies on n-grams, i.e., substrings of size n of a given sequence [Brown et al., 1992]. A popular similarity function using n-grams is the *Dice coefficient*, an extension of the Jaccard index defined as twice the number of shared n-grams over the total number of n-grams:

$$Sim_{Dice}(s, s') = \frac{2|ngrams(s) \cap ngrams(s')|}{|ngrams(s)| + |ngrams(s')|}$$

2. Note that, while substrings are consecutive parts of a string, subsequences need not be.

Other approaches compare sequences based on their *edit distance*, i.e., in terms of the cost of the atomic operations necessary to transform one sequence into another. Many edit distances have been proposed that differ on the number, type, and cost of the edit operations. The most popular are the *Levenshtein distance*, that allows insert, delete, and substitute, and the *sequence alignment distance*, that allows match, replace, delete, and insert [Cohen et al., 2003, Navarro, 2001].

Finally, in *two-level* approaches sequences are compared based on the similarity between their elements. A simple example is the Hausdorff distance between sets, that relies on the distance between elements of the set. In [Monge and Elkan, 1997] the similarity between sequences s and s' is the average of the highest similarities between pairs of elements of s and s':

$$Sim_{M\&E}(s, s') = \frac{1}{|s|} \sum_{s_i \in s} max_{s'_j \in s'} \{Sim_{elem}(s_i, s'_j)\}$$

where Sim_{elem} measures the similarity between single elements. In *soft tf-idf* [Cohen et al., 2003], the tf-idf weight is extended using the similarity of sequence elements; more precisely,

$$Sim_{soft}(s, s') = \sum_{s_i \in Close_\theta(s,s')} T(s_i, s) \cdot T(s_i, s') \cdot max_{s'_j \in s'} \{Sim_{elem}(s_i, s'_j)\}$$

where $T(s_i, s)$ is a normalized form of the tf-idf of element s_i within sequence s, θ is a threshold, and $Close_\theta(s, s')$ is the set of elements $s_i \in s$ such that there is at least an element $s'_j \in s'$ with $Sim_{elem}(s_i, s'_j) > \theta$. While the two previous two-level approaches do not consider the ordering of elements within sequences, the *Smith-Waterman algorithm* relies on element ordering; it can be used to efficiently find the best alignment between subsequences of two given sequences by ignoring the non-matching parts of the sequences [Smith and Waterman, 1981]. It is a dynamic programming algorithm based on a matrix H whose value in position (i, j) expresses the score for aligning subsequences of s and s' that end in elements s_i and s'_j, respectively. This matrix is recursively defined based on the following formula:

$$H(i,j) = max \left\{ \begin{array}{c} 0; \\ H(i-1, j-1) + Sim_{elem}(s_i, s'_j); \\ max_{k \geq 1}\{H(i-k, j) - cost_k\}; \\ max_{k \geq 1}\{H(i, j-k) - cost_k\} \end{array} \right\}$$

where $cost_k$ is the cost of introducing a gap of length k in the matching between s and s'. Note that, here, the similarity between two elements can be negative, to express that there is a mismatch between them; intuitively, the algorithm seeks an optimal trade-off between the cost for introducing a gap in the matching subsequences and the cost for including a poorly matching pair of elements.

1.4.2 Query Comparison Approaches

We can distinguish two main motivations for comparing database queries. The first one is query optimization, where a query q to be evaluated is compared to another query

q', with the goal of finding a better way of evaluating q. This motivation attracted a lot of attention, and covers classical problems like view usability [Garcia-Molina et al., 2008, Gupta and Mumick, 1999], query containment [Abiteboul et al., 1995], plan selection [Ghosh et al., 2002], view selection [Aouiche et al., 2006, Golfarelli, 2003], and data prefetching [Sapia, 2000]. The second, more recent, motivation is to suggest a query to the user without focusing on its evaluation. In this context, a query is compared to another one with the goal of helping the user exploring or analyzing a database. This includes query completion [Yang et al., 2009] and query recommendation [Stefanidis et al., 2009, Drosou and Pitoura, 2011, Chatzopoulou et al., 2009, Chatzopoulou et al., 2011, Akbarnejad et al., 2010, Giacometti et al., 2009].

From a technical point of view, the approaches found in the literature can be classified according to (i) the *query model* they adopt, i.e., the structure used to compactly represent queries; (ii) the *information source* from which the representation of each query is derived; and (iii) the *function* used to compute similarity.

Query models range from a string corresponding to the uninterpreted SQL sentence [Yao et al., 2005] to the set of tuples resulting from the query evaluation [Stefanidis et al., 2009, Drosou and Pitoura, 2011]. Queries can also be modeled as vectors of features with either a score or a Boolean for each feature [Akbarnejad et al., 2010, Agrawal et al., 2006, Aouiche et al., 2006, Ghosh et al., 2002], or as sets of *fragments*, each representing a particular part of the query, such as the attributes required in output (SELECT clause) or the table names in the cross product (FROM clause) [Sapia, 2000, Aligon et al., 2011]. Finally, queries are sometimes modeled as graphs, following the database schema like in [Yang et al., 2009].

As to the information source, it can be the query expression, e.g., the uninterpreted query text [Yao et al., 2005] or the list of query fragments (selection predicates, projection, etc.) [Garcia-Molina et al., 2008, Yang et al., 2009]. When fragments are used, only some of them may be taken into account; for instance, only the selection attributes are used by [Agrawal et al., 2006] and [Yang et al., 2009] whereas all fragments are used by [Garcia-Molina et al., 2008] and [Gupta and Mumick, 1999]. The information source can also be related to the database queried; more precisely, it can be:

- The database instance, e.g., the query result or the active domain of the database attributes [Agrawal et al., 2006, Chatzopoulou et al., 2009, Chatzopoulou et al., 2011, Giacometti et al., 2009, Stefanidis et al., 2009, Drosou and Pitoura, 2011]. In the former case, the query can be evaluated either fully [Stefanidis et al., 2009, Drosou and Pitoura, 2011] or partially [Giacometti et al., 2009]. In this category we also include an approach for measuring similarity between multidimensional cubes [Baikousi et al., 2011], because obviously an OLAP query returns a multidimensional cube.
- The statistics used by the query optimizer, like table sizes and attribute cardinalities [Ghosh et al., 2002].
- The database schema, e.g., the keys defined or the index used to process a selection [Ghosh et al., 2002, Golfarelli, 2003].
- The query log, if the query model relies on other queries that have previously been launched on the same database. For instance, [Chatzopoulou et al., 2009], [Chatzopoulou et al., 2011], [Akbarnejad et al., 2010], [Aouiche et al., 2006], and

Table 1.4: Query comparison approaches at a glance

Ref.	*Motivation*	*Model*	*Source*	*Similarity Function*
[Gupta and Mumick, 1999]	optimization	sets	S, P, C	fragment tests
[Chatzopoulou et al., 2011]	recommend.	vector	db instance, log	cosine
[Akbarnejad et al., 2010]	recommend.	vector	S, P, log	cosine
[Agrawal et al., 2006]	optimization	vector	S, db instance	cosine
[Aouiche et al., 2006]	optimization	vector	S, P, log	Hamming distance
[Ghosh et al., 2002]	optimization	vector	S, C, db statistics	Hamming distance
[Stefanidis et al., 2009] (1)	recommend.	vector	log	inner product
[Stefanidis et al., 2009] (2)	recommend.	set	db instance	Jaccard index
[Giacometti et al., 2009]	recommend.	set	db instance	Hausdorff distance
[Sapia, 2000]	optimization	sets	S, P	query repres. equality
[Golfarelli, 2003]	optimization	set	P, db schema & statistics	group-by lattice
[Yao et al., 2005]	recommend.	string	SQL sentence	entropy
[Yang et al., 2009]	recommend.	graph	S, P, C	query repres. equality

[Stefanidis et al., 2009] model a query in terms of its links with other queries or how many times it appears in the log.

Finally, the result of query comparison can be a Boolean or a score, usually normalized in the [0..1] interval. The first case applies when queries are tested for equivalence [Abiteboul et al., 1995] or view adaptation [Gupta and Mumick, 1999], or when the goal is to group queries based on some criteria [Sapia, 2000, Yang et al., 2009]. In this case, the comparison can be a simple equality test of the query representations [Sapia, 2000, Yang et al., 2009] or it can be based on separate tests of query fragments [Gupta and Mumick, 1999]. In the second case, the comparison is normally based on classical functions applied to the query representations. For instance, if the query is modeled as a vector, cosine [Agrawal et al., 2006, Akbarnejad et al., 2010, Chatzopoulou et al., 2009, Chatzopoulou et al., 2011], inner product [Stefanidis et al., 2009], or Hamming distance [Aouiche et al., 2006] can be used; if the query is modeled as a set, the Jaccard index [Stefanidis et al., 2009] or the Hausdorff distance [Giacometti et al., 2009] can be used. Sometimes, more sophisticated similarity functions are used. For instance, [Yao et al., 2005] use a measure based on entropy to cluster queries modelled as strings. In [Golfarelli, 2003], similarity between OLAP queries is computed based on the relative position of the query group-by sets within the group-by lattice.

Table 1.4 summarizes the approaches reviewed in this Section. Note that [Stefanidis et al., 2009] propose two ways of comparing queries: (1) based on the frequency of the query in the log, and (2) based on the query result. Letters S, P, and C indicate the fragments used by the approach (S for selection, P for generalized projection —including the group-by set and the aggregation operator—, and C for cross-product).

Among the query similarity functions proposed in the OLAP area, the one that captures the above requirements at best is [Aouiche et al., 2006]. In that approach, similarity between queries q and q' is based on the number of attributes they share within their SELECT, WHERE, and GROUP-BY clauses; the normalized form we adopt here for comparison purposes (Section 4.2) is

$$\sigma_{AJD}(q, q') = \frac{|L \cap L'|}{|L \cup L'|}$$

where L and L' are the attributes appearing in q and q', respectively.

1.5 Discussion: Requirements for Similarity-based Recommendation of OLAP Sessions

This section discusses the requirements for the recommendation approach developed in this dissertation. These requirements are defined Section 1.5.1 and especially indicate that the sequential aspect of the sessions is essential to consider in a recommender system. That is why specific requirements are defined for session similarity in Section 1.5.2.

1.5.1 Requirements for OLAP Session Recommendation

The previous approaches of recommendation in a data warehouse context, reviewed in Section 1.3.2, often consider collaborative filtering or hybrid solutions but rarely consider sequential aspects of the former or current sessions. Moreover, no approach attempts to recommend a sequence of queries. Recommendations are rarely synthesized queries that are not already present in a log. Indeed, the recommendations are either deduced from the best former queries matching a current user context or from the best predictions of former queries in stochastic techniques.

This section lists a number of requirements to be used for recommendation in an OLAP context to overcome the drawbacks of the solutions:

♯1 Sessions are recommended rather than single queries. We consider that an OLAP session issued by a skilled user is not just a path aimed at leading her to a single, valuable query (for instance, the one at the end of the session). Indeed, the whole sequence of queries belonging to a session is valuable in itself, because it gives the user a different and complemental view of information.

♯2 Consistently with collaborative filtering and hybrid approaches, the knowledge acquired by other users during previous sessions is reused.

♯3 The recommendation is based on the query expressions rather than tuples. Indeed, as for collaborative filtering approaches, former sessions are compared to a current user context in an online step. This comparison is often costly if tuples are considered and could cause problems of response time.

♯4 Current sessions are matched with former sessions using a similarity measure. This similarity measure has to consider the query order and give a preference to the queries shared by the two sessions, while being able to include queries that are potential recommendations.

♯5 The recommended sessions have to share similar context with various sessions present in a log.

♯6 The recommended sessions have to be close to the analysis context of the current session.

We now give observations on the features for recommendations:

– According to requirement ♯1, sessions are preferred for recommendation. Consequently, this choice excludes to consider Markov models to recommend sessions (like in [Sapia, 2000] or [Aufaure et al., 2013]). Indeed, Markov models are generally of order 1 (models with higher orders are very costly), i.e. the probability to reach a state in a model only depends of the previous state. Thus, it is difficult to predict the n+1 states following a current state. Moreover, the states can mix several sessions.
– According to requirements ♯5 and ♯6, quality criteria assessing the suitability of recommendations have to be defined. In particular, requirement ♯5 supposes to estimate the *relevance* of the recommendations among the former sessions that could be interesting for users. Requirement ♯6 supposes to propose four more quality criteria. A first should measures the $foresight$ of the recommendation, by identifying the distance separating the current session and the recommendation. The second should measure the *novelty* of the recommendation to the former sessions. Indeed, a user is often interested by recommendations providing new information. This requirement also insists on the distance to the analysis conducted by a current user. This can be measured with two more criteria. The first measures the *adaptation* of the recommendation to the current session, i.e. how is similar the analysis sequence of the recommendation with the current session, whereas the second measures the *obviousness* of the recommendation, i.e. if the recommended sequence shares queries closed to the current session.

1.5.2 Requirements for OLAP Session Similarity

This Section lists a number of requirements to be used for (i) understanding which approaches, among all those proposed in the literature for query and sequence comparison (see Section1.4), are eligible for the OLAP context; and (ii) driving the adaptation and extension of the eligible approaches towards the development of an original approach to OLAP session comparison.

We start by proposing a set of requirements, suggested by the specific features of the OLAP context:

♯7 Multidimensional databases store huge amounts of data, and OLAP queries may easily return large volumes of results. Computing similarity at the extensional level, i.e., by comparing the data resulting from queries, would pose serious efficiency problems in this context, and would discourage the use of the approach for recommendation and personalization — that require a fast interaction with users. Indeed, as noted by [Chatzopoulou et al., 2011] in the case of recommendation of SQL queries, there is a clear trade-off between efficiency and effectiveness, when a fragment based model or a tuple based model is used. For this reason we compute similarity at the intensional level, i.e., considering only query expressions.

♯8 It is unlikely that two OLAP sessions share identical queries; this feature is better managed by having comparisons of single queries result in a score rather than in a Boolean.

♯9 A typical OLAP query is defined by the fact to be analyzed, one or more measures to be computed, a set of hierarchy levels for aggregating measure values, a predicate for

filtering a subset of events, and a presentation. Though the presentation chosen for displaying the results of an OLAP query (e.g., a cross-tab or a pie-chart) certainly has an influence on how easily users can interpret these results, it does not affect the actual informative content, so it should not be considered when comparing queries.

To discover additional requirements for OLAP sessions similarity, we conducted a user study. We prepared a questionnaire asking to give a qualitative evaluation of the similarity between couples of OLAP queries and couples of OLAP sessions over a simple multidimensional schema (more details will be given in Section 4.2). The questionnaire [3] was submitted to all the teachers and PhD students of the First European Business Intelligence Summer School (eBISS 2011) [4], as well as to the master students of two specialistic courses on data warehouse design at the Universities of Bologna (Italy) and Tours (France). All people involved had some experience as OLAP users, most of them had some practice of multidimensional design too. Overall, 41 answers were collected. The additional requirements emerging from an analysis of the questionnaire results can be summarized as follows:

♯10 The selection predicate is the most relevant component in determining the similarity between two OLAP queries, followed by the group-by set. The less important component is the set of measures to be returned.

♯11 The order of queries is relevant in determining the similarity between two sessions, i.e., two sessions sharing the same queries but in different orders have low similarity.

♯12 Recent queries are more relevant than old queries in determining the similarity between two OLAP sessions. Since the time actually elapsed between two consequent queries in a session depends on several unpredictable factors (e.g., the query execution time, the size and complexity of the data returned, the user's query formulation skills), only the *order* of queries will be considered.

♯13 The longest the matching fraction of two sessions, the highest their similarity.

♯14 Two sessions that match with one or more gaps (i.e., one or more non-matching queries are present) are similar, but their similarity is lower than the one of two sessions that match with no gaps.

In particular, as to point ♯10, in Figure 1.7 we show the percentages of users that perceive a given level of similarity for couples of queries that only differ in either their measure sets, or their selection predicates, or their group-by sets. Apparently, measures are the less important component in determining similarity since most users perceive as highly similar two queries that only differ in their measures. The opposite holds for the selection predicate component.

We now give observations on the features for query comparison approach that should have to be used for OLAP queries:

– Following requirement ♯7, we solely rely on query expressions to derive query representations. Then we exclude the approaches based on query evaluation [Giacometti et al., 2009, Stefanidis et al., 2009, Drosou and Pitoura, 2011], those depending on database instances [Chatzopoulou et al., 2009,

3. Available at `http://www.julien.aligon.fr/recherche/similarityform.aspx`.
4. `http://cs.ulb.ac.be/conferences/ebiss2011/`

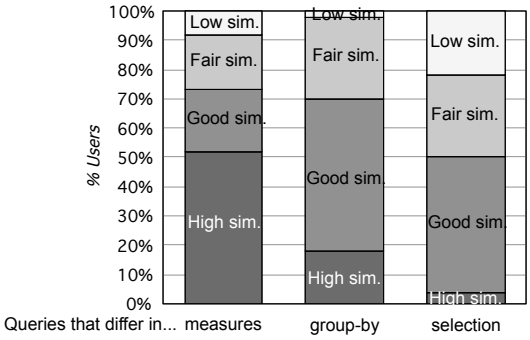


Figure 1.7: Perceived similarities for OLAP queries only differing in one of their three main components

Chatzopoulou et al., 2011, Agrawal et al., 2006, Baikousi et al., 2011], and those using query logs [Aouiche et al., 2006, Akbarnejad et al., 2010, Stefanidis et al., 2009].

- Our goal is not query optimization, so we drop the approaches aimed at optimization like [Ghosh et al., 2002]. In that particular work, the idea is to reuse execution plans, that heavily rely on "physical" properties (like statistics and presence of indexes); thus, query similarity is more related to how queries are evaluated than to what they mean to users. This means that two queries that should be very similar for our purposes could be found to be very dissimilar using that approach if their execution plans are different (for instance, if one has a WHERE clause and the other does not).
- According to requirement ♯8, query comparison should result in a score. So, Boolean approaches like [Gupta and Mumick, 1999] and [Yang et al., 2009] are less relevant in our context.
- OLAP queries are expressed using a friendly visual interface, and the syntax of the underlying query language (e.g., MDX) is typically transparent to users. This discourages the adoption of uninterpreted approaches like [Yao et al., 2005].
- According to requirement ♯9, the OLAP semantics is carried by a number of different components (e.g., the aggregation level), which encourages the adoption of a fragment-based query model like in [Sapia, 2000], also taking into account the peculiarities of the multidimensional model like in [Golfarelli, 2003].

The features for sequence comparison approach that should have to be used for OLAP sessions are:

- In OLAP sessions, the order of queries is relevant (requirement ♯11), which discourages from taking token-based approaches.
- Mostly, OLAP sessions do not share the very same queries (requirement ♯8). This makes two-level approaches, that take advantage of a similarity function for OLAP queries, more suitable for our purposes.
- Following requirement ♯14, it is important to be able to determine similar regions in two globally different sessions, which favors a sequence alignment approach.

1.6 Conclusion

This state of the art discussed recommendation based on usage mining. In the OLAP context, this research area is in expansion and most works use logs and sessions to recommend queries or tuples using stochastic or similarity-based processes. But several drawbacks have been identified such as the sequential aspect is rarely addressed and is never a basis to recommend query expressions. Moreover, no approach ever considered recommending synthesized sequence of queries.

To answer these lacks, we propose a set of requirements to take into account in a recommendation context. In particular, sessions have to be recommended. Indeed, the whole sequence of queries belonging to a session is valuable in itself, because it gives the user a different and complemental view of information. A similarity-based recommendation is preferred to a Markov model since sessions are recommended. Also, query expressions are preferred to tuples, in order to compute the recommendations in an online step (discussed in Chapter 3). A set of criteria is also proposed to assess the effectiveness of recommendations such as a relevance or novelty measure against former sessions or foresight, obviousness and adaptation measures against current sessions (defined in Chapter 4).

Since the recommendation system proposed in this dissertation is based on similarity measures between queries and sessions, a state of the art of classical measures in Information Retrieval is presented. To take into account the OLAP context in which these measures will be used, a set of requirements is also defined, based on questionnaires answered by PhD students and Master's students. In particular, query similarity has to give a more important weight for selection predicates rather than group-by set or measures. Session similarity has to consider an order between the queries composing the sessions to compare. Recent queries are especially considered as more relevant than old queries. This similarity measure has also to be organized in a two-level approach, including the query similarity. (defined in Chapter 2)

Chapter 2

Defining Similarities for OLAP sessions

This chapter introduces the definitions required to propose similarity measures for OLAP sessions. First of all, Section 2.1 formally defines our data model as well as the models of query, session and log required in this dissertation. The next sections are devoted to a three-level approach to compare OLAP logs. Indeed, the approach considers an interlinking between different types of similarity measures. More precisely, the comparison of logs depends of the comparison of sessions that depends of the comparison between queries. The first one, defined in Section 2.2, compares queries and is specifically devised to take the query model into account. The second one, defined in Section 2.3, compares sessions and proposes an adaptation, in the OLAP context, of different classical measures that can be found in information retrieval context. The last one, defined in Section 2.4, compares logs from classical measures comparing sets.

2.1 Modeling Multidimensional Log

This section is devoted to the model of multidimensional log used in this dissertation. More precisely, the definition of our multidimensional data model is given in Section 2.1.1. Sections 2.1.2.1 and 2.1.3 describe the models of query, session and log.

2.1.1 Modeling Multidimensional Data

In this Section, we give the formal definition of the multidimensional model. In what follows, basic knowledge in databases (see [Abiteboul et al., 1995]) and data warehouses (see [Golfarelli and Rizzi, 2009]) are required.

2.1.1.1 Modeling Members, Levels and Hierarchies

Definition 1 (Levels and Members)**.** *Let L be a set of attributes called levels, and for $l \in L$, a member is an element of $Dom(l)$.*

Roll-up and Drill-down are two partial mappings from L to L defined by: Given two levels l_j and l_k, $Rollup(l_j) = l_k$ if there exists a functional dependency $l_j \to l_k$ or undefined otherwise, and $Drilldown(l_j) = l_l$ if there exists a functional dependency $l_l \to l_j$ or undefined otherwise.

Definition 2 (Hierarchy)**.** *A hierarchy h_i is a set $Lev(h_i) = \{l_0, ..., l_d\}$ of levels together with a roll-up total order $\succeq_{h_i}$ of $Lev(h_i)$, which is such that, for any l_j and l_k in $Lev(h_i)$, $l_j \succeq_{h_i} l_k$ if $Rollup(l_k) = l_j$.*

For each hierarchy h_i, the bottom level l_0 of the hierarchy, denoted by ALL_i, has a single possible value and determines the coarsest aggregation level. Conversely, the top level l_d, denoted by DIM_i, determines the finest aggregation level for the hierarchy. For simplicity, we define hierarchies as total orders instead of partial orders, i.e., we will assume hierarchies have no branches.

2.1.1.2 Multidimensional schema and group-by sets

Definition 3 (Multidimensional Schema)**.** *A* multidimensional schema *(or, briefly, a* schema*) is a triple $\mathcal{M} = \langle L, H, M \rangle$ where:*

- *$L = \{l_1, \ldots l_p\}$ is a finite set of* levels*, i.e., categorical attributes;*
- *$H = \{h_1, \ldots, h_n\}$ is a finite set of* hierarchies*, each characterized by (1) a subset $Lev(h_i) \subseteq L$ of levels and (2) a* roll-up *total order $\succeq_{h_i}$ of $Lev(h_i)$;*
- *$M = \{m_1, \ldots, m_l\}$ is a finite set of* measures*, i.e., numerical attributes.*

A group-by set includes one level for each hierarchy, and defines a possible way to aggregate data.

Definition 4 (Group-by Set)**.** *Given schema $\mathcal{M} = \langle L, H, M \rangle$, let $Dom(H) = Lev(h_1) \times \ldots \times Lev(h_n)$; each $g \in Dom(H)$ is called a* group-by set *of $\mathcal{M}$.*

In particular, for a group-by set $g = \langle l_1, \ldots, l_n \rangle$ we note $l \in g$ if $l = l_i$ with $i \in [1..n]$.

Let $\succeq_H$ denote the product order[1] *of the roll-up orders of the hierarchies in H. Then, $(Dom(H), \succeq_H)$ is a lattice, that we will call* group-by lattice*, whose bottom and top elements are $G^{\perp} = \langle DIM_1, \ldots, DIM_n \rangle$ and $G^{\top} = \langle ALL_1, \ldots, ALL_n \rangle$, respectively.*

Example 2.1.1. *We consider the* CENSUS *schema given in Example 1.3.4. As a reminder, the* CENSUS *multidimensional schema has five hierarchies, namely* RACE, TIME, SEX, OCCUPATION, *and* RESIDENCE*, and measures (aggregated either by Sum, Max, Min or Avg)* Income, PropInsr *(property insurance cost),* PerWt *(person weight),* CostGas, CostWtr, *and* CostElect*. It is* City $\succeq_{\text{RESIDENCE}}$ State *(the complete roll-up orders are shown in Figure 1.3);*

1. The product order of n total orders is a partial order on the Cartesian product of the n totally ordered sets, such that $\langle x_1, \ldots, x_n \rangle \succeq \langle y_1, \ldots, y_n \rangle$ iff $x_i \succeq y_i$ for $i = 1, \ldots, n$.

Possible group-by sets are:

$g_1 = \langle \mathsf{State}, \mathsf{Race}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{Occ} \rangle$
$g_2 = \langle \mathsf{State}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{Occ} \rangle$
$g_3 = \langle \mathsf{Region}, \mathsf{AllRaces}, \mathsf{Year}, \mathsf{Sex}, \mathsf{Occ} \rangle$
$g_4 = \langle \mathsf{AllCities}, \mathsf{AllRaces}, \mathsf{AllYears}, \mathsf{AllSexes}, \mathsf{AllOccs} \rangle$

The last group-by set specifies total aggregation.

2.1.2 Query Model and Query Manipulation Operators

We develop in this section our query model and our query manipulation operators. We present in Section 2.1.2.1, the query model based on the intensional level thats represents the query expression written in a particular query language (like the MDX language, a de facto standard in OLAP, see [Microsoft, 2009]). We motivate this decision by the fact that this dissertation is dedicated to a recommendation solution using similarity measures (developed Chapter 3). As explained in the requirements of Section 1.5.2, using the extensional level, i.e., by comparing the result of queries, would pose serious efficiency problems in this context.

The query operators are described in Section 2.1.2.2. These operators allow to derive a query expression from another one by affecting a particular component of our query model. In fact, the operators are similar to those traditionally used in OLAP (such as roll-up, drill-down, or slice-and-dice) on cubes. Indeed, instead of manipulating a cube, we only modify a query expression using a group-by set, a set of predicates or a set of measures, as defined in our query model. In particular, these operators form the core of our different synthetic log generators (as described in Section 4.1.1).

2.1.2.1 Modeling Queries

We consider a basic form of OLAP query centered on a single schema and characterized by an aggregation and a set of selection predicates. To be independent of the details related to logical design of multidimensional schemata and to specific query plans, we express queries using an abstract syntax. In a relational implementation, a multidimensional schema is translated into a star schema; in this case, the queries we consider can be classified as *GPSJ - Generalized Projection / Selection / Join* queries [Gupta et al., 1995], based on a star join between the fact table and the dimension tables. We opt for a fragment-based query model with three components as defined below:

Definition 5 (OLAP Query)**.** *A* query *on schema* $\mathcal{M} = \langle L, H, M \rangle$ *is a triple* $q = \langle g, P, Meas \rangle$ *where:*

1. $g \in Dom(H)$ *is the query group-by set;*
2. $P = \{p_1 = l_1 \in X_1, \ldots, p_n = l_n \in X_n\}$ *is a set of predicates, one by hierarchy, whose conjunction is of the form* $l_1 \in X_1 \wedge \ldots \wedge l_n \in X_n$*, where* l_j *is a level and* X_j *is a set of members for that level. Conventionally,* $p_i = TRUE_i$ *if no selection on* h_i *is made in* q*;*

3. $Meas \subseteq M$ *is the measure set whose values are returned by* q*.*

Example 2.1.2. *Suppose a user wishes to obtain the average cost of water and electricity, in 2005, for each type of races in the different states and occupations. Using our query model, he can formulate his need as follows:*

$$q_1 = \langle\langle \mathsf{State}, \mathsf{Race}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{Occ}\rangle, \{TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2005\}, TRUE_{OCCUPATION}, TRUE_{SEX}\}, \{AvgCostWtr, AvgCostElect\}\rangle$$

2.1.2.2 Query Manipulation Operators

We present the different query operators, each defined on a specific component of the query (group-by set, selection set or measure set), according to the query model given in Definition 5.

We propose two operators allowing to modify the group-by set g, by changing a level, for a given hierarchy h_i, to a coarser ($Rollup_{int}$) or finer ($Drilldown_{int}$) granularity level. Let $q = \langle g, P, Meas\rangle$ be a query. Let h_i be the hierarchy where a Rollup or Drilldown operator will be applied on a level $l_i \in g$. The $Rollup_{int}$ operation is defined when $l_i \neq ALL_i$ and the $Drilldown_{int}$ operation id defined when $l_i \neq DIM_i$.

$Rollup_{int}(q, h_i) = \langle g', P, Meas\rangle$ *where* $l_i \in Lev(h_i)$ *and* $g' =< l_1, \ldots, Rollup(l_i), \ldots, l_n >$ $Drilldown_{int}(q, h_i) = \langle g', P, Meas\rangle$ *where* $l_i \in Lev(h_i)$ *and* $g' =< l_1, \ldots, Drilldown(l_i), \ldots, l_n >$

The next two operators modify the selection set P. The first operator adds a member c'_i of a level l'_i to a set of members X_i of the same level as l'_i of a predicate $p_i \in P$, for a given hierarchy h_i. Let $q = \langle g, P, Meas\rangle$ be a query.
Let $l_i \in \{c'_i\}$ be the selection predicate where c'_i is the member to add to $p_i \in P$ with $p_i = l_i \in X_i$. In particular, the operator is defined when $c'_i \notin X_i$.

$AddSelection(q, l_i \in \{c'_i\}) = \langle g, P', Meas\rangle$ *where* $p_i \in P'$ *with* $p_i = l_i \in X'_i$ *such as* $X'_i = X_i \cup \{c'_i\}$

The second operator removes a member c'_i of a level l'_i to a set of members X_i of the same level as l'_i of a predicate $p_i \in P$, for a given hierarchy h_i. Let $q = \langle g, P, Meas\rangle$ be a query. Let $l_i \in \{c'_i\}$ be the selection predicate where c'_i is the member to remove from $p_i \in P$ with $p_i = l_i \in X_i$. In particular, the operator is defined when $c'_i \in X_i$.

$RemoveSelection(q, l_i \in \{c'_i\}) = \langle g, P', Meas\rangle$ *where* $p_i \in P'$ *with* $p_i = l_i \in X'_i$ *such as*
$X'_i = X_i - \{c_i\}$

The next two operators modify the measure set $Meas$. The first operator, allowing to add a measure fragment $m \notin Meas$, is defined below: Let $q = \langle g, P, Meas\rangle$ be a query.

Let m a measure to add in $Meas$. In particular, this operator is defined when $m \notin Meas$.

$AddMeasure(q, m) = \langle g, P, Meas' \rangle$ *where* $Meas' = Meas \cup \{m\}$

The second operator, allowing to remove a measure fragment $m \in Meas$, is defined below:
Let $q = \langle g, P, Meas \rangle$ be a query.
Let m be a measure to remove from $Meas$. In particular, this operator is defined when $m \in Meas$.

$RemoveMeasure(q, m) = \langle g, P, Meas' \rangle$ *where* $Meas' = Meas - \{m\}$

2.1.3 Modeling Sessions and Logs

An OLAP session is an ordered sequence of queries formulated by a user on a schema; typically (but not necessarily), each query in a session is derived from the previous one by applying an OLAP operator.

Definition 6 (OLAP Session)**.** *An* OLAP session *of length v is a sequence $s = \langle q_1, \ldots, q_v \rangle$ of v queries on schema $\mathcal{M}$.*

Given a session s, we will denote with $length(s)$ the number of queries in s, with $s[w]$ $(1 \leqslant w \leqslant length(s))$ the w-th query of s, and with $s[v, w]$ $(1 \leqslant v \leqslant w \leqslant length(s))$ the subsession of s spanning from its v-th query to the w-th one. The last query of s, $s[length(s)]$, is briefly denoted with $s[.]$, so $s[v, .]$ is the subsession of s spanning from its v-th query to the end. Finally, we will write $s' \sqsubset s$ to denote that s' is a subsession of s.

Definition 7 (OLAP Log)**.** *An* OLAP log *is a set $\mathcal{L}$ of OLAP sessions.*

Example 2.1.3. *All the examples in this Section will be based on a simple log that consists of three sessions:*

$$
\begin{aligned}
s &= \langle q_1, q_2, q_3 \rangle \\
s' &= \langle q_4, q_5, q_6, q_7, q_8 \rangle \\
s'' &= \langle q_9, q_{10} \rangle
\end{aligned}
$$

Table 2.1 represents each query in terms of our query model; the involved group-by sets are those used in Example 1.3.4, while the selection predicates are:

$$
\begin{aligned}
P_1 = \{&TRUE_{\mathsf{RESIDENCE}}, TRUE_{\mathsf{RACE}}, (\mathsf{Year} \in \{2005\}), \\
&TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\} \\
P_2 = \{&TRUE_{\mathsf{RESIDENCE}}, (\mathsf{RaceGroup} \in \{Chinese\}), TRUE_{\mathsf{TIME}}, \\
&TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\} \\
P_3 = \{&TRUE_{\mathsf{RESIDENCE}}, (\mathsf{RaceGroup} \in \{Chinese\}), (\mathsf{Year} \in \{2005\}), \\
&TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}
\end{aligned}
$$

The query expression, in MDX formulation, of query q_4 is:

Table 2.1: Queries for Example 2.1.3

		Queries									
		q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
Group-by set		g_1	g_2	g_2	g_2	g_2	g_3	g_3	g_2	g_1	g_1
Measures	AvgCostWtr	✓	✓	✓	✓	✓	✓	✓	✓		
	AvgCostElect	✓	✓	✓		✓	✓		✓		
	AvgCostGas			✓						✓	✓
	AvgIncome							✓	✓		✓
Selection predicates		P_1	P_1	P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1

```
SELECT AvgCostWtr ON COLUMNS,
    Crossjoin(OCCUPATION.Occ.members,
        Crossjoin(TIME.Year.members,RESIDENCE.State.members)) ON ROWS
FROM CENSUS WHERE RACE.RaceGroup.[Chinese]
```

2.2 Query Similarity

In this Section we define the similarity function used in our three-level approach to compare OLAP queries. As remarked in Section 1.5.2, this function must consider the peculiarities of the multidimensional model, be computable based on query expressions only, and result in a score. Consistently with Definition 5, the function we propose is a combination of three components: one related to group-by sets, one to selection predicates, and one to measure sets.

To define group-by set similarity, we first introduce the notion of distance between levels in a hierarchy.

2.2.1 Similarity between group-by sets

Definition 8 (Distance between hierarchy levels)**.** *Let $\mathcal{M} = \langle L, H, M\rangle$ be a schema, $h_i \in H$ be a hierarchy, and $l, l' \in Lev(h_i)$ be two levels. The* distance *between l and l', $Dist_{lev}(l, l')$, is the difference between the positions of l and l' within the roll-up order $\succeq_{h_i}$.*

Definition 9 (Group-by set similarity)**.** *Let q and q' be two queries, both on schema $\mathcal{M}$, with group-by sets g and g', respectively, and let $g.h_i$ ($g'.h_i$) denote the level of h_i included in g (g'). The* group-by set similarity *between q and q' is*

$$\sigma_{gbs}(q, q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{lev}(g.h_i, g'.h_i)}{|Lev(h_i)|-1}}{n}$$

where n is the number of hierarchies in $\mathcal{M}$.

2.2.2 Similarity between selection sets

Our definition of selection similarity takes into account both the levels and the constants that form the selection predicates. In particular, for each hierarchy, two identical clauses are given maximum similarity, and non-identical clauses are given decreasing similarities according to the distance between the hierarchy levels they are expressed on.

Definition 10 (Distance between selection clauses)**.** *Let $\mathcal{M} = \langle L, H, M \rangle$ be a schema, and $p_i = l_i \in X_i$ and $p'_i = l'_i \in X'_i$ be two predicates over hierarchy $h_i \in H$. The* distance *between p_i and p'_i is*

$$Dist_{pred}(p_i, p'_i) = \begin{cases} 1 - \frac{|X_i \cap X'_i|}{|X_i \cup X'_i|}, \text{ if } l_i = l'_i; \\ Dist_{lev}(l_i, l'_i) + 1, \text{ otherwise} \end{cases}$$

According to this definition, the distance between two selection clauses on h_i is 0 if they are expressed on the same level and the same set of members, between 0 (excluded) and 1 if they are defined on the same level with different but non disjoint sets of members, greater than 1 if they are defined on different levels.

Definition 11 (Selection similarity)**.** *Let q and q' be two queries, both on schema $\mathcal{M}$, with selection predicates P and P', respectively, with $P = \{p_1, \ldots, p_n\}$ and $P' = \{p'_1, \ldots, p'_n\}$. The* selection similarity *between q and q' is*

$$\sigma_{sel}(q, q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{pred}(p_i, p'_i)}{|Lev(h_i)|}}{n}$$

2.2.3 Similarity between measure sets

Finally, to define the measure similarity, we use the Jaccard index.

Definition 12 (Measure similarity)**.** *Let q and q' be two queries, both on schema $\mathcal{M}$, with measure sets $Meas$ and $Meas'$, respectively. The* measure similarity *between q and q' is*

$$\sigma_{meas}(q, q') = \frac{|Meas \cap Meas'|}{|Meas \cup Meas'|}$$

2.2.4 Similarity measure between queries

Definition 13 (Similarity of OLAP queries)**.** *Let q and q' be two queries, both on schema $\mathcal{M}$. The* similarity *between q and q' is*

$$\sigma_{que}(q, q') = \alpha \cdot \sigma_{gbs}(q, q') + \beta \cdot \sigma_{sel}(q, q') + \gamma \cdot \sigma_{meas}(q, q')$$

where α, β, and γ are normalized to 1.

Table 2.2: Query similarities for Example 2.2.1

	q_4	q_5	q_6	q_7	q_8
q_1	0.694	0.927	0.844	0.622	0.866
q_2	0.716	0.950	0.866	0.644	0.888
q_3	0.661	0.838	0.755	0.616	0.833

Example 2.2.1. *The similarity between queries q_1 and q_4 of Example 2.1.3 is computed as follows:*

$$\begin{aligned}\sigma_{gbs}(q_1, q_4) =& 1 - \frac{(0/3 + 1/3 + 0/1 + 0/1 + 0/1)}{5} = 0.933\\ \sigma_{sel}(q_1, q_4) =& 1 - \frac{(0/4 + 3/4 + 2/2 + 0/2 + 0/2)}{5} = 0.650\\ \sigma_{meas}(q_1, q_4) =& \frac{1}{2} = 0.500\\ \sigma_{que}(q_1, q_4) =& 0.694\end{aligned}$$

(assuming for simplicity $\alpha = \beta = \gamma = 0.333$). The overall query similarities for sessions s and s' are summarized in Table 2.2.

2.3 Similarity measures between sessions

In this section we define different two-level approaches to compare OLAP sessions, from query similarity defined in the previous section. Indeed, several approaches are possible to answer to the requirements issued for session similarity in the OLAP context (see Section 1.5.2),. Each of them coming from classical measures discusses in Section 1.4.1, but are adapted to the OLAP context.

2.3.1 Extension of the Dice Coefficient

An n-gram is a substring of size n of a given string [Brown et al., 1992]. A popular string similarity function based on n-grams is the *Dice coefficient*, an extension of the Jaccard index defined as twice the number of shared n-grams over the total number of n-grams in the two strings.

In the OLAP context, the concept of "shared" n-grams becomes that of "similar" n-grams. Two n-grams r and r' are similar if their queries are pairwise similar, i.e., if their similarity is above threshold θ. To ensure symmetry while being consistent with the original definition, in our two-level extension similarity is defined as follows.

Definition 14 (Subsequence-Based Similarity of OLAP Sessions)**.** *Let s and s' be two OLAP sessions on schema $\mathcal{M}$, and $n \geq 1$. Given a matching threshold θ, the* subsequence-based similarity *between s and s' is*

$$\sigma_{sub}(s, s') = \frac{2 \times min\{|SNgram_\theta(s, s')|, |SNgram_\theta(s', s)|\}}{|Ngram(s)| + |Ngram(s')|}$$

where $Ngram(s)$ is the set of n-grams of s and $SNgram_\theta(s, s') \subseteq Ngram(s)$ is the set of n-grams of s that have a similar n-gram in s':

$$SNgram_\theta(s, s') = \{r \in Ngram(s) | \exists r' \in Ngram(s'), \sigma_{que}(r_i, r'_i) \geq \theta \; \forall i = 1, \ldots, n\}$$

The complexity of this function is that of finding the n-grams of the two sessions, which is $O(v)$ (where v is the length of the longest one), plus that of computing the sets $SNgram_\theta(s, s')$, which is $O((v-n)^2)$.

Example 2.3.1. *Applying the above definition to Example 2.1.3, with n=1, we obtain* $\sigma_{sub}(s, s') = \frac{2 \times min\{1,2\}}{1+2} = 0.67$.

2.3.2 Extension of the Tf-Idf

In the tf-idf approach, the similarity between two sets of tokens (in information retrieval applications, tokens are lemmas and sets of tokens are documents) depends on both the frequency of each token in the sets and its frequency in a corpus. In our context, this approach can be adopted if the OLAP sessions to be compared are taken from a log, to penalize the non-distinctive queries (i.e., those that are more frequent in the log) when assessing similarity.

To propose an extension of the tf-idf method we start by applying the definition of *soft tf-idf* given by [Moreau et al., 2008]:

$$Sim_{soft}(s, s') = \sum_{s_i \in Close_\theta(s,s')} T(s_i, s) \cdot T(s'_{j_i}, s') \cdot \sigma_{que}(s_i, s'_{j_i})$$

where θ is a threshold,

$$Close_\theta(s, s') = \{s_i \in s | \exists s'_j \in s', \sigma_{que}(s_i, s'_j) > \theta\},$$
$$T(s_i, s) = \frac{tfidf(s_i, s)}{\sqrt{\sum_{s_k} tfidf(s_k, s)^2}},$$
$$tfidf(s_i, s) = tf(s_i, s) \cdot idf(s_i, s) = \frac{n_{s_i,s}}{|s|} \cdot log \frac{|L|}{|\{s \in L | s_i \in s\}|},$$
$$s'_{j_i} = argmax_{s'_j \in s'} \{\sigma_{que}(s_i, s'_j)\},$$

$n_{s_i,s}$ is the number of times s_i appears in s, and L is the set of OLAP sessions in the log. Intuitively, $Close_\theta(s, s')$ is the set of queries in sessions s that have some similarity to a query in session s'; $tfidf(s_i, s)$ is directly proportional to the frequency of query s_i in session s and inversely proportional to the frequency of s_i in the log L ($tfidf(s_i, s) = 0$ when all session in L include s_i); $T(s_i, s)$ is a normalized form of $tfidf(s_i, s)$; s'_{j_i} is the query in s' that is most similar to s_i.

This definition cannot be immediately used in our case for the following reasons:

1. It uses the "crisp" definition of tf-idf in the definition of T whereas in our case, given that it is unlikely to find the same query twice in an OLAP log, a "soft" version (i.e., one based on query similarity) should be used instead.

2. The soft tf-idf is not symmetric, which is not desirable for a similarity function.
3. There may be more than one query s'_{j_i} in s' that maximizes σ_{que} with s_i, which may not be relevant in the context of named entity matching [Moreau et al., 2008], but is definitely relevant in the OLAP context.
4. As pointed out by [Moreau et al., 2008], there is a problem with counting that makes the similarity not normalized.

To cope with the first issue, we inject the similarity σ_{que} in the definition of tf-idf. By replacing equality with similarity, a two-level tf-idf can be computed as:

$$tfidf_2(s_i, s) = \frac{|Close_\theta(s_i, s)|}{\sum_{s_k \in Q} |Close_\theta(s_k, s)|} \cdot log \frac{|L|}{|\{s \in L | Close_\theta(s_i, s) \neq \emptyset\}|}$$

where Q is the set of all queries in L and $Close_\theta(s_i, s)$ is the set of queries of s that are similar to s_i.

Symmetry can be achieved by modifying the definition of similarity to work on pairs of queries, each relating a query in one session with one of its closest queries in the other session. This set of pairs is defined by:

$$R_\theta(s, s') = \{\langle s_i, s'_k \rangle | s_i \in s, s'_k \in Closest_\theta(s_i, s'))\} \cup \{\langle s_l, s'_j \rangle | s'_j \in s', s_l \in Closest_\theta(s'_j, s)\}$$

where $Closest_\theta(s_i, s)$ is the set of queries of s that have maximum similarity with s_i. Note that a query in a session appears more than once in $R_\theta(s, s')$ if there is more than one query in the other session with maximum similarity. This solves the third issue.

Finally, to cope with the fourth issue, the similarity is computed as the cosine of the two vectors obtained by taking the $tfidf_2$ of all the first (respectively, second) queries of the pairs.

Definition 15 (Log-Based Similarity of OLAP Sessions)**.** *Let s and s' be two OLAP sessions on schema $\mathcal{M}$. The* log-based similarity *between s and s' is*

$$\sigma_{log}(s, s') = \sum_{\langle s_i, s'_j \rangle \in R_\theta(s, s')} T_2(s_i, s, s') \times T_2(s'_j, s', s) \times \sigma_{que}(s_i, s'_j)$$

where

$$T_2(s_i, s, s') = \frac{tfidf_2(s_i, s)}{\sqrt{\sum_{\langle s_i, s'_j \rangle \in R_\theta(s, s')} tfidf_2(s_i, s)^2 + \sum_{Closest_\theta(s_i, s') = \emptyset} tfidf_2(s_i, s)^2}}$$

$$T_2(s'_j, s', s) = \frac{tfidf_2(s'_j, s')}{\sqrt{\sum_{\langle s_i, s'_j \rangle \in R_\theta(s, s')} tfidf_2(s'_j, s')^2 + \sum_{Closest_\theta(s'_j, s) = \emptyset} tfidf_2(s'_j, s')^2}}$$

The complexity of this function should obviously be expressed not only in terms of the sessions to be compared but also in terms of the size of the log; it turns out that the complexity of computing $R_\theta(s, s')$ is $O(v^2)$, while that for computing all the $tfidf_2$ terms it is $O(v \times |Q|)$ where v the length of the longest session in the log.

Note that, as any cosine similarity, σ_{log} can be easily turned into the angle distance $arcos(\sigma_{log})$, which is a metric [Bustos and Skopal, 2011].

Example 2.3.2. *With reference to Example 2.1.3, we focus on computing the log-based similarity between s and s'. The set of query pairs used in the computation of $\sigma_{log}(s, s')$ is $R_{0.7}(s, s') = \{\langle q_1, q_5\rangle, \langle q_2, q_5\rangle, \langle q_3, q_5\rangle, \langle q_2, q_4\rangle, \langle q_2, q_6\rangle, \langle q_2, q_8\rangle\}$; the two components of the $tfidf_2$ weights for each of these queries are as follows:*

$$\begin{aligned}
tf_2(q_1, s) &= 0.333\,, & idf_2(q_1, s) &= 0.176\\
tf_2(q_2, s) &= 0.333\,, & idf_2(q_2, s) &= 0.176\\
tf_2(q_3, s) &= 0.333\,, & idf_2(q_3, s) &= 0.000\\
tf_2(q_4, s') &= 0.117, & idf_2(q_4, s') &= 0.176\\
tf_2(q_5, s') &= 0.235, & idf_2(q_5, s') &= 0.176\\
tf_2(q_6, s') &= 0.235, & idf_2(q_6, s') &= 0.176\\
tf_2(q_8, s') &= 0.235, & idf_2(q_8, s') &= 0.000
\end{aligned}$$

Note that, though q_3 and q_8 are similar (same group-by set, same selection predicate, and nearly the same set of measures) and should positively contribute to the similarity of s and s', they do not actually enter in the computation of $\sigma_{log}(s, s')$. Indeed, queries similar to q_3 and q_8 can be found in each session of the log, making their idf weight 0. By applying Definition 15 we get $\sigma_{log}(s, s') = 0.479$, while $\sigma_{log}(s, s'') = \sigma_{log}(s', s'') = 0$.

2.3.3 Extension of the Levenshtein Distance

The Levenshtein distance compares two strings in terms of the cost of the atomic operations (typically insertion, deletion, and substitution of a character) necessary to transform one string into another [Ristad and Yianilos, 1998]. Given two strings s and s' of v and v' characters, respectively, a $(v + 1) \times (v' + 1)$ distance matrix D of reals is recursively defined in terms of the deletion, insertion, and substitution costs; the Levenshtein distance between s and s' is found in the bottom-right cell of D, that represents the minimum sum of the operation costs to transform s in s'.

In the traditional formulation, an operation is applied in absence of a perfect match (i.e., of an identity) between the compared characters. In our case this is too restrictive, because OLAP queries are complex objects whose match is not effectively captured by identity (see requirement ♯8). So we consider two queries as matching when their similarity is above a given threshold θ, and we apply a transformation operation when the similarity is under θ. Besides, we normalize distances using the length of the longest of the two sessions involved, so that the cost of a single mismatch is lower for longer sessions.

Definition 16 (Edit-Based Similarity of OLAP Sessions)**.** *Let s and s' be two OLAP sessions on schema $\mathcal{M}$, of lengths v and v' respectively. Given a matching threshold θ, the distance matrix for s and s' is a $(v+1) \times (v'+1)$ matrix D_θ of reals recursively defined as*

follows:

$$D_\theta(i,j) = \begin{cases} 0, \text{ when } i = 0 \text{ or } j = 0 \\ D_\theta(i-1, j-1), \text{ when } i,j > 0 \text{ and } \sigma_{que}(s_i, s'_j) \geq \theta \\ min \left\{ \begin{array}{c} D_\theta(i-1,j) + 1; \\ D_\theta(i,j-1) + 1; \\ D_\theta(i-1,j-1) + 1 \end{array} \right\}, \text{ when } i,j > 0 \text{ and } \sigma_{que}(s_i, s'_j) < \theta \end{cases}$$

where s_i is the i-th *query of session s. The* edit-based similarity *between s and s' is:*

$$\sigma_{edit}(s, s') = 1 - \frac{D_\theta(v, v')}{max\{v, v'\}}$$

Note that, like in most applications of the Levenshtein distance, all transformation costs are set to 1.[2] As to complexity of this function, in the general case it is $O(v \cdot v')$ where v and v' are the lengths of the two sessions [Wagner and Fischer, 1974].

Example 2.3.3. *With reference to Example 2.1.3 and using $\theta = 0.7$, the minimum cost to transform s' to s is obtained by matching queries as follows: $\langle q_1, q_5 \rangle, \langle q_2, q_6 \rangle, \langle q_3, q_8 \rangle$ and deleting q_4 and q_7. Thus, it is $\sigma_{edit}(s, s') = 1 - \frac{2}{5} = 0.60$.*

2.3.4 Extension of the Sequence Alignment

As emerged in Section 1.4.1, a comparison of OLAP sessions should support subsequence alignment, keep query ordering into account, and allow gaps in the matching subsequences. The Smith-Waterman algorithm mentioned in Section 1.4.1 has all these features. It relies on a distinction between *matching* elements (whose similarity is positive) and *mismatching* elements (whose similarity is negative), and is based on a matrix whose cells show the score for aligning two sequences starting from a specific couple of elements. Each score is the result of a trade-off between the cost for introducing a gap in the matching subsequences and the cost for including a mismatching pair of elements.

Unfortunately, none of the implementations available in the literature can be directly applied here for different reasons:

- The algorithm was originally aimed at molecular comparison, so sequence elements were taken from a set that is known a priori (the set of all amino acids). This allows matching and mismatching pairs to be enumerated and a similarity score to be assigned in advance to each possible couple of elements. In the OLAP context matching elements are queries, and the domain of the possible OLAP queries is huge (requirement ♯8); besides, the similarity between two queries is always positive, so separating matching and mismatching queries requires the adoption of a threshold.
- For the same reason mentioned above, in all previous implementations the cost for introducing a gap could be assigned in advance to each possible couple of elements. Conversely, in our case it must be determined at runtime based on the two specific sessions being compared (requirement ♯14).

2. In the formula, the three rows of the *min* argument deal with deletions, insertions, and substitutions, respectively.

– In all previous implementations all matchings were considered to be equally important, while in OLAP sessions a matching between recent queries should be given more relevance (requirement ♯12).

To address all these issues, we propose an extension of the Smith-Waterman algorithm that relies on the matrix defined below. The value in position (i, j) of this matrix is a score that expresses how "well" two sessions s and s' match when they are aligned ending in queries s_i and s'_j. Intuitively, each score is recursively calculated by progressively adding the similarities between all pairs of matching queries in the two sessions. Threshold θ is used to distinguish matches from mismatches; a *time-discounting* function $\rho(i, j)$ is used to promote alignments based on recent queries; finally, a *gap penalty* δ is used to discourage discontinuous alignments.

Definition 17 (OLAP Session Alignment Matrix)**.** *Let s and s' be two OLAP sessions on schema $\mathcal{M}$, of lengths v and v' respectively. Given a matching threshold θ, the* (OLAP session) alignment matrix *for s and s' is a $(v+1) \times (v'+1)$ matrix A of reals recursively defined as follows:*

$$A(i,j) = \begin{cases} 0, \text{ when } i = 0 \text{ or } j = 0 \\ max \left\{ \begin{array}{c} 0; \\ A(i-1, j-1) + (\sigma_{que}(s_i, s'_j) - \theta) \cdot \rho(v-i, v'-j); \\ max_{1 \le k < i}\{A(k,j) - \delta \cdot (i-k)\}; \\ max_{1 \le k < j}\{A(i,k) - \delta \cdot (j-k)\} \end{array} \right\}, \text{ else} \end{cases}$$

where δ is the average similarity between all couples of queries in s and s' whose similarity is above θ:

$$\delta = avg_{(i,j):\sigma_{que}(s_i,s'_j) \ge \theta}\{\sigma_{que}(s_i, s'_j)\} ,$$

ρ is a two-dimensional logistic sigmoid function:

$$\rho(i,j) = 1 - \frac{1 - \rho_{min}}{1 + e^{slope - i - j}} ,$$

ρ_{min} is the minimal value assumed by ρ (i.e., the maximum time discount), and slope rules the position where the slope is steepest (Figure 2.1).

Some observations on the above definition:

– The use of the term $\sigma_{que}(s_i, s'_j) - \theta$ implies that query pairs whose similarity is above (below) θ are considered as matches (mismatches). Although a "sharp" threshold is used, the score of a matching pair and the cost of a mismatching pair turn out to be proportional to the distance of that pair similarity from θ.

– The definition given of the gap penalty δ is such that it guarantees a gap penalty to be payed if it enables a good match (i.e. a match higher than the average). Note that a penalty only related to the threshold could lead to underestimating or overestimating the impact of a gap on the overall similarity.

– The time-discounting function ρ leads match and mismatch scores to decay when moving backwards along the two sessions; it is maximum and equal to 1 for the ending queries of the two sessions.

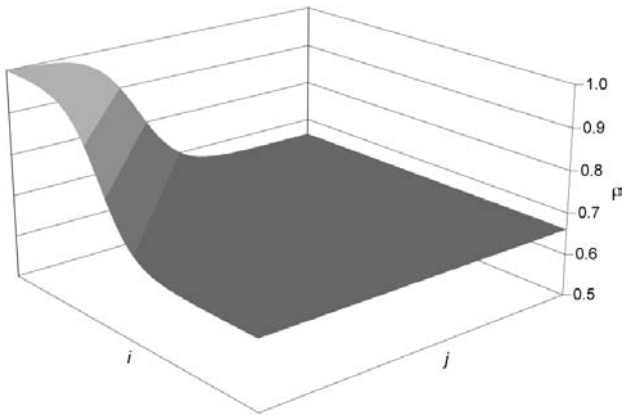


Figure 2.1: The time-discounting function $\rho(i, j)$ with $\rho_{min} = 0.66$ and $slope = 4$

Table 2.3: Threshold-filtered and discounted query similarities, $(\sigma_{que}(s_i, s'_j) - \theta) \cdot \rho(v - i, v' - j)$, for Example 2.3.4

	q_4	q_5	q_6	q_7	q_8
q_1	-0.004	0.171	0.120	-0.071	0.160
q_2	0.013	0.208	0.151	-0.053	0.186
q_3	-0.032	0.126	0.053	-0.082	0.132

The optimal alignment between s and s' is determined by the highest value in A, $\overline{A}$, that we call *alignment score*. The positions $\overline{i}$ and $\overline{j}$ such that $A(\overline{i}, \overline{j}) = \overline{A}$ mark the end of the matching subsequences of s and s'.

The alignment score is not really a similarity value, since it is not limited in the interval [0..1]. This creates problems when comparing sessions with difference length. Then we define OLAP session similarity by normalizing the alignment score:

Definition 18 (Alignment-Based Similarity of OLAP Sessions)**.** *Let s and s' be two OLAP sessions on schema $\mathcal{M}$, of lengths v and v' respectively (with $v \leq v'$), and let $\overline{A}$ be the alignment score for s and s'. The* alignment-based similarity *between s and s' is*

$$\sigma_{ali}(s, s') = \frac{\overline{A}}{(1-\theta)\sum_{k=1}^{v} \rho(k, k)}$$

where the normalizing factor is the alignment score for two identical sessions of length v.

Like for edit-based similarity, the complexity of this function is known to be $O(v \cdot v')$ where v and v' are the lengths of the two sessions [Li and Durbin, 2010].

Example 2.3.4. *Again we focus on comparing s and s' of Example 2.1.3. Table 2.3 reports the results obtained by filtering query similarities with $\theta = 0.7$ and applying the time-discounting function ρ as shown in Definition 17. Note that a negative value represents a mismatch, and a positive one a match. Table 2.4 shows the OLAP session alignment matrix for s and s'; the cells in bold denote alignments between two queries (e.g., q_1 is*

Table 2.4: OLAP session alignment matrix for Example 2.3.4

	q_4	q_5	q_6	q_7	q_8
q_1	0.000	**0.171**	0.120	0.000	0.160
q_2	0.013	0.208	**0.322**	*0.191*	0.186
q_3	0.000	0.139	0.261	0.241	**0.323**

aligned with q_5), those in italics refer to gaps. Alignments on recent queries are favored, so q_3 is aligned with q_8. Query q_4 is not involved in the alignment due to the low similarity it has with the other queries in s. In q_7, a gap penalty is paid to gain the good match between q_3 and q_8. The overall similarity between s and s' is 0.323 (the highest value in the matrix). After normalization, we obtain $\sigma_{ali}(s, s')$=0.387.

The properties of the proposed similarity function can be evaluated in terms of the distance function it induces using the standard transformation $\sigma_{ali} = 1/(1 + Dist_{ali})$. As stated by [Bustos and Skopal, 2011] for the original Smith-Waterman approach, $Dist_{ali}$ is not a metric because, while it is non-negative and symmetrical, it is not reflexive and it does not satisfy the triangular inequality as shown in Example 2.3.5. In particular, the triangular inequality cannot be satisfied because this approach is based on a local alignment.

Example 2.3.5. *Let $s = \langle q_1, q_2 \rangle$, $s' = \langle q_1, q_2, q_3, q_4 \rangle$, and $s'' = \langle q_3, q_4 \rangle$ be three sequences, where $\sigma_{que}(q_i, q_j) = 0$ if $i \neq j$. It is*

$$Dist_{ali}(s, s'') = \infty \ , \ Dist_{ali}(s, s') = Dist_{ali}(s', s'') = 0$$

which obviously contradicts the triangle inequality axiom. Besides, s' has zero distance from both s and s'' though $s \neq s' \neq s''$.

2.4 Similarity between OLAP Logs

In this Section we define different three-level approaches to compare OLAP logs. We recall that a log is composed of a set of sessions (see Definition 7) and consequently, the similarity between logs will be based on a session similarity measure. In Section 2.4.1, an Accuracy-based similarity is defined and extends classical Precision and Recall measures. A similarity based on the classical Hausdorff distance is also proposed in Section 2.4.2. Finally, a similarity based on the Jaccard Coefficient is defined in Section 2.4.3.

2.4.1 Accuracy-based Similarity

Accuracy-based Similarity is measured by extending the classical precision and recall measures to take similar sessions into account.
Let $\mathcal{L}$ and $\mathcal{L}'$ be two logs to compare. The more $\mathcal{L}$ and $\mathcal{L}'$ share similar sessions, the higher similarity value is.

We define the set TP_{sim} of true positives by $\{s \in \mathcal{L} | s' \in \mathcal{L}' \wedge \sigma_{session}(s, s') >= \tau_{session}\}$, i.e. the set of sessions that are in $\mathcal{L}$ and similar to those in $\mathcal{L}'$. The set of false positives FP_{sim} is $\mathcal{L} \setminus TP_{sim}$ and the set of false negative FN_{sim} is $\mathcal{L}' \setminus \{s' \in \mathcal{L}' | s \in$

$\mathcal{L} \wedge \sigma_{session}(s, s') >= \tau_{session}\}$. Note that $\sigma_{session}$ is a similarity measure between sessions as those defined in Section 2.3. Two sessions are considered as similar if their similarity score is higher than $\tau_{session}$.

We defined precision and recall extending for similar sessions as:

- $\sigma_{Precision} = \frac{|TP_{sim}|}{|TP_{sim}|+|FP_{sim}|}$
- $\sigma_{Recall} = \frac{|TP_{sim}|}{|TP_{sim}|+|FN_{sim}|}$

The global accuracy is measured as the F1-score:

$\sigma_{Accuracy}(\mathcal{L}, \mathcal{L}') = 2 * \frac{\sigma_{Precision} * \sigma_{Recall}}{\sigma_{Precision} + \sigma_{Recall}}$

We can note that the $\sigma_{accuracy}(\mathcal{L}, \mathcal{L}')$ is not a symmetric measure.

2.4.2 Similarity based on the Hausdorff Distance

The classical Hausdorff distance compares two sets, based on the distance between the elements of these sets. We adapt this distance for defining a similarity adapted to an OLAP context.

Let $\mathcal{L}$ and $\mathcal{L}'$ be two logs to compare.

$$\sigma_{Hausdorff}(\mathcal{L}, \mathcal{L}') = min\{min_{s \in \mathcal{L}}\ max_{s' \in \mathcal{L}'}\ \sigma_{session}(s, s'), min_{s' \in \mathcal{L}'}\ max_{s \in \mathcal{L}}\ \sigma_{session}(s, s')\}$$

2.4.3 Jaccard Similarity Coefficient

The classical Jaccard coefficient identifies the ratio of similar elements between two sets. We adapt this measure to compare two OLAP logs, identifying the ratio of similar sessions.

Let $\mathcal{L}$ and $\mathcal{L}'$ be two logs to compare.

$$\sigma_{Jaccard}(\mathcal{L}, \mathcal{L}') = \frac{|\mathcal{L} \bigcap_{\sigma_{session}(s \in \mathcal{L}, s' \in \mathcal{L}) >= \tau_{session}} \mathcal{L}'|}{|\mathcal{L} \bigcup_{\sigma_{session}(s \in \mathcal{L}, s' \in \mathcal{L}) >= \tau_{session}} \mathcal{L}'|}$$

2.5 Conclusion

This chapter developed the formal definitions for queries, sessions and logs. Our query model is composed with sets of fragments (group by sets, selection predicate sets, measure sets) allowing to consider the user's intension, without evaluating the query. The use of this model in user-centric approaches, like recommendation presented in this dissertation, will ensure efficient approaches.

We also presented in this Chapter, different similarity measures between queries and sessions, that are extensions of classical measures in information retrieval (Dice Coefficient, Sequence Alignment, TF-IDF, Levenshtein Distance). These measures can be applied in clustering techniques to identify, for instance, profiles in an optimization context [Golfarelli, 2003] or user-centric context [Yao et al., 2005], [Giacometti et al., 2009]. From the requirements considered in Section 1.5.2 and subjective and objective tests (given in Sections 4.2.1 and 4.2.2), the alignment-based measure seems a relevant candidate for a recommendation system.

Finally, different similarity measures between logs complete this Chapter. Each of

these measures compares sets of OLAP sessions by extending classical measures that are Accuracy, Hausdorff Distance and Jaccard Coefficient. These measures will allow to define quality measures from the criteria discussed in Section 1.5.1, to assess the relevance of the sessions obtained by the recommendation system (given in Chapter 4).

Chapter 3

SROS System

This chapter is devoted to our recommendation system, named *SROS* (Similarity-based Recommendation of OLAP Sessions). Section 3.1 introduces the approach by giving the intuitions and the sequencing between the different phases that compose the system. Each phase of the *SROS* system is explained in the following sections. Section 3.2 develops the *Selection* phase allowing to obtain a set of possible recommended sessions. Section 3.3 presents the *Ranking* phase determining the most relevant base recommendation. The *Tailoring* phase, adapting the base recommendation to the current session, is explained in Section 3.4. Note that a running example illustrates each phase of the *SROS* system.

3.1 Principle

We introduce in this section the *SROS* system, and more specifically the different phases to recommend a sequence of queries for a current session. Figure 3.1 depicts the principle of *SROS*. A user conducts an OLAP session for which *SROS* will recommend a sequence of queries leveraging former sessions devised by previous users. The *SROS* system is organized in three phases:

1. A *Selection* phase that identifies in a log a set of sessions that constitute relevant futures for the current session.
2. A *Ranking* phase that determines among these sessions one of them whose portion will be the base recommendation.
3. A *Tailoring* phase that adapts the base recommendation to the current session.

Algorithm 2 sketches the overall sequencing of the system, whose *Selection*, *Ranking* and *Tailoring* functions are presented, in the next sections.

3.2 Selection of Futures

The goal of this first phase is to identify in a log $\mathcal{L}$ a set F of sessions that would provide relevant futures for the current session. Such sessions are found by matching each session of $\mathcal{L}$ with the current session.

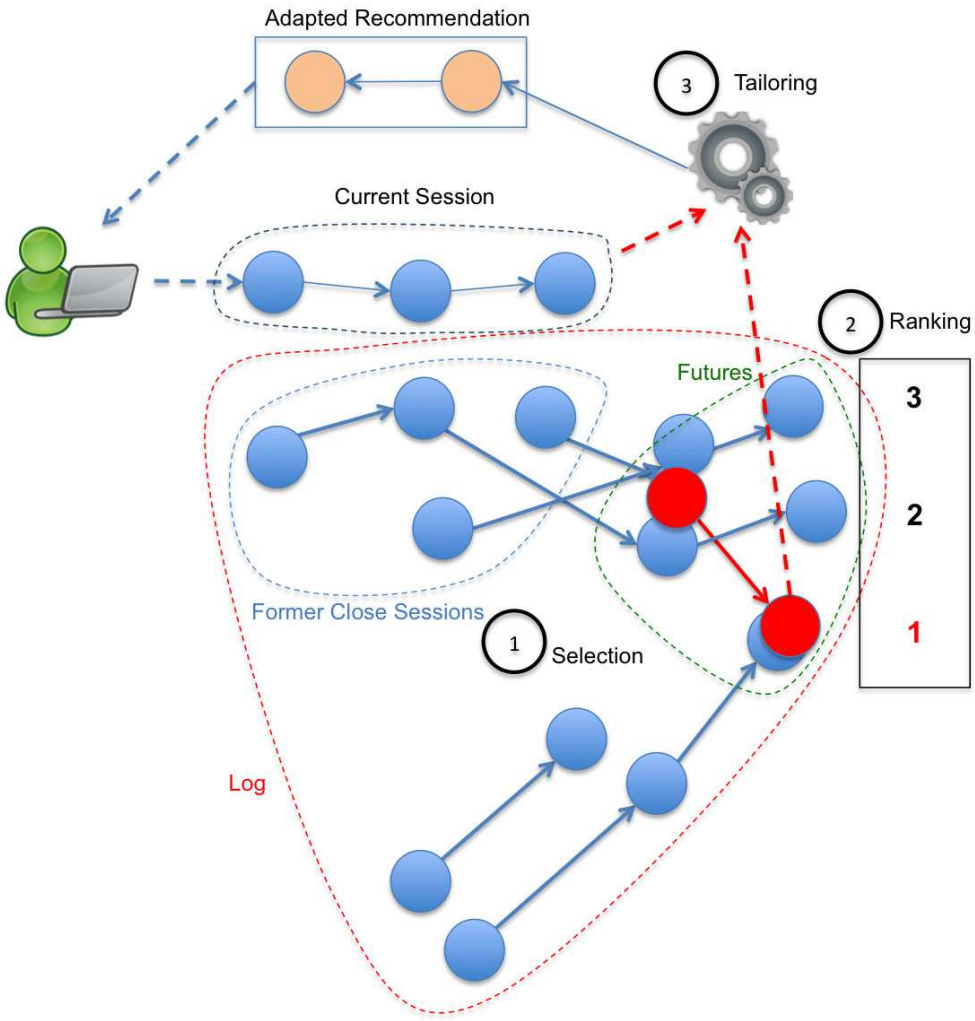


Figure 3.1: Principle of *SROS*

Algorithm 2 SROS

INPUT: s_{curr}: the current session, $\mathcal{L}$: a log;
OUTPUT: s_{reco}: the session to recommend;
VARIABLES: F: the set of futures; s_{base}: the base recommendation; s_{reco}: the recommended session;
1: $F \leftarrow Selection(s_{curr}, \mathcal{L})$
2: $s_{base} \leftarrow Ranking(F, \mathcal{L})$
3: $s_{reco} \leftarrow Tailoring(s_{base}, s_{curr})$
4: **return** s_{reco}

3.2.1 Selecting Log Sessions

The aim is to find the largest subsession of the current session matching some subsessions located as close to the beginning of the log sessions, as possible. To match log sessions with the current session, similarity measures between query sequences can be used (as those defined in Section 2.3). The comparison between subsessions allows to give more flexibility for the match, since the queries of the current session do not need to be absolutely similar with those of the log session to compare. Moreover, it is better if the current session matches with subsessions located as close to the beginning of the log sessions, as possible. Indeed, the sequence following the subsession of the log session will be considered as a possible future for the current session for which it is important to maximize the number of queries.

The use of Sequence Alignment given by the Smith-Waterman algorithm seems relevant since the comparison between two sequences is based on the best subsession alignment. However, the extension of the Sequence Alignment in the OLAP context, proposed in Definition 18, has to be modified. Indeed, the sigmoid function proposed in this extension (see Definition 17) favors alignments between the ends of the sessions to compare. Whereas, the aim of this phase is to promote alignments between the end of a current session and the beginning of the log sessions. To cope with this, we propose a new definition of the sigmoid function, given below. Its behavior is illustrated in Figure 3.2, that depicts the function applied to a current session of length 10 and a log session of length 15.

Definition 19 (Sigmoid Function for Recommendation)**.** *Let l_s be a log session and s_{curr} be the current session. ρ_{reco} is a two-dimensional logistic sigmoid function:*

$$\rho_{reco}(i,j) = 1 - \frac{1 - \rho_{reco-min}}{1 + e^{\frac{-20}{|l_s|}*j + \frac{5}{|s_{curr}|}*i + \frac{10}{|s_{curr}|}}} ,$$

where i is a query position in the current session s_{curr}, j is a query position in log session l_s, $\rho_{reco-min}$ is the minimal value assumed by ρ_{reco} (i.e., the minimal weight given for query alignments considered as irrelevant). Note that the constants have been experimentally defined in order to answer to specific desired behaviors:

- *$\frac{-20}{|l_s|}$ defines the proportion of queries in the current session whose alignment with the first queries of the log session has to be favored.*
- *$\frac{5}{|s_{curr}|}$ defines the proportion of queries in the log session whose alignment with the last queries of the current session has to be favored.*
- *$\frac{10}{|s_{curr}|}$ defines a minimal weight to consider between the first queries of the current and log sessions.*

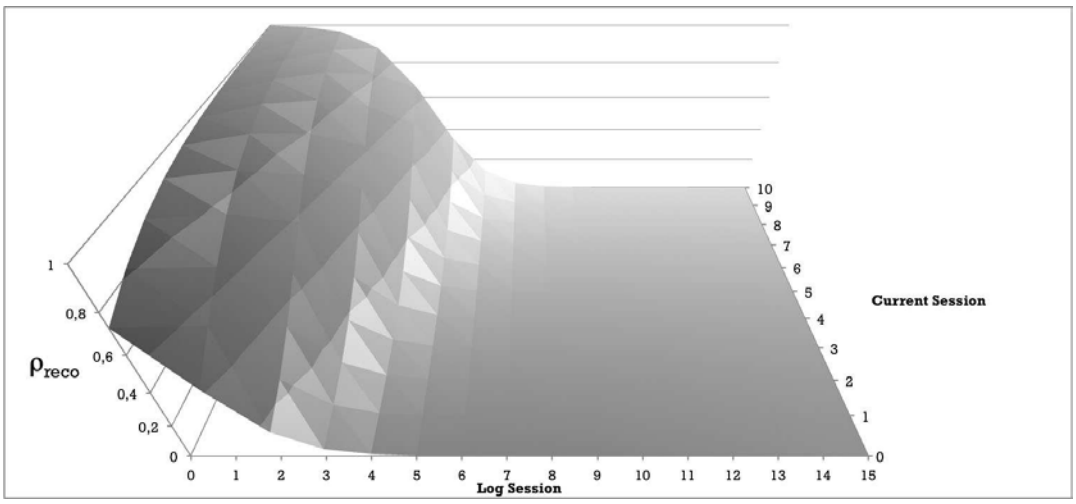


Figure 3.2: The Sigmoid Function for Recommendation, used to align Current and Log Sessions

We recall that the output of the Smith-Waterman algorithm is an *alignment* a. An alignment of s_i and s_j is defined by two matching subsessions $s_i[v_1, v_2]$ and $s_j[w_1, w_2]$ and by their similarity σ. We will denote the starting and ending positions of the matching subsessions as $a^i.from = v_1$, $a^i.to = v_2$, $a^j.from = w_1$, and $a^j.to = w_2$, and the similarity with $sim(a) = \sigma$.

3.2.2 Determining Futures

The set F of possible futures for the current session, initially empty, is computed as follows. Each session s_l of the log S is aligned with the current session s_{cur}, resulting in an alignment a, where subsessions $s_{cur}[v1, v2]$ and $s_l[w1, w2]$ match. If $sim(a) > 0$ and $length(s_{cur}) - \tau_{align} \times length(s_{cur}) \leqslant v2 \leqslant length(s_{cur})$, then $s_l[w2, .]$ is added to F.

The parameter τ_{align} indicates the percentage of last queries of the current session that can be left non aligned with queries of the log sessions. Indeed, the alignment being based on subsessions, it is possible that queries of the current session are not aligned. However, it seems important to align a large number of last queries of the current session with queries of the log session. That is why, the value given for this parameter has to be restrictive enough to avoid a future deriving too much from the last queries of the current session.

The pseudocode of the *Selection* phase to obtain a set of futures is given in Algorithm 3.

Considering the CENSUS multidimensional schema presented in Example 1.3.4, we motivate the *SROS* system by a running example, given below.

Example 3.2.1. *Let L be a log of two sessions, $s_1 = \langle q_{11}, q_{12}, q_{13}, q_{14}\rangle$ and $s_2 = \langle q_{15}, q_{16}, q_{17}, q_{18}, q_{19}\rangle$, generally focused on the study of average income or cost of water for female in different years and different types of race. Let $s_{curr} = \langle qc_1, qc_2\rangle$ be the*

Algorithm 3 Selection

INPUT: s_{curr}: the current session, L: the log;
OUTPUT: F: set of sessions;

```
F ← ∅                                                          ▷ Initialize F
for each s ∈ L do
    a ← SW_cand(s_cur, s)                                      ▷ Align s with s_cur
    if a ≠ NULL and s_cur^(a)[.] = s_cur[.] and s^(a)[.] ≠ s[.] then   ▷ An alignment is found ending in s_cur[.]
        v ← position in s of the last query in s^(a)
        f ← s[v + 1, .]                                        ▷ Find the candidate recommendation
        F ← F ∪ {f}
return F
```

Table 3.1: Queries for Example 3.2.1

		Queries										
		q_{11}	q_{12}	q_{13}	q_{14}	q_{15}	q_{16}	q_{17}	q_{18}	q_{19}	qc_1	qc_2
Group-by set		g_5	g_6	g_7	g_7	g_6	g_7	g_7	g_7	g_7	g_5	g_6
Measures	AvgCostWtr	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	AvgCostElect										✓	✓
	AvgIncome				✓	✓	✓			✓		
Selection predicates		P_1	P_1	P_4	P_4	P_5	P_5	P_5	P_6	P_6	P_7	P_7

current session composed of two queries and analyzing both the average costs of water and electricity in 2002 for different types of race.

Table 3.1 represents the queries of s_1, s_2 and s_{curr}.

The involved group-by sets are:

$g_5 = \langle \mathsf{AllCities}, \mathsf{Race}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{AllOccs} \rangle$
$g_6 = \langle \mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{AllOccs} \rangle$
$g_7 = \langle \mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{Sex}, \mathsf{AllOccs} \rangle$

Table 3.2: OLAP session alignment matrix between s_{curr} and s_1 for Example 3.2.1

	q_{11}	q_{12}	q_{13}	q_{14}
qc_1	**0.157**	0.037	0	0
qc_2	0.138	**0.335**	0.215	0.095

Table 3.3: OLAP session alignment matrix between s_{curr} and s_2 for Example 3.2.1

	q_{15}	q_{16}	q_{17}	q_{18}	q_{19}
qc_1	0.036	0	0.001	0	0
qc_2	**0.076**	0.034	0.057	0.015	0

While the selection predicates are:

$$P_1 = \{TRUE_{\mathsf{RESIDENCE}}, TRUE_{\mathsf{RACE}}, \mathsf{Year} \in \{2005\}, TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}$$
$$P_4 = \{TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2005\}, TRUE_{OCCUPATION}, \mathsf{Sex} \in \{Female\}\}$$
$$P_5 = \{TRUE_{\mathsf{RESIDENCE}}, TRUE_{\mathsf{RACE}}, \mathsf{Year} \in \{2004\}, TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}$$
$$P_6 = \{TRUE_{\mathsf{RESIDENCE}}, TRUE_{\mathsf{RACE}}, \mathsf{Year} \in \{2004\}, TRUE_{\mathsf{OCCUPATION}}, \mathsf{Sex} \in \{Female\}\}$$
$$P_7 = \{TRUE_{\mathsf{RESIDENCE}}, TRUE_{\mathsf{RACE}}, \mathsf{Year} \in \{2002\}, TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}$$

We first apply the Aligning phase. We align all the sessions of the log with the current session. Note that we set $\tau_{align} = 1$ for this example, i.e. the last query of s_{curr} has to be aligned with a query of the log sessions. The session alignment between s_c and s_1 is given in Table 3.2. We can note that $\langle qc_1, qc_2\rangle$ is aligned with $\langle q_{11}, q_{12}\rangle$ of s_1, with a score of 0.335, The session alignment between s_c and s_2 is given in Table 3.3. We can note that $\langle qc_2\rangle$ is aligned with $\langle q_{15}\rangle$ of s_2, with a score of 0.076.

Consequently, the set of possible futures is $F = \{future_1 = \langle q_{13}, q_{14}\rangle, future_2 = \langle q_{16}, q_{17}, q_{18}, q_{19}\rangle\}$

3.3 Ranking Futures and Extraction of the Base Recommendation

We detail in this section the phase allowing to obtain a base recommendation. The first step of this phase, presented in Section 3.3.1, ranks the futures returned by the *Selection* phase. Then, a second step, described in Section 3.3.2, extracts a subsession for the future, having the best score, to form the base recommendation.

3.3.1 Ranking Candidate Futures

Given a log session $s_i \in L$, we will refer to $f_i \sqsubseteq s_i$ as the $future$ of s_i, i.e., the subsession $s_i[v, .]$ such that $s_i[v-1]$ is the query of s_i aligned with the last query of the current session s_{cur}.

The result of the previous phase is the set F of possible futures for s_{cur}. The goal of this phase is to examine F in order to determine a base recommendation r, which will be refined in the next phase. Consistently with collaborative filtering approaches, our goal is to identify the densest areas in F so as to determine r as the most relevant subsession in F. More precisely, this step is composed of two separate stages:

1. A *ranking* stage, in which we calculate a relevance score for each single query $q \in f_i$, $f_i \in F$.
2. An *evaluation* stage, in which we calculate a relevance score of each possible subsession in f_i by averaging the relevance of its queries; finally, we choose r as the subsession with the highest relevance.

The pseudocode for this phase is sketched in Algorithm 4. During the ranking stage (lines from 1 to 10) we calculate the relevance of every $q \in f_i$ by initializing it to 0 and increasing it proportionally to the similarity between q and the queries in the other sessions. To this end, we compute the pairwise alignment between all futures in F, making use of the scoring version of the Smith-Waterman algorithm (see Definition 18). This version of the Smith-Waterman algorithm identifies a set A of alignments between two sessions f_i and f_j; for each $a \in A$, the relevance of every aligned query in f_i and f_j is increased by the alignment score, i.e., $sim(a)$.

Once the pairwise alignments have been computed, the evaluation stage starts (line 11). The relevance of each f_i is computed (that is the average relevance of the queries of f_i) and the one having the maximally score is selected. Then, the *Recommendation Length* step (line 12) is applied on this future, whose principle is explained below.

3.3.2 Extracting the Base Recommendation

The aim of this step it to extract a subsession from the future, obtained previsouly, that forms the base recommendation. This subsession is obtained by choosing the sequence of queries having a maximally average relevance from the most relevant query. The most relevant query, simply MRQ, is the query having the highest score among all the queries of the future.

The number of queries in the session to recommend is determined by picking the neighbor queries to the MRQ and stoping if their score decreases too much. The pseudocode of this step is given in Function 5.

In particular, the average score gap λ_{gap} is computed between two consecutive queries in the future (line 1). The queries of the base recommended session are found by starting from the MRQ, and λ^0_{score} is the average score of the recommended session, at first that of the MRQ (lines from 2 to 6). Then, the query immediately before this recommended session and the one immediately after the recommended session are considered (lines from 7 to 8). The query maximizing the average score of the recommended session λ^t_{score} is added only if this average score is greater than $\lambda^{t-1}_{score} - \lambda_{gap}$. This principle is repeated until no more query can be added. Finally, the set of queries chosen delimitates the subsession of the future to extract as the base recommendation.

Considering the futures obtained in Example 3.2.1, we now apply the *Ranking* phase.

Algorithm 4 Ranking

Input: F: set of futures of sessions aligned with s_{curr};
Output: r: base recommendation
Variables: A: set of alignments;

for each $f_i \in F$ **do** ▷ Initialize query relevance
  for each $q \in f_i$ **do**
    $q.relevance \leftarrow 0$
for each $f_i \in F, f_j \in F, f_i \neq f_j$ **do**
  $A \leftarrow SW_{sco}(f_i, f_j)$ ▷ Compute pairwise alignments...
  for each $a \in A$ **do** ▷ ...and update query relevance
    for each $q \in f_i[a^i.from, a^i.to]$ **do**
      $q.relevance \leftarrow q.relevance + sim(a)$
    for each $q \in f_j[a^j.from, a^j.to]$ **do**
      $q.relevance \leftarrow q.relevance + sim(a)$
$future \leftarrow getFutureWithMaxAvgRelevance(F)$ ▷ Find maximally relevant future...
$r \leftarrow Length(future)$
return r ▷ Return the base recommendation

Function 5 Length

Input: $future$: relevant future;
Output: r: base recommendation;

$\lambda_{gap} \leftarrow averageGap(F)$ ▷ Computation of the average gap of scores in r
$id_{MRQ} \leftarrow getId_{MRQ}(future)$ ▷ Position of the most relevant query, MRQ
$r[1] \leftarrow future[id_{MRQ}]$
$\lambda \leftarrow 0$
$\lambda^0_{score} \leftarrow MRQ.relevance$
$id_{left} \leftarrow id_{MRQ} - 1$ ▷ Position of the left query from MRQ
$id_{right} \leftarrow id_{MRQ} + 1$ ▷ Position of the right query from MRQ
while $\lambda <= \lambda_{gap}$ **do**
  $q_{adjacent} \leftarrow max_{relevance}(future[id_{left}], future[id_{right}])$
  $\lambda^t_{score} \leftarrow avgRelevance(r, adjacent])$ ▷ Average relevance including $q \in r$ and $q_{adjacent}$
  $\lambda \leftarrow \lambda^{t-1}_{score} - \lambda^t_{score}$
  if $\lambda <= \lambda_{gap}$ **then**
    if $q_{adjacent}$ *is* $future[id_{left}]$ **then**
      $r.add(1, q_{adjacent}])$ ▷ Adding the query at the beginning of the sequence r
      $id_{left} \leftarrow id_{left} - 1$
    else
      $r.add(q_{adjacent})$ ▷ Adding the query at the end of the sequence r
      $id_{right} \leftarrow id_{right} + 1$
return r ▷ Return the base recommendation

Example 3.3.1. *Each query of each possible future is scored in order to identify densest area into the log. The query alignment scores between* $future_1$ *and* $future_2$ *is given in Table 3.5. The details for each query of each future are given in Tables 3.5 and 3.6, respectively. Thus, the average score given to* $future_1$ *is 0.889 whereas the score of* $future_2$ *is 0.444. Consequently,* $future_1$ *is selected to be the base recommendation. Note that the Recommendation Length step returns* $\langle q_{13}, q_{14} \rangle$.

3.4 Tailoring the Base Recommendation

The goal of this phase is to adapt the selected future to the current session. Such an adaptation is achieved by constructing profiles for the current session and the log session used to select the future, and to use these profiles to transform the queries of the base recommendation. To do so, we adapt the technique of [Aligon et al., 2011] and extract

Table 3.4: Query Alignment Scores between $future_1$ and $future_2$ for Example 3.3.1

	q_{16}	q_{17}	q_{18}	q_{19}
q_{13}	-	-	0.889	-
q_{14}	-	-	-	0.889

Table 3.5: Score for each query of $future_1$ for Example 3.3.1

	q_{13}	q_{14}	**Average**
Score	0.889	0.889	**0.889**

associations rules from the current session and the log session, to constitute the profiles. We define two types of rules. The first type of rules represents the differences between the current session and the log session, while the second type of rules models the user's behavior during the current session. These rules are then applied to each query of the recommended session, to modify this query. More precisely, the rules modify the fragments that are shared by all queries of the recommended session. This ensures that the adaptation phase does not produce two identical queries into the recommended session and respects the browsing logic of the recommended session. Type 1 rules are applied first, then type 2 rules are applied. This principle is described by Algorithm 6, and we detail below the two types of rules.

3.4.1 Extraction of Association Rules of Type 1

Type 1 rules aim at turning a fragment of the log session into a fragment of the current session, provided there exists a strong correlation between the two fragments. Thanks to session alignment, each query q_i of the current session is aligned with a query q_j of the log session. The set of the so-formed couples (q_i, q_j), called C, is mined for association rules of the form $x \rightarrow y$ with x and y fragments (of the same type, i.e. either measures, levels or selection predicates on the same hierarchy), x in the log session and y in the current session c_s. The extracted rules are ranked according to the geometric mean of the following values:

1. the support value of the rule, i.e. $supp(x \cup y) = \frac{|\{(q_s, aligned(q_s)) | q_s \in s_{curr}, aligned(q_s) \in l, x \in aligned(q_s), y \in q_s\}|}{|C|}$
2. the confidence value of the rule, i.e. $conf(x \rightarrow y) = \frac{supp(x \cup y)}{supp(x)}$
3. the average position in the current session where the head fragment y appears (to favor recent fragments),
4. the support value of the head fragment y into the current session (to favor frequent fragments).

The intuition underlying these rules is to substitute the fragment x presents in a query of the base recommendation by fragment y.

The pseudocode applying the rules of type 1 is given in Algorithm 6 (lines from 4 to 11).

Table 3.6: Score for each query of $future_2$ for Example 3.3.1

	q_{16}	q_{17}	q_{18}	q_{19}	**Average**
Score	0	0	0.889	0.889	**0.444**

Type 1 rules are applied in descending order, to each query of the recommended session, only on the body fragments x that do exist in all the queries of the base recommendation, but that do not exist in the current session. If these conditions hold, this fragment is replaced by y (lines from 5 to 9). Every modified fragment is marked so as not to be adapted any more (line 10).

Considering the base recommendation obtained in Example 3.3.1, the *Tailoring* phase is applied in order to adapt $future_1$ to the current session s_{curr}.

Example 3.4.1. *Let $\mathcal{R}_{Type1}$ be the ordered set of association rules of Type 1. Only a sample of these rule set is given below. Note that the score given for each rule is the geometric mean of support and confidence values but also position and support value of the head fragment in the current session.*

$$\begin{array}{lll} rule_1: & \mathsf{AllCities} \rightarrow \mathsf{AllOccupation} & (0.93) \\ rule_2: & \mathsf{AvgCostWatr} \rightarrow \mathsf{AvgCostElect} & (0.93) \\ rule_3: & \mathsf{Year} \in \{2005\} \rightarrow \mathsf{Year} \in \{2002\} & (0.93) \\ rule_{28}: & \mathsf{RaceGroup} \rightarrow \mathsf{RaceGroup} & (0.70) \end{array}$$

The rules are applied in descending order, for each query of $future_1$. The set of fragments shared by all the queries of the base recommendation is $Fr = \{\mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{Sex}, \mathsf{AllOccs}, \mathsf{Year} \in \{2005\}, \mathsf{Sex} \in \{Female\}, \mathsf{AvgCostWtr}\}$.

Regarding q_{13}, $rule_1$ cannot be applied because the hierarchies of the body and head fragments are not the same (respectively RESIDENCE *and* OCCUPATION *hierarchies). Rule $rule_2$ could be applied since the body fragment* AvgCostWatr *exists in q_{13}. But* AvgCostWatr *also exists in the current session. Consequently, $rule_2$ is not applied. The body fragment* $\mathsf{Year} \in \{2005\}$ *of r_3 matches q_{13}, therefore* $\mathsf{Year} \in \{2005\}$ *is replaced by* $\mathsf{Year} \in \{2002\}$. *Finally, $rule_{28}$ is not considered because the body and head fragments (*RaceGroup*) are the same.*

Regarding q_{14}, again only rule $rule_3$ is applied.

Consequently, the adapted recommendation, named r', becomes at this step:

$$\begin{aligned} r' = \langle q'_{13} = \langle \langle &\mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{Sex}, \mathsf{AllOccs} \rangle, \\ &\{TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2002\}, \\ &\quad TRUE_{OCCUPATION}, \mathsf{Sex} \in \{Female\}\}, \\ &\quad \{\mathsf{AvgCostWrt}\} \rangle \\ q'_{14} = \langle \langle &\mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{Sex}, \mathsf{AllOccs} \rangle, \\ &\{TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2002\}, \\ &\quad TRUE_{OCCUPATION}, \mathsf{Sex} \in \{Female\}\}, \\ &\quad \{\mathsf{AvgCostWrt}, \mathsf{AvgIncome}\} \rangle \rangle \end{aligned}$$

3.4.2 Extraction of Association Rules of Type 2

Type 2 rules aim at finding "invariants" of the current session c_s, i.e. fragments used frequently in the current session. The rules have the form $X \rightarrow y$ with X a set of fragments and y a fragment. The extracted rules are ranked according to the geometric mean of following values:

1. the support value of the rule, i.e. $supp(X \cup y) = \frac{|q_s | q_s \in s_{curr}, X \cup \{y\} \subseteq q_s\}|}{|c_s|}$
2. the confidence value of the rule, i.e. $conf(X \rightarrow y) = \frac{supp(X \cup y)}{supp(X)}$
3. the average position in the session where the head fragment appears (to favor recent fragments).
4. the support value of the head fragment y into the current session (to favor frequent fragments).

The intuition underlying these rules is to identify the queries of the base recommendation matching with the body fragments X and to add the head fragment y into these queries.

The pseudocode applying the rules of type 2 is given in Algorithm 6 (lines from 12 to 21).

Type 2 rules are applied in descending order, to each query of the recommended session, only if rule's body X is included in the set of fragments of the query. If the rule's head y is a selection predicate or a level and is not already present in the query, then it is substituted to the corresponding fragment (i.e., of the same hierarchy) of the query (lines from 13 to 18). If y is a measure that is not already present in any query of the base recommendation, then it is added into the measure set of the query. This solution allows to avoid identical queries in a same recommended session (lines from 20 to 21). As for type 1 rules, once a fragment is modified, it is marked so as not to be modified any more (line 22).

Considering the recommendation, obtained in Example 3.4.1 and tailored with rules of type 1, we now apply the rules of type 2.

Example 3.4.2. *Let $\mathcal{R}_{Type2}$ be the ordered set of association rules of Type 2. Only a sample of these rule set is given below. Note that the scores given for each rule is the geometric mean of support and confidence values but also position of the head fragment in the current session.*

$$
\begin{array}{ll}
rule'_1: & \mathsf{Year} \in \{2002\} \rightarrow \mathsf{Year} \quad (0.93) \\
rule'_2: & \mathsf{AllCities}, \mathsf{Year} \rightarrow \mathsf{AvgCostElect} \quad (0.93) \\
rule'_3: & \mathsf{AvgCostWtr}, \mathsf{Year} \rightarrow \mathsf{AllSexes} \quad (0.93) \\
rule'_4: & \mathsf{AllOccs} \rightarrow \mathsf{AvgCostElect} \quad (0.93)
\end{array}
$$

As explained in Section 3.4.2, the rules are applied in descending order, for each query of r'.

Regarding q'_{13}, $rule'_1$ cannot be applied because the head fragment Year *already exists in q'_{13}. $rule'_2$ is applied because the body fragments* AllCities *and* Year *match with q'_{13} and the measure* AvgCostElect *does not exist for each query of r'. Consequently, the measure* AvgCostElect *is added into q'_{13}. The body fragments* AvgCostWtr *and* Year *of $rule'_3$ match*

Algorithm 6 Tailoring

INPUT: s_{curr}: the current session, l: the log session, r: the base recommendation,
OUTPUT: $r' = r$ adapted to c

```
 1: r' ← r
 2: T1 ← extractType1Rules(s_cur, l)                          ▷ Extract rules
 3: T2 ← extractType2Rules(s_cur)
 4: Fr ← ⋂_{q∈r'} q
 5: while Fr ≠ ∅ and T1 ≠ ∅ do                                ▷ Apply Type 1 rules
 6:     (x → y) ← max_rank(T1)
 7:     for i = 1 to length(r') do
 8:         if x ∈ Fr ∩ r'[i] and ∀q_c ∈ s_cur, x ∉ q_c then
 9:             r'[i] ← r'[i] \ {x} ∪ {y}
10:     Fr ← Fr \ {x}
11:     T1 ← T1 \ {x → y}
12: while Fr ≠ ∅ and T2 ≠ ∅ do                                ▷ Apply Type 2 rules
13:     (X → y) ← max_rank(T2)
14:     if y is a level or a selection predicate then
15:         if ∃z ∈ Fr corresponding to y then
16:             v ← z
17:     for i = 1 to length(r') do
18:         if X ⊆ r'[i] then
19:             if v = z and v ∈ r'[i] then
20:                 r'[i] ← r'[i] \ {v} ∪ {y}
21:             else
22:                 if ∀q_r ∈ r', y ∉ q_r' then
23:                     r'[i] ← r'[i] ∪ {y}
24:     Fr ← Fr \ {v}
25:     T2 ← T2 \ {x → y}
        return r'
```

with q'_{13}. *Thus, the head fragment* AllSexes *replaces the corresponding fragment in the query, that is the level* Sex. $rule'_4$ *cannot be applied due to the fact that the measure* AvgCostElect *has already been added by the rule* $rule'_2$.

Regarding q'_{14}, *as for* q'_{13}, *the rules* $rule'_2$ *and* $rule'_3$ *are applied.*

Consequently, the final recommendation is:

$$\begin{aligned}\langle q''_{14} = \langle\langle &\mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{AllOccs}\rangle,\\ &\{TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2002\},\\ &\quad TRUE_{OCCUPATION}, \mathsf{Sex} \in \{Female\}\},\\ &\quad \{\mathsf{AvgCostWtr}, \mathsf{AvgCostElect}\}\rangle\\ q''_{15} = \langle\langle &\mathsf{AllCities}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{AllOccs}\rangle,\\ &\{TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2002\},\\ &\quad TRUE_{OCCUPATION}, \mathsf{Sex} \in \{Female\}\},\\ &\quad \{\mathsf{AvgCostWtr}, \mathsf{AvgCostElect}, \mathsf{AvgIncome}\}\rangle\rangle\end{aligned}$$

As we can see, this final recommendation preserves important invariants of the current session such as the study over the selection $\mathsf{Year} = \{2002\}$ *and measures* AvgCostWtr *and* AvgCostElect. *Moreover, the recommendation also proposes new focus coming from the log session, such as selection* $\mathsf{Sex} \in \{Female\}$ *or measure* AvgIncome.

3.5 Conclusion

We showed in this chapter the principle of the recommendation system. Three phases compose the system. The first one aligns the log sessions with the current session using the Smith-Waterman algorithm. A new version of the sigmoid function, promoting alignments between the end of a current session and the beginning of the log sessions, is proposed. The queries following the aligned subsessions of the log sessions are considered as potential futures to recommend. The second phase ranks each future by identifying densest areas of similar queries in the log sessions. The future having the best score is selected for extracting the base recommendation whose length is determined dynamically. The last phase adapts the base recommendation by modifying or adding fragments in its queries. The queries are modified using two types of association rules respectively extracted from:

- the log session (whose base recommendation comes from) and the current session.
- only the current session in order to identify fragments used frequently in the current session.

A running example has also been presented in this chapter and motivates our approach by clearly showing that the *SROS* system can preserve frequent fragments of the current session while adding new fragments. However, the quality of the recommended sessions has to be assessed in order to answer to the quality criteria expressed in Section 1.5.1. This is discussed in Chapter 4.

Chapter 4

Assessing the quality of the recommender system

This chapter reports the experiments assessing the relevance of the *SROS* recommendation approach proposed in this dissertation (see Chapter 3).

A part of this chapter is also devoted to obtaining OLAP query logs. Indeed, for assessing the relevance of history based user-centric solutions (our recommendation system and our proposals of various similarity measures), the use of former OLAP sessions is essential. In particular, "objective" generations of logs are proposed in Section 4.1 by producing a set of synthetic sessions imitating user behaviors. Section 4.1.2 discusses on a "subjective" generation of logs developed by Master's students in Business Intelligence.

Section 4.2 describes the tests conducted with the similarity measures for queries and sessions (formalized in Sections 2.2 and 2.3). The aim of these tests is to find the best measure, between queries and sessions, to use in *SROS* and respecting the requirements given in Section 1.5.2. In particular, subjective tests, based on user questionnaires, indicate how OLAP session similarities are perceived by users. The objective tests show the behaviors of our similarity measures through different types of session templates, for assessing the capabilities of each of them and answering to the requirements.

Section 4.3 focuses on the tests of our recommendation system using the best similarity measure identified previously. These tests assess the efficiency and the effectiveness of *SROS*, based on recommendation criteria listed in Section 1.5.2.

4.1 Getting Logs

The use of OLAP sessions, conducted by professional analysts, is certainly the best way to assess the relevance of OLAP solutions based on former queries (in particular with user-centric approaches, like recommendation or personalization of queries). However, for academic research teams, obtaining such logs is often difficult since these data can have a strategic interest for a company. Consequently, the access to these data is limited. Filtering such data could be an option, but the risk is to denature the analysis that are devised.

To cope with this, we report in this section two different approaches for obtaining

OLAP sessions and logs, allowing to asses our recommendation system.

The first approach, presented in Section 4.1.1, focuses on synthetic data generators and proposes different solutions for obtaining various analysis behaviors adhering to pre-defined templates. The second approach, described in Section 4.1.2, reports a feedback about a test conducted with Master's students in order to obtain non synthetic logs.

4.1.1 Synthetic Data Generation

Synthetics logs generation allows to generate sessions from specific behaviors such as the number of OLAP operations separating two successive queries or, in a more global perspective, a sequence of queries reaching a particular goal. This allows to assess user-centric solutions with objective criteria. Section 4.1.1.1 gives a first proposal of generation, based on the *Shortest OLAP Path*, which is the basis of the other generators described in Sections 4.1.1.2 and 4.1.1.4. Section 4.1.1.3 describes process based on the query result, unlike the previous generators that are based on the query expression.

4.1.1.1 Shortest OLAP Path

The log generation principle by *Shortest OLAP Path* produces a session starting from an initial query and ending in a final query, both obtained by randomly choosing a group-by set, a selection predicate in each hierarchy, and a subset of measures. Intermediate queries are then generated by applying, one at a time in a random order, the minimal atomic OLAP operations that transform the initial query into the final one.

The OLAP operations are those described in Section 2.1.2.2 that are: change level along one hierarchy in the group-by set, add or remove a selection value in a predicate, and add or remove a measure.

Example 4.1.1. *Let q_1 and q_4 be the initial and final queries respectively:*

$$\begin{aligned} q_1 = \langle\langle &\mathsf{State}, \mathsf{Race}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{Occ}\rangle, \\ \{&TRUE_{RESIDENCE}, TRUE_{RACE}, \mathsf{Year} \in \{2005\}, \\ &TRUE_{OCCUPATION}, TRUE_{SEX}\}, \\ &\{AvgCostWtr, AvgCostElect\}\rangle \\ q_4 = \langle\langle &\mathsf{State}, \mathsf{RaceGroup}, \mathsf{Year}, \mathsf{AllSexes}, \mathsf{Occ}\rangle, \\ \{&TRUE_{RESIDENCE}, \mathsf{RaceGroup} \in \{Chinese\}, TRUE_{TIME}, \\ &TRUE_{OCCUPATION}, TRUE_{SEX}\}, \\ &\{AvgCostWtr\}\rangle \end{aligned}$$

Figure 4.1 illustrates one possible path of operations between q_1 and q_4. In this example, four operations are possible from q_1 to q_4:

- *Applying a Rollup operation over level* Race
- *Removing measure* AvgCostElect
- *Adding selection* RaceGroup $\in \{Chinese\}$
- *Removing selection* Year $\in \{2005\}$

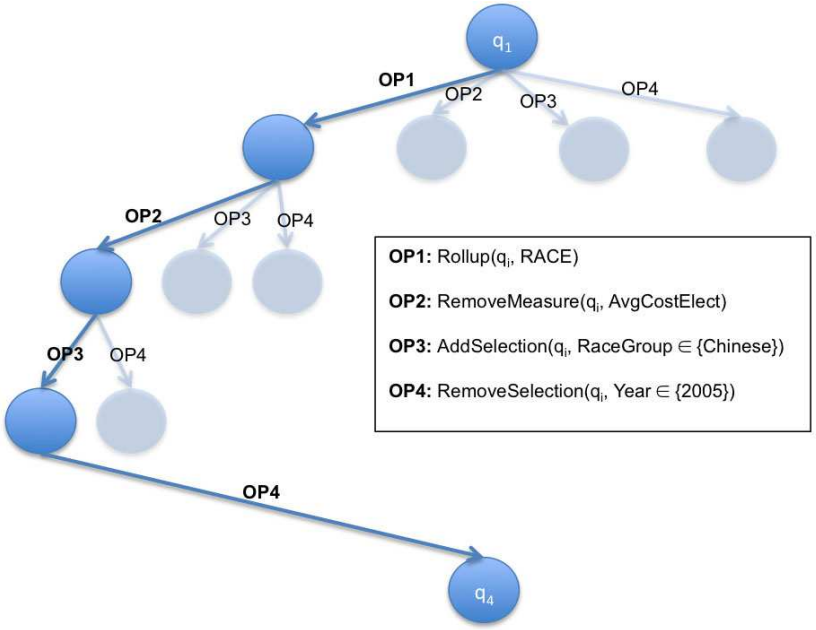


Figure 4.1: *Shortest OLAP Path* principle of Example 4.1.1

4.1.1.2 Behavior Generation

The *Behavior Generation* uses two different types of templates, each of them modeling a particular analysis behavior.

The first one produces *Explorative* sessions. As depicted in Figure 4.2, a first *Initial Query* is obtained randomly. Then, another random query is generated and considered as a *Surprising Query*. A path between the *Initial Query* and the *Surprising Query* is created using the *Shortest OLAP Path* principle (see Section 4.1.1.1). Finally, the obtained sequence deviates to a random direction by applying a sequence of random OLAP operations, creating a new query for each operation, until reaching the desired number of queries.

The second template generates *Goal Oriented* sessions. As depicted in Figure 4.3, a first *Initial Query* is obtained randomly. Then, the *Shortest OLAP Path* principle is used to reach a *Final Query*, randomly generated. If the length of the session is too short (specified by a parameter), a random deviation is added between queries of the session. The deviation produces a sequence of random OLAP operations that each one forms a query.

4.1.1.3 Mining Query Result

Another approach of synthetic generation, based on the query results, is given below. This principle is only an example illustrating the generation of logs representing better a discovery driven analysis.

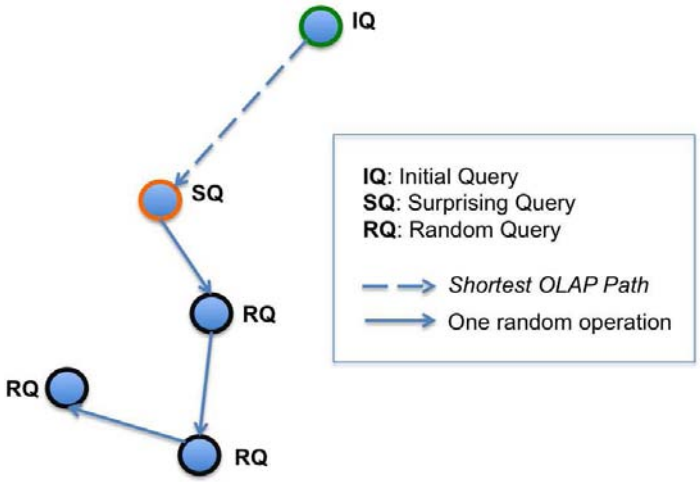


Figure 4.2: Explorative Template

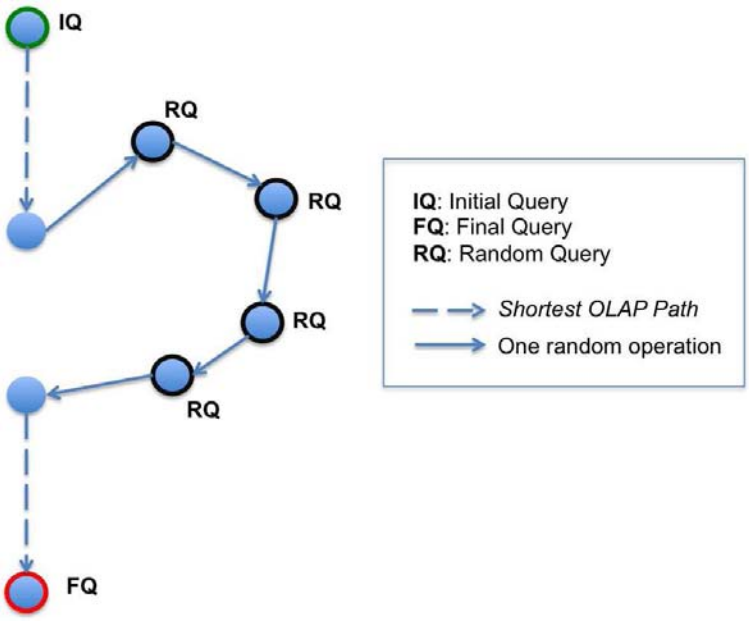


Figure 4.3: Goal-Oriented Template

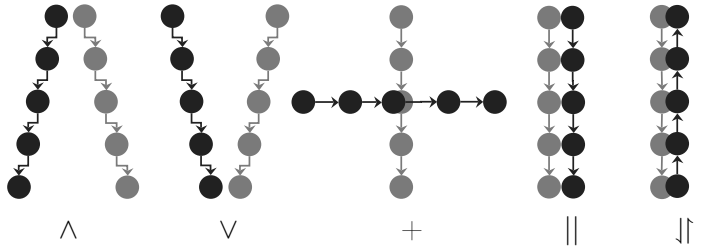


Figure 4.4: The templates used to generate sessions. Overlapping circles represent identical queries, near circles represent similar queries. For template ||, the queries are pairwise separated by one atomic OLAP operation

Note that this process of log generation based on the answers of OLAP queries has been detailed and implemented in [Negre, 2009] and uses in [Aligon et al., 2011]. It aims at simulating analysis behavior. The proposal has then been adapted for managing different multidimensional schema. The overall principle is the following.

- A first initial query is randomly generated and executed,
- A random choice between the $Diff$ and $Relax$ operators (described in [Sarawagi, 1999] and [Sathe and Sarawagi, 2001]) is applied on the query answer. These operators can automatically explore a cube by a sequence of drill-downs ($Diff$) or roll-ups ($Relax$), identifying interesting results of the query,
- A new query is created for each interesting results,
- Repeat the process until reaching the desired number of sessions.

4.1.1.4 Session-Pair Generation

This generator produces pairs of sessions according to specific templates. Each template models intuitive notions of what similar sessions might look like. These different templates are used in Section 4.2 for testing the similarity measures between sessions proposed in Section 2.3.

A session is generated according to the *Shortest OLAP Path* principle, described in Section 4.1.1.1.

A pair of sessions is generated using one of the five templates depicted in Figure 4.4:

- In template ∧, the two sessions have similar starting queries then they diverge to radically different queries.
- In template ∨, the two sessions have radically different starting queries then they converge to similar ending queries.
- In template +, the two sessions converge to the same query then they diverge.
- In template ||, the second session is constructed by "shifting" all queries in the first session by one OLAP operation.
- In template ⇃↾, the two sessions have the same queries in reverse order.

Algorithm 7 SPaG Generation

Input: t: a template, m: the number of session sets to generate, n: the number of sessions to generate for a set;
Output: L: the log;

1: $L \leftarrow \emptyset$
2: **while** $|L| <= (m * n)$ **do**
3: Generate a pair of sessions, s and s', that respect t
4: $S \leftarrow \{s, s'\}$
5: **while** $|S| <= n$ **do**
6: $q_{initial} \leftarrow derive(s[0], 3)$ ▷ Query derived from $s[0]$ applying 3 OLAP operations
7: $q_{final} \leftarrow derive(s[|s|], 3)$
8: $s'' \leftarrow shortestOLAPPath(q_{initial}, q_{final})$
9: $S \leftarrow S \cup \{s''\}$
10: $L \leftarrow L \cup S$
11: **return** L

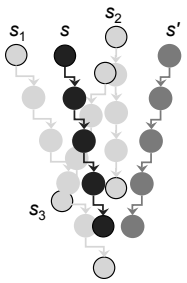


Figure 4.5: The seed session s (in black), its mate s' according to template $\wedge$ (in dark gray), and three random sessions (in light gray). The first and last queries of sessions are circled.

4.1.1.5 Log Generation

From the session generations described in the previous sections, different principles of log generation can be stated.

The first log generation, abbreviated *SPaG*, is based on the *Session-Pair* generation principle described in Section 4.1.1.4. The pseudo-code is given in Algorithm 7. Given one of the five templates, we generate a log as follows (see also Figure 4.5 for an example). We first generate a pair of sessions, s and s', that respect the template (line 3). Then, n more sessions $s_1, \ldots s_n$ are generated using s as a seed. The first and the last query of s_i are obtained by applying n_{op} random OLAP operations to the first and the last query of s, respectively (lines 6 and 7); then, the intermediate queries of s_i are generated using the *shortest OLAP Path* principle (line 8, see Section 4.1.1.1).

The second one, shortly named *BeG*, is based on the *Behavior* generation principle given in Section 4.1.1.2. The pseudocode is given in Algorithm 8. The log is generated as follows. Random queries are computed to obtain a set of *Initial Queries* (*IQ*), *Final Queries* (*FQ*) and *Surprising Queries* (*SQ*) respectively (lines from 2 to 4). Then, k sessions are generated from each *IQ*. A random template is also chosen between *Explorative* and *Goal-Oriented*. A random length of session is also chosen between 10 and 20 (lines 9 and 10). Finally, the session is generated considering a random surprising or final query depending on the template to apply (lines 11 to 16).

Algorithm 8 BeG Generation

Input: n: number of initial queries, m: number of final queries, l: number of surprising queries, k: number of sessions for each initial query;
Output: L: the log;

$L \leftarrow \emptyset$
$Q_{IQ} \leftarrow n$ *random initial queries*
$Q_{FQ} \leftarrow m$ *random final queries*
$Q_{SQ} \leftarrow l$ *random surprising queries*
for each $q \in Q_{IQ}$ **do**
  $S \leftarrow \emptyset$
  while |S| <= k **do**
    $s \leftarrow \emptyset$
    $template \leftarrow random(Explorative, Goal\text{-}Oriented)$ ▷ A random template
    $lengthSession \leftarrow random(10, 20)$ ▷ A random length for s between 10 and 20
    if template is $Explorative$ **then**
      $q' \leftarrow randomQuery(Q_{SQ})$ ▷ A random query $q \in Q_{SQ}$
      $s \leftarrow explorative(q, q', lengthSession)$
    else
      $q' \leftarrow randomQuery(Q_{FQ})$ ▷ A random query $q \in Q_{FQ}$
      $s \leftarrow goal\text{-}oriented(q, q', lengthSession)$
    $S \leftarrow S \cup \{s\}$
  $L \leftarrow L \cup S$
return L

4.1.2 Gathering Real Logs

Logs produced by users allow to consider subjective criteria when sessions are devised (for instance sessions answered to questions having different difficulty degrees). We report in this section a feedback from real OLAP sessions developed by Master's students in Business Intelligence. By proposing a test conducted with Master's students, we can intuitively think that the analysis sessions are necessarily of lower quality than sessions provided by professional analysts. However, [Runeson, 2003] supposes that graduate students can devise analysis sessions as good as experimented analysts. Indeed, [Runeson, 2003] showed that graduate students can devise similar sessions to industry people but more investigation is needed since the study has not enough data to confirm the trend. In contrast, he clearly demonstrated that freshmen students are not good candidates. In the context of relational databases, a test in [Khoussainova et al., 2011] has already been conducted with students to demonstrate that browsing through past SQL query sessions helped speed up query composition.

This feedback reports the design of questionnaires (Section 4.1.2.1) and the use of an original user interface to easily conduct OLAP sessions (presented in Section 4.1.2.2). Statistical results on the log obtained and a work about log filtering are also presented in Section 4.1.2.3.

4.1.2.1 Design of the Questionnaires

We describe here the design of the questionnaires [1] which the students have to answer, for producing several analysis sessions. The questionnaires are based on the CENSUS schema, given in Example 1.3.4.

1. All questionnaires are available at http://www.julien.aligon.fr/index.php/research-activities/real-olap-logs/#test

As a reminder, the CENSUS multidimensional schema has five hierarchies, namely RACE, TIME, SEX, OCCUPATION, and RESIDENCE, and measures (aggregated either by Sum, Max, Min or Avg) Income, PropInsr (property insurance cost), PerWt (person weight), CostGas, CostWtr, and CostElect. The complete roll-up orders are shown in Figure 1.3.

Different requirements have to be expressed in the questionnaires in terms of diversity and complexity of analysis. Thus, three types of needs have been proposed:

- the *individual profile* analysis. A profile is defined as the combination of SEX and RACE hierarchies, i.e. the study requires a crossed analysis between the levels of SEX and RACE.
- the OCCUPATION analysis.
- the *mixed* analysis, i.e. an analysis is not specifically related to an *individual profile* or *occupation*.

For each analysis described previously, two different measures, can be analyzed:

- the Income measure (measuring the personal income)
- the energy measures (i.e CostGas, CostWtr and CostElect)

Consequently, 6 questionnaires have been designed for the tests with the students. For each of these questionnaires, different versions (4 or 5) have been designed including different questions.

In particular, each questionnaire includes 5 questions, divided in three levels of difficulties:

- Basic needs (2 questions). For this level, the needs are explicitly given. For instance, a question is: *Is there a trend in the evolution of the average cost of gas for some profiles?*
- Intermediate needs (2 questions). For this level, the needs are less explicit than the basic needs but not too complex. A question is: *Compare the evolution of the minimum of energy costs, for the highest income, with the evolution of the maximum energy costs for the lowest incomes.*
- Advanced needs (1 question). For this level, the needs are deliberately fuzzy. A question is: *Where is it better to live in terms of incomes, for an occupation?*

A total of 26 different questions have been asked among all the questionnaires, 11 for basic needs, 9 for intermediate needs, 6 for advances needs.

4.1.2.2 Graphical User Interface for OLAP Session Design

We now present the user interface for querying an OLAP cube, that is required for especially taking into account the query model used in our recommendation approach. We can note that few works have focused on the combination between the HCI and structured data (see [Li and Jagadish, 2012] and [Nandi and Jagadish, 2011]). The Graphical User Interface is depicted in Figure 4.6.

Because the students are not familiar with a particular language (like the MDX language) for designing queries, we chose to abstract this by implementing a user interface allowing to graphically design OLAP queries. This functionality can be seen in part 1 of Figure 4.6. The interface is inspired by the Dimension Fact Model (DFM, detailed in [Golfarelli and Rizzi, 2009]). It allows to design a query respecting the formal model defined in Section 2.1.1. A group-by set is created by linking a level between each hierarchy.

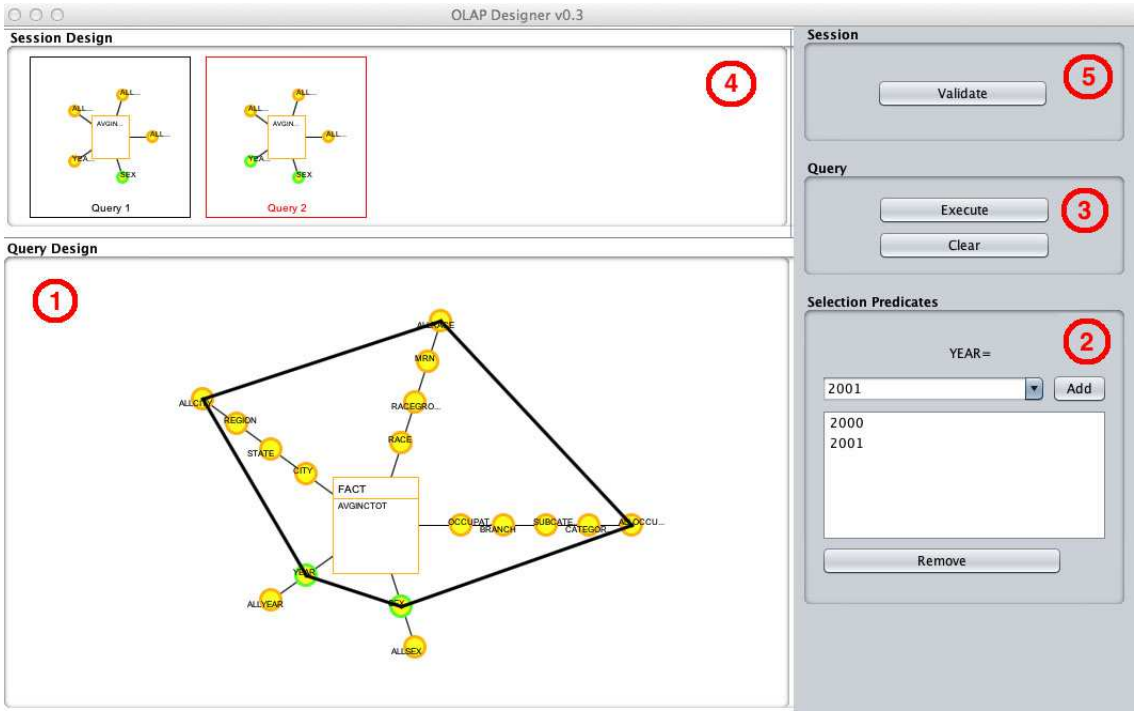


Figure 4.6: User Interface for Designing OLAP Sessions

A selection predicate can be added by selecting a level and the desired values (part 2 of Figure 4.6). The same principle allows to add the measures. When a query is designed, the user has to execute it (part 3 of Figure 4.6). The query result is displayed to the user and the query is added to the current session (part 4 of Figure 4.6). Once the user considers his need is answered, he validates her session (part 5 of Figure 4.6) which is automatically added in the log. Note that we assume that at least three queries are needed to form a session.

4.1.2.3 Characteristics of the logs obtained

In this section, we discuss the characteristics of the logs[2] obtained from the tests conducted by the Master's students in Business Intelligence from the University Francois Rabelais of Tours and the University of Bologna. 40 students participated to the tests (18 from France and 22 from Italy) by answering to a questionnaire. We present the characteristics of the sessions obtained but also of the query components (i.e. the elements of the group-by set, measure set or selection set, see Definition 5), named *fragment*.

The log is composed of 810 queries, distributed among 182 sessions (85 from France and 97 from Italy). Each questionnaire has been answered 4 or 5 times. Figure 4.7 shows the number of sessions of the log for each complexity of question. We can note that the number of sessions for the advanced questions is half as large as the basic or intermediate

2. All the results are in http://www.julien.aligon.fr/index.php/research-activities/real-olap-logs/#test

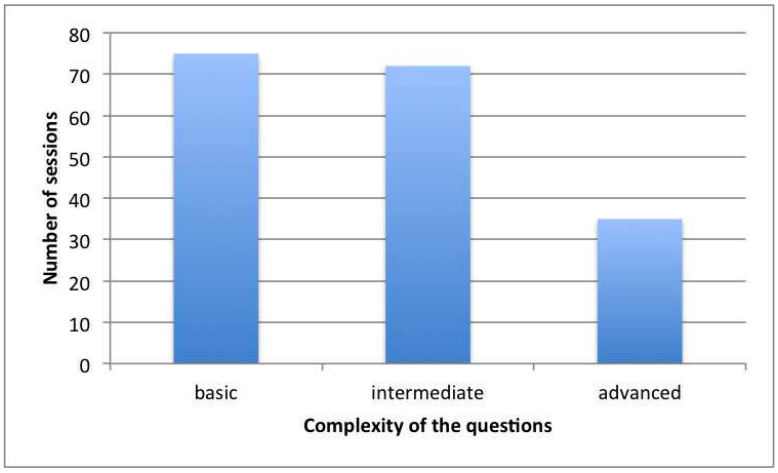


Figure 4.7: Number of sessions per complexity of questions

questions. This is simply because the number of advanced questions in the questionnaires is less important than the others.

Figure 4.8 reports the average number of queries for each complexity of question. For each level of difficulty, the average number seems low but similar between them. This result can seem strange when, intuitively, we could think that the more difficult the question is, the more important the number of queries should be. An answer could be that the advanced questions were too difficult for the students or the questionnaires were too long (tiring the students). Another answer could be that the students do not understand the difficulty behind the advanced questions.

Figure 4.9 indicates the average time for designing the sessions for each complexity of question. We can see that the time for designing the sessions for the basic questions is the highest. This is due to the fact that the basic questions were the first conducted by the students and a period of adaptation to the tool and the exercise was probably necessary. The low period of time for devising the answers to the advanced questions seems to confirm the previous comment about Figure 4.8.

We describe below more detailed statistics about fragments of queries (the elements included in the group-by set, measure set or selection set).

Figure 4.10 refers to the number of fragments for each complexity of question. We can notice that the number of fragments for the advanced questions is very low compared to the others. Consistently with the results of Figures 4.8 and 4.9, we can assume that the advanced questions were less well addressed than others.

Figure 4.11 shows the number of fragment types. We notice that there are more projection fragments than measures. In the same way, there are more measure fragments than selections. These disproportions seem expected. Indeed each OLAP query must include a group-by set (in our case, composed with 5 levels) and at least one measure (if the user does not specify a measure, the default measure is used) whereas the selection set can be left empty.

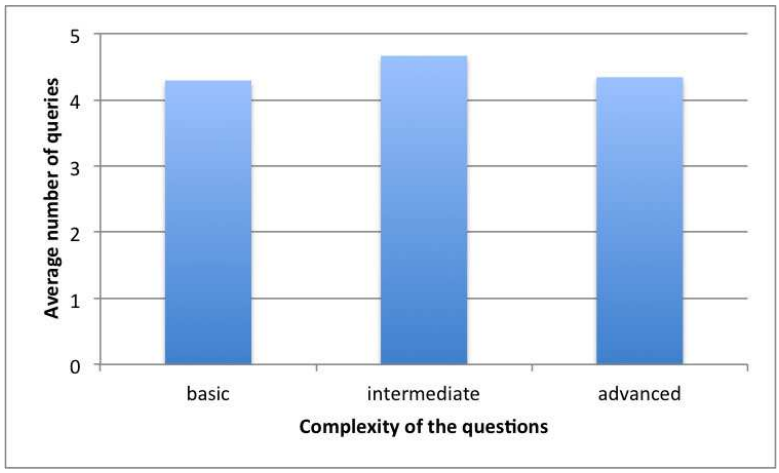


Figure 4.8: Average number of queries per complexity of questions

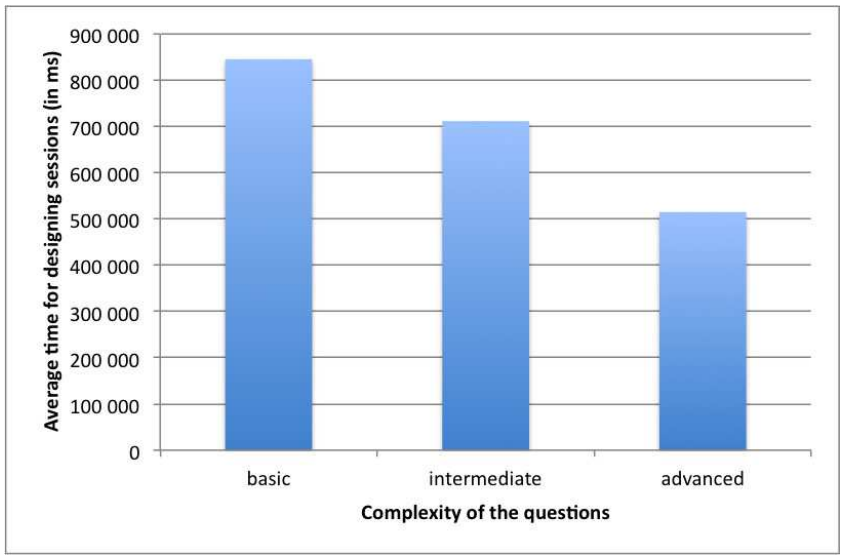


Figure 4.9: Average time per complexity of questions

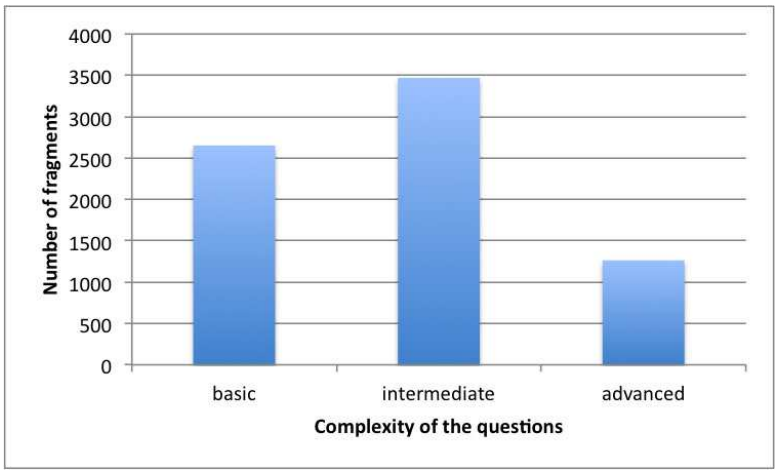


Figure 4.10: Number of fragments per complexity of questions

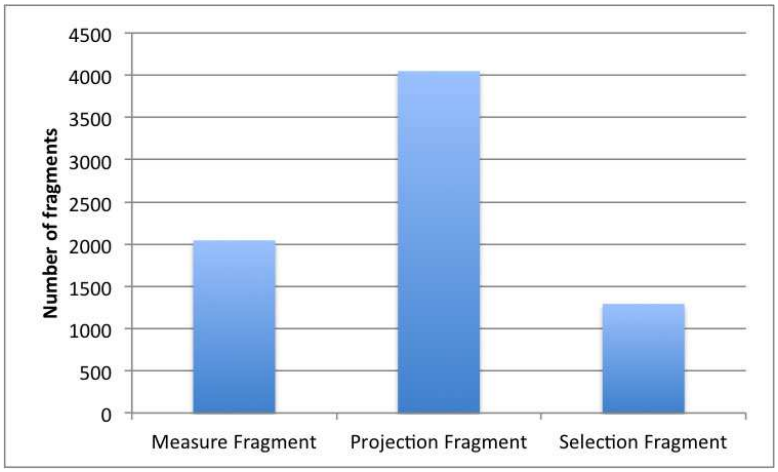


Figure 4.11: Number of fragment type

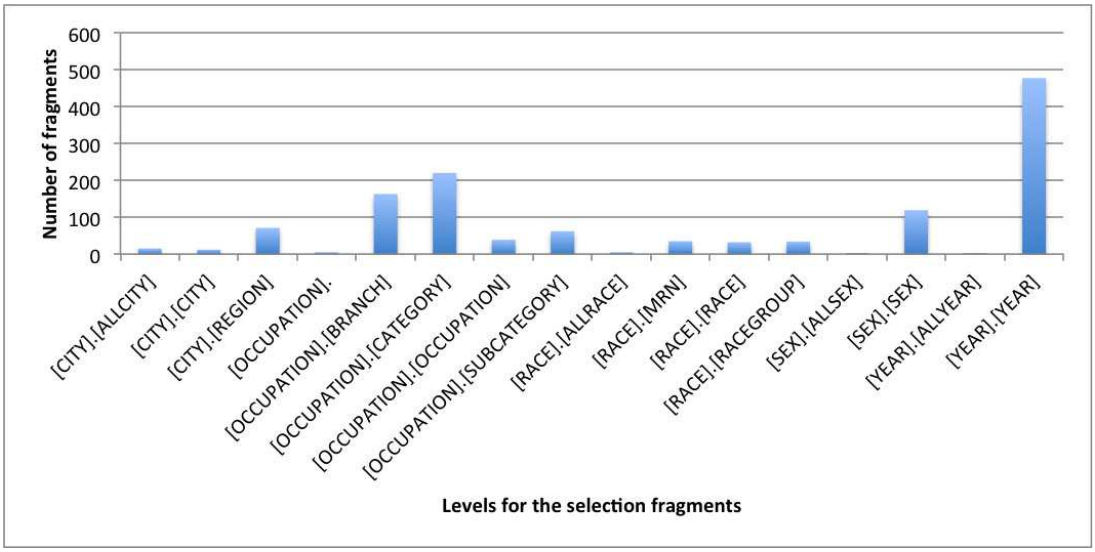


Figure 4.12: Number of fragments per level of selections

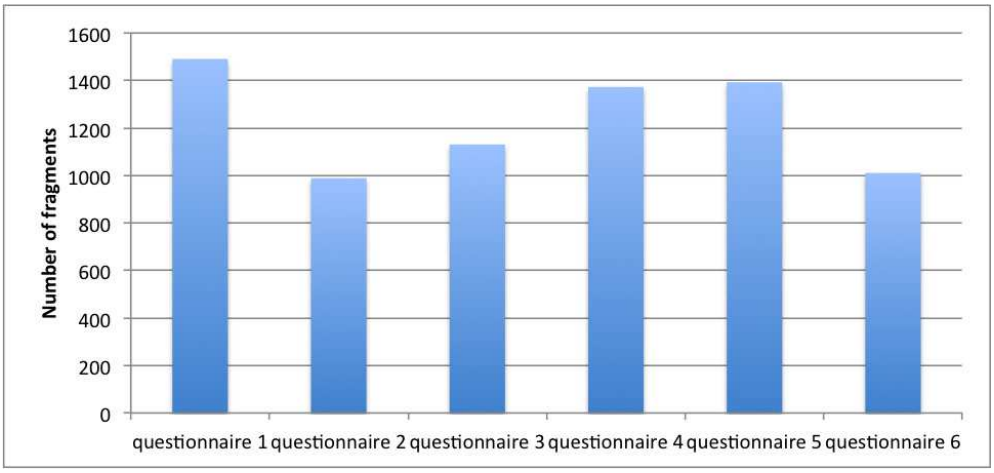


Figure 4.13: Number of fragments per questionnaires

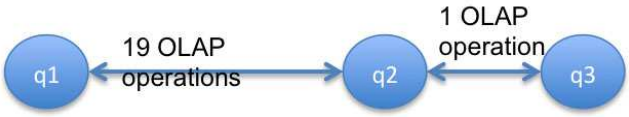


Figure 4.14: Session devised by a student with high variations of OLAP operations.

These results show that some sessions can potentially not meet the needs expressed in the questionnaires. Therefore, it is interesting to identify and remove these sessions in order to avoid to use them in our *SROS* system (which could propose no relevant recommendations). We note two problems identified in the logs:

– Problems in composing the session.
– A problem in composing the query.

Regarding the problems in composing the session, two behaviors have been identified. The first one is that students devised identical successive queries in a session. A solution could be to delete them since these queries do not provide new information. Among the student logs, 47 sessions include repetition of queries (about 25% of the sessions). Removing these queries may produce sessions having less than 3 queries, violating the constraint expressed in Section 4.1.2.2. Thus, these sessions have to be dropped. This affects 25 sessions (about 14% of the sessions) among all the logs. The second identified behavior is that high variations of OLAP operations between successive queries are possible (an example of session with such high variation is given Figure 4.14). To identify variations, a solution is to compute the minimal number of OLAP operations that transform a given query into its next query (according to the principle given in Section 4.1.1.1) and to compare it to the average number of the OLAP operations separating successive queries for a same question among all the student logs. A standard deviation is also computed. Thus, the sessions are dropped if their number of OLAP operations, between two queries, exceeds the average and the standard deviation. If we consider that 6 OLAP operations is the maximum between two successive queries, 12 sessions have to be dropped.

Regarding the problem of query composition, one particular behavior has been identified. Indeed, it is possible to find queries having each member of a given level l in the selection set. A solution is to delete this set since the query result is exactly the same. 128 queries have been identified among the student logs (about 15% of all queries).

4.1.3 Conclusion

We conclude this section by giving the characteristics of the synthetic logs obtained using the generators defined in Section 4.1.1 and the logs devised by the students, presented in Section 4.1.2. These logs are either used to assess the session similarities defined in Section 2.3 or the *SROS* system described in Chapter 3.

Regarding the *SPaG* log generation, one log is generated for each one of the five templates available in Figure 4.4. Algorithm 7 is set with the parameters m=5 (number of session set) and n=5 (number of sessions for a set). Thus, 35 sessions are available in each log.

Regarding the *BeG* log generation, one log is generated including both templates *Explorative* and *Goal-Oriented* (see Section 4.1.1.2). Algorithm 8 is set with the parameters n=10 (number of initial queries), m=10 (number of final queries), l=4 (number of surprising queries) and k=20 (number of sessions for each initial query). A total of 200 sessions (2950 queries) are present in the log, 99 *Explorative* and 101 *Goal-Oriented* sessions. We shortly name this log: Log_{BeG}.

Regarding the logs devised by the Master's students (see Section 4.1.2), 182 sessions

are available. However, 37 sessions (about 20% of the sessions) have to be dropped (as discussed in Section 4.1.2.3). Consequently, 145 sessions can be considered as workable. We shortly name this log: $Log_{student}$.

4.2 Assessing Session Similarities

This section discusses the outcomes of the tests about similarity measures (described in Sections 2.2 and 2.3) we run to answer three main questions: *Do the proposed solutions properly capture the idea of similarity as perceived by the users? Do they adequately express the similarity criteria proposed in Section 1.5.2? What are their discriminant capabilities?* While the first question will be answered in Subsection 4.2.1, the remaining two questions will be discussed in Subsection 4.2.2.

4.2.1 Subjective Tests

As stated in Section 1.5.2, we submitted a questionnaire to 41 persons with different OLAP skills. The results have been used in the first stages of this work to understand how OLAP session similarity is perceived by users, and they will be used here to verify if the proposed methods capture the users' perception of similarity. To enable a better interpretation of the results, for each questionnaire test we show the *consensus* ϕ, i.e., the degree of agreement among raters, defined as the percentage of users who gave the majority judgement.

The first four tests of the questionnaire were focused on OLAP query comparison. In each test the users were asked to rate the similarity between a given query q_c and three other queries $\{q_1, q_2, q_3\}$ in both absolute (using four scores: *low*, *fair*, *good*, and *high*) and relative terms (i.e., by ranking queries in order of similarity). All queries were focused on the complete CENSUS schema (including 5 hierarchies and a subset of 6 measures, see Example 1.3.4); they were basic OLAP queries as of Definition 5 and were presented in a graphical way. We used the results obtained in two ways: (i) to compare σ_{que} with function σ_{AJD} mentioned in Section 1.4.2 in terms of compliance with the users' judgments; and (ii) to set the weights of the three components of our query similarity function σ_{que}.

As to (i), we defined two matching factors as follows:

- The *score matching factor SM* for σ is the percentage of times the score given by a user is the same returned by σ. To compute it, we first discretized the values returned by σ in ranges corresponding to *low*, *fair*, *good*, and *high*.
- The *rank matching factor RM* for σ is the percentage of cases in which the rankings σ provides match with those given by users (e.g., q_c was judged to be more similar to q_i than to q_j, and $\sigma(q_c, q_i) > \sigma(q_c, q_j)$).

As to (ii), we tuned the weights through an optimization process whose goal function was the maximization of the correspondence with the questionnaire results. To avoid overfitting we used a ten folds cross-validation approach. The ranges for the weights were chosen consistently with requirement ♯10 in Section 1.5.2: $\alpha \in [0.2, 0.5]$, $\beta \in [0.35, 0.75]$, $\gamma \in [0.05, 0.45]$. The function to be optimized was the average value of RM for σ_{que} in Tests 1 to 4, that measures the percentage of cases in which the rankings provided by

Table 4.1: Consensus and matching factors for OLAP query comparison user tests

	Consensus		σ_{AJD}		σ_{que}	
	ϕ_{score}	ϕ_{rank}	SM	RM	SM	RM
Test 1	70%	94%	70%	94%	70%	94%
Test 2	56%	70%	56%	56%	56%	70%
Test 3	41%	64%	34%	57%	41%	64%
Test 4	73%	93%	49%	93%	59%	93%

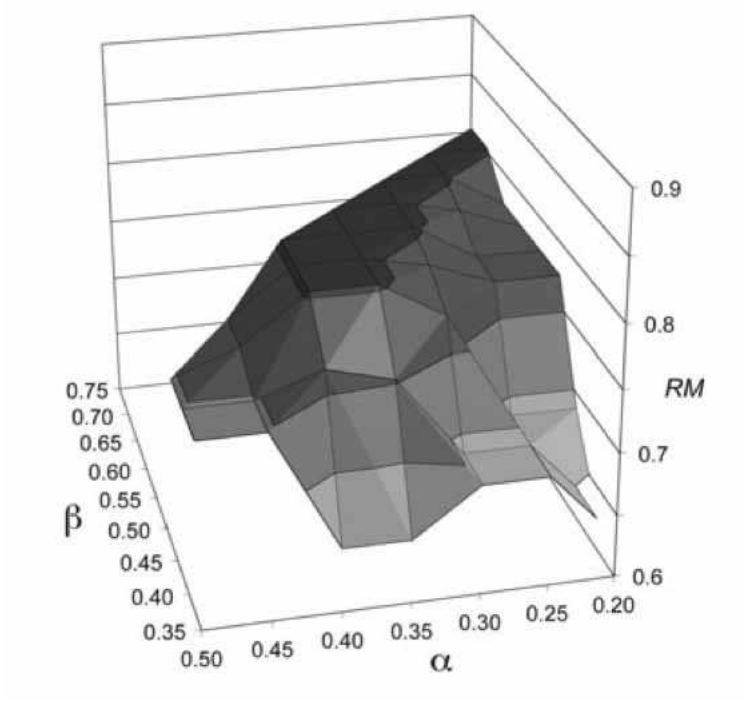


Figure 4.15: Questionnaire matching for σ_{que} as a function of weights α and β

σ_{que} match with those given by users. Figure 4.15 shows the average RM as a function of α and β (γ is set so that they sum up to 1). The optimal weights turned out to be $\alpha = 0.35$, $\beta = 0.5$, and $\gamma = 0.15$ ($\beta > \alpha$, consistently with requirement ♯10); noticeably, RM smoothly decreases for increasing distances from these optimal values, which proves that the setting is robust.

The comparison results are reported in Table 4.1. For all the tests, σ_{que} matches the users' judgement at least like σ_{AJD} thanks to its fine-grained definition. In particular, σ_{que} returns the same answers given by the majority of the users (i.e. the highest possible values for SM and RM) in Tests 1, 2, and 3, while σ_{AJD} returns the same answers only in Test 1. Note that σ_{AJD} falls short both when there is high user consensus (Test 4) and when user consensus is low because queries are very similar to each other (Tests 2 and 3). Overall, these results confirm a strong correlation between the query similarity computed through σ_{que} and the one perceived by users. Since σ_{que} is more sensitive than σ_{AJD} and it shows better results, in the remaining tests we will focus on the former.

Table 4.2: Consensus and matching factors for OLAP session comparison user tests

	Consensus		σ_{edit}		σ_{sub}		σ_{log}		σ_{ali}	
	ϕ_{score}	ϕ_{rank}	SM	RM	SM	RM	SM	RM	SM	RM
Test 1	51%	75%	51%	-	29%	-	51%	75%	51%	71%
Test 2	43%	70%	33%	-	9%	-	39%	70%	43%	70%
Test 3	51%	64%	41%	-	4%	-	51%	46%	51%	46%
Test 4	36%	80%	19%	-	26%	-	35%	65%	35%	65%
Test 5	38%	78%	33%	-	13%	-	33%	70%	33%	70%

The second part of the questionnaire included five more tests focused on OLAP session comparison. In each test, the users were asked to evaluate the similarity of a given session s_c against three candidate sessions $\{s_1, s_2, s_3\}$ in absolute and relative terms. Sessions were graphically presented to users as sequences of queries, emphasizing the OLAP operator used to move from one query to the next one. The results are summarized in Table 4.2 for the four functions described in Section 2.3, by applying SM and RM to sequences rather than to single queries. Note that the edit-based and the subsequence-based approaches, that do not directly incorporate the σ_{que} score in their definitions, are not sensitive enough to rank the sessions proposed in our tests. In fact, they return the same similarity for most sessions involved in each test, so their RM cannot be determined. This also penalizes SM, that is significantly low.

Conversely, both the log-based and the alignment-based approaches perform very well and the scores returned are, in most cases, those of the majority of users (i.e., $SM = \phi_{score}$ and/or $RM = \phi_{rank}$, that is the maximum attainable). The errors always involve sequences that are quite similar, making the comparison more subjective. Note that the absolute consensus is always much lower than the relative one; this can be explained considering that scoring entails a 4-valued choice, while ranking only requires choosing between two alternatives (s_c is either more similar to s_i than s_j or not), thus making inter-user agreement more likely. Some more detailed comments for single tests of log-based and alignment-based approaches follow:

- In test 1, candidate sessions differ in the length of the match. s_1 and s_2 are very similar to each other and determine a long match with s_c, while s_3 is quite different from the others. While the log-based approach returns the same results as the majority of users, the alignment-based approach returns an inverted ranking between s_1 and s_2, which is a minor issue due to their strong similarity.
- In test 2, candidate sessions differ in the position of the match. The log-based approach returns a score that is slightly different from the one of the majority group since it does not give different relevance to matches of recent and old queries.
- In test 3, all three candidate sessions are quite similar to each other and to s_c, leading to a difficult ranking operation for both functions.
- In test 4, each candidate session differs from the reference only for one of the components of its queries (group-by set, predicates, and measures). Both approaches agree with the users majority in indicating the session that differs in their selection predicates as the less similar to the reference session. However, both approaches return an inverted ranking between the sessions that differ in their group-by sets and in their predicates, respectively. This is probably due to the weight we use for measure

Table 4.3: Ratio τ for template-based OLAP session comparison objective tests

Log	σ_{edit}	σ_{sub}	σ_{log}	σ_{ali}
$\wedge$	1.39	1.16	1.39	2.32
$\vee$	1.46	1.52	1.31	3.21
$+$	1.44	1.23	1.32	2.15
$\|\|$	1.79	1.57	1.51	5.23
$\downarrow\uparrow$	1.08	1.57	1.42	0.78
average	1.40	1.35	1.35	2.51

similarity, $\gamma = 0.15$, that in this particular case is not low enough to counterbalance the relevant difference on measure sets.

– In test 5, session s_1 is very similar to s_c; s_2 and s_3 are similar to each other and quite different from s_c. Both approaches agree with the users majority in indicating s_1 as the most similar to s_c, but they disagree in ranking the other two sessions. This is actually not surprising in light of the low relative consensus ($\phi_{rank}(s_2, s_3) = 61\%$).

4.2.2 Objective Tests

In this section we compare the four functions described in Section 2.3; for subsequence-based similarity we use 3-grams (empirically tested for best results). All tests were conducted on a 64-bits Intel Xeon quad-core 3GHz, with 8GB RAM, running Windows 7 pro SP1; the similarity threshold was tuned to $\theta = 0.8$ to achieve the best results.

Our benchmark includes a set of synthetic sessions over the CENSUS schema (see Example 1.3.4) with our own log generator developed in Java. To generate logs we considered the five templates described in Section 4.1.1.4.

In light of the requirements expressed in Section 1.5.2, some of these templates should yield higher similarities. In particular, we want template $\vee$ to yield higher similarities than $\wedge$ due to requirement ♯12. For requirement ♯13, we also expect $||$ to yield higher similarities than $\vee$, $\wedge$, and $+$. As to $\downarrow\uparrow$, requirement ♯11 imposes that it yields low similarities.

The first test assesses the capabilities of the similarity functions. In this test, we use the five logs generated with the *SPaG* generation principle, described in Section 4.1.1.5, whose characteristics are given in Section 4.1.3.

Then, for each log and each similarity function, we computed the ratio τ of the average similarity σ_t between the two sessions respecting the template and the average similarity σ_r between each seed and the 5 sessions generated from it; the higher τ, the better the function can distinguish a template from the background. Table 4.3 reports the results. Noticeably, the alignment-based approach largely outperforms the others; besides yielding an average τ that is almost twice that of the other approaches, it meets the expectations as to template similarities. Template $||$ is correctly recognized as the one with highest similarity; $\vee$ clearly yields higher similarities than $\wedge$, while $\downarrow\uparrow$ yields low similarities since it does not fulfill requirement ♯11 about query ordering. The only other function that captures requirement ♯11 is σ_{edit}. Noticeably, though all the other functions return an

Table 4.4: Ratio τ for increasing distances in the || template

|| dist	σ_{edit}	σ_{sub}	σ_{log}	σ_{ali}
1	1.79	1.57	1.51	5.23
2	1.91	1.55	1.51	3.78
3	1.86	1.56	1.45	3.48
4	1.81	1.52	1.42	2.80
5	1.81	1.52	1.55	2.68

average ratio τ higher than 1, they are not sensitive enough to distinguish and rank the different templates.

The purpose of the second objective test is to discover how sensitive each function is to the distance between the two sessions that form template ||; to this end, the number of atomic OLAP operations that separate these two sessions is varied from 1 to 5 (using the same log-generation algorithm explained for the first test). Even in this test σ_{ali} turns out to be more effective than the other functions. Indeed, as shown in Table 4.4, the ratio τ for σ_{ali} progressively decreases for increasing distances, while for the other functions it is almost constant. This is because σ_{ali} is sensitive to the specific values of similarity between each couple of queries, while for the other functions each couple of queries either match or do not match.

The next test measures the time for computing each similarity function. For this test we generated a log, randomly chose one session s, and compared all prefixes of s with 10 other sessions randomly chosen from the log. Note that, for log-based similarity, we disregard the time for building the frequency matrix used in the computation of all the idf's. We report the results for a minimum prefix of 1 query and a maximum prefix of 13 queries. As expected, the subsequence-based approach is the most efficient (from 0.4 ms to 3.6 ms for a single comparison), followed by the alignment-based approach (from 1.1 ms to 7.1 ms) and by the edit-based approach (from 1.3 ms to 8.3 ms). Log-based similarity is the less efficient (from 30.4 to 75.1 ms).

We close this section with a final remark related to efficiency. OLAP sessions are inherently interactive; to understand to what extent our approach can realistically be adopted to compare sessions at user-time, we made two tests using the same protocol adopted for the test above:

- We measured how many comparisons can be made for each similarity function during 100 ms, which is usually considered to be the maximum interactive response time [Khoussainova et al., 2010]. The number of comparisons ranges from 109 for subsequence-based similarity to 3 for log-based similarity, with alignment-based and edit-based similarity scoring 32 and 31 comparisons, respectively.
- We measured how many comparisons can be made during the average time it takes to evaluate a query. To this end we randomly chose a session in the log and computed the average execution time for its queries, expressed in MDX; we used real data extracted from the IPUMS database [Minnesota Population Center, 2008], corresponding to about 500,000 facts stored on Oracle 11g. The average query execution time turned out to be 553.46 ms, which corresponds to 607 comparisons for subsequence-based

similarity, 177 and 175 comparisons for alignment-based and edit-based similarity respectively, and 18 comparisons for log-based similarity.

4.2.3 Conclusion

This section presented experimental results assessing different similarity measures between queries and sessions for defining a similarity function to compare OLAP sessions, based on the requirements deduced from a user study conducted with practitioners and researchers. We considered and compared two functions for OLAP query similarity and four functions for OLAP session similarity; in particular, the latter were obtained by extending popular approaches for string comparison. Overall, the experimental results we obtained show that the alignment-based approach (an extension of the Smith-Waterman algorithm, coupled with a three-component query similarity function) is the one that best matches the users' judgements. It is also the one that clearly gives best results on a synthetic benchmark in terms of sensitivity and capability of correctly ranking different templates of session similarity. Finally, from the point of view of efficiency, the time required for comparing two sessions is perfectly compatible with complex applications.

Finally, these tests confirm that the alignment-based similarity is the ideal candidate to be used in our recommendation system. Thus, the *Selecting* and *Ranking* phases of *SROS* system will make use of this measure.

4.3 Assessing the Recommendation Approach

This section presents the tests conducted for assessing the relevance of our recommendation system. The efficiency of our approach is tested in Section 4.3.1 while Section 4.3.2 is devoted to effectiveness tests. In particular, effectiveness tests will assess the recommender system against the criteria presented Section 1.5.1.

Note that all the tests were conducted on a 64-bits Intel Core i5 2.5GHz, with 16GB RAM, running Mac OSX Mountain Lion. Two types of logs are used for the tests : Log_{BeG} and $Log_{student}$, whose characteristics are described in Section 4.1.3.

To assess the *SROS* system, one session is extracted from the log where the first-third part is the current session and the rest is the expected recommendation. The other sessions form the log used by *SROS* to recommend a session. This process is repeated until each session of the log provided the current session and the expected recommendation.

4.3.1 Efficiency

The efficiency test has been conducted using Log_{BeG}. The test assesses the computation time of our recommendation method for different lengths of log (using 25%, 50%, 75% and 100% of the log, i.e. 50, 100, 150 and 200 sessions present in the log or, in terms of queries, 726, 1474, 2224 and 2950 respectively).

Figure 4.16 shows the evolution of the average computation time to obtain a recommendation with the *SROS* system. The time to compute a recommendation is more than 1 second for a log size of 25% and more than 3 seconds for the complete log.

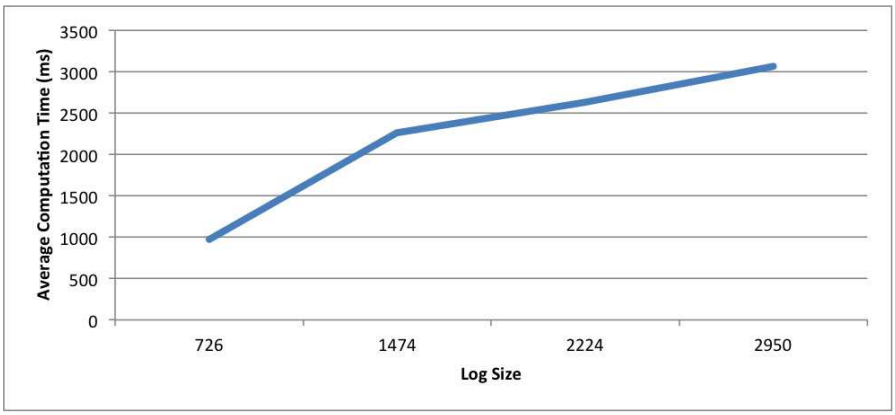


Figure 4.16: Average computation time for obtaining a recommendation.

4.3.2 Effectiveness

We present the tests of effectiveness of the *SROS* system, conducted with logs Log_{BeG} and $Log_{student}$. Beforehand, we define measures to assess the effectiveness of the *SROS* system in Section 4.3.2.1. The results of the different measures are given Sections 4.3.2.2 and 4.3.2.3.

4.3.2.1 Definitions of Effectiveness Measures

We propose a set of quality measures to assess the recommended sessions produced by the *SROS* system. Each of the quality measure answers to a particular criteria introduced in Section 1.5.1.

As a reminder, the quality criteria are:
- *Relevance*
- *Foresight*
- *Novelty*
- *Adaptation*
- *Obviousness*

Relevance is measured by the score given to the recommended session by the ranking phase of the approach (see Section 3.3).

Foresight measures how "far" from the current query c_s the first query of the recommended session r_s is. Formally, the measure is defined as:
$foresight(c_s, r_s) = 1 - \sigma_{query}(c_s[length(c_s)], r_s[1])$.

Novelty measures how distant is the recommended session r_s from the sessions in a log. We adapt the similarity based on the Hausdorff Distance (defined in Section 2.4.2), comparing two logs L and L', to compute *Novelty*. Since one session (the recommended session) is compared to a set of sessions (the log), we consider that $L = \{r_s\}$ and L' is the log. Consequently, the definition of *novelty* can be simplified as:
$novelty(r_s, L') = min_{l \in L'}(1 - \sigma_{SW}(l, r_s))$.

Figure 4.17: Recommendation Length

Adaptation measures how well the recommended session r_s fits the current session c_s with the recall of c_s's fragments. We use the Recall measure to compute the number of shared fragments between r_s and c_s and recalled in c_s. The measure is defined as: $adaptation(r_s, c_s) = \frac{|fragments(r_s) \cap fragments(c_s)|}{|fragments(c_s)|}$.

Obviousness measures how many queries in the recommended session r_s appear in the current session c_s. As for *Adaptation* measure, we use the Recall measure to compute the number of shared queries between r_s and c_s and recalled in r_s. The measure is defined as: $obviousness(r_s, c_s) = \frac{|\bigcup_{q \in c_s} \cap \bigcup_{q \in r_s}|}{|\bigcup_{q \in r_s}|}$.

In addition to the quality criteria, the effectiveness tests include an indicator measuring the closeness of the set of recommended sessions, named RS, with the set of expected recommendations, named FCS. For this, we use the Accuracy-based Similarity (defined in Section 2.4.1) to compare the two sets. As a reminder, the Accuracy-based Similarity is defined as: $\sigma_{Accuracy}(\mathcal{L}, \mathcal{L}') = 2 * \frac{\sigma_{Precision} * \sigma_{Recall}}{\sigma_{Precision} + \sigma_{Recall}}$ We also define a coverage measure between the two sets that is: $coverage(RS, FCS) = \frac{|RS|}{|FCS|}$.

4.3.2.2 Quality Criteria

The assessment of recommendations, based on the quality criteria, are given in this section. Each of the results are tested with scores of relevance increasingly demanding, i.e. only the recommendations having a relevance score greater than a given threshold are considered. Regarding Log_{Beg}, the maximum threshold for relevance score is 32 while it is 17 for $Log_{student}$, since no recommendation is found with a higher threshold. Note that in the following figures, we will not represent the threshold of relevance scores just including one recommendation. Indeed, these recommendations are not enough relevant to analyze overall behaviors of the different quality measures. Thus, the maximum thresholds are 31 and 16 for Log_{Beg} and $Log_{student}$ respectively.

Figure 4.17 gives the average length of the recommendation for different threshold of relevance scores. We can note that the average length decreases with high values of

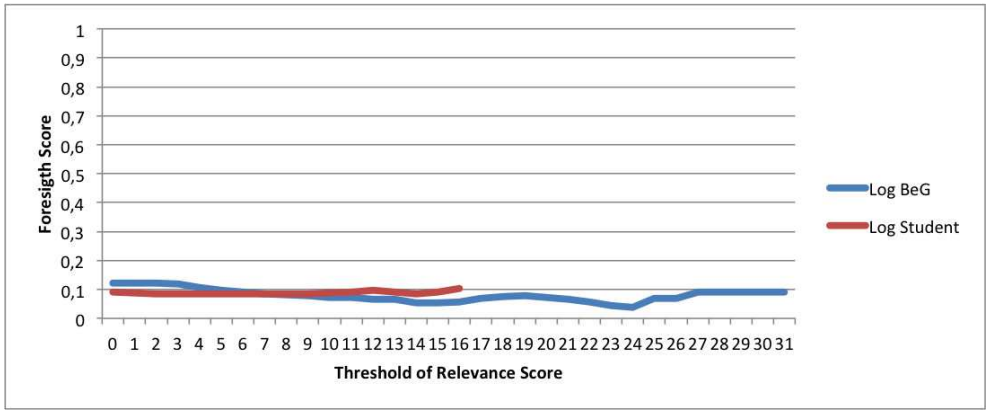


Figure 4.18: Foresight Measure

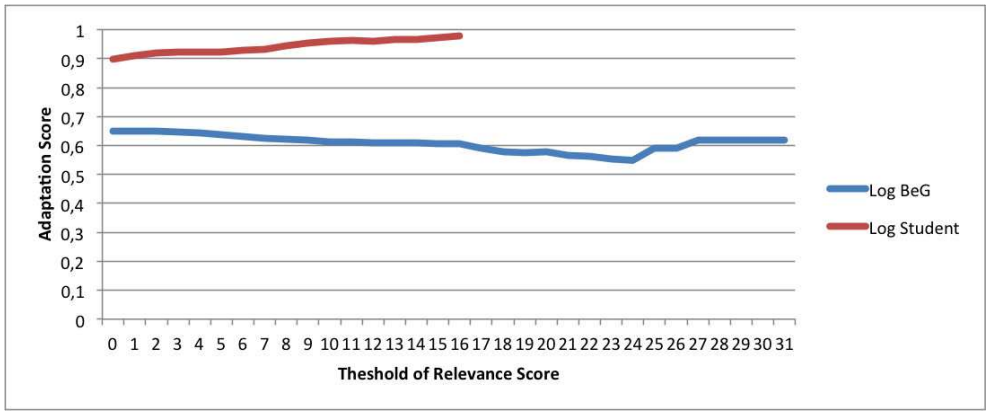


Figure 4.19: Adaptation Measure

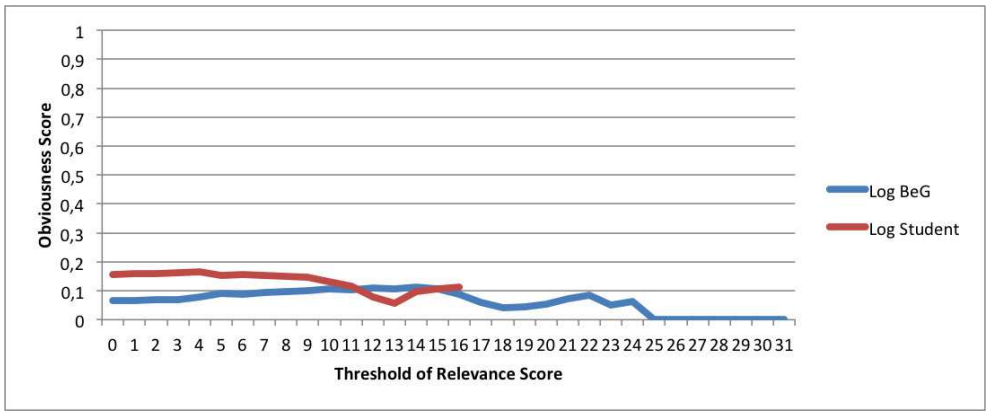


Figure 4.20: Obviousness Measure

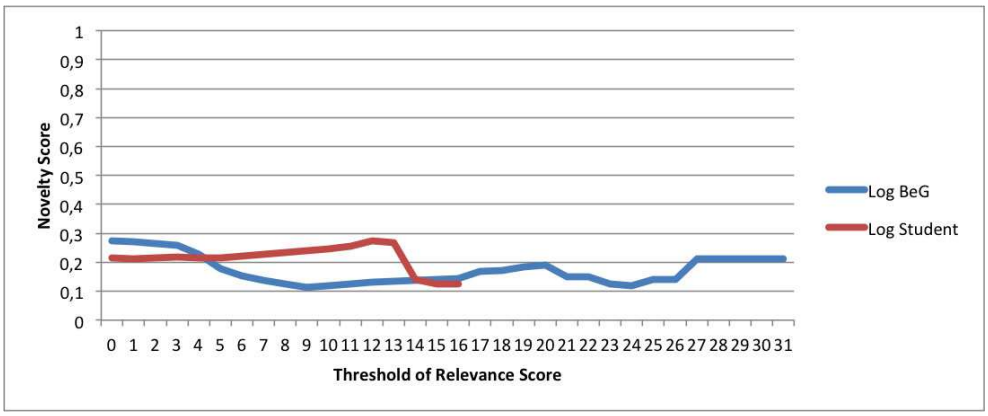


Figure 4.21: Novelty Measure

relevance scores for both types of logs. This result is expected since the relevance of a recommendation, just including MRQ query (that is the query with the highest relevance, see Section 3.3.2), is higher than a relevance score of a recommendation including several queries whose scores can significantly deteriorate it. The difference of recommendation length values between Log_{BeG} and $Log_{Student}$ is correlated with the average session length. Indeed, the length of the current sessions, coming from Log_{BeG}, is under 5 queries, while the number of queries coming from $Log_{student}$ is 1.5.

Figure 4.18 represents the average values of the foresight measure. The foresight scores are low for both logs (less than 0.12). This means that the recommendations are "close" to the current sessions. The low scores can be explained by the $Tailoring$ phase that modifies the recommendations with fragments of the current sessions. Moreover, a recommendation is a query sequence following a subsession, of a log session, aligned with the current session. Thus, it is not surprising to find a good similarity between the last query of the current session and the first query of the recommendation. Consequently, the results of the foresight measure show that the recommended sessions are not "visionary", even if the recommendations seem to follow the logic of the current sessions. Besides, the next result of the adaptation measure confirms this remark.

Figure 4.19 shows the scores of the adaptation measure for Log_{BeG} and $Log_{student}$. Regarding $Log_{student}$, the adaptation score increases from 0.9 to 1 with higher relevance scores. Regarding Log_{BeG}, the scores are quite stable, around 0.6. These results seem to be correlated with the properties of the logs. Indeed, $Log_{student}$ has short current and recommended session average lengths (respectively 1.5 and 1.3). During the $Tailoring$ phase, only the common fragments among the queries of the base recommendation can be modified. Thus, there is more chance to find common fragments within short recommendation. Consequently, the results of the adaptation measure show that the recommended sessions are well adapted to the current sessions.

Figure 4.20 indicates the obviousness scores, i.e. the average number of queries present in both recommendations and current sessions. The scores for Log_{BeG} and $Logstudent$ are under 0.2, (except for the last value of $Log_{student}$ since one query is recommended). These

scores mean that, despite the fact that the *Tailoring* phase replaces or add fragments from the current session, the recommended session includes a limited number of queries existing in the current session.

Figure 4.21 shows the novelty scores, i.e. the minimal average distances of the recommended sessions from the sessions of the log. The scores for Log_{BeG} are between 0.1 and more than 0.4. The scores for $Log_{Student}$ are between 0.1 and less than 0.3. This result indicates that the recommendations are rather new, even if, as also noted in Figure 4.18 and 4.20, we can see that the *Tailoring* phase gives recommended sessions closer to the current sessions than to the log sessions.

The results of the different measures studied in this section show that the recommendations have a good quality being adapted to the current session, without being obvious against the queries already devised by the current user and rather new compared to the log sessions.

4.3.2.3 Accuracy and Coverage

The last tests focuse on the different measures of Recall, Precision and Coverage (defined in Section 2.4.1). Notably in Figure 4.22 (focused on Log_{BeG}), we can note that the scores are high with a relevance score of 0 and decreases with higher relevance score. That is expected due to the fact that fewer sessions are recommended with higher relevance score. However, we can notice that the *Tailoring* phase increases the recall and precision scores of 10% (in the best case) compared to the scores given before the *Tailoring* phase. The same phenomenon can be seen for $Log_{student}$, in Figure 4.23. In particular the score of recall is improved by 20% whereas the precision score is improved up to 30% ! These results, although impressive at first glance, are relative. Indeed, and as described in Section 4.1.2.3, the analysis behaviors can be very different. Thus, the sessions of the log seem very distant from each other. This observation is confirmed by a test computing the average session similarities in this log, whose result is 0.04 which is very low, especially for a small average session lengths.

Consequently, the base recommendations are very distant from the expected recommendations (the recall score before *Tailoring* phase is only under 20% for the best case). When the *Tailoring* phase is applied on the base recommendation, it turns a lot of fragments of the base recommendation into fragments from the current session, making the recommendation and the expected session very close.

4.4 Conclusion

This chapter detailed the experiments conducted with similarity measures and the recommendation system.

Beforehand, this chapter also discussed different possibilities for obtaining logs allowing to test our proposals. Two synthetic log generators ($SPaG$ and BeG), based on different templates, create sessions reproducing analysis behavior we think are important to measure and consequently allow to assess the solution with objective criteria. A study is also given about sessions devised by Master's students from questionnaires, allowing to take into

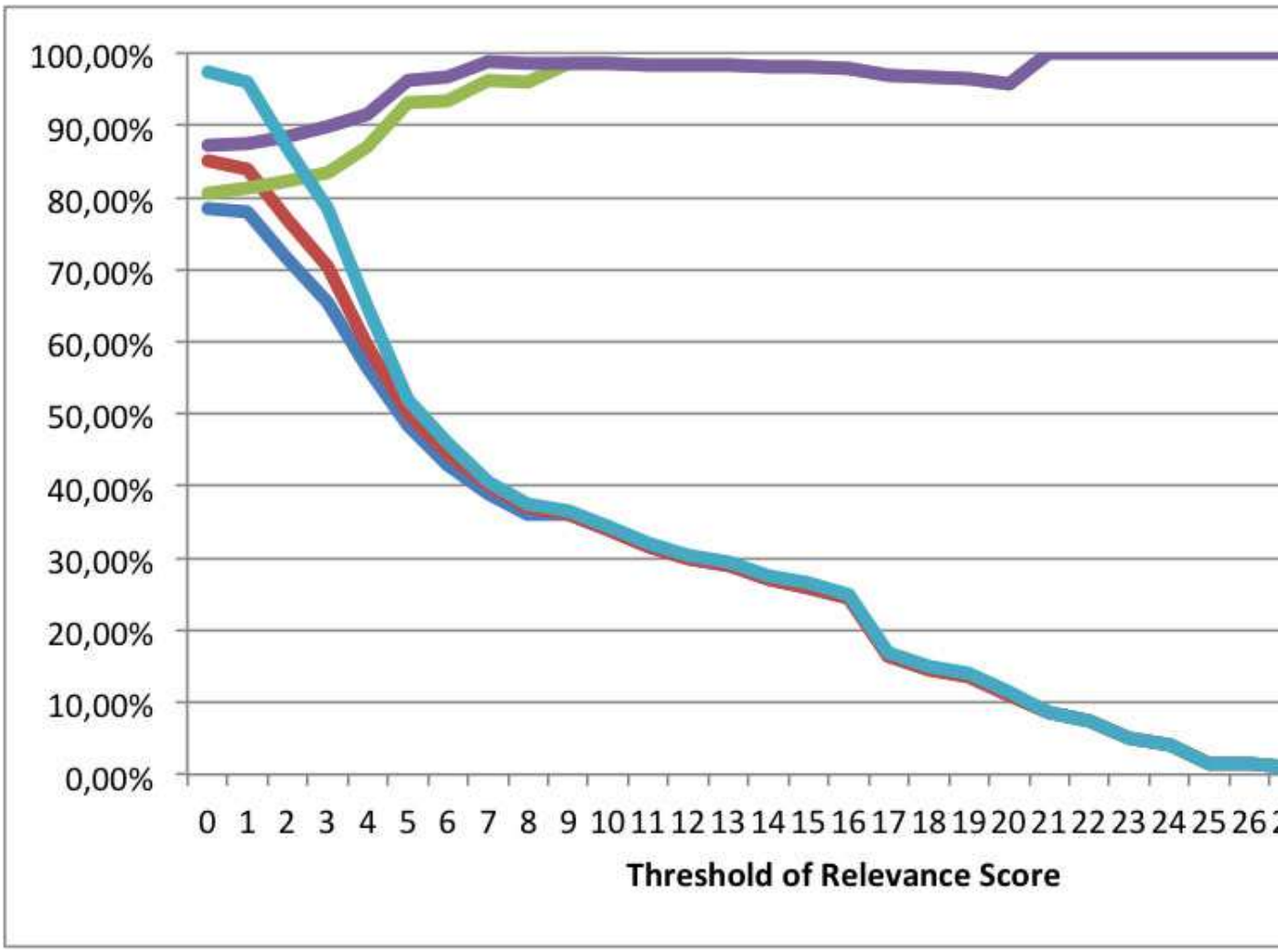


Figure 4.22: Recall, Precision and Coverage Measures with Log_{BeG}

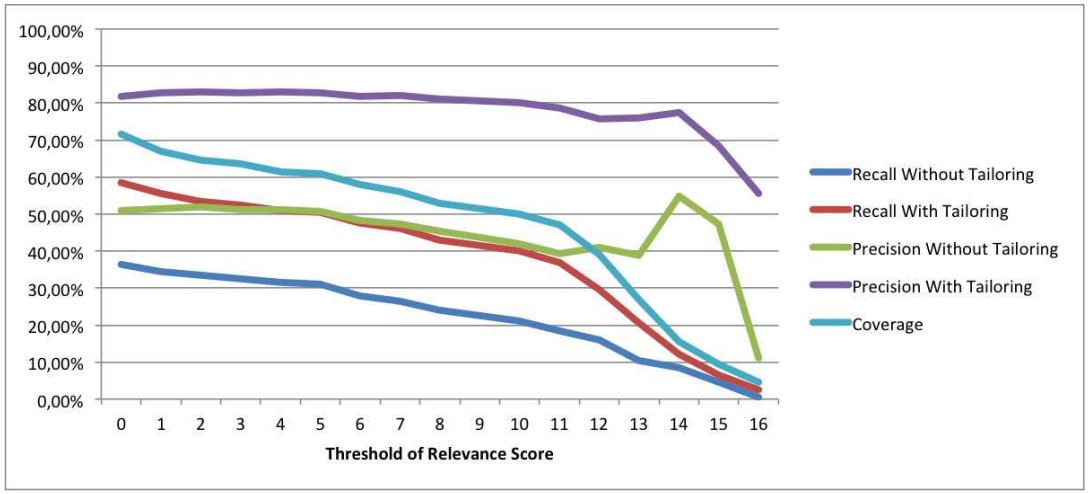


Figure 4.23: Recall, Precision and Coverage Measures with $Log_{student}$

account the subjectivity of the analysis sessions. The statistical results show that the obtained log is workable, even if a filtering step of the log is unavoidable to put aside sessions showing strange behaviors.

The tests shows that the alignment-based approach outperforms the other measures by achieving the best results (in terms of subjectivity and objectivity) and answering, at the best, to the criteria of similarity between sessions that we propose.

The recommendation system has been tested in terms of efficiency and effectiveness. The test of efficiency shows that our system produces recommendations under 3 seconds by considering a log of 200 sessions. The effectiveness tests have been conducted to assess our recommendation system against quality criteria that we issued. The results clearly demonstrate that *SROS* system proposes recommendations satisfying the criteria. Moreover, regarding the log produced by Master's students, the recommender system provides recommended sessions with high accuracy, in particular thanks to the *Tailoring* phase, even if log sessions are distant from each other. Consequently, our system is able to recommend sessions with a good quality, in very different contexts of log density.

Conclusion

Summary of the Contribution

This dissertation presents an approach for recommending OLAP sessions, in a collaborative filtering context, and based on similarity measures between queries and sessions. We give the different contributions of this work in each of the following sections. Note that this work is the subject of an ongoing work in [Aligon et al., 2013a].

Requirements for Recommendation and Similarity Measures

After briefly reviewing classical techniques for usage mining in Web Page Recommendation, a study of recommender systems in Databases and Data Warehouses allowed to identify several shortcomings. Indeed, sequential aspects are rarely addressed in these works and no approach ever considered to recommend sessions. Besides, queries are rarely synthesized for the recommendation and are often chosen among past queries. This dissertation answers these shortcomings by proposing a set of requirements to take into account in a recommendation context. In particular:

- Sessions have to be recommended.
- The recommender system should be based on similarity measures between sessions.
- Query expressions are preferred to tuples, in order to compute the recommendations in an online step.
- Quality criteria have to be defined to assess sessions recommended by the system, that are: relevance, novelty, foresight, obviousness and adaptation.

Since the recommender system is based on a similarity measures, a study of classical measures in information retrieval has been presented. To extend these measures in an OLAP context, several requirements are proposed:

- Query similarity has to give a more important weight for selection predicates rather than group-by set or measures
- Session similarity has to consider an order between the queries composing the sessions to compare
- Recent queries are especially considered as more relevant than old queries

Note that the extension of similarity measures for OLAP sessions is published in [Aligon et al., 2013b].

Definitions of Three-Levels Similarity Measures

This dissertation proposed several similarity measures organized in a three-level approaches between OLAP logs. Similarity measures between logs depend on similarity measures between sessions that depend on similarity measures between queries. In particular, the similarity measure between queries only depends on the query definition, that is a fragment-based structure composed of a group-by set, a selection predicate set and a measure set. The similarity measures between sessions are extensions, in the OLAP context, of classical measures in information retrieval that are Dice Coefficient, Sequence Alignment, TF-IDF, and Levenshtein Distance. Notably, the Sequence Alignment is the basis of the recommender system. The similarity measures between logs extend classical measures between sets, that are Accuracy, Hausdorff Distance and Jaccard Coefficient. They are the basics of the quality measure definitions for recommended sessions and answering to the quality criteria.

Proposal for a Similarity-based Recommendation of OLAP sessions

A recommender system based on similarity measure between sessions is proposed. Note that this proposal is an ongoing work in [Aligon et al., 2013a]. Three phases compose this system:

- The first phase aligns the log sessions with the current session, based on a modified version of Sequence Alignment. The queries following the aligned subsessions of the log sessions are considered as potential futures to recommend.
- The second phase ranks each future by identifying densest areas of similar queries in the log sessions.
- The last phase adapts the future ranked first by modifying or adding fragments in its queries, using patterns extracted from the log and the current session, and recommends it. In particular, this phase adapts the technique of [Aligon et al., 2011].

Assessing the Similarity Measures and the Recommender System

The recommender system is assessed in terms of efficiency and effectiveness with sessions coming from synthetic log generations or logs whose sessions have been devised by Master's students. Two synthetic generators, based on different templates, create sessions reproducing analysis behaviors. The sessions devised by students, whose a feedback is published in [Aligon, 2013], allow to use the subjectivity of analysis sessions for testing the recommender system, even if a pre-filtering of the sessions is even so needed. The tests showed that the similarity measure based the subsequence alignment outperforms the other approaches. The tests on effectiveness of the recommender system showed that the recommended sessions satisfy the quality criteria expressed for recommendation.

Perspectives

We present in the following sections short-terms and long-terms perspectives of the works presented in this dissertation.

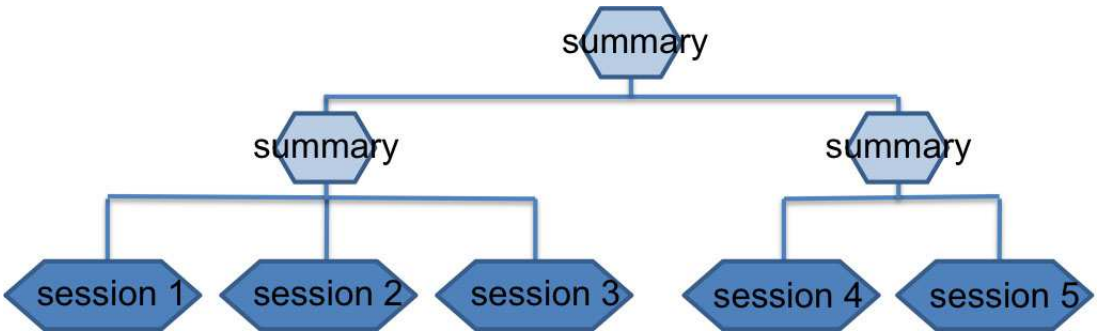


Figure 4.24: Agglomerative Hierarchical Clustering of Log and Summaries

A Tool for Session Design Assistance

When a user just begins to devise her session, the recommender system generally cannot propose queries since no current session or not enough queries are available. Well known in the recommendation context, this *Cold Start* problem could be addressed using the works proposed in [Aligon and Marcel, 2012] and [Aligon et al., 2012]. Indeed, the idea would be that a user explores the query logs in order to identify particular queries that she could reuse to initiate her session. Since a log is very large, a user would be overwhelmed by the amount of information to explore. Thus, only the most relevant queries of a log should be presented to a user and organized to provide easy navigation between them.

To cope with the problem of query navigation, a solution could be to identify different groups of similar queries for different levels of closeness density. Typically, an agglomerative hierarchical clustering could produce clusters, organized in a tree structure. To browse between the nodes, operators for selecting nodes and moving between them could be proposed. Unfortunately, the nodes of the tree can be very hard to analyze since many sessions can be present. A line of thought, presented below, is to produce a relevant and concise representation of a group of sessions, named *summary* (see Figure 4.24).

To cope with the problem of representation of groups of sessions, a solution is proposed in [Aligon and Marcel, 2012] and [Aligon et al., 2012] that addresses the problem of summarization of query expressions. More precisely, a framework is introduced in which summarization operators over queries, sessions and logs are defined. The intuition of the log summarization is that the form of the summary should be the same as the form of the original log and that summary may loose information. Moreover, the sequentiality of a session has to be reflected in the summary. This is captured by proposing specialization relation between sessions, another specialization relation between queries, and a cover relation between queries and sessions. More precisely, as depicted in Figure 4.25, a session summarizes another session if the former is more general than the latter, i.e., each of its queries covers a subsession of the original session. And, a query covers a sequence of queries if it is more general than each query of the sequence. An Algorithm allows to automatically compute a summary of log where quality measures ensure that the summary does not loose too much information. The overall principle of log summarization is given Figure 4.26 and shows the two steps of the algorithm that are:

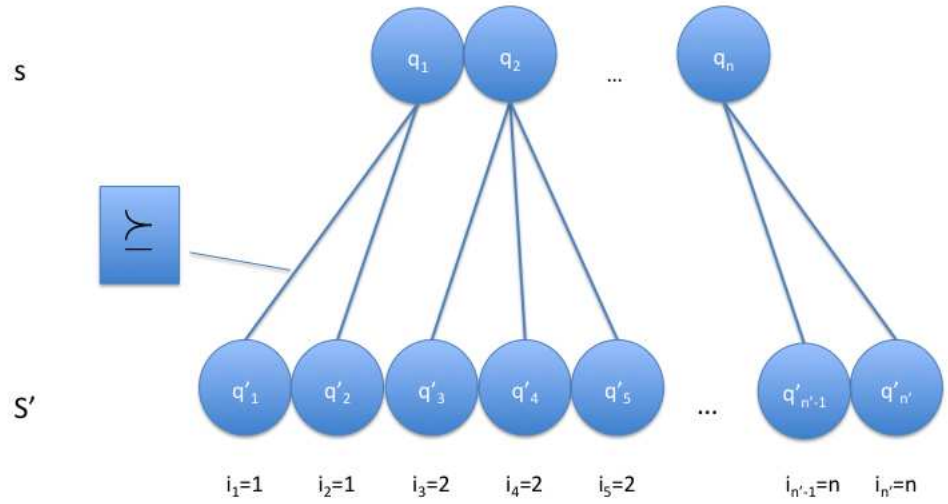


Figure 4.25: Specialization relation over Sessions

– The reduction of the number of queries per session.
– The merging of sessions reducing the number of sessions by generalizing pairs of sessions by one session.

Considering both the hierarchical structure of the log sessions and the summarization technique for obtaining an overall view of each cluster, a user should be able to select queries to reuse in her current session. Finally, when the number of queries in the current session is important enough, the recommender system takes over.

A Benchmark for OLAP sessions

To conduct and simulate tests, a perspective is the development of a platform for assessing the quality of an analysis session over a cube. Indeed, although data quality is a well studied area of data management, the quality of the querying process has not yet been investigated. Data quality is modeled according to quality dimensions (like e.g., accuracy.), each of them grouping quality factors (like e.g., semantic correction, or syntactic correction), each of them measured by various metrics [Berti-Equille et al., 2011]. A similar approach could be used to model the quality of analysis queries and the quality of analytical sessions. In addition, such a model would allow the development of a platform for assessing and validating user-centric approaches in the context of OLAP.

Notably, in domains like Information Retrieval, benchmarks are used [(NIST), 2012] to validate approaches aiming at supporting browsing and exploratory search results. In

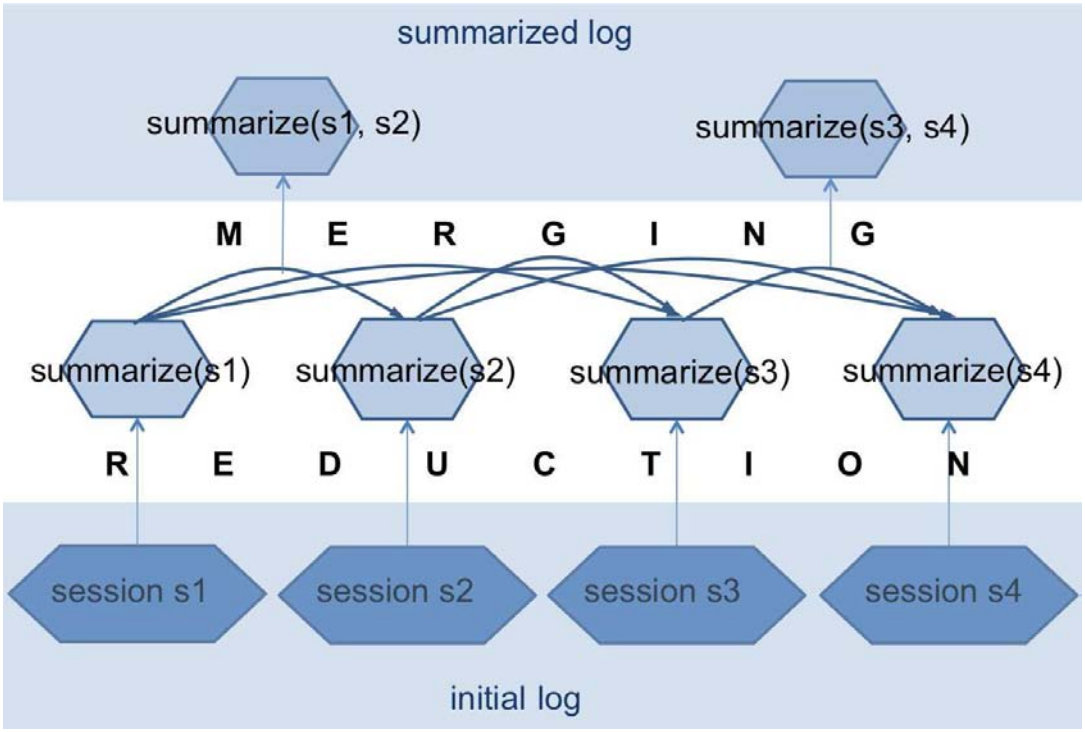


Figure 4.26: Log Summarization Principle

the domain of data warehousing, the existing benchmarks are exclusively focused on performance (see for instance [(TPC), 2012, O'Neil et al., 2009, Darmont et al., 2007]), and effectiveness of the querying process in terms of how successful the analytical session is, is simply not addressed. So far, validating personalization or recommendation approaches is based on synthetic query logs [Mishra and Koudas, 2009] or the processing of existing logs with no guarantee of being realistic [Chatzopoulou et al., 2009].

A platform for assessing user-centric approaches should allow to measure to which extent approaches are effective in that answers found are relevant, the effort spent to conduct the analysis is reduced, etc. Such a platform could benefit from previous effort to develop a benchmark to validate personalization approaches in databases [Peralta et al., 2009].

The development of such a platform supposes to extend the session definition by including as much information as possible, like for instance, the OLAP operations.

Adaptation of the Recommender System in other Contexts

Another long-term perspective is to adapt the recommender system in contexts other than OLAP. Indeed, the recommender system can be transposable in contexts where sequences of complex operations are possible and where analysis are required. Applications of the recommender system can be found in Databases of course but also in Data Mining where complex sequences can be viewed as sequences of data mining tasks. These complex sequences have to be adapted with the session similarity, by taking into account the

specific aspects of each domain. In the web, the system could also manage sequences of web search or analysis sequences over social networks. In particular, the recommendation system could also take into account the relationships between users and could include a similarity measure between users. Indeed, the system proposed in this dissertation only uses sessions independently of a user, without beforehand grouping similar sessions into user profiles.

Bibliography

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749.

[Agrawal et al., 1993] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216.

[Agrawal et al., 2006] Agrawal, R., Rantzau, R., and Terzi, E. (2006). Context-sensitive ranking. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 383–394, Chicago, IL.

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA. IEEE Computer Society.

[Akbarnejad et al., 2010] Akbarnejad, J., Chatzopoulou, G., Eirinaki, M., Koshy, S., Mittal, S., On, D., Polyzotis, N., and Varman, J. S. V. (2010). SQL QueRIE recommendations. *PVLDB*, 3(2):1597–1600.

[Aligon, 2013] Aligon, J. (2013). Gathering real olap analysis sessions: A feedback. In *Proceedings EDA*, Blois, France.

[Aligon et al., 2013a] Aligon, J., Gallinucci, E., Golfarelli, M., Marcel, P., and Rizzi, S. (2013a). Olap's a journey, not a destination: An approach for recommending olap sessions.

[Aligon et al., 2011] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E. (2011). Mining preferences from OLAP query logs for proactive personalization. In *Proceedings ADBIS*, pages 84–97, Vienna, Austria.

[Aligon et al., 2013b] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E. (2013b). Similarity measures for olap sessions. *KAIS*, 34(3).

[Aligon et al., 2012] Aligon, J., Li, H., Marcel, P., and Soulet, A. (2012). Towards a logical framework for OLAP query log manipulation. In *PersDB 2012, 6th International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (invited paper)*.

[Aligon and Marcel, 2012] Aligon, J. and Marcel, P. (2012). A framework for user-centric summaries of olap sessions. In *Proceedings EDA*, pages 103–117, Bordeaux, France.

[Aouiche et al., 2006] Aouiche, K., Jouve, P.-E., and Darmont, J. (2006). Clustering-based materialized view selection in data warehouses. In *Proceedings ADBIS*, pages 81–95, Thessaloniki, Greece.

[Aufaure et al., 2013] Aufaure, M.-A., Beauger, N. K., Marcel, P., Rizzi, S., Vanrompay, Y., et al. (2013). Predicting your next olap query based on recent analytical sessions. In *Proceedings DaWaK*, Prague, Czech Republic.

[Baeza-Yates et al., 2004] Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. In *Proceedings of the 2004 international conference on Current Trends in Database Technology*, EDBT'04, pages 588–596, Berlin, Heidelberg. Springer-Verlag.

[Baikousi et al., 2011] Baikousi, E., Rogkakos, G., and Vassiliadis, P. (2011). Similarity measures for multidimensional data. In *Proceedings ICDE*, pages 171–182, Hannover, Germany.

[Banerjee and Ghosh, 2001] Banerjee, A. and Ghosh, J. (2001). Clickstream clustering using weighted longest common subsequences.

[Beeferman and Berger, 2000] Beeferman, D. and Berger, A. (2000). Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 407–416, New York, NY, USA. ACM.

[Berti-Equille et al., 2011] Berti-Equille, L., Comyn-Wattiau, I., Cosquer, M., Kedad, Z., Nugier, S., Peralta, V., Cherfi, S. S.-S., and Thion-Goasdoué, V. (2011). Assessment and analysis of information quality: a multidimensional model and case studies. *IJIQ*, 2(4):300–323.

[Borges and Levene, 2000] Borges, J. and Levene, M. (2000). Data mining of user navigation patterns. In *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling*, WEBKDD '99, pages 92–111, London, UK, UK. Springer-Verlag.

[Brown et al., 1992] Brown, P. F., Pietra, V. J. D., de Souza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

[Bustos and Skopal, 2011] Bustos, B. and Skopal, T. (2011). Non-metric similarity search problems in very large collections. In *Proceedings ICDE*, pages 1362–1365, Hannover, Germany.

[Chatzopoulou et al., 2011] Chatzopoulou, G., Eirinaki, M., Koshy, S., Mittal, S., Polyzotis, N., and Varman, J. S. V. (2011). The QueRIE system for personalized query recommendations. *IEEE Data Eng. Bull.*, 34(2):55–60.

[Chatzopoulou et al., 2009] Chatzopoulou, G., Eirinaki, M., and Polyzotis, N. (2009). Query recommendations for interactive database exploration. In *Proceedings SSDBM*, pages 3–18, New Orleans, LA.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *Proceedings IJCAI-03 Workshop on Information Integration on the Web*, pages 73–78, Acapulco, Mexico.

[Darmont et al., 2007] Darmont, J., Bentayeb, F., and Boussaid, O. (2007). Benchmarking data warehouses. *IJBIDM*, 2(1):79–104.

[Drosou and Pitoura, 2011] Drosou, M. and Pitoura, E. (2011). ReDRIVE: result-driven database exploration through recommendations. In *Proceedings CIKM*, pages 1547–1552, Glasgow, United Kingdom.

[Drosou and Pitoura, 2013] Drosou, M. and Pitoura, E. (2013). Ymaldb: exploring relational databases via result-driven recommendations. *The VLDB Journal*, pages 1–26.

[Eirinaki et al., 2013] Eirinaki, M., Abraham, S., Polyzotis, N., and Shaikh, N. (2013). Querie: Collaborative database exploration. *IEEE Transactions on Knowledge and Data Engineering*.

[Fonseca et al., 2005] Fonseca, B. M., Golgher, P., Pôssas, B., Ribeiro-Neto, B., and Ziviani, N. (2005). Concept-based interactive query expansion. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 696–703, New York, NY, USA. ACM.

[Garcia-Molina et al., 2008] Garcia-Molina, H., Ullman, J. D., and Widom, J. D. (2008). *Database Systems: The Complete Book, Second edition*. Prentice Hall.

[Ge et al., 2010] Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *RecSys*, pages 257–260.

[Ghosh et al., 2002] Ghosh, A., Parikh, J., Sengar, V. S., and Haritsa, J. R. (2002). Plan selection based on query clustering. In *Proceedings VLDB*, pages 179–190, Hong Kong, China.

[Giacometti et al., 2009] Giacometti, A., Marcel, P., and Negre, E. (2009). Recommending multidimensional queries. In *Proceedings DaWaK*, pages 453–466, Linz, Austria.

[Giacometti et al., 2011] Giacometti, A., Marcel, P., Negre, E., and Soulet, A. (2011). Query recommendations for OLAP discovery-driven analysis. *IJDWM*, 7(2):1–25.

[Girvan and Newman, 2002] Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826.

[Glover et al., 2002] Glover, E. J., Tsioutsiouliklis, K., Lawrence, S., Pennock, D. M., and Flake, G. W. (2002). Using web structure for classifying and describing web pages. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 562–569, New York, NY, USA. ACM.

[Golfarelli, 2003] Golfarelli, M. (2003). Handling large workloads by profiling and clustering. In *Proceedings DaWaK*, pages 212–223, Prague, Czech Republic.

[Golfarelli and Rizzi, 2009] Golfarelli, M. and Rizzi, S. (2009). *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

[Golfarelli et al., 2011] Golfarelli, M., Rizzi, S., and Biondi, P. (2011). myOLAP: An approach to express and evaluate OLAP preferences. *IEEE TKDE*, 23(7):1050–1064.

[Gunawardana and Shani, 2009] Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10:2935–2962.

[Gupta et al., 1995] Gupta, A., Harinarayan, V., and Quass, D. (1995). Aggregate-query processing in data warehousing environments. In *VLDB*, pages 358–369.

[Gupta and Mumick, 1999] Gupta, A. and Mumick, I. (1999). *Materialized views: techniques, implementations, and applications*. MIT Press.

[Han, 2005] Han, J. (2005). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

[Jagadish et al., 2007] Jagadish, H. V., Chapman, A., Elkiss, A., Jayapandian, M., Li, Y., Nandi, A., and Yu, C. (2007). Making database systems usable. In *Proceedings SIGMOD*, pages 13–24, Beijing, China.

[Jerbi et al., 2009] Jerbi, H., Ravat, F., Teste, O., and Zurfluh, G. (2009). Applying recommendation technology in OLAP systems. In *Proceedings ICEIS*, pages 220–233, Milan, Italy.

[Jones et al., 2006] Jones, R., Rey, B., Madani, O., and Greiner, W. (2006). Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA. ACM.

[Kersten et al., 2011] Kersten, M. L., Idreos, S., Manegold, S., and Liarou, E. (2011). The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB*, 4(12):1474–1477.

[Khalil et al., 2006] Khalil, F., Li, J., and Wang, H. (2006). A framework of combining markov model with association rules for predicting web page accesses. In *Proceedings of the fifth Australasian conference on Data mining and analystics - Volume 61*, AusDM '06, pages 177–184, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Khoussainova et al., 2009] Khoussainova, N., Balazinska, M., Gatterbauer, W., Kwon, Y., and Suciu, D. (2009). A case for a collaborative query management system. In *Proceedings CIDR*, Asilomar, CA.

[Khoussainova et al., 2010] Khoussainova, N., Kwon, Y., Balazinska, M., and Suciu, D. (2010). SnipSuggest: Context-aware autocompletion for SQL. *PVLDB*, 4(1):22–33.

[Khoussainova et al., 2011] Khoussainova, N., Kwon, Y., Liao, W.-T., Balazinska, M., Gatterbauer, W., and Suciu, D. (2011). Session-based browsing for more effective query reuse. In *Proceedings SSDBM*, pages 583–585, Portland, OR.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[Li and Jagadish, 2012] Li, F. and Jagadish, H. V. (2012). Usability, databases, and hci. *IEEE Data Eng. Bull.*, 35(3):37–45.

[Li and Durbin, 2010] Li, H. and Durbin, R. (2010). Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, 26(5):589–595.

[Liu et al., 1999] Liu, B., Hsu, W., and Ma, Y. (1999). Mining association rules with multiple minimum supports. In *Proceedings of the fifth ACM SIGKDD international*

conference on Knowledge discovery and data mining, KDD '99, pages 337–341, New York, NY, USA. ACM.

[MacQueen, 1967] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

[Madria et al., 1999] Madria, S. K., Bhowmick, S. S., Ng, W. K., and Lim, E.-P. (1999). Research issues in web data mining. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*, DaWaK '99, pages 303–312, London, UK, UK. Springer-Verlag.

[Microsoft, 2009] Microsoft (2009). MDX reference. http://msdn.microsoft.com/en-us/library/ms145506.aspx.

[Minnesota Population Center, 2008] Minnesota Population Center (2008). Integrated public use microdata series. http://www.ipums.org.

[Mishra and Koudas, 2009] Mishra, C. and Koudas, N. (2009). Interactive query refinement. In Kersten, M. L., Novikov, B., Teubner, J., Polutin, V., and Manegold, S., editors, *Proceedings EDBT*, volume 360 of *ACM International Conference Proceeding Series*, pages 862–873. ACM.

[Mobasher et al., 2001] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on Web information and data management*, WIDM '01, pages 9–15, New York, NY, USA. ACM.

[Monge and Elkan, 1997] Monge, A. E. and Elkan, C. (1997). An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proceedings Workshop on Research Issues on Data Mining and Knowledge Discovery*.

[Moreau et al., 2008] Moreau, E., Yvon, F., and Cappé, O. (2008). Robust similarity measures for named entities matching. In *Proceedings International Conference on Computational Linguistics*, pages 593–600, Manchester, UK.

[Nandi and Jagadish, 2011] Nandi, A. and Jagadish, H. V. (2011). Guided interaction: Rethinking the query-result paradigm. *PVLDB*, 4(12):1466–1469.

[Navarro, 2001] Navarro, G. (2001). A guided tour to approximate string matching. *ACM Comput. Surveys*, 33(1):31–88.

[Negre, 2009] Negre, E. (2009). *Exploration collaborative de cubes de données*. PhD thesis, Université François Rabelais Tours.

[(NIST), 2012] (NIST) (2012). Text retrieval conference (trec) home page. http://trec.nist.gov/.

[Ögüdücü, 2010] Ögüdücü, S. G. (2010). *Web Page Recommendation Models: Theory and Algorithms*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

[Ögüdücüıand Özsu, 2006] Ögüdücüı, c. G. and Özsu, M. T. (2006). Incremental click-stream tree model: Learning from new users for web page prediction. *Distrib. Parallel Databases*, 19(1):5–27.

[O'Neil et al., 2009] O'Neil, P. E., O'Neil, E. J., Chen, X., and Revilak, S. (2009). The Star Schema Benchmark and Augmented Fact Table Indexing. In *Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers*, pages 237–252.

[Peralta et al., 2009] Peralta, V., Kostadinov, D., and Bouzeghoub, M. (2009). Apmd-workbench: A benchmark for query personalization. In *Workshop on Contextual Information Access, Seeking and Retrieval Evaluation (CIRSE)*.

[Ristad and Yianilos, 1998] Ristad, E. S. and Yianilos, P. N. (1998). Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532.

[Runeson, 2003] Runeson, P. (2003). Using students as experiment subjects - an analysis on graduate and freshmen psp student data. In *Proceedings of 7th International Conference on Empirical Assessment & Evaluation in Software Engineering*, pages 95–02.

[Sapia, 2000] Sapia, C. (2000). PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems. In *Proceedings DaWaK*, pages 224–233, London, UK.

[Sarawagi, 1999] Sarawagi, S. (1999). Explaining differences in multidimensional aggregates. In *Proceedings VLDB*, pages 42–53, Edinburgh, Scotland.

[Sarawagi, 2000] Sarawagi, S. (2000). User-adaptive exploration of multidimensional data. In *VLDB*, pages 307–316.

[Sarawagi and Sathe, 2000] Sarawagi, S. and Sathe, G. (2000). i^3: Intelligent, interactive investigaton of olap data cubes. In *SIGMOD Conference*, page 589, Dallas, Texas.

[Sarwar et al., 2000] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T. (2000). Application of dimensionality reduction in recommender system - a case study. In *In ACM WebKDD Workshop*.

[Sathe and Sarawagi, 2001] Sathe, G. and Sarawagi, S. (2001). Intelligent rollups in multidimensional olap data. In Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., and Snodgrass, R. T., editors, *VLDB*, pages 531–540. Morgan Kaufmann.

[Smith and Waterman, 1981] Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, pages 3–17, London, UK, UK. Springer-Verlag.

[Stefanidis et al., 2009] Stefanidis, K., Drosou, M., and Pitoura, E. (2009). "You May Also Like" results in relational databases. In *Proceedings International Workshop on Personalized Access, Profile Management and Context Awareness: Databases*, Lyon, France.

[Steinhaus, 1956] Steinhaus, H. (1956). Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804.

[Stone, 1974] Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147.

[Tao et al., 2003] Tao, F., Murtagh, F., and Farid, M. (2003). Weighted association rule mining using weighted support and significance framework. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 661–666, New York, NY, USA. ACM.

[Töscher et al., 2009] Töscher, A., Jahrer, M., and Bell, R. M. (2009). The bigchaos solution to the netflix grand prize. *Netflix prize documentation*.

[(TPC), 2012] (TPC) (2012). Tpc benchmark ds (tpc-ds): The new decision support benchmark standard. http://www.tpc.org/tpcds/.

[Wagner and Fischer, 1974] Wagner, R. and Fischer, M. (1974). The string-to-string correction problem. *Journal ACM*, 21(1):168–173.

[Wang and Zaïane, 2002] Wang, W. and Zaïane, O. R. (2002). Clustering web sessions by sequence alignment. In *DEXA Workshops*, pages 394–398, Aix-en-Provence, France.

[Wen et al., 2001] Wen, J.-R., Nie, J.-Y., and Zhang, H.-J. (2001). Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 162–168, New York, NY, USA. ACM.

[Yang et al., 2009] Yang, X., Procopiuc, C. M., and Srivastava, D. (2009). Recommending join queries via query log analysis. In *Proceedings ICDE*, pages 964–975, Shanghai, China.

[Yao et al., 2005] Yao, Q., An, A., and Huang, X. (2005). Finding and analyzing database user sessions. In *Proceedings DASFAA*, pages 851–862, Beijing, China.

French Abstract

L'OLAP (On-Line Analytical Processing) est le paradigme principal pour accéder aux données multidimensionnelles dans les entrepôts de données. Pour obtenir une haute expressivité d'interrogation, malgré un petit effort de formulation de la requête, OLAP fournit un ensemble d'opérations (comme drill-down et slice-and-dice) qui transforment une requête multidimensionnelle en une autre, de sorte que les requêtes OLAP sont normalement formulées sous la forme de séquences appelées *Sessions OLAP*. Lors d'une session OLAP l'utilisateur analyse les résultats d'une requête et, selon les données spécifiques qu'il voit, applique une seule opération afin de créer une nouvelle requête qui lui donnera une meilleure compréhension de l'information. Les séquences de requêtes qui en résultent sont fortement liées à l'utilisateur courant, le phénomène analysé, et les données. Alors qu'il est universellement reconnu que les outils OLAP ont un rôle clé dans l'exploration souple et efficace des cubes multidimensionnels dans les entrepôts de données, il est aussi communément admis que le nombre important d'agrégations et sélections possibles, qui peuvent être exploités sur des données, peut désorienter l'expérience utilisateur.

Cette thèse présente une approche pour recommander des sessions OLAP, dans un contexte de filtrage collaboratif, et fondé sur des mesures de similarité entre les requêtes et les sessions. Après une brève étude des techniques classiques d'extraction de l'expérience utilisateur dans le domaine des recommandations de page web, une étude sur les systèmes de recommandation dans les bases et entrepôts de données permet d'identifier plusieurs lacunes. En effet, les aspects séquentiels sont rarement pris en compte dans ces travaux et aucune approche n'a déjà envisagé de recommander des sessions. Par ailleurs, les requêtes sont rarement synthétisées pour la recommandation et sont souvent choisis parmi les requêtes passées. Cette thèse répond à ces inconvénients en proposant un ensemble d'exigences à prendre en compte dans un contexte de recommandation. Puisque le système de recommandation est basée sur une mesure de similarité, une étude des mesures classiques en recherche d'information est également présentée. Par la suite plusieurs mesures de similarité sont étendus dans un contexte OLAP et sont organisés dans une approche à trois niveaux entre les logs OLAP. Les mesures de similarité entre les logs dépendent de mesures de similarité entre les sessions qui dépendent de mesures de similarité entre requêtes. Ensuite, un système de recommandation, basé sur une mesure de similarité entre sessions, est proposé. Trois phases composent ce système. La première phase aligne les sessions du log à la session courante et identifie d'éventuelles recommandations. La deuxième phase classe chaque recommandation en identifiant les zones les plus denses de requêtes similaires dans les sessions du log. La dernière phase adapte la recommandation, ayant le meilleur score de classement à la session courante, en utilisant des motifs extraits du log et de la session courante, et la recommande. Enfin, le système de recommandation est évaluée en termes d'efficacité et de pertinence avec des sessions provenant de générateurs de logs synthétiques ou de logs dont les sessions ont été conçues par les étudiants de Master en Aide à la Décision.

Finalement, plusieurs perspectives de recherche sont présentées. En particulier, une proposition pour palier au problème du *Dmarrage Froid*, lors de la composition de sessions, est décrite. Une discussion est aussi exprimée sur le besoin d'un benchmark pour les sessions OLAP mais aussi sur l'adaptation du système de recommendation à d'autres

contextes que l'OLAP.

Mots clés :

Système de recommandation, Log, Session OLAP , Mesures de Similarité OLAP

Abstract :

OLAP (On-Line Analytical Processing) is the main paradigm for accessing multidimensional data in data warehouses. To obtain high querying expressiveness despite a small query formulation effort, OLAP provides a set of operations (such as drill-down and slice-and-dice) that transform one multidimensional query into another, so that OLAP queries are normally formulated in the form of sequences called *OLAP sessions*. During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, applies one operation to determine a new query that will give her a better understanding of information. The resulting sequences of queries are strongly related to the issuing user, to the analyzed phenomenon, and to the current data. While it is universally recognized that OLAP tools have a key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses, it is also commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating.

This dissertation presents an approach for recommending OLAP sessions, in a collaborative filtering context, and based on similarity measures between queries and sessions. After briefly reviewing classical techniques for usage mining in Web Page Recommendation, a study of recommender systems in Databases and Data Warehouses allows to identify several shortcomings. Indeed, sequential aspects are rarely addressed in these works and no approach ever considered to recommend sessions. Besides, queries are rarely synthesized for the recommendation and are often chosen among past queries. This dissertation answers these shortcomings by proposing a set of requirements to take into account in a recommendation context. Since the recommender system is based on a similarity measures, a study of classical measures in information retrieval is also presented. Afterward several similarity measures are extended in an OLAP context and are organized in a three-level approaches between OLAP logs. Similarity measures between logs depend on similarity measures between sessions that depend on similarity measures between queries. Then, a recommender system based on similarity measure between sessions is proposed. Three phases compose this system. The first phase aligns the log sessions with the current session and identifies possible recommendations. The second phase ranks each recommendation by identifying densest areas of similar queries in the log sessions. The last phase adapts the recommendation, ranked first to the current session, using patterns extracted from the log and the current session, and recommends it. Also, the recommender system is assessed in terms of efficiency and effectiveness with sessions coming from synthetic log generations or logs whose sessions have been devised by Master's students in Business Intelligence.

Finally, several research perspectives are presented. In particular, a proposal to over-

come the *Cold Start* problem, during session design, is described. A discussion is also expressed the need of a benchmark for OLAP sessions as well as the adaptation of the recommender system in contexts other than OLAP.

Keywords :

Recommender System, Log, OLAP Session, OLAP Similarity Measures