User Intent based Recommendation for Modern BI Systems
Contents

1 Introduction ............................................. 17
   1.1 Exploring evergrowing data .................. 17
   1.2 Problems that BI analysts are facing in industry .... 18
   1.3 Our challenges .................................. 19
   1.4 An eye outside BI .............................. 20
   1.5 Our contributions .............................. 20

2 State of the Art ........................................ 23
   2.1 Introduction ................................. 23
   2.2 Main concepts in Recommender Systems .............. 24
       2.2.1 Recommender Systems function and models .... 24
       2.2.2 Modern Web search recommendations .......... 26
       2.2.3 Composite Items Recommendations .......... 28
       2.2.4 Evaluating Recommender Systems .......... 29
   2.3 Recommender Systems for BI ................. 30
       2.3.1 BI interactions ............................ 30
       2.3.2 Discovering and representing User Intents in BI ... 30
       2.3.3 BI Recommender Systems ................. 31
       2.3.4 Our positioning in BI Recommender Systems ... 32
   2.4 Recommender Systems for Web search .......... 34
       2.4.1 Search interactions ...................... 34
       2.4.2 Discovering User Intents in Web search .... 34
       2.4.3 Web search Recommender Systems based on User Intents ... 39
   2.5 Recommending Composite Items .............. 41
       2.5.1 Principles .................................. 42
       2.5.2 Advantages and disadvantages of bundle building approaches ... 46
       2.5.3 A focus on Cluster And Pick method .......... 47
   2.6 Conclusions .................................... 48
3 Identifying User Intents 51
  3.1 Formal model of BI interactions ........................................... 52
    3.1.1 BI Questions, Suggestions and Queries .............................. 52
    3.1.2 Observations, Interactions and User Intents ....................... 54
    3.1.3 Running example ..................................................... 55
  3.2 Characterizing and clustering User Intents ........................... 56
    3.2.1 User Intent description features .................................. 57
    3.2.2 Clustering User Intents ............................................. 59
  3.3 Evaluation of User Intent detection ..................................... 60
    3.3.1 Experimental protocol .............................................. 61
    3.3.2 Lessons learned ................................................... 63
  3.4 Chapter conclusions ..................................................... 69

4 Intent-based recommender system 71
  4.1 Introduction ............................................................. 71
  4.2 Item-based Recommender System ....................................... 71
    4.2.1 Learning the Markov model ....................................... 72
    4.2.2 The prediction algorithm ......................................... 72
  4.3 Evaluating the Intent-based Recommender System ..................... 73
    4.3.1 Experimental protocol .............................................. 73
    4.3.2 Lessons learned ................................................... 77
  4.4 Chapter conclusions ..................................................... 81

5 Recommending Complementary Items 83
  5.1 Introduction ............................................................. 83
  5.2 User Intents space to compare entities ................................ 84
    5.2.1 Identifying User Intents .......................................... 85
    5.2.2 Data model ......................................................... 86
  5.3 Problem statement ...................................................... 88
    5.3.1 Hard and soft constraints identification .......................... 88
    5.3.2 Bundle building as a constrained clustering problem ............ 89
    5.3.3 Constraints formalization as clustering penalty terms ........... 90
    5.3.4 Objective function ............................................... 92
  5.4 Bundle building algorithm ............................................. 93
    5.4.1 Membership and center update ................................... 93
    5.4.2 Greedy heuristic for bundle construction ........................ 94
    5.4.3 Main algorithm ................................................... 95
  5.5 Testing our bundle building approach .................................. 97
CONTENTS

5.5.1 Dataset ................................................. 97
5.5.2 Assessing the User Intents quality ....................... 98
5.5.3 Evaluation of BIbR ........................................ 103
5.6 Chapter conclusions ........................................... 119

6 Conclusions and future work .................................. 121
  6.1 Conclusions ................................................. 121
  6.2 Future work ................................................. 122

A Business needs .................................................. 125

B Metrics to evaluate the quality of discovered intents .... 127

C Stages of prototype testing by internal SAP users ........ 129

Index ................................................................. 146
List of Tables

1.1 Example of an exploratory search of a user interrogating a BI platform using keywords. There are several repetitive questions that ask for the revenues and the growth of sales for certain media in Spain. 18

2.1 Discovering user intents techniques in BI. These approaches use in general the current session or query to recommend, and some of them exploit past logs. 32

2.2 Some examples of the groups of features implemented in feature-based metrics of state-of-the-art approaches. They are separated in two categories of features learned over data (Data Features) and derived ones (Derived Features). The groups of features here include several features of the same type, that in general explore the same log detail, as for example: [Wang et al., 2013] employs features related to query words and the session details of observations. 36

2.3 Discovering user intents techniques in Web. 39

2.4 Soft and hard constraints of CI approaches implemented by the state-of-the-art works. 46

3.1 Basic characteristics of user intents 56

3.2 Summary of observations in the running example 56

3.3 Characteristics of user interests in the running example 57

3.4 Features considered 58

3.5 Analysis of business needs 61

3.6 Clustering results with distinct subset of features on COMPLETE data set. For short, Rec, Prec and ARI denote respectively recall, precision and ARI scores. 64

3.7 Feature weights 65

3.8 Behaviour of GF2 set of features with PAM clustering when learning weights over observations or analyses. Column “Weighting” indicates whether weights are learned over observations or analyses. 65

3.9 Average dissimilarities between clusters 66
LIST OF TABLES

3.10 Silhouette coefficients and diameter for 10 clusters obtained with PAM using GF2 based dissimilarities. ............................................................... 66
3.11 Generalization of our approach. Each test correspond to the training of the metric and discovery of user intents on different subsets of business needs. ............................................................... 67
3.12 Comparison of our metric based on GF2 features with other metrics when paired with PAM clustering. ALL denotes the set of 15 features, [Aligon et al., 2014b] is the state-of-art metric and “+” indicates a metric with added features and corresponding weights. ............................................................... 68
3.13 Increasing the number of clusters to detect intra-interaction intents. Dense UI indicates the number of clusters with more than 5 different users. Expertise indicates the number of clusters with both types of users (beginners and experts). ............................................................... 69
4.1 Transition probabilities for the 10 states (clusters) of the recommender’s Markov model ............................................................... 78
5.1 An overview of the datasets that are used to learn user intents and to test our recommender system. ............................................................... 99
5.2 Features description and their associated metrics ............................................................... 100
5.3 Weights of features for groups GB1-GB4, learned on D1 dataset with a linear SVM. The results show that folder and LDAqueryParts are the two most decisive features, independently of the number or the other features that are implemented in LFCMD. ............................................................... 101
5.4 Comparison of different feature groups to discover 18 intents. Finer intents show better the impact of the choice of features in the quality of the discovered user intents. Here it is clearly visible the advantage of G4 to the other groups. ............................................................... 102
5.5 Comparison of different feature groups to discover 10 intents. The results are very similar, as the decreasing number of clusters yield more homogeneous intents, but we can identify G4 as the group of features slightly more accurate than the others. ............................................................... 102
5.6 Comparison of algorithms for the identification of 10 intents ............................................................... 103
5.7 Comparison of algorithms for the identification of 18 intents ............................................................... 103
5.8 Some of the best combinations of constraint’s weight ............................................................... 106
5.9 Constraints’ weights for each of the 5 bundles we propose to complete a user report online. ............................................................... 108
5.10 Constraints weights to evaluate the contribution of uniformity and diversity 118
List of Figures

2.1 Document-to-query transition. In this example, the most likely transition from document D2 is to Q3, because the edge has 70 clicks (as opposed to 50 to Q2 and 10 to Q1). ................................................................. 36

3.1 An example of the Search Panel in a BI Platform, where the users can write questions in natural language. They are then mapped to formal queries, which generates 5 suggestions and visualize the result of the chosen queries. 53

3.2 The analogy of the observation model of Web, presented by [Guha et al., 2015], to BI interaction. ................................................................. 54

4.1 Coverage and accuracy for IbR1, IbR2, Falseto and QueRIE. ............... 79
4.2 Expected diversity and expected user for IbR1, IbR2, Falseto and QueRIE. 79
4.3 Expertise and expertise benefit for IbR1, IbR2, Falseto and QueRIE. ....... 80
4.4 Accuracy for QueRIE on the entire log and on each user intent ............. 81
4.5 Coverage and accuracy for Falseto on the entire log and on each user intent 81

5.1 User intent discovery framework as presented in Chapter 3 .................. 85
5.2 Caption for LOF ................................................................. 86
5.3 Distribution of global scores for all testers and each bundle strategy. .... 109
5.4 Distribution of uniformity scores for all testers and each bundle strategy. 110
5.5 Distribution of diversity scores for all testers and each bundle strategy. ... 110
5.6 Distribution of personality scores that all testers gave to each bundle. This box-and-whisker plot shows as well the minimum and the maximum score attributed to them. ................................................................. 111
5.7 Personalization of recommendations for each user. ............................ 112
5.8 Precision for different similarity thresholds, for a simulated ideal user IU, whose profile is degraded with 20% and 40% of noise, to be compared with a real user profile RU. ................................................................. 114
5.9 The degradation of the context profile decreases the precision of the recommender system. ................................................................. 115
5.10 Integrating user and context profile in objective function. .................... 116
LIST OF FIGURES

5.11 Comparing the integration of a user profile with both a good (C1) and a
deteriorated (C2) context. ................................................... 116
5.12 Comparing the recommended bundle with the real future, considering the
order of queries suggested. ................................................... 117
5.13 Balancing uniformity and the diversity constraints gives more precise bundles. 118
5.14 BOBO vs CPS vs BiBr ....................................................... 120

C.1 The open page of our testing application, where the user can choose the
report for which she wants to consult recommended bundles. ............... 129
C.2 Caption for LOF ............................................................ 130
C.3 The user can click on 5 tabs to visit the 5 bundles generated from 5 different
strategies, each composed of a list of 5 queries, that we propose for the given
report. ................................................................................. 130
C.4 The form the user has to fill to rate each bundle with 1 to 5 points for
each constraint. We asked a global score as well to evaluate the tuning of
constraints’ weights. ........................................................... 131
Acknowledgments

A doctoral thesis is often perceived as a solitary endeavor, as it requires time and dedication for all its duration. But in the end, I am glad to see the long list of people I have met, and I am very thankful for their collaboration.

First, I would like to express my sincere gratitude to my advisors, for their continuous support and their guidance, improving my knowledge in machine learning, databases, and all the research relevant to this Ph.D. I want to thank Nicolas Labroche for his academic support, his explanations, patience and his sense of humor that made me enjoy this research project and made it less stressful. Thank you a lot for all the motivation from the day of my first interview to the final day of the thesis defense! For this, and all the discussions and help I got, I could not imagine having a better thesis director and mentor for my Ph.D. I appreciate a lot the job that Patrick Marcel has done with me, all the ideas, immense knowledge and critics. Under his supervision, I learned how to define a research problem, how to be critical to its solutions, and finally publish the results. Besides research guidance, on a personal level, I want to thank Veronika Peralta for her care and hospitality and for her support during tough moments of this journey. She is an inspiration for me, as a woman dedicated to research in informatics, and for her communication skills. The joy and enthusiasm they all have for research was contagious and motivational for me. It has been an honor for me to be one of their Ph.D. students.

Not less important was my project manager on the industry side, Alexis Naibo. I am indebted to all his support and encouragement to pursue this Ph.D study. My sincere thanks for allowing me to fully work over my research and fighting to find all the possible resources to exploit at SAP. Thank you for believing in me and making me believe in myself, I specially appreciate your advices for my professional and personal development.

Besides my advisors, I would like to thank the rest of my thesis committee members: Sihem Amer-Yahia, Prof. Jean-Marc Petit, Prof. Gilles Venturini and Prof. Christophe Marsala, for their insightful comments and for their questions, which I appreciated a lot, and allowed me to explain details of my work and widen my research from various perspectives.

Both the BTNL team at the University of Tours in Blois and the team I joined at SAP, made my time there enjoyable. I would like to express my sincere gratitude to all of them, for making me part of their team, their support and for the access they gave me to
laboratory and enterprise resources. The work of Julien Aligon was one of the bases for my research. Thank you for your support, collaboration and your encouragement. Not less important were Arnaud G., Thomas, Jean-Yves, Béatrice, Arnaud S., Dominique, Agata and Adnan, Marwa, Frédéric, Mahfoud that accompanied my time in Blois. I want to thank specially two students of the University of Tours, Alexandre Chanson et Ben Crulis for all the scientific discussions we had, their contribution in the last chapter of my thesis and also for the fun we had together in the last stressful months.

User experiments and the analysis at SAP would have not been possible without the help of Bruno Dumant, Gregoire Byrka, Kevin Le Fur, Alexandre Fortin and Pierre Saurel. I am grateful to everyone that I worked in every project I participated at SAP and continuously supported me to pursue my research. The timely completion of this dissertation would have been impossible without their help. Besides research, it was a real pleasure to exchange and to co-teach Data Science knowledge with Yann Le Biannic and Bertrand Lamy. A special note of thanks for Xiaohui, Valeria, Ludovic, Alexis, Simon, Eric, Florent, Olivier, Peter and Gildas for their encouragement, confidence, the sportive and joyful time together.

In the end, I would like to thank my family, who was in fact the first who trusted me, without even going through interviews. This thesis is a dedication to my parents, who raised me with a love for science, made this choice of pursuing my studies with a doctoral research look natural and taught me the way to go through each challenge; and to my sister, who was my inspiration of hardworking and well organizing. Thank you for your patience and for supporting me throughout this thesis. I want to specially thank Vilma for her support, appreciation of my work and for helping me with the final language corrections of this thesis.
Abstract

The shift of companies to the Cloud encourages the storage of a big amount of data from different sources. The incorporation of multi-sourcing and unstructured data is bringing new challenges to Business Intelligence (BI) systems and complicating user interactions with the system. In general, BI users interact with the system the same way, i.e., they navigate large datasets through sequences of analytical queries. Different users accessing a searching system can formulate different set of queries, that can be issued several times, to satisfy the same information need. Or, on the contrary, they may expect different answers to the same query, as they have different profiles, preferences or they are situated in a different context. Even modern BI systems are not capable of identifying users’ intents, which may lead to a series of repetitive and long questions towards the expected solution.

Recommender systems appear as a natural solution to help the users complete their analysis in BI systems. In general, they try to discover user behaviors from the past logs and to predict to users personalized actions in their future interactions, or to use environment information to filter the recommended items. Most of recommender systems consider either the past logs or the short-lived environment information, which leads to ambiguous predictions. Traditional recommender system applies data analysis techniques to the problem of helping users find the items that correspond to their research by predicting lists of likeness scores, which may lead to redundant recommendations. Nowadays, diversity is becoming increasingly essential and getting more and more attention for improving users’ satisfaction. Recent research on recommender systems has been focused on improving the profitability of the suggested items, which actually depends on the other items recommended as a set. Thus, a special interest is dedicated today to complementary recommendation, that beyond providing a list of personalized items, satisfies user with new, diverse and contextual information.

This thesis being a joint partnership between SAP Labs in France and the University of Tours, we had the opportunity to study two concrete data exploration problems from two different BI systems at SAP. The first problem is related to the difficulty of users to reach an information need navigating in a BI Platform. The second use case focuses on proactively assisting the user completing a BI report in editing mode. A BI report is a dynamic report composed of graphical query results that respond to a user analysis, whose content is updated each time the interrogated databases are refreshed. For both cases, it is tedious for the new users, not familiar with the data, to continue completing their sessions or reports with relevant queries. We need a personal assistance that actively or proactively
help, respectively, searchers and report editors to complete their tasks easier and faster, and provide new updated information.

Inspired from the resemblance of modern BI systems with Web search engines, we aim at finding, adapting and applying successful models of Web in BI systems to yield them intelligent and more user friendly. We study and develop the user intent detection in BI platforms and how it helps to devise recommendations of queries, for the user to pursue more efficiently her search in the current session. To leverage the identified user intents for the purpose of query recommendation, we present two recommender systems. The first, an original reactive collaborative Intent-based Recommender system, named IbR, that recommends sequences of queries for the user to pursue her analysis. The second recommender, named Bundle Intent-based Recommender (BibR), is close to proactive systems and proposes a bundle of queries to complete user analysis in a BI report, based on the user short and long term intents. BibR adopts the composite item recommender of Task Composition approaches to recommend bundles of complementary visualized queries, exploring user preferences and the report context. This research problem is completely new in the domain of BI.
Abstract

La transition des entreprises vers le cloud permet le stockage d’une grande quantité de données provenant de sources différentes. L’incorporation de données non structurées et multi-sources apporte de nouveaux défis aux systèmes de Business Intelligence (BI) et complique les interactions des utilisateurs avec le système. Cependant, tous les utilisateurs de BI interagissent avec le système de la même manière, à savoir qu’ils naviguent dans des jeux de données volumineux au moyen de séquences de requêtes analytiques. Différents utilisateurs accédant à un système de recherche peuvent formuler différents ensembles de requêtes pour satisfaire le même besoin d’information, ou, au contraire, ils peuvent s’attendre à différentes réponses pour la même requête, car ils ont des profils, des préférences ou des contextes différents. Même les systèmes de BI modernes ne sont pas capables d’identifier les intentions des utilisateurs, ce qui peut conduire à un processus long et répétitif de questions vers la solution attendue.

Les systèmes de recommandation apparaissent comme une solution naturelle pour aider les utilisateurs à aboutir dans leur analyse au travers des systèmes de BI. En général, ces systèmes essaient de découvrir le comportement des utilisateurs à partir de données historiques, et de prévoir des actions personnalisées pour les utilisateurs lors de leurs interactions futures, ou d’utiliser des informations d’environnement pour filtrer les éléments recommandés. Le système de recommandation traditionnel applique des techniques d’analyse de données pour aider les utilisateurs à trouver les éléments correspondant à leur recherche, en prédisant des listes de notes de similarité, ce qui peut entraîner des recommandations redondantes. De nos jours, la diversité des résultats est de plus en plus importante et devient essentielle pour améliorer la satisfaction des utilisateurs. De récentes recherches sur les systèmes de recommandation se sont focalisées sur l’amélioration de la rentabilité des éléments suggérés, qui dépend en réalité des autres éléments recommandés en tant qu’ensemble. La tendance est aujourd’hui aux recommandations complémentaires, qui au-delà d’une liste d’articles personnalisés, satisfont les utilisateurs avec de nouvelles informations, diverses et contextuelles.

Cette thèse étant un partenariat entre SAP Labs in France et l’Université de Tours, nous avons eu l’occasion d’étudier deux problèmes concrets d’exploration de données provenant de deux systèmes de BI différents chez SAP. Le premier problème est lié à la difficulté des utilisateurs à répondre à un besoin d’information en navigant dans une plateforme de BI. Le second consiste à aider de manière proactive l’utilisateur à compléter un rapport BI en mode édition. Un rapport BI est un rapport dynamique composé de résultats de requêtes...
graphiques répondant à une analyse utilisateur, dont le contenu est mis à jour chaque fois que les bases de données interrogées sont actualisées. Dans les deux cas, il est recommandé aux nouveaux utilisateurs, non familiarisés avec les données, du contenu pour poursuivre leur session ou des rapports contenant des requêtes pertinentes. Nous avons besoin d’une assistance personnelle qui aide de manière active ou proactive, respectivement, les analystes et les éditeurs de rapports à effectuer leurs tâches plus facilement et plus rapidement, ainsi que de fournir de nouvelles informations mises à jour.

Inspirés par la similarité des systèmes de BI modernes avec les moteurs de recherche Web, nous allons étudier, adapter et appliquer des modèles reconnus du Web dans les systèmes de BI pour les rendre plus intelligents et plus conviviaux. Nous étudions et développons la détection de l’intention utilisateur dans les plateformes de BI et la manière dont elle aide à élaborer des recommandations de requêtes, pour que l’utilisateur puisse mieux poursuivre sa recherche dans la session en cours. Afin de tirer parti des utilisateurs identifiés pour ces recommandations de requête, nous présentons deux systèmes de recommandation. Le premier, un système de recommandation réactif original basé sur l’intention des utilisateurs, appelé IbR, recommande des séquences de requêtes à l’utilisateur pour poursuivre son analyse. Le second système, appelé Recommender-based Bundle (BIbR), est proche de systèmes proactifs et propose un ensemble de requêtes permettant de compléter l’analyse de l’utilisateur dans un rapport BI, en fonction d’intentions à court et long terme de l’utilisateur. BIbR adopte les approches de recommandation d’éléments composites pour proposer des ensembles de requêtes visualisées complémentaires, en explorant les préférences de l’utilisateur et le contexte du rapport. Ce problème de recherche est complètement nouveau dans le domaine de la BI.
Chapter 1

Introduction

This chapter introduces the problems this thesis tackles in the Business Intelligence (BI) domain and the specific context of our work. Section 1.1 describes today situation of growing data and the difficulties we are facing to explore and learn data to make user searching less tedious. Section 1.2 gives more details about particular examples of concrete needs for the BI platform at SAP, of intelligent personal assistants to help analysts in their everyday tasks. We present the challenges we identify in BI domain in Section 1.3 and inspirations we find in Web Search to resolve them, in Section 1.4. By the end of this thesis introduction, we precise our contributions in Section 1.5.

1.1 Exploring evergrowing data

The shift of companies to the Cloud encourages the storage of a big amount of data from different sources. Users of Business Intelligence (BI) systems, that benefit from this centralized information to take their decisions more efficiently, range from executives to data enthusiasts. In general, they all interact with the system the same way, i.e., they navigate large datasets through sequences of analytical queries. Different users accessing a searching system can formulate different set of queries, that can be issued several times, to satisfy the same information need. Or, on the contrary, they may expect different answers to the same query, as they have different profiles, preferences or they are situated in a different context.

The incorporation of multi-sourcing and unstructured data is bringing new challenges to BI and complicating user interactions with the system. More and more professionals, analysts or decision makers, are facing the problem of analyzing evergrowing amounts of data through the existing BI systems to understand the user intents, which are characterized by the ensemble of the user past queries and past consulted results and express a single user need. Even modern systems are not capable of identifying users’ intents, which may lead to a series of repetitive and long questions towards the expected solution. It is well documented that it usually takes many interactions with the system to satisfy an information need, and the overall session is often a tedious process, especially when the information need is not even clear for the user [Sarawagi, 1999, Sapia, 2000, Zhao et al., 2017, Milo et Somech, 2018].
Recommender systems appear as a natural solution to help the users complete their analysis in BI systems. In general, they try to discover user behaviors from the past logs and to predict to users personalized actions in their future interactions, or to use environment information to filter the recommended items. Most of recommender systems consider either the past logs or the short-lived environment information, which leads to ambiguous predictions [Ustinovsky et Serdyukov, 2013].

Traditional recommender system applies data analysis techniques to the problem of helping users find the items that correspond to their research by predicting personalized lists of likeness scores, which may lead to redundant recommendations. Nowadays, diversity is becoming increasingly essential and getting more and more attention for improving users’ satisfaction. Recent research on recommender systems has been focused on improving the profitability of the suggested items, which actually depends on the other items recommended as a set. Thus, a special interest is dedicated today to complementary recommendation, that beyond providing a list of personalized items, satisfies user with new, diverse and contextual information.

1.2 Problems that BI analysts are facing in industry

This thesis being a joint partnership between SAP Labs in France and the University of Tours, we had the opportunity to study two concrete data exploration problems from two different BI systems at SAP.

The first problem is related to the difficulty of users to reach an information need navigating in a BI Platform. To simplify user interactions, SAP provides a modern BI system\(^1\), that features graphical interfaces and keyword search engines. They hide the real complexity of querying data and the organization of multiple data sources, offering users simple navigation frames. For example, Table 1.1 shows a user interaction and how she may express her information needs via keywords, letting the system infer from them the most likely formal queries (generally MDX or SQL) to be sent to the underlying data sources (generally data warehouses or databases).

<table>
<thead>
<tr>
<th>BI Platform interaction</th>
<th>Query 1</th>
<th>Sales Revenue, Genre, media format for Country='Spain'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 2</td>
<td>Sales Revenue, Genre, media format for Country='Italy', 'France', 'Spain'</td>
<td></td>
</tr>
<tr>
<td>Query 3</td>
<td>Sales Revenue / Sales Target Revenue, Retailer for Year='2013'</td>
<td></td>
</tr>
<tr>
<td>Query 4</td>
<td>Sales Revenue growth 2013 as Date Year for Genre='drame'</td>
<td></td>
</tr>
<tr>
<td>Query 5</td>
<td>Revenues, media format for Country='Spain'</td>
<td></td>
</tr>
<tr>
<td>Query 6</td>
<td>Sale Target, film, Genre for Country='Spain'</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: Example of an exploratory search of a user interrogating a BI platform using keywords. There are several repetitive questions that ask for the revenues and the growth of sales for certain media in Spain.

\(^1\)Here is the patent reference of the SAP BI system that permits interrogating the databases in natural language, similar to a search engine : us 20180157734A1: Business Intelligence System Dataset Navigation Based on User Interests Clustering
1.3. OUR CHALLENGES

The second use case focuses on proactively assisting the user completing a BI report in editing mode. A **BI report** is a dynamic report composed of graphical query results that respond to a user analysis, whose content is updated each time the interrogated databases are refreshed. These reports are organized in documents shared between users, which can be viewed, edited or deleted. Users can add new queries in their reports by drag-and-dropping query parts from a business layer, built upon a database. A report can be accessed and modified from multi users and it may include queries originating from different data sources. Thus, it is tedious for the new users, not familiar with the report, to continue completing them with queries relevant to the report and to their own research.

For both use cases, we need a **personal assistance** that actively or proactively help, respectively, searchers and report editors to complete their tasks easier and faster, and provide new updated information.

1.3 Our challenges

Being able to automatically identify user intents from past logs is a challenging problem as intents are hidden in the interactions, and two users with the same interest would probably interact with the system differently depending on their previous knowledge or exploration context. User intents have many potential applications, such as the suggestion of interesting data found by other users, repetitive task prediction or alert triggering, that would help reduce the analysis’ tediousness. In all cases, it is the users’ historic search profiles that help them resume ongoing tasks [Kotov et al., 2011] or complete tasks by leveraging the interactions of other users, stored by the system from their previous searches [White et al., 2013].

Modern BI system makes it possible to exploit a rich basis for discovering user intents such as: composing keywords, sources, formal queries, etc. The difficulty stands on the choice of the algorithm to discover user intents, as well as on the evaluation techniques to verify that they really correspond to what was expected by the users. Last but not less important is the requirement of providing a generic approach, easily adaptable in different BI systems.

Recommending relevant queries for BI interactions has attracted much attention recently, either in the context of On-Line Analytical Processing (OLAP) explorations (see e.g., [Aligon et al., 2015]) or SQL sessions (see e.g., [Eirinaki et al., 2014]). However, to the best of our knowledge, none of the proposed approaches in this domain extract and use explicitly the user intents to generate recommendations. In BI context, our recommendations aim at suggesting the desired information to make the data exploration less tedious and to facilitate the completing of BI reports. Our main objective is to demonstrate the practical benefit of clustering user intents in the context of BI interactions and, more specifically, in implementing intent-based query recommender systems.
1.4 An eye outside BI

BI systems try to facilitate users’ interactions by simplifying the queries they trigger and hiding the complexity of the data sources, resembling more to Web search engines. Similarly, users on Web typically need to repeatedly query the search engine to find interesting content, but differently from BI systems, Web engines are becoming more intelligent and more aware of the users’ global intents, intentions and their surrounding when performing searches [White, 2016]. This was made possible by several improvements in user modeling, semantic understanding of queries and results through machine learning and artificial intelligence. In Web Search, the literature approaches [Guha et al., 2015, Song et Guo, 2016, Yang et al., 2016] characterize user intents by means of features extracted from user traces and classify them in order to group queries related to the same information needs.

[White, 2016] considers search systems close to personal assistants, as they learn from a variety of signals from their environment that define the user context, to support the current searcher. Assistants react to queries provided by the searcher, anticipating future needs and making recommendations based on intents, explicitly communicated by the search or inferred by the system based on people’s activity.

Modern recommender system should satisfy other user constraints rather than just suggesting relevant content to the user and her context. The fine balance between the relevance and the diversity is a new challenge of package recommendation systems. The difficulty of the problem stands on finding a technique that suggests a set of items, that is relevant to the user profile, relevant to the report, but in the same time is diverse and fit to the current content of the report. There are relatively few works dedicated to recommending Composite Items in crowd-sourcing ([Alsayasneh et al., 2018, Amer-Yahia et al., 2016]) and to our knowledge this is a new problem presented in BI domain.

1.5 Our contributions

Our first contribution is the formalization of some concepts in the context of BI. We start by database entities that make our logs, and then we describe all components and metrics of our recommender systems. As the problems we treat are new in the BI domain, we have defined them and the data they explore formally. Inspired from the resemblance of modern BI systems with Web search engines, in this thesis, we aim at finding, adapting and applying successful models of Web in BI systems to yield them intelligent and more user friendly. We study and develop the user intent detection in BI platforms and how it helps to devise recommendations of queries, for the user to pursue more efficiently her search in the current session. Contrary to Web search engines which consider all the users equally, in a BI context we aim at helping novice analysts with recommendations from experts’ actions. With that in mind, our objective is to provide recommendations for both type of users: experts and beginners, and to solve two problems: continuing a BI exploration in a session and completing reports with queries. To be more precise, we recommend existing queries from other reports, which results are
associated with a particular graphical representation. In the end, what the user will see is a set of visualizations for the suggested queries.

To leverage the identified user intents for the purpose of query recommendation, we present two recommender systems. The first, an original reactive intent-based recommender system, named \( \text{IbR} \), that recommends sequences of queries for the user to pursue her analysis. It is a collaborative recommender system that adopts a technique first proposed in the literature for data prefetching [Sapia, 2000] and later extended for query recommendation [Aufaure et al., 2013]. The second recommender, named Bundle Intent-based Recommender (\( BIbR \)), is close to proactive systems and proposes a bundle of queries to complete user analysis in a BI report, based on the user short and long term intents. \( BIbR \) adopts the composite item recommender of [Amer-Yahia et al., 2016] to recommend bundles of complementary visualized queries, exploring user preferences and the report context. This research problem is completely new in the domain of BI. The bundle creation problem is treated as a constrained clustering problem, and the recommended bundle should fulfill all of them. We adapt and extend this work, compared to [Alsayasneh et al., 2018], to provide contextual and ordered suggestions, particular constrained for our use case. All the constraints are integrated in an objective function that we aim to optimize in order to provide qualitative bundles.

More precisely, the contributions of this thesis include:

- the learning of a similarity measure based on a set of features for characterizing BI user intents,
- an approach to automatically discover user intents based on our measure and an off-the-shelf clustering algorithm,
- \( \text{IbR} \), a simple yet effective user intent-based collaborative recommender system inspired by earlier works [Aufaure et al., 2013] in OLAP query prediction, specifically designed to take advantage of the clusters identified, that can recommend a sequence of queries to complete an existing interaction,
- \( BIbR \), a composite item approach that recommend a bundle of ordered visualized queries to complete a report, based on user intents and her context,
- an extensive set of experiments for the tuning and validation of our approaches offline and through a user study, including the comparison of our similarity measure with a state-of-the-art metric tailored for OLAP queries [Aligon et al., 2014b] and the comparison of our recommender systems with state-of-the-art recommender systems agnostic of user interests, \( \text{IbR} \) to [Eirinaki et al., 2014, Aligon et al., 2015] and \( BIbR \) to [Amer-Yahia et al., 2014, Bota et al., 2014].
1.5. OUR CONTRIBUTIONS

This thesis is organized as follows:

Chapter 2 gives an overview of related work. It starts by a general review of recommender systems, their classification and some essential concepts. This chapter introduces the type of logs we can explore, the current challenges of learning user intents and some first attempts of personalizing predicted result, both in a BI and Web environment. At the end, this chapter presents the latest trends of building composite items that are complementary to the current context, and our positioning between BI and Web systems.

Chapter 3 presents our formal model of BI interactions and explains how we identify user intents by characterizing the intent of the interaction with the BI system. We identify a set of features for an adequate description of intents, and a similarity measure for grouping logged observations into coherent intents. We validate experimentally our approach with a user study and we compare it to a state-of-the-art approach.

Chapter 4 introduces our intent-based recommender system, designed to exploit the clusters that represent user intents. It recommends a sequence of queries representing the sequence of moves that is expected to best complete the beginning of an interaction. This recommender stands on an order-1 Markov model to probabilistically represent user behaviors, where each state is a cluster constructed from a set of past user interactions. It can be seen as a model that guides the user’s next moves based on the probabilities of moving between discovered user intents.

Chapter 5 presents a second recommender system that proactively help users to complete BI reports with an adequate set of visualized queries. Similarly to the system introduced in the previous chapter, this is another application based on the discovery of user intents, but augmented with constraints pertaining to the user’s short-term and long-term interests, diversity of query results, etc., that tries to improve the effectiveness of the recommended queries and make the data search process less tedious. It shows how we use the identified user intents to represent entities over a uniform vectorial space.

Chapter 6 concludes the thesis and opens discussions for future works. It positions our proposed recommended systems amongst the state-of-the-art approaches in the BI domain. We resume the results of our experiments and we ask new questions about the future and the evolution of these systems.
Chapter 2

State of the Art

2.1 Introduction

This thesis aims at simplifying and making the users’ analysis more effective, proposing an Intelligent Personal Assistant to BI analysts, decision makers and beginners as well, as a personalized recommender. We focus on exploratory search in the BI context and more specifically on how it is possible to make the exploration process less tedious, by recommending queries to an analyst based on her preferences, interests, habits or context. Exploratory search can be used to describe both an information-seeking problem context that is open-ended, persistent, and multi-faceted, and an information-seeking process that is opportunistic, iterative and multi-tactical [White, 2016]. The emphasis in exploratory search is on learning about the topic of search, understanding collection content, and capitalizing on serendipitous opportunities to pursue particular directions when they emerge, differently from iterative search, where the target of the search is known.

Our business use cases motivated two aspects of our research work: (1) how to personalize and contextualize user interactions during BI exploration and in proactive systems, and consequently (2) how to model user intents in such context. To personalize and contextualize BI exploration, we rely on recommender systems that involve various machine learning techniques, capable to learn from past data. Beyond their generalization and prediction advantages, machine learning models heavily depend on the quality and the quantity of training data, the usage and the quality of user annotations.

Our work targets modern BI platforms that are close to Web applications, letting the users explore business objects such as documents or visualized queries, through natural language questions. On the other hand, many works in the field of recommendation are applied to Web applications [White, 2016], which gave us the opportunity to find a variety of solutions for problems similar to ours. However, their direct application in BI is limited due to the specific business use cases and the fact that, in general, no additional information is provided about the BI users or their environment. In our BI context, the user intents can only be discovered from the logs of past user actions and known business objects’ metadata. One of the challenges in our context will be the extraction of descrip-
2.2 MAIN CONCEPTS IN RECOMMENDER SYSTEMS

tive and discriminating features from logs, that should be representative of the type of problems that is addressed.

This literature survey starts by a general review of recommender systems and some essential concepts in Section 2.2. Section 2.3 introduces the first relevant attempts of recommending queries in a BI environment. It gives an overview of the type of the data logged and the current challenges of learning user intents. The following Section 2.4 presents Web recommendation, what are the detailed inputs that can be logged in Web applications, how they are explored to learn user behaviors, and how we can benefit from learned user intents to personalize recommendations. A particular focus is given in Section 2.5 on recommendation techniques using composite items. Section 2.6 concludes this chapter by detailing our contributions at the crossroads of BI and Web Search.

2.2 Main concepts in Recommender Systems

This section outlines the main concepts of Recommender Systems, their functionalities and the different types of recommenders. The goal is to get familiar with these techniques and to have a global overview of the solutions they offer to solve different cases. We start by recalling some basic concepts in Section 2.2.1, we stress the importance of contextual and personalized recommendations in 2.2.2 and the need of recommending composite items instead of an item list in Section 2.2.3. Finally, we conclude with some evaluation metrics of the recommendation quality in Section 2.2.4.

2.2.1 Recommender Systems function and models

[White, 2016] defines the recommender system as a subclass of information filtering systems, that predicts the preference or the rating that a person would give to an item. It uses historical user ratings to produce an ordered list of recommendations where the most interesting and relevant content to the user needs is suggested [Adomavicius et Tuzhilin, 2005]. Recommenders are now well established as a field of research, with application in a wide range of domains [Adomavicius et Tuzhilin, 2005, Burke et al., 2011], as for example designing the forthcoming query in an OLAP platform [Giacometti et al., 2008], suggesting movies in Netflix [Bell et Koren, 2007], recommending documents [Bennett et al., 2012] or recommending applications that users probably need to complete a task in proactive systems [Song et Guo, 2016]. These last ones are systems that push recommendations without an explicit user request. [Adomavicius et Tuzhilin, 2005] define an utility function $f_u$ of a recommender, that assuming there is a set $Users$ of all the users and a set $Items$ of all items available to recommend, it measures the usefulness of recommended item $i \in Items$ to user $u \in Users$ is defined as:

$$f_u : Users \times Items \rightarrow R \quad (2.1)$$

where $R$ is an ordered set of non-negative numbers, with highest scores corresponding to the most relevant items to user $u$. Finally, we want to choose for each user $u \in Users$ such items $I \in Items$ that maximize the user’s utility.
The utility function $f_u$ can be adapted according to the use case and the item or user profile. The complexity of a utility function varies, from the systems imposing single comparative criteria [Adomavicius et al., 2011] that consider the ratings a user gave to items, to more complex multi-criteria functions that estimate the preferences of a user towards items measuring multiple parameters. The experiments of [Adomavicius et al., 2011] over big or sparse data show that multi-criteria information is advantageous over rating. The use of multi-criteria opens several possibilities, such as recommending to a group of users instead of a final one, modeling their preferences by aggregating the users’ individual atomic preferences [Jameson et Smyth, 2007] or recommending to different stakeholders [Abdollahpouri et al., 2017]. The functions that use a single criteria are heuristic techniques that compute the similarity between items and users based on explicit ratings stored in the logs. They need well completed matrices with ratings over existing items to compute the relevance of items. Contrary to this, model-based techniques learn a predictive model using machine-learning methods. They learn preferences over multiple descriptive criteria of the favorite items, using a linear or non-linear regression, and use this model to predict relevance scores of unknown items. Regression analysis is a collection of statistical techniques that serve as the basis to estimate the linear or non-linear relationship among two or more variables [Golberg et Cho, 2010]. These models provide comparative metrics that stand on the basis of traditional recommender systems.

From different problems of recommending and exploring multiple sources emerged the need of several types of recommenders. They all share the same goal of predicting the missing ratings of unknown items for the user. Classical recommender systems are classified into these following groups [Adomavicius et al., 2011]:

Collaborative filtering techniques [Schafer et al., 2007] recommend based on the information about other users with similar preferences, built over the ratings that they have supplied for common items in the past. It is often implemented for predicting missing ratings in a user-item matrix recording user preferences. It can favor the recommendation of a top set of the most consumed items, preventing the users from seeing the items added recently. In general, these methods cannot handle new users or new items in the system.

Content-based approaches [Ricci et al., 2011] take advantage of the details of items that users liked the most in the past. Ideally, every item should be described with the same amount of details, comparable to each other, which is not always the case. To predict the score of an unknown item for a user, these approaches rely on the ratings she attributed to other observed items that share similar characteristics with it. The challenge of this approach is the extraction of descriptive features and the definition of the appropriate similarity metrics between them.

Knowledge-based systems use user and items’ insights to find the items that meet the user requirements. It is considered as a richer, sub-type of content-based approach [Burke et al., 2011]. The knowledge of items is extended to user preferences. The value of knowledge-based approaches is that they provide a model of users or items that can be
2.2. MAIN CONCEPTS IN RECOMMENDER SYSTEMS

built over annotations and tags (folksonomy) [Shepitsen et al., 2008], or their preferences can be described in terms of semantic technologies, concepts and ontologies [Cantador et al., 2008, Ghani et Fano, 2002]. The difficulty of this method stands in collecting and treating information to build and update a good user profile.

Hybrid approaches combine the previous techniques in many different ways, in order to make them overcome their disadvantages and to profit from the capacities of the other techniques. [Burke, 2002] introduces a new hybrid system that combines knowledge-based recommendations and collaborative filtering. It improves the system performance showing that semantic ratings obtained from the knowledge-based part of the system enhance the effectiveness of collaborative filtering.

2.2.2 Modern Web search recommendations

[Teevan et al., 2005] indicate that there is an opportunity to achieve significant improvement by adjusting search results to individuals, as different people have been shown to expect different results for the same queries. Modern web search engines now integrate various techniques for achieving such adjustments, detecting multi-tasking (Section 2.2.2.1), using personalization and contextualization (Section 2.2.2.2), reactivity and proactiveness (Section 2.2.2.3) or long or short-term intent detection (Section 2.2.2.4).

2.2.2.1 Exploring multi-task sessions

Historically, a session is defined as a set of queries issued temporally close to one-another to satisfy a single information need [Li et al., 2014, Jiang et al., 2011]. It is recognized as an accurate atomic unit for modeling user intent. [Lucchese et al., 2013] propose the concept of a “task-based session” for queries within the same session that serve to a common search intent. There are simple task models, often performed within a single search session [White, 2016]. Session is used as a crude approximation for a search task in the absence of more sophisticated models. It can be true for simple needs, i.e. searching between favorite documents, which can be satisfied with a few queries, but on current systems, there are seldom users that log in and out on a task-by-tasks basis. Other works [Lucchese et al., 2011] show that users tend to perform more than one task at the same time, involving a multi-tasking activity. This is the ability to simultaneously handle the demands of multiple tasks through task switching in a session. Identified intents can be placed over multiple days, multiple sessions, as well as multiple intents which may occur in the same session. Multi-tasking is common in exploratory search [Spink et al., 2006] and can lead to interwoven task histories that are difficult to disentangle. [Joachims, 2002, Radlinski et Joachims, 2005, White et al., 2013] leverage the aggregated historic behavior of many users to identify resources of interest to the current user given their task.

It is common in Web search to find users engaged in multiple tasks simultaneously [Spink et al., 2006], that may want to examine multiple items or mitigate networks constraints by requesting many services concurrently. For example, Web browsers support simultaneous navigation in multiple Web pages, e.g. listening to music while playing and
chatting in other pages. Session multi-tasks indicate the user context and can help the system to better understand her intents to further individualize the system to the current user.

### 2.2.2.2 Personalization and contextualization

In general, literature works introduce two axes for individualizing the search experience to a particular search engine user: personalizing to user behavior and using current context. [White, 2016] describes **personalization** as a process of modeling searchers’ activity over time, or other sources of data such as the content of accessed documents or metadata information on the user’s machine. Importantly, personalizing users experience requires tracking their long-term intents.

In contrast to personalization, that involves tailoring individual user experience and collecting it implicitly over a prolonged period of time, **contextualization** is presented in [White, 2016] as the process of modeling the user situation and environment, rather than the user profile. It is defined by the ensemble of recent interactions from the beginning of the current session, beyond evidence gathered from queries and click data alone. Contextualization is usually characterized by additional signals, as geographic location, time, and recent interaction behavior, representing the user’s short-term intents.

The experiments of [Wen et al., 2018] demonstrate how recommendation performance for different users may be affected by time-sensitive historical user data filtering. They found that incorporating outdated data or different filtering percentage of user past data is likely to degrade performance because user’s preferences and her context change over time. Personalization and contextualization have gained a lot of attention over the past years because of the growth of Intelligent Personal Assistants (IPAs), which are **reactive and proactive systems** that are able to provide recommendations without an explicit user request.

### 2.2.2.3 Reactive and Proactive Intelligent Personal Systems

The growth of mobile devices has increased the interest in IPAs, a new paradigm of recommendation well known in mobile applications, i.e. Google Now [Guha et al., 2015], Cortana [Microsoft, 2014] (both reactive and proactive systems, respectively, responding to requests and pushing recommendations) or Siri [Apple, 2011] (currently reactive only). They are armed with knowledge about the searcher’s situation and preferences, and capable of helping her to find the information needed, to complete tasks and to manage her everyday activity more effectively [White, 2016]. IPAs track users’ activity and try to understand their intents, to then provide individual services and information, such as news, weather, scheduled events etc., accessing different sources. It is essential for the mobile personal assistants to have at disposal a good user model and to be capable of identifying the user context, in order to provide coherent and contextual recommendations. Models can be constructed from previous queries and pages the searcher has browsed, irrespective of the explicit usage of search engines to reach those resources.
The goal of proactive systems is to reduce the number of explicitly issued queries [White, 2016]. On the other hand, recommenders in reactive systems start predicting future queries once the user is logged in the system and iteratively formulates queries to respond to one or more needs. Experience shows that users rarely give explicit feedback over their preferences or suggestions made to them previously. Using only the most recent queries in isolation offers limited information about the user intents. Thus, to understand the user behavior and needs, almost all the reviewed methods exploit the hidden information in the log files, which in general consists of unorganized data. Discovering the user intents over the entire past query sessions is a possible way of overcoming the limited user opinion and boosting the system performance. As a consequence, there is a need for learning long-term intents that can be learned over logs, as well as to identify the short-term user context intents in the actual session, to suggest items that are coherent with her current situation.

2.2.2.4 Short and long-term user intents

As can be seen in the previous paragraph, IPAs rely heavily on the ability of recommender systems to build upon short-term or contextual information and long-term or user search intents and habits. These notions are explained in [Guha et al., 2015] and [Wang et al., 2013] where, differently from the works [Joachims, 2002, Radlinski et Joachims, 2005, White et al., 2013] that focus on in-session tasks, here the authors look at several months of history, ingesting Web search history for signed-in users, and identifying coherent contexts that correspond to tasks, interests, and habits. These are known as cross-session approaches. One advantage of this approach is that it allows the building of intents: either that span over several user sessions in the case of long-term intents, or only a portion of one session in the case of short-term intents. Thus, intents provide insights on short and long term information needs and user habits, to build accurate user profiles.

In [Lucchese et al., 2013], authors adopt a cross-session approach to identify task aspects that are common to several users that search for the same information and group them into a single collective intent. In an attempt to clarify things, in what follows, we will refer to user intents when referring to collective behavior and user interests when restricting to a particular user, even if literature is not definitive about these concepts.

2.2.3 Composite Items Recommendations

The objective of recommender systems is to propose items close to users’ intents, conducive to the increase of views and sales in e-commerce, or to help the user get the information needed faster and more easily. Traditional recommender systems present generally a ranked list of items as the answer to user queries [Bota et al., 2014]. On the other hand, user experience in the examples of [Zhu et al., 2014], has shown that people prefer to consume items together, as they better perceive the relevance or profitability of one item according to the other items when assembled in a set, instead of the classic ranked lists. This helps and saves time as the user has all the information needed at once. In Web search, search engines are increasingly going beyond the pure relevance of search results [Bordino et al., 2016], as what users expect from a Web query nowadays is not just some relevant answers, but
2.2. MAIN CONCEPTS IN RECOMMENDER SYSTEMS

multi-modal, interesting and surprising information that can be extracted and synthesized from numerous sources [Lalmas, 2017].

In response to such a need, recent works, in various domains, studied the generation of personalized cohesive packages of items, in a way that allows to compare and combine items in a meaningful manner and fulfill the constraints and requirements of the user [Amer-Yahia et al., 2016, Alsayasneh et al., 2018, Leroy et al., 2015, Bota et al., 2014]. In the literature, we found this package under the generic name of bundle, which will be used from now on to describe this set of items. Section 2.5 is dedicated to these methods and provide more information about their applications and implementations.

2.2.4 Evaluating Recommender Systems

Recommender systems are essentially prediction engines, and as such were initially evaluated on their prediction power — their ability to accurately predict the user’s choices. However, it is now widely agreed that recommender systems are increasingly used for discovering and rapidly exploring diverse items [Gunawardana et Shani, 2015], and can therefore be seen as a particular type of search systems. In search system evaluation, [White, 2016] separates the evaluation methodologies from the evaluation measures, and distinguishes measures pertaining to the outcome from that pertaining to the process.

In terms of methodology, [Gunawardana et Shani, 2015] distinguish three types of experiments: i) offline setting, where recommendation approaches are compared without user interaction, ii) user studies, where a small group of subjects experiment with the system and report on the experience, and iii) describe large scale online experiments, where real user populations interact with the system.

Offline techniques usually employ cross-validation to evaluate performance. It stands on a multi-run partitioning of an available dataset such that in each run, a percentage of the dataset is used as a training set and the recommendations generated are compared to the remaining subset, which is considered as the ground truth. The recommendation process can be assessed through process measures, like for instance coverage, defined as the proportion of items or users for which it is possible to recommend, or with outcome measures, like relevance, i.e., comparing the sets of recommendations and the expected future with Recall, that measures the capacity of the system to retrieve all expected items, and Precision that measures the capacity of the system to retrieve only expected items. Many other measures exist beyond relevance and coverage, like novelty, diversity, serendipity, etc. [White, 2016].

User studies or online experiments try to evaluate the performance of a system from the user’s point of view, through explicit or implicit feedback. This is generally believed to be the best way of measuring user satisfaction, even they may be prone to experimental biases (laboratory studies), poor data quality (crowdsourcing) and lack of rationale (large-scale implicit feedback). Nevertheless, user studies are particularly relevant when testing systems devoted to personalization. In general direct feedback is collected through questionnaires, where users give a score for the system outcome or are asked to
compare experimentally the quality and the efficiency with several baselines. Differently from offline techniques, subjective measures can be employed, like for instance satisfaction, success, cognitive load, enjoyment or serendipity.

2.3 Recommender Systems for BI

The typical interaction between the user and the system in Business Intelligence consists of a sequence of queries answering a business question. Studying such sequences of past interactions can help in suggesting what the next query of the session could be. First of all, we introduce the interaction types in Section 2.3.1. Section 2.3.2 presents the existing approaches in BI in discovering user behaviors and how they are used to recommend in Section 2.3.3. In the end, in Section 2.3.4, we analyze our own challenge in BI platform and we take position in BI recommender systems.

2.3.1 BI interactions

The simplest logs consist of only the queries written by the users, which is typical for BI platforms [Aligon et al., 2014b, Giacometti et al., 2008, Sapia, 2000]. This interaction mode assumes that users are to some extent familiar with the content of the database and also have a quite clear understanding of their information needs [Stefanidis et al., 2009]. These queries that people issue to search systems are explicit, usually short, statements of search interests.

Works that collect formal queries, as [Aligon et al., 2014a, Giacometti et al., 2008, Chatzopoulou et al., 2011], compare them at the level of query composing parts (or query parts for short): measures, dimensions and filters. In addition to simple logged queries, most work enrich the user interactions with details of query parts sources and hierarchies to add more domain knowledge. In addition to the enriched user input queries, [Stefanidis et al., 2009] consider information from external sources related to the database, and [Nguyen et al., 2015] consider database areas containing the information extracted by user queries.

In the literature of BI systems we mainly find coarse comparisons of queries. There is a lack and a need for finer models of BI interactions that can further explore the user intention behind the query.

2.3.2 Discovering and representing User Intents in BI

While many recent works propose techniques to assist query formulation for the purpose of interactive database exploration [Idreos et al., 2015], few works address the issue of discovering user intents based on past queries without asking for explicit feedback. Many approaches are more focused on extracting navigational behavior than user intents. The SnipSuggest approach [Khoussainova et al., 2010] analyzes a log of past queries to extract correlations between SQL clauses, and use them to suggest SQL snippets for auto-completing SQL queries on the fly. [Giacometti et al., 2008] build algorithms that find matching patterns of sequences of switching interests, performing a K-medoid [Kaufman et Rousseeuw, 1987] over the whole sequence of queries.
2.3. RECOMMENDER SYSTEMS FOR BI

In the DICE approach [Jayachandran et al., 2014], the query log is used to prioritize speculative queries transparently sent for the purpose of pre-fetching. With the same pre-fetching motivation, the Promise approach [Sapia, 2000] builds Markov models out of a query log to represent a user’s behavior when querying a data cube. Two Markov models are built. In the first one, the states represent query patterns, i.e., groups of queries with the same group by set. This model allows the probabilities of a pattern to be obtained when given another one. The second Markov model focuses on the selection predicates’ values. This model obtains the probabilities to change a selected value by another one. At run time, these two models are used to obtain the query most likely to follow the query just executed. The Promise approach has influenced later work on query recommendation [Aufaure et al., 2013].

QueRIE [Eirinaki et al., 2014] decomposes each query into basic elements that capture the essence of the query’s logic. They introduce the notion of a session summary to summarize the characteristics of the queries posed in the session. This session summary can be represented in two ways: a vector of the frequencies of database tuples accessed by the user, or a vector of the frequencies of query fragments, that are syntactical features of the queries in the session, like attributes, tables, join and selection predicates. Different comparative metrics and importance scores for database tuples or query fragments are incorporated in each case. Similarly, [Nguyen et al., 2015] focus on the database areas accessed by queries to discover intents, based on the characteristics of these spaces, correlate them to queries and identify empty non-used spaces. They use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) data clustering algorithm [Ester et al., 1996a] to discover user intents. Their notion of user intents rely on the set of tuples that are more frequently accessed.

2.3.3 BI Recommender Systems

Many recent proposals investigate the use of database query logs for query recommendation in the context of interactive database exploration. One of the most prominent proposals is the QueRIE approach [Eirinaki et al., 2014], where different hybrid approaches are used to recommend a set of SQL queries. QueRIE constructs a matrix with logged sessions, where all past queries and sessions are vectorized using the database tuples or the SQL fragments (selections, projections, filters). When a new session is created and a recommendation for it is sought, a K-Nearest Neighbor (KNN) algorithm is used with classic Jaccard or Cosine similarities to find the queries closest to this session in the matrix. [Drosou et Pitoura, 2013] focus on the current state approaches, which consider recent user interactions, and present a method that explores the database schema by expanding the original current query through joins with other appropriate relations and the database content through value correlations.

A survey of query recommendation approaches for BI is proposed in [Aligon et al., 2015], showing that user interest is poorly considered in state-of-the-art approaches. [Aligon et al., 2015] propose a recommender system tailored for exploratory OLAP queries over a datacube, whose sessions can be seen as a particular type of BI interactions, where queries are regular dimensional queries [Golfarelli et Rizzi, 2009]. This system recommends a
sequence of queries and proceeds in three steps. When a new session is created and a recommendation for it is sought, a KNN approach is used, with a customized session similarity, to search in the log for the closest past sessions. Then one subsequence that best fits the new session is determined. Finally, this subsequence is adapted to the new session using patterns identified in the new session and the subsequence. This approach builds upon a review of query similarity measures and the proposal of two similarity measures: one tailored for OLAP queries and another tailored for OLAP sessions [Aligon et al., 2014b]. The authors showed through user studies that the proposed measures better respect the similarity perceived by users over the other measures reviewed.

Given a user query, DICE [Jayachandran et al., 2014] enumerates possible speculative queries using a faceted exploration model and calculates their corresponding likelihood from the query logs. Speculative queries are run greedily: the order is based on a combination of the likelihood of the query occurring and the estimated gain in accuracy from the query result.

[Aufaure et al., 2013] probabilistically model user behavior using a Markov model built on top of query clustering using the similarity metric proposed in [Aligon et al., 2014b]. Identified clusters are the states of the Markov model and log sessions define transition probabilities. This model is used at runtime to find the most likely query to follow a given current query.

We mention some of the first attempts in BI contexts, summarized in Table 2.1, that try to understand the user behaviors, more than really identifying user intents. Most of them search similar queries through sessions, comparing query parts and mining sequential patterns. Clustering queries based on these similarities, permits identifying global user behaviors. To recommend queries to continue the exploration in a new session, in general, they find a past session close to the current session, and extract or adapt its following queries. A notable exception is [Drosou et Pitoura, 2013], that needs no log but relies on precomputed correlations over the database instance.

As a conclusion, we observe that none of the studied works use as input BI reports or rely on models of complex, search engine-like interaction.

<table>
<thead>
<tr>
<th>Work</th>
<th>Short-term intent</th>
<th>Long-term intent</th>
<th>Input</th>
<th>Model</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Giacometti et al., 2008]</td>
<td>✓</td>
<td>✓</td>
<td>OLAP queries, DB tuples</td>
<td>clustering</td>
<td></td>
</tr>
<tr>
<td>[Khoussainova et al., 2010]</td>
<td>✓</td>
<td>×</td>
<td>SQL queries</td>
<td>SQL fragments</td>
<td>pattern mining</td>
</tr>
<tr>
<td>[Drosou et Pitoura, 2013]</td>
<td>✓</td>
<td>×</td>
<td>SPJ queries</td>
<td>tuples</td>
<td>correlations</td>
</tr>
<tr>
<td>[Aufaure et al., 2013]</td>
<td>✓</td>
<td>✓</td>
<td>OLAP queries</td>
<td>query parts</td>
<td>clustering</td>
</tr>
<tr>
<td>[Birman et al., 2014]</td>
<td>✓</td>
<td>×</td>
<td>SQL queries, DB tuples</td>
<td>SQL fragments, tuples</td>
<td>KNN</td>
</tr>
<tr>
<td>[Jayachandran et al., 2014]</td>
<td>✓</td>
<td>×</td>
<td>OLAP queries</td>
<td>query parts</td>
<td>classification</td>
</tr>
<tr>
<td>[Aligon et al., 2015]</td>
<td>✓</td>
<td>×</td>
<td>OLAP queries</td>
<td>query parts</td>
<td>KNN, pattern mining</td>
</tr>
</tbody>
</table>

Table 2.1: Discovering user intents techniques in BI. These approaches use in general the current session or query to recommend, and some of them exploit past logs.

### 2.3.4 Our positioning in BI Recommender Systems

Our objective is to build recommender systems for BI Platforms to help BI analysts run faster and easier their business tasks, taking advantage of the knowledge of experts. Per-
2.3. RECOMMENDER SYSTEMS FOR BI

Personalizing recommendations needs information about the user’s interests. In BI systems, it is not common to rate items or to give explicit feedback. Our purpose is to understand implicit user preferences and to deal with the problem of recommending new queries that are created every day in the system. For the not-rated items, as for the new ones, we should predict the probable score the user would give to these items and not interpret them as not-preferred items. Modeling items permits to fully understand users’ preferences and to compare them using a multi-criteria metric of comparison.

BI sessions are characterized by sequences of queries and each may respond to different business needs, as shown in the following Example 1, which is extracted from a real user interaction in a SAP BI platform. In this session, a user has started by analyzing movie sales (Q°1), continuing with other type of sales for a specific retailer (Q°2,3), followed by analyzes over population and GDP (Q°4,5), to finish again with movie sales(Q°6). We have observed not only the presence of multi-tasking in a session, but also the combination of tasks. The session contains a set of mixed queries from different tasks, meaning that we can possibly return to the same intents several times inside the session. This motivates our orientation towards approaches that treat multi-task session problems. As the same tasks are found in different sessions repeatedly, we treat the entire log, crossing logs and users to learn their intents.

Example 1 Here are some typical queries of a session in our logs:

(Q°1) Sales Revenue, Genre, media format for Country=’Iceland’
(Q°2) Sales Revenue / Sales Target Revenue, Retailer for Year=’2013’
(Q°3) Sales Revenue growth 2015 as Date Year for Retailer=’Alpen’
(Q°4) GDP for Country=’United States’
(Q°5) GDP, Income Group
(Q°6) Sale Target, film, Genre for Country=’Spain’

In this research work, we discuss two different situations where user may need assistance. First, we try to resolve the problem of continuing a user session in a BI search engine, with queries close to user interests. This is similar to the reactive search system in the Web, where we can write queries in natural language. Secondly, we aim at proactively help users in creating or editing business reports, with personalized and contextual queries shared between them.

The lack of advanced multi-tasking systems, of support of short and long-term intents, the limited interaction modeling etc. in BI approaches, lead our work to a richer domain of machine learning techniques applied on Web. The following two sections 2.4 and 2.5 are dedicated to the literature of Web solutions addressed, respectively, to our first problem of recommending in a BI search engine, and to our challenge of completing BI reports.
2.4 Recommender Systems for Web search

This section gives a global view of personalized recommender systems in the Web. The new range of mechanisms collecting and interpreting user natural inputs [White, 2016], offers more complete and rich models of interactions. Section 2.4.1 outlines different types of interactions in Web search engines. We evoke a variety of techniques that aim at discovering user intents in Section 2.4.2 and how they are applied in Web recommender systems in Section 2.4.3.

2.4.1 Search interactions

Different platforms collect different types of logs, based on user actions, data and metadata they access. Some works like [Bennett et al., 2012, Wang et al., 2013, Mei et al., 2008, Lucchesse et al., 2013] use external sources, such as Open Directory Project (ODP) [Chaudhuri et Kaushik, 2009], WordNet [Miller, 1995], Wiktionary 1 or Wikipedia 2.

Beyond the exploration of user issued queries and metadata related to them, in Web we find several approaches [Joachims, 2002, Jiang et al., 2011, Guha et al., 2015, Ustinovsky et Serdyukov, 2013, Bennett et al., 2012] that model interactions including the query result and the user clicks on search result, that correspond to URL pages or documents consulted. It is the most widely used implicit feedback [Cho et Roy, 2004]. Clickthrough data in search engines [Ustinovsky et Serdyukov, 2013] can be thought of as the triplet $(q, S(q), C(q))$ defined as follows:

**Definition 1** Let $L = \{(q, S(q), C(q))\}$ be a set of records from a clickthrough log where each record contains the user issued query $q$, search engine result page (SERP) $S(q) = \{s_1, \ldots, s_n\}$ and her clicks $C(q) = \{c_1, \ldots, c_n\}$ on the SERP, $c_i \in \{0, 1\}$, where $n$ is the number of top results returned by the search engine. [Guha et al., 2015] call such a record an observation.

2.4.2 Discovering User Intents in Web search

Searching user intents, in-session or cross-sessions, demands the definition of a similarity or a distance metric between the log entries and machine learning algorithms to discover groups of queries related to the same need. This section starts by recalling some metrics developed to compare the complex interactions presented in Web (2.4.2.1) and the second part (2.4.2.2) describes how they are used by techniques that discover user intents.

2.4.2.1 Metrics

Different methods are used to compare different types of logs. In this section, we present firstly a group of metrics that compare directly the similarity between interactions and secondly a more sophisticated metric, that creates models based on the first coarse methods.

---

1https://www.wiktionary.org/
2https://www.wikipedia.org/
Comparative measurements between log entities. One of the most important decisions that must be made when modeling search behavior is the unit of analysis at which those models will be constructed [White, 2016]. The complex interactions in Web favor finer comparisons that go beyond the direct comparison of queries, as we noticed in BI. From the possibility of writing queries in Web search engines in natural language, emerged new similarity metrics. For example, [Guha et al., 2015] add the keywords extracted from queries written in natural language in their definition of clickthrough interactions and use them to model interaction similarity metric. [Bennett et al., 2012, Wang et al., 2013, Mei et al., 2008] do not simply compare the composing words, but explore additional describing information over query words to automatically identify the user intents. For example, [Bennett et al., 2012] classify query words in different predefined categories, generated from a text-based classifier. Similarly to categories defined using external sources, manual annotations over queries or clicked pages are used to learn a classifier [Jones et Klinkner, 2008, Guha et al., 2015] to identify the group of items representing the same user intent. They group items annotated with the same terms, altogether characterizing a single topic. Furthermore, [Lucchese et al., 2011] are interested in finding semantic relations between queries, using external sources that increase the meaningfulness of queries, as Wikipedia or Wiktionary. They put together the lexical content and the semantic expansion via a convex combination:

\[
\text{dist}(q_1, q_2) = \alpha \cdot \text{dist}_{\text{content}}(q_1, q_2) + (1 - \alpha) \cdot \text{dist}_{\text{semantic}}(q_1, q_2)
\]

where \text{dist}_{\text{content}} and \text{dist}_{\text{semantic}} are the distances calculated between two queries \(q_1, q_2\) based on their membership to predefined categories and semantic relation.

[Li et al., 2014] model the query co-occurrences inside a session and label queries or links that are seen frequently together using Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. It is a generative probabilistic model for collections of discrete data such as text corpora, and models an item of a collection as a finite mixture over an underlying set of topics. [Li et al., 2014] believe that the occurrence of one query raises the probability that the other query will be issued in the near future.

Another approach that treats especially natural language queries [Hakkani-Tür et al., 2016] propose a holistic multi-domain and multi-task modeling approach to estimate domain classification of phases and their intents to best fill their slots, helped by NLP over phases’ words.

One of the advantages of Web tracking activity is that all information can be collected from a single user click. [Jiang et al., 2011] capture the user preferences mining web search context, looking at the query reformulation and user’s clickthroughs. [Joachims, 2002] and [Craswell et Szummer, 2007] explore the relation between clickthrough data, defining transition probabilities to build a Markov random walk based on highest probabilities to pass from a query to a document or otherwise. Based on these probabilities, they produce a likeliness ranking of unseen documents for a given query. Figure 2.1 shows a graph of walks from document to queries. Similarly is built the graph of walks (clicks) from query-to-document.
2.4. RECOMMENDER SYSTEMS FOR WEB SEARCH

Figure 2.1: Document-to-query transition. In this example, the most likely transition from document D2 is to Q3, because the edge has 70 clicks (as opposed to 50 to Q2 and 10 to Q1).

Feature-based metrics. The metrics mentioned above are directly employed to calculate the similarity or dissimilarity between the interactions, or they serve as a feature of a feature-based approach. Descriptive criteria of SERP interactions are defined as features and can be used to compare them. Each feature may be defined on its own space of possible values and thus implements a specific comparative measure, like Cosine Similarity, Euclidean Distance or Jaccard Similarity. Different works in the literature experiment with different groups of features, according to the problem context or the nature of collected interactions. In the Table 2.2 below we mention some of the classic group of features.

<table>
<thead>
<tr>
<th>Groups of features</th>
<th>Data features</th>
<th>Derived features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>temporal</td>
<td>query words</td>
</tr>
<tr>
<td>Anick, 2003</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Jones et al., 2008</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ustinovsky et al., 2013</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Goh et al., 2015</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Song et al., 2016</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Our results</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2.2: Some examples of the groups of features implemented in feature-based metrics of state-of-the-art approaches. They are separated in two categories of features learned over data (Data Features) and derived ones (Derived Features). The groups of features here include several features of the same type, that in general explore the same log detail, as for example: [Wang et al., 2013] employs features related to query words and the session details of observations.

The first list of groups, Data Features, consists of the direct comparison of interaction dimensions and metadata. Temporal features help to learn a good threshold of inactivity for identifying goal boundaries. Query words features discover related queries that share the same query words or query parts. Some features explore the search result and depend on the document or information content retrieved by the search engine for issued queries. Similarity between pairs of queries’ results is measured by commonalities among the terms.
or characteristics of their corresponding results. Session features compare queries based on the sessions they are included in.

The second list of groups, Derived Features, needs additional calculations over logs to express the user behavior. The development of these features is typically done by hand based on human intuition [White, 2016]. It includes: topics that the queries have been projected to, query logs features discover related co-occurring queries through several statistical and probabilistic measurements, and query properties compare queries based on several query characteristics, as their length, their position in the session etc. In general, the choice of features [White, 2016] is mostly based on their interpretability, model training time, and the reduction of over-fitting to the training data.

[Guha et al., 2015] present a set of descriptive features that, implemented in their personal assistant, yield better precision/recall than past work. Their features are specialized in the Web environment including keywords of user questions, formal query parts, user clicks and domain, user location and other temporal or session details. They take the concept of clickthrough and model observation similarly. Several functions are employed to calculate the similarity between two observations over each feature dimension. [Guha et al., 2015] represent the total similarity for each pair of observations as a weighted sum over all features’ similarities. To learn the feature weights, they use an off-the-shelf linear SVM (Support Vector Machine) classifier, which is a type of learning machine that allows to carry out regression estimation, using support vector techniques on some annotated pairs of observations [Pedregosa et al., 2011]:

\[
Sim(O_1, O_2) = \sum_{f=1}^{n_f} \omega_f v_f(O_1, O_2)
\]  

(2.3)

where \( O_1, O_2 \) are two observations, \( n_f \) is the number of features, \( v_f \) is the similarity measure for feature \( f \) and \( \omega_f \) is a weight representing this feature’s importance in the comparison.

2.4.2.2 Discovering intents methods

Mining large Web usage repositories has the potential of revealing information useful for understanding user behavior [Fournier-Viger et al., 2017], and earlier works aiming at discovering user intent relied on pattern mining. [Cooley et al., 1999] use association rule and sequential pattern mining to identify similar transaction cross-sessions. Furthermore, for Web personalization [Mobasher et al., 2002] present an efficient framework based on sequential and non-sequential pattern mining from clickthrough data.

In the following paragraphs we present recent methods to learn the intents, based on one supervised technique, classification, and on one unsupervised technique, clustering. At the end, Table 2.3 presents a summary of the main works that combine different interaction types, metrics and these techniques together.
2.4. RECOMMENDER SYSTEMS FOR WEB SEARCH

**Supervised classification.** Support Vector Machine (SVM) is one of the most commonly adopted learning method in search interaction that involves supervised methods, in which the desired output is known and it is the objective of the system to learn a general model based on provided training data [White, 2016]. [Jiang et Cheng, 2016] perform SVM to classify sessions and queries into predefined ODP (Open Directory Project) categories, based on composing words of the URLs the users have clicked among the suggestions of their query results. In both cases, a class or a predefined category, represent a user intent. [Jiang et al., 2011] uses domain related metrics to compare URLs, queries or sessions. For example, it classifies in the same intent URLs of the same host or two sessions whose first/last queries are similar. [Wang et al., 2013] adopt bestlink SVM, a structural learning method with latent variables, to realize the bestlink modeling assumption, to find which queries of the same tasks are strongly connected, over a graph of connected queries, with edges weighted by its nodes’ similarity.

**Clustering.** Clustering is the most common form of unsupervised learning [Manning et al., 2008]. The cluster algorithms’ goal is to create groups of items that are coherent internally, but clearly different from each other. These methods on intent identification [Jones et Klinkner, 2008, Lucchese et al., 2011, Wang et al., 2013] generally solve two subproblems sequentially: (1) use queries’ textual information and semantic meaning to cluster them into search topics inside observed query sequences, and (2) use obtained clusters, together with temporal information, to separate query sequences into search tasks.

[Guha et al., 2015] describe a near-linear approximation to a Hierarchical Agglomerative Clustering problem (HAC) [Koga et al., 2007] that groups queries presenting the same information need. Its similarity measure is learned over a set of descriptive features, as for example: the stemmed query words, top 10 web results for the queries or the stemmed words in the titles of clicked URL. Other works, as [Cutting et al., 1992], prefer generating these groups on the basis of the query content rather than relying on metadata.

[Lucchese et al., 2011, Lucchese et al., 2013] experiment with a variation of clustering techniques such as: K-means, HAC, DBSCAN or Query Clustering over Weighted Connected Components (QC-WCC) of a graph to identify tasks inside sessions. QC-WCC builds a graph for each time-based session, where nodes are the queries, and edges link together any pair of consecutive queries, weighted by the probability of the two queries being task-related. Their algorithm finds chains of consecutive queries in each session, and performs HAC [Koga et al., 2007] to merge similar chains. The construction of the graph can differ based on different calculated associations rules inside sessions, that define queries connection. [Wang et al., 2013] propose a solution to optimize HAC performing bestlink supervised clustering. Instead of calculating all the similarities between pairs of queries (all-link, sort of HAC average link), they propose an adapted HAC single link algorithm that searches to find at least one query of a cluster, which is strongly associated with the candidate query (the bestlink) [Jain et al., 1999].
Table 2.3 summarizes state-of-the-art works and techniques to discover the user intents. They collect and explore different data and meta data and build metrics that perform different similarity functions.

<table>
<thead>
<tr>
<th>Work</th>
<th>Model</th>
<th>Technique</th>
<th>Input</th>
<th>Logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Jones et Klinkner, 2008]</td>
<td>feature-based model</td>
<td>classification</td>
<td>queries, SERP annotated sessions</td>
<td>cross-sessions</td>
</tr>
<tr>
<td>[Lucchese et al., 2013]</td>
<td>lexical query content</td>
<td>pattern mining</td>
<td>queries</td>
<td>in-session</td>
</tr>
<tr>
<td>[Lucchese et al., 2011]</td>
<td>content &amp; semantic distance</td>
<td>clustering</td>
<td>queries, External Sources</td>
<td>in-session</td>
</tr>
<tr>
<td>[Bennett et al., 2012]</td>
<td>feature-based model</td>
<td>classification</td>
<td>queries, SERP, ODP</td>
<td>cross-sessions</td>
</tr>
<tr>
<td>[Song et Guo, 2016]</td>
<td>feature-based model</td>
<td>clustering</td>
<td>clickthrough</td>
<td>cross-sessions</td>
</tr>
<tr>
<td>[Li et al., 2016]</td>
<td>feature-based model</td>
<td>clustering</td>
<td>queries, SERP</td>
<td>cross-sessions</td>
</tr>
<tr>
<td>[Wang et al., 2013]</td>
<td>feature-based model</td>
<td>clustering</td>
<td>queries, SERP</td>
<td>cross-sessions</td>
</tr>
<tr>
<td>[Jiang et Cheng, 2016]</td>
<td>feature-based model</td>
<td>classification</td>
<td>clickthrough, ODP</td>
<td>in-session</td>
</tr>
<tr>
<td>[Guha et al., 2015]</td>
<td>feature-based model</td>
<td>clustering</td>
<td>clickthrough</td>
<td>cross-sessions</td>
</tr>
</tbody>
</table>

Table 2.3: Discovering user intents techniques in Web.

2.4.3 Web search Recommender Systems based on User Intents

From the beginning of recommender systems, several works tried to personalize the proposed suggestions, as the same recommendation will not satisfy different users in different contexts. [Wildemuth, 2004] found that novices and experts, with similar profiles, converge towards the same search patterns as they are exposed to a topic. Personalized behavioral patterns can be mined for users, and then used to identify frequent search habits, in order to provide personalized recommendations. Even though these personalizing tentatives improved the relevance of the suggestions, [Teevan, 2007] noticed that 27% of repeated searches had clicks on new results, so users need support for both finding and re-finding search results.

The experiments of [Awadallah et al., 2010] show that models that are based on user behavior achieve more success than those using pure document relevance to the user current query. Thus, in this section we will focus in literature works that profit from the discovered user intents to build recommenders to user interests, to be able to offer personalized proactive assistance, and to identify and integrate the user current context, as suggested in a recent report that summarizes strategic directions in information retrieval [Allan et al., 2012]. These approaches enhance search effectiveness beyond what is possible to achieve from popularity alone [Tan et al., 2006, Shen et al., 2005].

**Long and short term intents.** Most of works that search identifying long-term intents [Song et Guo, 2016] address the problem of predicting task repetition, i.e., whether a task represents a one-time information need or exhibits recurrent patterns. [Craswell et Szummer, 2007] propose to project queries and corresponding clicked results into click-through graphs (Figure 2.1). Mining graph relations permits discovering close queries, maybe never found in the same session, but that lead to clicks on the same set of results. Rather than focusing on individual query-click (ex. query-page, query-document)
pairs, [Radlinski et Joachims, 2005] create a model of query sequences directing to a document. They build Markov models based on the probability of a query leading to a document, and this document leading to a subsequent query, and use these behavioral sequences to recommend them once the user starts asking a similar set of queries.

In addition to previous works, experiments of [Palmisano et al., 2008] show how much the contextual information matters in building customer models in personalized applications. [Sun et al., 2016] is interested in contextual intent, with a particular emphasis on the external physical environment, and intent tracking is done in real-time. But, not every system can access environment information, so short-term intents are mainly defined over the user activity during the current session [Teevan et al., 2005, Ustinovsky et Serdyukov, 2013], as written queries or selected documents. [Teevan et al., 2005] rank the original recommendations from the closest to the user model, built from both usage information from the search engine, such as previously issued queries and visited Web pages, and other information about the user meta data, such as documents and emails the user has read and created. [Ustinovsky et Serdyukov, 2013] address the problem of short-term personalization based on current session. They use the short-term history to determine the context of a given query, restricting their attention to the first queries of the session. This allows the automatic learning of a filter that distinguishes recommended queries to re-rank the result in a post-treatment phase, using some personalization scores related to user short-term preferences.

Integrating separately long or short-term information may be useful in personalizing the recommendation, but it does not resolve the problem of unspecific, ambiguous queries. Thus, modern search engines tend to employ not only the knowledge about searcher’s profile, but also the query information and the context of the current query. This involves modeling the previous interactions and the content of browsed data, of pages, documents or chosen items [White et al., 2010, Bennett et al., 2012], extracting information from user’s browsing long and short-term history.

[Bennett et al., 2012] resume from their experiments that long-term intents, extracted from historic behavior, are useful at the start of a session and short-term intents provide more precise prediction as the session proceeds. They provide a first study to assess how short-term and long-term intents interact, and how each may be used in isolation or in combination to optimally contribute to gain in relevance of personalized exploration. They combine different groups of features to identify long or short term intents, learning which features to ignore for historic or contextual data.

**Recommending future intents instead of items.** Differently from the literature works mentioned in the previous sections, that calculate personalization scores and recommending specific items, there are some works [Lucches et al., 2013, Jiang et Cheng, 2016, White et al., 2010] that firstly discover the user intent that follows her search navigation, based on the current context and her historic, and, in a second phase, choose the most representative queries of this intent. [Li et al., 2016] tend, furthermore, to discover a “chain” of intents, based on the probabilities of user tendencies to pass from one intent to
2.5. RECOMMENDING COMPOSITE ITEMS

another within a session. They propose an unsupervised approach to identify search tasks via topic membership along with topic transition probabilities. Different from previous studies that classify a query to a single intent, they present a membership vector of the query to each discovered intent. [Li et al., 2016] presents a novel hidden semi-Markov model to model topic transitions by considering not only the semantic information of queries but also the latent search factors originated from user search behaviors. Different from hard clustering that strictly relate each query to a topic, they propose a soft or fuzzy clustering that consists on a membership matrix of participation of each query to each topic.

In addition to context, [Jiang et al., 2015] go further and include the diversification in the production phase of algorithm, which produces novel and interesting recommendations for the final user. From the rising number of constraints to consider during recommendation, separately or integrated together, emerged the need of more complex methods. In the next session, we will evoke these types of algorithms that build Composite Items, in order to recommend a set of complementary items, relevant to the user and her situation.

2.5 Recommending Composite Items

This section is dedicated to a relatively new problem in recommendation domain that aims at grouping personalized complementary items to recommend. The main works in this specific problem apply these recommendations to suggest a touristic tour of a city [Benouaret et Lenne, 2016], a travel itinerary [Taylor et al., 2018], items to sell in addition to an article in Web [Zhang et al., 2014] or a group of composite tasks in crowdsourcing to assign to workers [Amer-Yahia et al., 2016, Alsayasneh et al., 2018]. This kind of recommender responds to a problem similar to our second use case that aims at completing BI reports with a set of sorted business objects.

Composed set of items are recommended instead of ranked lists. This gives a clearer and a full view of the central axes of the analysis, as well as the compatibility and consistency of items to each other, which better satisfies the users’ anticipation. These compositions take the form of a star [Roy et al., 2010], with a central item and other satellite compatible items; the form of a chain in touristic tours cases [De Choudhury et al., 2010] or the form of snowflakes for tasks crowdsourcing [Amer-Yahia et al., 2014]. [Amer-Yahia et Roy, 2018] shows how specific use cases and constraints applied in bundle creation define its shape and complexity, which presents a solution that may range from graph traversal to merging lists, and clustering.

We start by presenting the basic principles of bundle construction, insisting on the integration of different constraints to satisfy the user needs, and detailing state-of-the-art approaches of bundle construction in Section 2.5.1. Section 2.5.2 summarizes and compares existing methods of bundle building, while Section 2.5.3 focuses on the most suitable approach for the context of this thesis.
2.5. RECOMMENDING COMPOSITE ITEMS

2.5.1 Principles

In this section we will present some typical approaches that propose packages of items. We can identify for each of these techniques new constraints they introduce to filter the recommender result to better fit to user needs, and different methods of integrating these constraints together. They differentiate mainly by their way of constructing the recommended packages, the complexity and the penalty terms to respect.

2.5.1.1 Input representation and relevance through merging lists of results

In e-commerce search engines or enterprises, the data is in general distributed in multiple, separate sources. Query results may cover several sources the user can access, searching simultaneously in all of them to retrieve the appropriate information corresponding to the user need. Federated search techniques [Shokouhi et Si, 2011] can provide parallel search over multiple collections. They query several data sources and rank the items retrieved from each of them. The selection of suitable collections is based on some knowledge about the content of each source, creating the “representative problem”. Representativeness enforces the proposed suggestion to provide a good coverage of the input. More recent works dedicated to Composite Items (CIs), as [Amer-Yahia et al., 2016, Alsayasneh et al., 2018], ensure representativeness thanks to the clustering algorithms based on the distance between items.

Traditional federated [Arguello et al., 2009, Shokouhi et Si, 2011] and aggregated [Zhou et al., 2012] search systems, rather than building bundles, aim to simply merge items from all different result lists into a central index and order them by a traditional ranking function. This was a first approach that gave the possibility to merge results originating from different catalogs, calculated separately and presented under the same list. The challenges that arise in realizing such systems are the selection of a vertical, that represents a dimension over which the items are differentiated such as: a catalog, a document type etc., and the choice of the final ranking of selected items, as they are heterogeneous, coming from different sources and not-comparable between them. In federated and aggregated search, the relevance is simply estimated as the pertinence of the result to the current user query, but in other contexts its measurement can be more complicated. It can be related to the set of queries user performed in the near past and her choices (clicks), in addition to the current user query. All these elements represent together the user short-term exploring interest. In Web and proactive systems, the relevance is evaluated over the similarity to the current context, enriched by the user location, schedule or other environmental indications. While in exploratory search, the relevance consists in the similarity between the suggestion and the current user session.

2.5.1.2 Bundles to integrate item cohesiveness and diversity

PAC (Produce and Choose) approaches [Bota et al., 2014, Amer-Yahia et al., 2014, Feuerstein et al., 2016, Benouaret et Lenne, 2016] consist in the creation of an exhaustive number of valid bundles, from which we choose the ones that better optimize the objective func-
2.5. RECOMMENDING COMPOSITE ITEMS

tion. PAC approaches intend to produce cohesive and intra-diverse bundles, which make the principal constraints of data selection. Cohesiveness guarantees a minimal demanded similarity among suggested items. In general, it is measured as the average similarity between all pairs of items. The underlying assumption is that the utility of one item depends on the other items shown to the user, so these approaches consider the whole set of recommended items as a bundle, instead of treating them independently. On the other hand, PAC techniques try to diversify the produced bundles. Diversity may be defined over distinct item types, different semantic or other metadata characteristics. Being two opposite concepts, cohesiveness and diversity are measured over different data descriptive dimensions. Bundle intra-diversity defines a distance function between the items of a single bundle and inter-diversity defines the distance between bundles themselves, in the cases when the recommender proposes more than one bundle of items.

There are two main algorithms in the category of PAC methods: BOBO (Bundles One-by-One) [Amer-Yahia et al., 2014, Feuerstein et al., 2016, Benouaret et Lenne, 2016] and CPS (Central-Plus-Satellite) [Bota et al., 2014], an extension of BOBO adapted for the heterogeneous cases, where the information is retrieved from different sources and items describe different type of data, as: text, image, video etc. Recommended bundles generated from these algorithms take the shape of snowflake [Amer-Yahia et Roy, 2018].

BOBO produces a set of candidate bundles inspired by $KNN$-clustering: an initial accessible item, called pivot, is chosen at each step and a valid bundle is built around this pivot. At first, each element of the itemset is considered as a candidate pivot and, once a new item is chosen as a pivot, the building routine greedily keeps picking the closest element to it. The similarity between items is provided explicitly as an input to the algorithm, or computed implicitly from the representation of the items [Benouaret et Lenne, 2016]. The items added to the pivot maximize the internal cohesion during the bundle production, which is one of the criteria to valid candidate bundles. The production phase includes the creation of several bundles that are valid when they satisfy constraints of complementarity and budget, defined as follows [Amer-Yahia et al., 2014]:

**Definition 2 Complementarity:** given a a bundle $B_i$ and a property $p$ of the items, no two items in a bundle $B_i$ exhibit the same value for that property: i.e. $\forall u,v \in B_i$ and $u \neq v, u.p \neq v.p$

**Definition 3 Budget:** given a set of bundles $B = \{ B_1, \ldots, B_k \}$ built over a set of items $V$, a set-valued non-negative and monotone function $\text{funct} : 2^V \rightarrow \mathbb{R}^+$ and a budget threshold $\#\text{budget}$, we require that $\forall B_i \in B, \text{funct}(B_i) \leq \#\text{budget}$.

Typical examples of budget are simply the number of items forming a bundle or an upper-bound on the sum of the costs of items forming the bundle, given a cost attribute.

There exists several adapted versions of BOBO in [Benouaret et Lenne, 2016] that take into account the popularity of items and user preferences, and RBOBO [Leroy et al., 2015], that optimizes BOBO removing the candidates that contribute the least to decreasing the distances of data points to their closest item.
2.5. RECOMMENDING COMPOSITE ITEMS

The second algorithm, CSP model, consists in several verticals, each handling different types of items. A central vertical is used to produce homogeneous bundles using BOBO and then items of other selected verticals are attached into the produced bundles. Bundle cohesiveness is ensured adding items that share with other existing items in the bundle a percentage of common entities, greater than a specified threshold. Differently from BOBO, that chooses items from different verticals to diversify the result inside a single bundle (bundle intra-diversity), CPS consider Maximal Marginal Relevance (MMR) [Carbonell et Goldstein, 1998] ranking strategy to optimize bundles inter-diversity. This approach orders the list of recommended bundles in a way that each bundle is followed by the most dissimilar bundle to it. Similarly, [Bota et al., 2014] apply a post-diversification process that measures at each step the diversity between the previous bundle to other candidates to add and choose the one that is the most cohesive and relevant, but at the same time the most distant, measured over a different dimension.

Approaches presented in this section are very interesting for us as they provide cohesive and diverse bundles. The problem stands on the way they integrate these two concepts. They offer two constraints: cohesiveness and diversity, that are evaluated independently from each other and are only used as hard constraints: a bundle just above the budget will not be considered even if it is more coherent, by a large margin, than other candidate bundles. In our case, we want to use soft constraints that balance the ratio between the item cohesiveness and diversity. We do not evaluate items separately, we search the package of items, that together validate a bundle and optimize our objective function. Furthermore, these methods still miss some important aspects. BOBO suggests the same items for all users, not personalizing the result, which is very important in our business environment with restricted user rights, and where casual analysts collaborate with experts. Another proposition of [Zhu et al., 2014], Bundle Recommending Problem (BRP) favors the highest rating and popular elements, falling in repetition and not encouraging novelty. For these and other possible constraints, we need a more expressive method that we can extend with new constraints.

2.5.1.3 Constrained bundles around cluster centroids

CAP (Cluster and Pick) methods [Amer-Yahia et al., 2013, Amer-Yahia et al., 2014, Amer-Yahia et al., 2016, Alsayasneh et al., 2018, Roy et al., 2010, Leroy et al., 2015] search for representative clusters of input data and create around their centers valid recommending bundles. They solve a 2-way optimization problem, where the first stage aims to identify representative cluster centroids obtained through hard or fuzzy clustering, whereas the second stage ensures that the chosen items are close to centers to assure cohesiveness. A greedy procedure applied on each cluster ensures the quality of each bundle and the intra-bundle diversity. At the end, all produced bundles around centroids are valid bundles. In the Pick phase, the best possible subset is selected from the candidate bundles. At the bundle building stage, [Bota et al., 2014, Amer-Yahia et al., 2013] calculate the cohesion and intra-bundle diversity, based on data sources, to produce valid bundles, and at a second phase they integrate similarity to user profile and inter-bundles diversity, selecting
2.5. RECOMMENDING COMPOSITE ITEMS

packages related with different topics, to maximize the variety of proposed bundles. A bundle is created inside each cluster, respecting several conditions, from the budget or the number of items it should contain to the threshold scores of complementarity or similarity between items. In this section, we will present some of the most common algorithms of CAP approaches that integrate different penalty terms (constraints) to validate their bundles.

Different CAP methods use different constrained clustering algorithms to select the centers: hard clustering, as for example the Constrained Hierarchical Agglomerative (C-HAC) in [Amer-Yahia et al., 2014], or fuzzy clustering, as Fuzzy C-Means (FCM) in [Leroy et al., 2015]. C-HAC [Davidson et Ravi, 2005] is similar to classical agglomerative hierarchical clustering and it starts by considering each item as a separate cluster and it continues by merging small clusters of items that respect budget and complementarity constraints. In this algorithm, each cluster is always a valid bundle, so when it reaches the stopping condition, that may be the number of required bundles, it returns immediately the result with no need for any further check. C-HAC will strictly classify an item to a cluster.

Rather than this hard clustering, there are other preferred methods in bundle construction that use fuzzy clustering. They produce a membership matrix of assignment of each item to each cluster and permit to parametrize the level of fuzziness of clustering. Previous studies’ results have shown that the use of fuzzy instead of hard clustering, simultaneously optimize the representativeness of bundles. An extended version of a well known fuzzy clustering algorithms FCM is the constraint-based fuzzy clustering algorithm (KFC), implemented by several works [Leroy et al., 2015, Amer-Yahia et al., 2016, Amer-Yahia et al., 2014].

Differently from the works presented in previous sections, where we had limited criteria taken in consideration, the advantage of CAP is the opportunity to adapt or add all the necessary constraints, whether they are strong (number of items, budget etc.) or soft constraints (cohesion, personalization, diversity, etc.), to assure the best user experience. The greedy procedure that builds bundles around centroids is easily extensible and can be parametrized according to user preferences.

The complexity of the proposed technique depends on the complexity of demanded validation constraints. The list of soft constraints can be extended to improve the quality of composite items. [Amer-Yahia et al., 2016], in the context of assignment of tasks to workers for crowdsourcing, develop an approach that builds for each worker a set of representative and personalized Composite Tasks (CTs). Bundle validity, in this particular case, ensures that a CT contains the number of tasks requested for a work session and that the worker is qualified for each one of them. Several studies in Task Crowdsourcing [Amer-Yahia et al., 2016, Alsayasneh et al., 2018] show how including personalization and diversification in CT recommendation impacts directly the performance augmentation, by proposing to workers more relevant and interesting set of tasks, adequate with workers’ skills and preferences. [Alsayasneh et al., 2018] explore a trade-off between task uniformity and diversity within a CT. The choice of the diversity function is made along a dimension, different from those
used to measure uniformity, personalization or other constraints.

A major challenge brought by these approaches is the computation of similarities between items, knowing that the items of a bundle can originate from different sources and that they can have different types, and between items and users [Bota et al., 2014]. The similarity values may be provided explicitly in the input, or computed implicitly from the representation of the items [Amer-Yahia et al., 2014]. To be comparable, they should be projected over a uniform set of attributes, such as a common topical space where items and users alike may be represented.

Highly scored and diverse bundles are expected to be retrieved from the collection of $k$ candidate bundles, built inside of $k$ clusters. The quality of the chosen collection is given by a weighted combination of multiple constraints evaluation, related to the quality of each bundle, and the inter-bundle diversity. [Amer-Yahia et al., 2014] show that the performance of these methods depends basically on a parameter controlling the trade-off between the average score of the bundles and the diversity of the set of bundles.

Different methods prioritize different constraints, for example: the relevance to the query asked or to the user profile, as well as a high level of diversity or novelty, which lead to different choices according to the user need. When diversity is highly important, the comparing results [Amer-Yahia et al., 2014] show that the best performance is obtained using algorithms based on creating a global clustering of the items first and then choosing bundles that respect those clustering boundaries. This is the CAP approach. When diversity is less important, [Bota et al., 2014, Amer-Yahia et al., 2014] show that “local” methods (PAC), that construct more cohesive bundles around randomly chosen pivots, produce better results.

### 2.5.2 Advantages and disadvantages of bundle building approaches

In the previous sections we gave a panorama of the literature works that attempt to build bundle of items. They are summarized in the Table 2.4 below, giving more details about the constraints considered by some of the main approaches.

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Validity</th>
<th>Representativeness</th>
<th>Cohesiveness</th>
<th>Diversity</th>
<th>Personalization</th>
<th>Relevance</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federated &amp; Aggregated Search [Shokouhi et Si, 2011]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BOBO [Amer-Yahia et al., 2014] [Zhou et al., 2012]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CPS [Roy et al., 2010] [Bota et al., 2014]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FCM [Amer-Yahia et al., 2016] [Alsayasneh et al., 2018]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2.4: Soft and hard constraints of CI approaches implemented by the state-of-the-art works.
2.5. RECOMMENDING COMPOSITE ITEMS

Federated and Aggregated Search are the first approaches that try to provide a result originating from different sources, proposed as a merged list of items retrieved from each source. They are ranked based on item relevance to the user query, which risks to produce the same solution for different users. BRP, BOBO and CPS are the first methods that propose recommending bundles instead of ranked lists. Even though they improved sensibly the recommendation quality maintaining a trade-off between the cohesiveness and the diversity of items, their suggestions are still unpersonalized and the items are evaluated separately and not as a bundle. In general, BPR and PAC methods are limited in the constraints they integrate, which makes it difficult to insert new penalty terms in particular cases or to extend the model in the future. A bundle may be valid if its composing items come from different data sources, or if they respect a certain budget or a maximum bundle size. In the case of task crowdsourcing, a bundle proposed to a worker is valid if the composing tasks are conform to the worker qualification, are not banned by them and the profit of the whole bundle respond to the worker wage.

2.5.3 A focus on Cluster And Pick method

Here we will enter more into the details of a particular approach, CAP, which is easily extensible and as such seems more suitable for our case. We describe more in details how the problem of building bundles is treated as a constrained clustering problem.

Composite Item (CI) approaches [Benouaret et Lenne, 2016, Roy et al., 2010, Leroy et al., 2015] have shown to be very effective in solving problems of summarizing large collection of items or complex information needs such as planning a city tour, selecting books for a reading club, proposing restaurants respecting group constraints etc. [Amer-Yahia et al., 2016] present Composite Tasks (CTs), a specific problem of CIs. It examines the problem of producing, for each worker, a personalized summary of micro-tasks. Similar to them, we aim to group cohesive and relevant queries to user to recommend bundles of queries that resolve the problem of completing reports in our BI platform.

As a bundle is created inside each cluster and that several representative bundles are expected in the end, [Amer-Yahia et al., 2013] define this problem as a typical clustering problem. They implement a Fuzzy C-Means algorithm (FCM) to produce compact clusters and to cover all input data so the bundles will be representative of all the dataset. Another advantage of FCM is that is allows one item to belong to different clusters.

The quality of the built bundles is ensured by an objective function. The algorithm that resolves CT problem proposed by [Amer-Yahia et al., 2016] maps the objective function into FCM [Bezdek, 1981] to cluster tasks and seamlessly integrate the optimization goals as convex penalty terms. Then a new step is added to the traditional FCM to greedily select the optimal subset of items in each cluster forming a valid bundle.

FCM assigns tasks to a collection of $k$ fuzzy clusters, represented through their centroids and a weighted matrix of the membership of each task to each cluster. They present an algorithmic solution for the optimization of NP-hard problem of founding the best bundle of items that optimizes the objective function in each cluster.

47
2.6. CONCLUSIONS

The algorithm of bundle construction presented in [Alsayasneh et al., 2018] differs from the one in [Leroy et al., 2015] in that it solves a 3-way optimization problem. A part aims at identifying task representatives, another ensures that the representatives chosen are close to valid CTs and a last part ensures that the returned CTs are relevant to the user profile. Furthermore, the optimization objective is extended with task diversity assuring diversified CTs over another dimension, different from the one that measures their similarity. The objective function defined by [Alsayasneh et al., 2018] tries to maximize the distances between items, to ensure cohesion, personalization and diversity:

\[
\arg\max_{C,W} \sum_{i=1}^{K} \sum_{j=1}^{K} w_{ij} \alpha \left| x_{i}, c_{j}\right| + \sum_{x \in CT_j} \beta \left| x_{i}, c_{j}\right| + \frac{\delta}{\#_w - 1} \sum_{x,y \in CT_j, x < y} \left| x_{i}, c_{j}\right| + \sum_{x \in CT_j} \gamma \left| x_{i}, c_{j}\right|
\]

s.t. \forall i \in [1, |\chi|], \sum_{j=1}^{K} w_{ij} = 1

where \(x, y\) are tasks, \(C\) denotes the set of \(K\) clusters obtained by the FCM clustering of a set of tasks \(\chi\) and \(c_j\) the center of cluster \(j\). \(W\) is the membership matrix and \(w\) the membership score of an item \(i\) to a cluster \(j\). \(CT\) is the set of the Composite Tasks of size \(#_w\), where \(CT_j\) denotes the bundle issued from cluster \(j \in [1, K]\). \(\nu\) denotes valid CTs. The quality score estimated at each iteration takes into account the cohesion of items inside a bundle as represented by the function \(\text{sim}(\cdot)\) and the intra-bundle diversity, as denoted by the function \(\text{div}(\cdot)\). \(\alpha, \beta, \gamma\) and \(\delta\) represents the relative weights of each penalty term in the objective function.

The compatibility between the center and bundle items is measured based on their characteristics. For example, [Alsayasneh et al., 2018] and [Amer-Yahia et al., 2016] sort and compare tasks by dimensions such as creation date or reward amount, to facilitate the task assignment. The bundles that minimize the score of this function \(gf\), which means minimizing the distances between items and item to the user and maximizing the diversity, represent the subset of chosen elements for each cluster.

2.6 Conclusions

The particularity of our data and our business context do not allow to directly implement the methods evoked in this literature. We are inspired from several works resolving dif-
2.6. CONCLUSIONS

Different problems in the Web, which we aim to adapt and combine to respond to our two BI use cases: (1) completing user exploratory sessions with a sequence of queries, and (2) completing a BI report with a bundle of queries. For both of them, we need firstly to discover user intents. To the best of our knowledge, our work is the first attempt to automatically discover BI users’ intents in a multi-user environment and to show that detecting user intents in past user traces helps query recommendation.

Discovering user intents. The classic classification methods [Jiang et Cheng, 2016, Jones et Klinkner, 2008] to discover user intents, based on predefined categories of external sources or annotations, were excluded due to our lack of user annotations and the domain specific problem. [Guha et al., 2015] work with Web SERP observations and consider queries written in natural language, similar to our BI platform, where the business questions are mapped to formal queries. Inspired from this work, we try to adapt log descriptors as features to create a model-based metric for discovering the user intents. The advantage we have over the metrics in Web consists on the business knowledge we dispose, such as: details of query sources and their organization, business object dimensions and their functional dependencies. This permits us to propose a new group of features related to business objects’ metadata for our platform. We reuse the idea of accessed area of database comparison proposed by [Nguyen et al., 2015] as one of our features, and differently from most of the reviewed approaches, we consider the content of the query result. However, our work in discovering user intents deviates from [Guha et al., 2015] on major aspects. First, we present our own formal model tailored to BI interactions and we address a specific type of intent. Consistently, we use a specific set of features. Second, we focus more on the expressiveness of the model rather than on specific optimizations for scaling to web data volumes. Finally, our approach is entirely automatic, and we present our own evaluation of it, which includes a specific user study.

Intent-based recommender of a sequence of BI queries. Our first BI platform resembles to an exploratory search tool, and we aim at suggesting following queries to complete user navigation. Instead of analyzing and recommending specific final queries, our goal is to firstly understand the user actual intent and to predict a query from the following intents. Markov models [Aufaure et al., 2013] is a promising solution to discover these “chains” of intents. In BI environment we consider for the moment two groups of user expertise: expert analysts and novices, which can demand specialized recommendations. In the future, it can be employed for finer organizational groups of users. It can help resolving effectively cold-start problems of new users in the system.

Intent-based recommender of a bundle of BI queries. The second recommender system we envision, suggests proactively a bundle of items to complete a BI report. Our objective is beyond simply gathering cohesive and relevant items. Important indications of the quality of our recommendation are the diversification of suggested items, projected
2.6. CONCLUSIONS

over a different dimension from the cohesiveness. In our specific context, it is possible to measure the diversity of queries looking at the visualization types of their results. [Teevan et al., 2009, Netflix, 2018] examined how different representations of Web pages affected people’s ability to discover and recognize new relevant content and return to previously viewed Web pages. Furthermore, we want to include two new constraints: relevance to the editing report, evoking user short-term interests or context, and the ranking of items inside the bundle, that ensures a logical order between the suggested queries, aligned with the existing items of the current report.

The approach that best fits our needs is CAP, which ensures the representativeness of input data and permits to add more and different constraints in the phase of bundle production around centroids. Algorithm for building bundles of [Amer-Yahia et al., 2016] is flexible and can be easily adapted for our specific use case by modifying the greedy approach. We can add new constraints to represent new utilities and we can balance them through the penalty scores.
Chapter 3

Identifying User Intents

Current BI systems do not adequately detect and characterize user intents, which may lead to tedious and unproductive interactions. Being able to automatically identify user intents from past logs is a challenging problem as they are hidden in BI interactions.

In this chapter, we propose to identify such user intents by characterizing the intent of the interaction with the BI system. With an eye on user modeling for proactive search systems, we identify a set of features for an adequate description of intents, and a similarity measure for grouping logged observations into coherent intents. We validate experimentally our approach with a user study, where we analyze traces of BI navigation in a SAP platform that permits writing queries in natural language\footnote{Here is the patent reference of the SAP BI system that permits interrogating the databases in natural language, similar to a search engine: us 20180157734A1: Business Intelligence System Dataset Navigation Based on User Interests Clustering}. We show that our similarity measure outperforms a state-of-the-art query similarity measure [Aligon et al., 2014b] and yields a very good precision with respect to expressed user intents.

Our contributions include:

- a simple formal model tailored to BI interactions,
- the identification of a specific set of features for characterizing BI user intents,
- the learning of a similarity measure based on these features,
- an approach to automatically discover user intents based on our measure and an off-the-shelf clustering algorithm,
- an extensive set of experiments for the tuning and validation of our approach, the comparison of our measure with a state-of-the-art metric tailored for OLAP queries [Aligon et al., 2014b], and the study of its behaviour in various practical situations.

This chapter is organized as follows: Section 3.1 presents our formal model of BI interactions and user intents. Section 3.2 details the set of features used to characterize user intents and our algorithm for discovering coherent cross-interaction intents. Section 3.3 presents our experimental validation and Section 3.4 concludes the chapter.
3.1 Formal model of BI interactions

This section presents our model of BI interaction, which corresponds to a sequence of queries to resolve a user need. BI users can write queries using keywords to respond to their business questions. For each query entered in the BI engine, there are a set of formal queries generated by mapping keywords to business objects of the system business layer. Users can choose one of these propositions and display the query result.

Given the proximity of BI interactions in modern BI systems and web searches, our modeling of BI interactions is inspired by the modeling of web search sessions. Note that our model is independent of the mechanisms used by BI systems to generate formal (MDX or SQL) queries from keywords. Such mechanisms are out of the scope of this thesis.

3.1.1 BI Questions, Suggestions and Queries

Let $D$ be a database schema, $I_D$ an instance of $D$ and $Q$ the set of formal queries one can express over $D$. For simplicity, we consider relational databases under star schemata, queried with multidimensional queries [Vaisman et Zimányi, 2014], but our approach can be adapted to other database models as well. Without lack of generality, we consider $D$ to be a global schema resulting from the integration of several data sources. We simply note $\text{sources}(q)$ the set of sources used by a query $q$ over $D$.

Let $A$ be the set of attributes of the relations of $D$. Let $\text{Meas} \subset A$ be a set of attributes defined on numerical domains called measures. Let $H = \{h_1,\ldots,h_n\}$ be a finite set of hierarchies, each characterized by (1) a subset $\text{Lev}(h_i) \subset A$ of attributes called levels and (2) a roll-up total order $\succeq_{h_i}$ of $\text{Lev}(h_i)$. Consistent with the literature on database theory [Abiteboul et al., 1995], we denote by $\text{adom}(I_D)$ the set of all constants appearing in the instance $I_D$ of $D$, i.e., the constants that are used to form the tuples of instance $I_D$.

We call a database entity an element of the set $A \cup \text{adom}(I_D)$. Let $V$ be a n-dimensional building model, whose axis represent levels $L$ or measures $\text{Meas}$ of a schema $D_j$, that together can retrieve database information and represent it graphically to understand and explore the filtered results via visualization. According to the number and particularity of attributes, there are different visualization types as: Charts, Tables, Tag Clouds, Plots, Tree Map, Speedometer etc. Formally, a query $q$ over a database schema $D_j \in D$ is a tuple $\langle \text{Fil}, \text{Lev}, \text{Meas}, V, D_j \rangle$ where $\text{Fil}$ is a set of filters, $\text{Lev}$ is a set of levels, $\text{Meas}$ is a set of measures and $V$ is a visualization type. The result (or answer) of a query $q$ over a database instance $I_D$ is denoted by $q(I_D)$.

Let $T$ be a countably infinite set of keywords named tokens. A BI question (or question for short), $G$, is a set of tokens entered by a user, e.g. “Revenue for France as Country”. Note that tokens may contain stop words (e.g. “by” or “as”) that have no equivalent in formal queries but that the BI system may use in formal queries generation. The reason for including such stop words in our model is that the use of stop words may convey a certain user behavior or expertise with the system. Each token may be matched with the entities in $A \cup \text{adom}(I_D)$ to generate queries. To simplify, we describe a multidimensional query $q$ in $Q$ as a set of query parts, as in [Aligon et al., 2011]. A query part is either a level of a hierarchy in $H$ used for grouping, a measure in $\text{Meas}$,
or a simple Boolean predicate of the form \( a \ op \ v \), where \( a \) is a level of a hierarchy in \( H \), \( op \) is an operator in \( \{=, <, >, \leq, \geq, \neq \} \) and \( v \) is a constant in \( \text{adom}(I_D) \). In what follows, queries are confounded with their sets of query parts, unless otherwise stated.

**Example 2** Starting from the question “Revenue for France as Country” the following tokens \( G_1 = \{”Revenue”,”for”,”France”,”as”, ”Country”\} \) are identified. A corresponding formal query contains the following query parts: Revenue is a measure, Country a level in a hierarchy, and France is a constant, resulting in \( \text{Country}=\text{France} \) being a Boolean predicate.

If a query part \( p \) is a selection predicate of the form \( a \ op \ v \), or a grouping attribute \( a \) we use \( \text{level}(p) \) to denote attribute \( a \). Given two query parts \( p_1 \) and \( p_2 \), \( \text{FD}(p_1, p_2) \) denotes a functional dependency \( \text{level}(p_1) \rightarrow \text{level}(p_2) \). Given two queries \( q_1 \) and \( q_2 \), the Boolean expression \( \text{OP}(q_1, q_2) \) indicates if they differ in at most one query part. This allows for the detection of OLAP operations when users navigate along hierarchies or change selection conditions.

---

Figure 3.1: An example of the Search Panel in a BI Platform, where the users can write questions in natural language. They are then mapped to formal queries, which generates 5 suggestions and visualize the result of the chosen queries.

As keywords are entered, a BI system might on the fly suggest further tokens to complete the current ones, letting the user choose among them, as in web search engines. The underlying idea is that a suggestion completes the original BI question to obtain a well-formed query over a database, as shown in Figure 3.1.

We formalize the notion of suggestions as follows. A suggestion \( s \) is a triple \( \langle G, D, q \rangle \) where \( G \) is a BI question, \( D \) is a database schema and \( q \) is a query over \( D \). For short, given...
3.1. FORMAL MODEL OF BI INTERACTIONS

a suggestion $s = \langle G, D, q \rangle$, we note tokens($s$) for referring to $G$, query($s$) for referring to $q$, and sources($s$) for referring to sources($q$).

**Example 3** The question $G_1 = \{"Revenue", "for", "France", "as", "Country"\}$ is completed to focus on year 2017. The corresponding suggestion, $s_{11} = \langle G, D, q \rangle$, consists of question $G = \{"Revenue", "for", "France", "as", "Country", "and", "2017", "as", "Year"\}$, schema $D$, that includes a relation Sales, and the formal query $q$, represented by its three query parts \{Revenue, Country = France, Year = 2017\}, whose SQL code is: SELECT sum(Revenue) FROM Sales WHERE Country = 'France' AND Year = 2017;

3.1.2 Observations, Interactions and User Intents

In Web Search, search histories (i.e., interactions with a search engine) are analyzed to identify coherent information needs as basis for recommendation generation. For instance, [Guha et al., 2015] propose modeling information needs as sequences of observations, an observation being a search engine query with its associated web results (Search Engine Result Page or SERP for short) and clicks. We adapt the model proposed by [Guha et al., 2015] to model contexts of BI interactions. This adaptation relies on the following simple analogy: (i) the search engine query corresponds to the BI question, (ii) the SERP corresponds to the set of suggestions associated with the BI question, and (iii) a click on one SERP link corresponds to the choice of a suggestion and hence to the evaluation of the query associated with the suggestion. Figure 3.2 shows the analogy of these web concepts to BI domain.

![Figure 3.2: The analogy of the observation model of Web, presented by [Guha et al., 2015], to BI interaction.](image)

Formally, an observation $o$ is a triple $o = \langle G, S, s \rangle$ where $G$ is a question, $S = \{s_1, \ldots, s_d\}$ is a set of $d$ suggestions for question $G$, and $s \in \{s_1, \ldots, s_d\}$ is the suggestion selected by the user. Given an observation $o$, we note $G^o$ the question $G$ of $o$, suggestions($o$) its set of suggestions, and chosen($o$) the chosen suggestion. We note
query(o) = query(chosen(o)), the query of the chosen suggestion, and result(o) = query(o), the result set of the query over a database instance ID. All these components of observation are presented in Figure 3.1. In addition, we annotate each observation o with a binary property indicating the expertise of the user who interacted with the system, denoted by expertise(o).

An interaction of length v is a sequence of v observations i = ⟨o1, . . . , ov⟩ that represents the user interaction with the BI system.

Example 4 Consider question G2 = {"Revenue", "for", "France"} of an observation o2. Several suggestions are proposed, with respective questions: {"Revenue", "for", "France", "as", "Country"}, {"Revenue", "for", "France", "as", "Market Unit"}, {"Revenue Closed", "for", "France", "as", "Country"}, etc. Assume the first suggestion is chosen by the user, its formal query is evaluated and its result result(o2) is displayed to the user. Other questions {"Revenue", "for", "France", "2010"} or {"Revenue", "for", "France", "2015"}, {"Revenue Closed", "for", "France"} follow, to create a complete interaction of the user with the BI system, analyzing the economic growth of France.

Without loss of generality and to keep the formalism simple, we assume that an observation is part of only one interaction. The function interaction(o) returns the interaction to which o belongs. Given two observations ox and oy in an interaction, we say that oy refines (is a refinement of) ox if ox precedes oy and either Gox = Goy ∪ {t} or Goy = Gox ∪ {t}, where t, t′ ∈ T.

A user intent is a finite set U = {o1, . . . , ol} of l observations that represents one particular information need.

Table 3.1 presents the basic characteristics we use to describe user intents. Note that ∪B denotes bag union (preserving duplicates to compute frequencies), P is a set of query parts and matches(t, p) is a binary function indicating if token t matches query part p.

3.1.3 Running example

Consider two user intents U1 and U2, with U1 containing a unique observation o1 = ⟨G1, {s11}, s11⟩ of a unique interaction i1 (described in Example 3), and U2 containing two observations o2 = ⟨G2, {s21, s22, s23}, s21⟩ and o3 = ⟨G3, {s31, s32}, s32⟩ that are part of the interaction i2 (described in Example 4). The observations are summarized in Table 3.2 by listing questions and suggestions. For the sake of readability, Table 3.2 describes the suggestions only by means of their queries. Assume that all suggestions access the same database and the same underlying source Sales and that all users are beginners.

Table 3.3 presents the characteristics of both user interests. For the sake of space, we do not show the query results. RefToken is defined when an observation refines another one, in which case it contains the tokens involved in the refinement. MatchTok contains the tokens that match some set of query parts; in Table 3.3 we used P = {"Revenue", "Country = France", "Year = 2007"}, arbitrarily.
3.2 Characterizing and clustering User Intents

Table 3.1: Basic characteristics of user intents

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Definition</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>questions(U)</td>
<td>∪_{q ∈ U} G_o</td>
<td>all the questions</td>
</tr>
<tr>
<td>tokens(U)</td>
<td>∪_{o ∈ U} G_o</td>
<td>all the tokens</td>
</tr>
<tr>
<td>suggestions(U)</td>
<td>∪_{o ∈ U} suggestions(o)</td>
<td>all the suggestions</td>
</tr>
<tr>
<td>chosenSuggest(U)</td>
<td>∪_{o ∈ U} chosen(o)</td>
<td>all the chosen suggestions</td>
</tr>
<tr>
<td>queries(U)</td>
<td>∪_{o ∈ U} {query(o)}</td>
<td>all the chosen queries</td>
</tr>
<tr>
<td>interactions(U)</td>
<td>∪_{o ∈ U} interaction(o)</td>
<td>all the interactions</td>
</tr>
<tr>
<td>results(U)</td>
<td>∪_{o ∈ U} result(o)</td>
<td>all the results</td>
</tr>
<tr>
<td>sources(U)</td>
<td>∪_{o ∈ U} sources(chosen(o))</td>
<td>all the sources</td>
</tr>
<tr>
<td>expertise(U)</td>
<td>∪_{o ∈ U} expertise(o)</td>
<td>all the expertises</td>
</tr>
<tr>
<td>refTok(U)</td>
<td>{t ∈ tokens(U)</td>
<td>∃o, o' ∈ U, t ∈ (G_o \ G_{o'})}, o refines o'</td>
</tr>
<tr>
<td>matchTok(U, P)</td>
<td>{t ∈ tokens(U)</td>
<td>∃p ∈ P, matches(t, p)}</td>
</tr>
</tbody>
</table>

Table 3.2: Summary of observations in the running example

| o1 : G_1       | “Revenue for France as Country” |
| s_{11} :       | {“Revenue”, “Country=France”, “Year=2007”} |
| o2 : G_2       | “Revenue for France” |
| s_{21} :       | {“Revenue”, “Country=France”} |
| s_{22} :       | {“Revenue”, “Market Unit=France”} |
| s_{23} :       | {“Revenue Closed”, “Country=France”} |
| o3 : G_3       | “Revenue for France 2010” |
| s_{31} :       | {“Revenue”, “Year”, “Country=France”} |
| s_{32} :       | {“Revenue”, “Country=France”, “Year=2010”} |

3.2 Characterizing and clustering User Intents

Following the approach of [Guha et al., 2015], we formalize the problem of discovering coherent user intents as a clustering problem, for which a similarity measure is learned over a set of descriptive features. These features allow observations (and user intents) to be grouped based not only on their intentions expressed by the BI question but also based on their objectives as expressed by the chosen suggestion, and on their knowledge, as provided by the evaluation of the chosen query. To compare two user intents, a global similarity is computed as a weighted sum of feature-based similarity measures.

We first define the set of features we consider, together with their similarities, then explain how the features are weighted and how the contexts are clustered.
3.2. CHARACTERIZING AND CLUSTERING USER INTENTS

To provide the best characterization of user intents, we define a set of candidate features, that we subsequently analyze to identify those maximizing the accuracy from the user's perspective. We considered three groups of features, listed in Table 3.4. The first group of features relates to the BI questions and suggestions (features 1-6). The second group relates to the chosen suggestions, and especially their query parts (features 7-9). Both groups proved effective in identifying intents in the context of Web searches [Guha et al., 2015]. The third group consists of specific BI features and relates to formal queries and their answers (features 10-15).

Contrary to most works where features are descriptive and relate to each data object independently of the others, in our proposal, features should be understood as dimensions on which it is possible to compare two user intents, \( U_1 = \{ o_1^1, \ldots, o_l^1 \} \) and \( U_2 = \{ o_1^2, \ldots, o_m^2 \} \). As a consequence, each feature has a proper semantic attached to it, as it describes a particular aspect of a relation between the user intents - for example the number of occurrences of tokens in the questions of each user intent. Then, each feature \( f \) is also paired with a similarity measure denoted by \( v_f \in \mathbb{R}^+ \), which hereafter quantifies this relation.

Table 3.4 details the features by giving their formal definition and the feature-based similarity measure used for comparing two user intents. Given a bag of elements \( x \), \( freq(x) \) is a vector counting the number of occurrences of each element of \( x \). For each feature, we propose a similarity measure that is the most suited for it (e.g., cosine for vectors of frequencies, Jaccard for sets). We follow the same logic as in [Guha et al., 2015], and, in particular, the definition of similarity measures MaxFrac and NormInt are drawn from [Guha et al., 2015]. MaxFrac measures the maximum fraction of observations of each
3.2. CHARACTERIZING AND CLUSTERING USER INTENTS

<table>
<thead>
<tr>
<th>#</th>
<th>Feature</th>
<th>Formal definition</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frequency of tokens</td>
<td>( \text{freq(tokens(U_1))} )</td>
<td>Cosine</td>
</tr>
<tr>
<td>2</td>
<td>Frequency of refining tokens</td>
<td>( \text{freq(refTok(U_1))} )</td>
<td>Cosine</td>
</tr>
<tr>
<td>3</td>
<td>Suggestions</td>
<td>( \text{suggestions(U_1)} )</td>
<td>NormInt.</td>
</tr>
<tr>
<td>4</td>
<td>BI questions</td>
<td>( \text{questions(U_1)} )</td>
<td>NormInt.</td>
</tr>
<tr>
<td>5</td>
<td>( U_1 ) questions that are sub-questions in ( U_2 )</td>
<td>( { G \in \text{questions(U_1)} \mid \exists G' \in \text{questions(U_2)}, G' \subset G } )</td>
<td>MaxFrac.</td>
</tr>
<tr>
<td>6</td>
<td>( U_1 ) questions in the same interaction as a question in ( U_2 )</td>
<td>( { G^o \mid o \in U_1, \exists o' \in U_2, \text{interactions(o) = interactions(o')} } )</td>
<td>MaxFrac.</td>
</tr>
<tr>
<td>7</td>
<td>Frequency of chosen query parts</td>
<td>( \text{freq(qParts(U_1))} )</td>
<td>Cosine</td>
</tr>
<tr>
<td>8</td>
<td>Frequency of tokens of ( U_1 ) that match chosen query parts of ( U_2 )</td>
<td>( \text{freq(matchTok(U_1, qParts(U_2)))} )</td>
<td>Cosine</td>
</tr>
<tr>
<td>9</td>
<td>Chosen suggestions</td>
<td>( \text{chosenSuggest(U_1)} )</td>
<td>NormInt.</td>
</tr>
<tr>
<td>10</td>
<td>Levels in chosen query parts</td>
<td>( { \text{Level}(p) \mid p \in \text{qParts(U_1)} } )</td>
<td>Jaccard</td>
</tr>
<tr>
<td>11</td>
<td>Tuples retrieved by chosen queries</td>
<td>( \text{results(U_1)} )</td>
<td>NormInt.</td>
</tr>
<tr>
<td>12</td>
<td>Queries in ( U_1 ) that differ by one query part from a query in ( U_2 )</td>
<td>( { q \in \text{queries(U_1)} \mid \exists q' \in \text{queries(U_2)}, \text{OP}(q, q') } )</td>
<td>MaxFrac.</td>
</tr>
<tr>
<td>13</td>
<td>Sources</td>
<td>( \text{sources(U_1)} )</td>
<td>MaxFrac.</td>
</tr>
<tr>
<td>14</td>
<td>Attributes of ( U_1 ) functionally identifying attributes in ( U_2 )</td>
<td>( { \text{level}(p) \mid p \in \text{qParts(U_1)} } )</td>
<td>MaxFrac.</td>
</tr>
<tr>
<td>15</td>
<td>Expertise of users</td>
<td>( \text{expertise(U_1)} )</td>
<td>MaxFrac.</td>
</tr>
</tbody>
</table>

Table 3.4: Features considered

user intent that match an observation in the other user intent. Given two interests \( U_1 \) and \( U_2 \), it is defined by:

\[
\text{MaxFrac}(U_1, U_2) = \max \left( \frac{|O^s_1|}{|O_1|}, \frac{|O^s_2|}{|O_2|} \right)
\] (3.1)

where \( O^s \) are the observations that satisfy some property \( s \) over the total number of observations \( O_i \) of \( U_i \).

\[
\text{NormInt}(U_1, U_2) = \frac{|F_1 \cap F_2|}{\min(|F_1|, |F_2|)}
\] (3.2)

where \( F_i \) is the set of features of \( U_i \) and \( |F_i| \) is the cardinality of this set of features. It is important to note that, different from Jaccard similarity, NormInt favors the merging of \( U_1 \) and \( U_2 \) whenever \( U_1 \subseteq U_2 \).

**Example 5** Let consider the user interests described in the running example (Section 3.1.3) and the feature 1 from Table 3.4.
3.2. CHARACTERIZING AND CLUSTERING USER INTENTS

freq(tokens(U\textsubscript{1})) = <(Revenue, 1), (for, 1), (France, 1), (as, 1), (Country, 1)>

freq(tokens(U\textsubscript{2})) = <(Revenue, 2), (for, 2), (France, 2), (2010, 1)>

By following a bag-of-word representation for both user interests, we end-up with the vectors \( t_1 \) and \( t_2 \) representing respectively tokens’ frequency for users \( U_1 \) and \( U_2 \).

<table>
<thead>
<tr>
<th>Revenue</th>
<th>for</th>
<th>France</th>
<th>as</th>
<th>Country</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Following Table 3.4, the similarity for feature 1 is a cosine measure defined classically as:

\[
v_1(t_1, t_2) = \frac{\langle t_1, t_2 \rangle}{\sqrt{\langle t_1, t_1 \rangle} \ast \sqrt{\langle t_2, t_2 \rangle}} \quad (3.3)
\]

where \( \langle, \rangle \) denotes the inner product. Here, the similarity would be \( v_1(t_1, t_2) = 0.74 \), meaning that on this particular feature user interests \( U_1 \) and \( U_2 \) are relatively close to each other.

3.2.2 Clustering User Intents

Grouping observations into user intents, and then grouping similar user intents, requires addressing two problems: (i) determining a similarity measure between user intents and (ii) finding a clustering algorithm that can work on the sole basis of this similarity.

Regarding problem (i), our aim is to distinguish among the candidate features presented above, those who are the most suitable to identify coherent intents from a user standpoint. As we are expecting an understandable model that provides the relative importance of each feature in the process of comparing two user intents \( U_1 \) and \( U_2 \), we rely on a linear aggregation for our similarity \( Sim(U_1, U_2) \) defined as follows:

\[
Sim(U_1, U_2) = \sum_{f=1}^{n} \omega_f v_f(U_1, U_2) \quad (3.4)
\]

where \( n \) is the number of features, \( v_f \) is the similarity measure indicated in Table 3.4 for feature \( f \) and \( \omega_f \) is a weight representing this feature’s importance in the comparison.

With this formulation, the problem of designing a similarity naturally translates into a problem of determining the set of weights \( \omega_f \) paired with each similarity measure \( v_f \).

To this end, we formalize the problem of discovering \( \omega_f \) as a classification task, which proved effective in [Guha et al., 2015, Wang et al., 2013]. Indeed, we are able to train a classifier \((X, Y)\) in which each entry \( x \in X \) corresponds to a couple of observations, the descriptive features of each entry being the one introduced in Table 3.4 and the output \( y \in Y \) being set to 1 if these two observations relate to the same intent, and -1 otherwise.

We use an off-the-shelf SVM linear classifier paired with some ground truth knowledge about user intents to learn the predictive value of the feature. For a feature \( f \), the weight
3.3. EVALUATION OF USER INTENT DETECTION

$\omega_f$ is set to the conditional probability that two observations correspond to the same user intent knowing that they coincide on feature $f$. The absolute value of $\omega_f$ reflects how discriminant feature $f$ is (a large value indicates that feature $f$ is very influential in the decision process), while the sign of $\omega_f$ denotes that feature $f$ will either act in favor of grouping user intents or, conversely, to separate them. In particular, the descending list of the absolute value of weights ranks the features, starting from the most important one.

Noticeably, some preprocessing and optimizations have been performed to ensure that our SVM is accurate. First, our data set of observations’ pairs has been balanced to guarantee that there were the same number of couples related to the same user intent (labeled $1$) as the couples related to different user intents (labeled $-1$). Second, the hyper parameter, named $C$ in [Guha et al., 2015], which traditionally determines the balance between the flatness of solution weights and the amount up to which it is possible to deviate from the regulation term in the SVM optimization model, has been tuned to its best possible value by an extensive cross validated random search. Finally, we note that we disregard the weight $\omega_0$ learned by the linear SVM classifier, as it only plays the role of an offset in the similarity $\text{Sim}$.

The particularity of our log records and the descriptive features limited our tests with other possible methods to learn the similarity metric between observations. Most of existing methods take as input numeric features, so that, it is not possible to apply them directly over our data, as our features represent couples of observations and not single numeric data.

Problem (ii) is addressed by experimenting with off-the-shelf well-known and trusted relational clustering algorithms implementing different strategies, i.e., centroid-based clustering, connectivity-based clustering and density-based clustering, as explained in Section 3.3.1.4.

3.3 Evaluation of User Intent detection

This section presents the empirical evaluation of our approach for detecting user intents. It starts with the experimental protocol (Section 3.3.1) and exposes our results (Section 3.3.2).

Our first objective is to determine a metric based on the features introduced in Section 3.2.1 that allows, when paired with a clustering algorithm, the grouping of user observations into clusters that accurately reflect user intents. The main goal is to use these clusters for recommending queries that share the same user’s intents. In this regard, the first experiments (Section 3.3.2.1 to Section 3.3.2.4) aim at determining and validating the best subset of features from the set presented in Table 3.4. We test its sensitivity to the clustering algorithm as well as its behavior when confronted with observations or clusters of observations related to a business need. Then, a comparative experiment (Section 3.3.2.6) with the state-of-the-art similarity measure for OLAP sessions proposed in [Aligon et al., 2014b] shows the effectiveness of our proposal in the particular context of user intents discovery. Incidentally, our experiments also reveal that considering the reference metric [Aligon et al., 2014b] as a feature in our similarity measure in some cases improves the overall quality of our approach.
3.3. EVALUATION OF USER INTENT DETECTION

Finally, we propose two side experiments to further validate our clustering approach, as follows: (i) the behavior of our metric when confronted with unseen business needs (Section 3.3.2.5), and (ii) the behavior of our metric in detecting intra-interaction intents (Section 3.3.2.7).

3.3.1 Experimental protocol

3.3.1.1 Data set

The data used for our experiments consist of navigation traces of 14 volunteers at SAP, covering a range of skills in data exploration, divided into two groups, namely, beginners and experts, based on their position in the company. To evaluate to what extent actual user intents were discovered by our method, we set 10 business needs (named \(Q_1\) to \(Q_{10}\)), each corresponding to a specific user intent. Users were asked to analyze some of the 7 available data sources to answer each of the 10 business needs, using an SAP prototype that supports keyword-based BI queries\(^2\). The business needs were grouped in different business cases, such as: "For each European country, detect which genres of films did not reach the expected sales" or "In which income group would you classify a candidate country with a GDP of $6 billion?". All business needs are listed in Appendix A.

To be more realistic, business needs were defined to expect some overlap in terms of accessed data and queries. In the context of user intent discovery, the business needs \(Q_1\) to \(Q_{10}\) serve as our ground truth, our objective being to cluster together observations (potentially from different user interactions) that addressed the same business need.

<table>
<thead>
<tr>
<th></th>
<th>(Q_1)</th>
<th>(Q_2)</th>
<th>(Q_3)</th>
<th>(Q_4)</th>
<th>(Q_5)</th>
<th>(Q_6)</th>
<th>(Q_7)</th>
<th>(Q_8)</th>
<th>(Q_9)</th>
<th>(Q_{10})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty</td>
<td>low</td>
<td>med</td>
<td>med</td>
<td>med</td>
<td>low</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>med</td>
<td>high</td>
</tr>
<tr>
<td>Number of interactions</td>
<td>19</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Number of queries</td>
<td>84</td>
<td>65</td>
<td>60</td>
<td>41</td>
<td>50</td>
<td>43</td>
<td>61</td>
<td>51</td>
<td>26</td>
<td>49</td>
</tr>
<tr>
<td>Number of relevant queries</td>
<td>34</td>
<td>26</td>
<td>30</td>
<td>16</td>
<td>26</td>
<td>10</td>
<td>27</td>
<td>24</td>
<td>24</td>
<td>9</td>
</tr>
<tr>
<td>Queries / interaction</td>
<td>4.4</td>
<td>5.9</td>
<td>6.0</td>
<td>4.1</td>
<td>5.0</td>
<td>5.4</td>
<td>6.8</td>
<td>5.7</td>
<td>2.9</td>
<td>6.1</td>
</tr>
<tr>
<td>Relevant queries / interaction</td>
<td>1.8</td>
<td>2.4</td>
<td>3.0</td>
<td>1.6</td>
<td>2.6</td>
<td>1.25</td>
<td>3.0</td>
<td>2.7</td>
<td>2.7</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table 3.5: Analysis of business needs

In total, our data set, named COMPLETE hereafter, contains 24 user interactions, each one possibly concerning several business needs, accounting for 530 queries. Table 3.5 describes, for each business need, its difficulty, estimated by an expert (in terms of time, number of queries and exploited sources expected in its resolving), the number of interactions devised for solving it, the number of queries and the number of queries perceived as relevant by users in their own activity. To have several difficulty settings, we also built two reduced data sets named REDUCED 1 and REDUCED 2, each corresponding to 4 business needs and 4 distinct data sources, which in turn removes most of the potential overlap.

\(^2\)The patent reference of a SAP BI system that permits interrogating the databases in natural language: us 20180157734A1: Business Intelligence System Dataset Navigation Based on User Interests Clustering
3.3. EVALUATION OF USER INTENT DETECTION

Each of them contains 225 observations. Importantly, Reduced 1 and Reduced 2 are not related to the same business needs. When dealing with these data sets, only 4 well separated clusters are to be found, contrary to the COMPLETE data set in which 10 clusters with overlap are expected.

3.3.1.2 Assessing the user intents quality

Our objective is to build groups of observations that are only related to a single user intent. As in any clustering problem, there are two main solutions to assess the quality of the results, i.e., either based on some external knowledge of the ground truth clusters or based on some internal criterion evaluation. In our case, we evaluate both as follows.

Concerning the external evaluation, the main indicator of success in our case is the precision of the clustering when compared to the theoretical grouping of observations provided by the business questions. At a second level, recall allows for determining to which extent each cluster covers all of the observations related to a user intent. Finally, we use the classic Adjusted Rand Index (ARI) (see Appendix B) to evaluate the overall quality of the clustering. The values of this index range from below 0 (when the clustering performs badly and produces a partition close to a random clustering) and 1 (when the clustering is perfect) [Desgraupes, 2013].

The internal evaluation comprises measuring the quality of the obtained clusters through the classic Silhouette coefficient, described in Appendix B. We observe, for each cluster, the intra similarity between its composing observations and the inter-cluster similarity, so that we can conclude on the compactness of the clusters. Being aware of possible observation overlapping between clusters, we should ensure that the distance between the observations within a cluster is shorter compared to the distance with other clusters’ observations. It is expected that using clustering algorithms like DB-SCAN or hierarchical clustering with single-link criterion that may end-up with elongated clusters would, by definition, underscore the Silhouette coefficient.

3.3.1.3 Metric learning

The feature weights are learned over 50% of all observations chosen randomly, with a balance in the number of observations per business needs. Our objective is two-fold and aims at finding the smallest subset of features to avoid any problem of over-fitting when the number of dimensions increases, while still maximizing the quality of the discovery of user intents. To this aim, we tested several subsets of features and trained the weights of the metric with a linear SVM algorithm, as presented in Section 3.2.2, on the sole basis of these features. The subsets of features are selected as follows. We consider all 15 features described in Table 3.4 and learn the metric. The linear SVM outputs weights that traduce the relative importance of each feature. It is thus possible to order features by the absolute value of their weights. This ranking allows the forming of subsets of features starting from those with only highly weighted features to subsets that more widely cover the whole set of features. We give the results for the following meaningful subsets: GF2={1, 3, 7, 8, 9}, GF3=GF2∪{5, 10, 11, 13, 14} and ALL, that include the features with the highest relative importance (the top-5, top-10 and all features, respectively). We also constitute a group
3.3. EVALUATION OF USER INTENT DETECTION

GF1=\{7, 8, 9, 10, 13\} that includes the top-5 features selected by repetitively adding to the group those features that increase precision, similar to the work in [Guha et al., 2015]. Note that GF3 includes both GF1 and GF2. Finally, groups GF4=\{1, 2, 3, 4, 5, 6\} and GF5=\{7, 8, 9, 10, 12, 13, 14\} are specific groups of features related only to keywords (GF4) and query parts (GF5).

3.3.1.4 Choice of the clustering algorithms

As no hypothesis can a priori be made on the shape of expected groups of observations, we use in our tests various clustering algorithms that are representative of the diversity of common methods from the literature. The only constraint imposed by the formulation of our problem is that these methods must be relational, i.e., only based on the expression of a distance or dissimilarity between pairs of data instances. The first method we tested is the PAM algorithm [Kaufman et Rousseeuw, 1987], which is a k-medoid algorithm that finds hyperspherical clusters centered around the k most representative observations. We also use agglomerative hierarchical clustering algorithms [Kaufman et Rousseeuw, 1990] with single and complete linkage criterion to either allow for either elongated or compact clusters. Finally, we use the traditional DBSCAN algorithm [Ester et al., 1996b] that is not restricted to a specific cluster shape but constraints clusters to share the same density of points.

3.3.1.5 Implementation

Our approach is implemented in Java. We also use Python Scikit Learn [Pedregosa et al., 2011] linear SVM to learn the weights of our similarity measure and R clustering packages cluster for k-medoids and hierarchical clustering, as well as fpc for DBSCAN.

3.3.2 Lessons learned

3.3.2.1 Determining the best subset of features and the clustering algorithm

Table 3.6 shows that the quality of the discovered groups of observations heavily depends first on the subset of features, as expected, but also on the clustering algorithm used. It can be seen that approaches that allow for elongated clusters, such as hierarchical clustering with single link criterion and DBSCAN algorithms, achieve very poor precision results (\(\text{Prec} = 0.11\)). This can be explained by the fact that these two algorithms are sensitive to potential overlapping between clusters. In our case, similarities between user intents cause early unwanted merging between groups of observations. The stability in precision is because these two approaches constantly built a majority of mono-observation clusters and one cluster with almost all the observations, whatever the group of features considered. Conversely, clustering algorithms that favor compact clusters, such as hierarchical clustering with complete link or k-medoids PAM algorithms, perform better. PAM performs significantly better than the hierarchical complete link algorithm, knowing that standard deviations (not reported here for the sake of readability) do not exceed \(10^{-2}\) and are usually around \(10^{-3}\). Finally, when considering only PAM, it can be seen that the
3.3. EVALUATION OF USER INTENT DETECTION

subset of features $GF^2$ outperforms all the others. Interestingly, these features are those that had the most discriminating behavior based on the SVM weights observed on all our 15 features (see Section 3.3.1.3). Adding more features only slightly increases the recall. Other strategies (not mixing features from different specific groups or using the strategy proposed in [Guha et al., 2015]) can dramatically harm the precision. It is also important to note that subset $GF^2$ does not include BI specific features, which indicates that enough semantics is carried out by the other features in detecting user intents. From the previous findings, we define $GF^2$ as the set of features and we use PAM clustering in the remaining tests, unless otherwise stated.

<table>
<thead>
<tr>
<th>Features</th>
<th>H. Single</th>
<th>H. Complete</th>
<th>PAM</th>
<th>DBSCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>0.96 0.11</td>
<td>0.49 0.34</td>
<td>0.52</td>
<td>0.46 0.42</td>
</tr>
<tr>
<td>GF1</td>
<td>0.90 0.11</td>
<td>0.67 0.12</td>
<td>0.43</td>
<td>0.40 0.35</td>
</tr>
<tr>
<td>GF2</td>
<td>0.92 0.11</td>
<td>0.68 0.11</td>
<td>0.51</td>
<td>0.50 0.44</td>
</tr>
<tr>
<td>GF3</td>
<td>0.97 0.11</td>
<td>0.38 0.28</td>
<td>0.52</td>
<td>0.47 0.43</td>
</tr>
<tr>
<td>GF4</td>
<td>0.96 0.11</td>
<td>0.67 0.14</td>
<td>0.47</td>
<td>0.29 0.26</td>
</tr>
<tr>
<td>GF5</td>
<td>0.91 0.11</td>
<td>0.39 0.28</td>
<td>0.45</td>
<td>0.42 0.37</td>
</tr>
</tbody>
</table>

Table 3.6: Clustering results with distinct subset of features on COMPLETE data set. For short, $Rec$, $Prec$ and $ARI$ denote respectively recall, precision and ARI scores.

### 3.3.2.2 $GF^2$ metric interpretation

Several features were proposed to describe the differences between the observations. The first two groups presented in Table 3.4 are inspired by the features used in the Web while the last one is a new proposal to show specific differences related to the BI context. The succeeding experiments concluded that a particular group of features presented in Table 3.7, named $GF^2$ in Section 3.3.1.3, principally composed of features related to business objects and formalized queries, in collaboration with PAM, identifies better the user intents.

The BI tool used for these experiments assisted users to reach the information needed, despite their non-precise questions in natural language, by proposing well formalized query suggestions. The user intent is expressed by the keywords (Tokens (Feature 1)) and the suggestions (Feature 3) proposed, but the real difference between user observations is specified by the chosen suggestion (Feature 9) with the query parts composing it (Feature 7) and their matching tokens (Feature 8). Consequently, the features of $GF^2$ in Table 3.7 are the best selection that achieve identification of the user intents, as they directly represent the user choices. Adding more dimensions in the observations comparison reduces the accuracy of the measured dissimilarity between them, which leads to intents that are not well-defined.
3.3. EVALUATION OF USER INTENT DETECTION

<table>
<thead>
<tr>
<th>GF2 Features</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency of tokens</td>
<td>0.39</td>
</tr>
<tr>
<td>Suggestions</td>
<td>0.41</td>
</tr>
<tr>
<td>Frequency of chosen query parts</td>
<td>1.23</td>
</tr>
<tr>
<td>Frequency of tokens of $U_1$ that match chosen query parts of $U_2$</td>
<td>0.38</td>
</tr>
<tr>
<td>Chosen suggestions</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 3.7: Feature weights

3.3.2.3 GF2 metric behavior

While our metric is learned from observations, our experimental protocol aims at grouping together observations participating in the analysis of a business need. To understand the behavior of our GF2 metric, we tested how it degrades when applied to analyses and then to observations. Analyses are defined as sets of observations participating to answering the same need. This is unlikely to be detected in practice, and this information was explicitly asked to the users when they answered the different needs. Obviously, as shown in Table 3.8, when applied to analyses, our metric achieves optimal to very good performance. In the easiest case, when user intents are clearly distinct from each other and rich information is provided to our algorithm with analyses rather than observations, the clustering fits perfectly, with precision, recall and ARI scores equal to 1. Interestingly, when we cluster analyses based on the metric learned on observations, the results are identical to the previous results. In contrast, learning metric weights on the basis of analyses (although not realistic) do not lead to good clusters of observations with significantly lower scores. Therefore, this experiment validates our choice of learning weights on observations and our choice of the GF2 features.

<table>
<thead>
<tr>
<th>Input</th>
<th>Input Weighting</th>
<th>Complete Recall</th>
<th>Complete Precision</th>
<th>Complete ARI</th>
<th>Reduced 1 Recall</th>
<th>Reduced 1 Precision</th>
<th>Reduced 1 ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations</td>
<td>Observations</td>
<td>0.51</td>
<td>0.50</td>
<td>0.44</td>
<td>0.70</td>
<td>0.64</td>
<td>0.54</td>
</tr>
<tr>
<td>Analyses</td>
<td>Analyses</td>
<td>0.80</td>
<td>0.74</td>
<td>0.74</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Analyses</td>
<td>Observations</td>
<td>0.80</td>
<td>0.74</td>
<td>0.74</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Observations</td>
<td>Analyses</td>
<td>0.44</td>
<td>0.42</td>
<td>0.36</td>
<td>0.61</td>
<td>0.59</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Table 3.8: Behaviour of GF2 set of features with PAM clustering when learning weights over observations or analyses. Column “Weighting” indicates whether weights are learned over observations or analyses.
3.3. EVALUATION OF USER INTENT DETECTION

3.3.2.4 GF2 metric robustness

The choice of the clustering algorithm and the features to learn the similarity between observations is decisive, in the sense that they have to fit the data that we collected. The main goal is to separate observations in compact clusters, as distant as possible from each-other to identify clear user intents. As shown in Table 3.9, the intra clusters dissimilarities, presented in the diagonal, are lower than inter clusters dissimilarities. While these differences are not significant because of high standard deviation (not reported for the sake of readability), this result is confirmed by the Silhouette coefficient [Rousseeuw, 1987]. This coefficient is positive for all the clusters, as presented in Table 3.10, verifying that the built clusters are cohesive and the majority of observations in each of them are well classified in their own clusters. In view of the business questions corresponding to the cluster intents, we notice a higher compactness for the groups of observations responding to well separated questions, as in clusters 9 and 10. As expected, increasing the number of finer intents we want to discover, i.e., augmenting the number of clusters, results in more compact groups of observations. For instance, for 50 clusters, better Silhouette coefficients are obtained for most of the clusters, reaching the highest value of 0.5, with the notable exception of two small clusters (having 6 and 7 observations, respectively) that manifest a negative coefficient, close to 0, showing the inconsistency of the observations composing these clusters. Regarding the diameter, it is well balanced among the discovered clusters, with some minor differences being explained by outliers.

<table>
<thead>
<tr>
<th>Cluster IDs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.716</td>
<td>0.798</td>
<td>0.968</td>
<td>0.957</td>
<td>0.919</td>
<td>0.935</td>
<td>0.973</td>
<td>0.913</td>
<td>0.942</td>
<td>0.892</td>
</tr>
<tr>
<td>2</td>
<td>0.722</td>
<td>0.963</td>
<td>0.950</td>
<td>0.887</td>
<td>0.908</td>
<td>0.972</td>
<td>0.886</td>
<td>0.922</td>
<td>0.915</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.869</td>
<td>0.960</td>
<td>0.934</td>
<td>0.941</td>
<td>0.929</td>
<td>0.928</td>
<td>0.963</td>
<td>0.983</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>0.764</td>
<td>0.913</td>
<td>0.910</td>
<td>0.969</td>
<td>0.945</td>
<td>0.970</td>
<td>0.975</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>0.763</td>
<td>0.873</td>
<td>0.962</td>
<td>0.892</td>
<td>0.933</td>
<td>0.962</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.781</td>
<td>0.955</td>
<td>0.937</td>
<td>0.951</td>
<td>0.967</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.834</td>
<td>0.969</td>
<td>0.981</td>
<td>0.984</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.701</td>
<td>0.809</td>
<td>0.967</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.691</td>
<td>0.973</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.634</td>
</tr>
</tbody>
</table>

Table 3.9: Average dissimilarities between clusters.

<table>
<thead>
<tr>
<th>Cluster IDs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>#observations</td>
<td>48</td>
<td>77</td>
<td>53</td>
<td>74</td>
<td>58</td>
<td>60</td>
<td>78</td>
<td>34</td>
<td>30</td>
<td>18</td>
</tr>
<tr>
<td>Silhouette</td>
<td>0.09</td>
<td>0.08</td>
<td>0.01</td>
<td>0.14</td>
<td>0.09</td>
<td>0.08</td>
<td>0.1</td>
<td>0.13</td>
<td>0.15</td>
<td>0.27</td>
</tr>
<tr>
<td>Diameter</td>
<td>0.71</td>
<td>0.72</td>
<td>0.86</td>
<td>0.76</td>
<td>0.76</td>
<td>0.78</td>
<td>0.83</td>
<td>0.70</td>
<td>0.69</td>
<td>0.63</td>
</tr>
</tbody>
</table>

Table 3.10: Silhouette coefficients and diameter for 10 clusters obtained with PAM using GF2 based dissimilarities.
3.3. EVALUATION OF USER INTENT DETECTION

3.3.2.5 Handling unseen business needs

In this experiment, we study how our method handles previously unseen business needs and how general the metric learned on the GF2 features is. To this aim, we consider both Reduced data sets and use one to train the metric and the other to test with PAM clustering. Recall that reduced data sets cover different business needs, with no overlap among them. The results in Table 3.11 show that our metric is indeed general and can adapt to new business needs as there is no drop in performance between each of the generalization tests. Moreover, the results are comparable to those observed in previous tests, as reported in Table 3.12. Finally, it can be seen that testing on Reduced 2 leads to better results than with Reduced 1. This is expected, as Reduced 2 contains observations related to business need Q9, which has more relevant queries than Q10 contained in the Reduced 1 data set (see Table 3.5).

<table>
<thead>
<tr>
<th>Training</th>
<th>Testing</th>
<th>Recall</th>
<th>Precision</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced 2</td>
<td>Reduced 1</td>
<td>0.76</td>
<td>0.67</td>
<td>0.61</td>
</tr>
<tr>
<td>Reduced 1</td>
<td>Reduced 2</td>
<td>0.73</td>
<td>0.71</td>
<td>0.62</td>
</tr>
</tbody>
</table>

Table 3.11: Generalization of our approach. Each test correspond to the training of the metric and discovery of user intents on different subsets of business needs.

3.3.2.6 Comparative experiments

Table 3.12 shows how our metric compares to a reference metric that [Aligon et al., 2014b] designed for OLAP queries. This metric has been validated by user tests that showed its effectiveness in grouping queries in accordance with what a human expert would have done. Table 3.12 reveals 2 distinct behaviors depending on whether we consider the Complete data set or the Reduced 1 (where clusters are well separated). With the Complete data set, our metric with GF2 features performs better than the other metrics, as it only relies on the most discriminating features. Indeed, we know from the protocol that the groups of observations heavily overlap. Thus, our metric, based on SVM, cannot find a proper linear separation between observations related to different user interests. In this particular context, adding more features makes the problem even more complex to solve for SVM, as it has to determine a solution that compromises over 15 dimensions for ALL features rather than 5 in the case of GF2 features, and with only a few training instances. In contrast, with the Reduced 1 set of observations, groups are clearly separable, the problem is much easier for the linear SVM and adding features may help in finding a better solution by fine tuning the separation hyper plane. Consequently, in this case, slightly better results may be achieved with features other than GF2’s.

If we dig into the details of the features involved in each result, it can be seen that good results are achieved when the weights of the features related to queryParts, tokenQPart or
3.3. EVALUATION OF USER INTENT DETECTION

<table>
<thead>
<tr>
<th>Features</th>
<th>Recall</th>
<th>Precision</th>
<th>ARI</th>
<th>Recall</th>
<th>Precision</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete 10 clusters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>0.52</td>
<td>0.46</td>
<td>0.42</td>
<td>0.73</td>
<td>0.64</td>
<td>0.56</td>
</tr>
<tr>
<td>GF2</td>
<td>0.51</td>
<td>0.50</td>
<td>0.44</td>
<td>0.70</td>
<td>0.64</td>
<td>0.54</td>
</tr>
<tr>
<td>Metric [Aligon et al., 2014b]</td>
<td>0.39</td>
<td>0.20</td>
<td>0.14</td>
<td>0.41</td>
<td>0.33</td>
<td>0.10</td>
</tr>
<tr>
<td>ALL + [Aligon et al., 2014b]</td>
<td>0.40</td>
<td>0.40</td>
<td>0.32</td>
<td>0.78</td>
<td>0.65</td>
<td>0.63</td>
</tr>
<tr>
<td>GF2 + [Aligon et al., 2014b]</td>
<td>0.45</td>
<td>0.43</td>
<td>0.38</td>
<td>0.69</td>
<td>0.62</td>
<td>0.52</td>
</tr>
</tbody>
</table>

Table 3.12: Comparison of our metric based on GF2 features with other metrics when paired with PAM clustering. ALL denotes the set of 15 features, [Aligon et al., 2014b] is the state-of-art metric and “+” indicates a metric with added features and corresponding weights.

Chosen are important. For the COMPLETE data set, this is observed for GF2 features that reach the best overall clustering results. In the case of ALL features, queryParts is slightly below, which may explain the small difference in performance with GF2 features. Adding the similarity proposed by [Aligon et al., 2014b] to GF2 or ALL decreases the weights of queryParts or tokenQPart, which in turn degrades the results. These observations are coherent with what is observed for the REDUCED 1 data set. In this case, adding the similarity proposed by [Aligon et al., 2014b] to GF2 only slightly lowers the score on tokenQPart and queryPart, which again may explain the small difference observed. Finally, with ALL features, it can be seen that adding the similarity proposed by [Aligon et al., 2014b] causes a drop in the previous important features, but that is compensated by the emergency of new features, such as the OLAP operation. However, we expect our approach to be the most efficient in any scenarios and the hypothesis that clusters of observations are clearly separated is too strong in practice. Thus, the metric based on GF2 features seems to be the most appropriate among those that we evaluated, in particular when compared to the state-of-the-art metric [Aligon et al., 2014b].

3.3.2.7 Discovering intra-interaction intents

In this test, we successively increase the number of clusters and we check how many users of different expertise are represented in each cluster. The aim is to show that our metric is good not only at grouping observations that participate in the resolution of a particular business need but also at identifying parts of the resolution that are shared by users with different expertise. To emphasize the evolution of precision (which indicates the coherence of clusters), we use the (GF2 + [Aligon et al., 2014b]) configuration, which is a good compromise in the previous experiment, and test it on the well separated REDUCED 1 data set, starting from the beginning with an increased number of clusters, with 10 clusters in a space that we expect only 4. The results reported in Table 3.13 show how the mixing of users decreases while the precision increases (and consequently recall and ARI
3.4. CHAPTER CONCLUSIONS

decrease) as we increase the number of clusters. It can be noted that for high precisions, the composition of clusters in terms of users with different expertise remains very acceptable. For instance, when precision reaches 95%, more than 63% of clusters have users with different expertise. In other words, this shows that our metric can be used to identify shared sub-tasks (or intra-interaction intents), where some experts’ queries could be recommended to beginner users having to solve the same business need.

<table>
<thead>
<tr>
<th># clusters</th>
<th>Recall</th>
<th>Precision</th>
<th>ARI</th>
<th>Dense UI</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.35</td>
<td>0.86</td>
<td>0.41</td>
<td>10 (100%)</td>
<td>10 (100%)</td>
</tr>
<tr>
<td>15</td>
<td>0.24</td>
<td>0.90</td>
<td>0.31</td>
<td>14 (93.3%)</td>
<td>14 (93.33 %)</td>
</tr>
<tr>
<td>20</td>
<td>0.20</td>
<td>0.92</td>
<td>0.26</td>
<td>14 (70%)</td>
<td>18 (90 %)</td>
</tr>
<tr>
<td>25</td>
<td>0.18</td>
<td>0.92</td>
<td>0.24</td>
<td>13 (52%)</td>
<td>19 (76 %)</td>
</tr>
<tr>
<td>30</td>
<td>0.17</td>
<td>0.95</td>
<td>0.23</td>
<td>13 (43.3%)</td>
<td>19 (63.33 %)</td>
</tr>
<tr>
<td>35</td>
<td>0.16</td>
<td>0.95</td>
<td>0.22</td>
<td>12 (34.3%)</td>
<td>19 (54.29 %)</td>
</tr>
<tr>
<td>50</td>
<td>0.14</td>
<td>0.96</td>
<td>0.19</td>
<td>11 (22%)</td>
<td>20 (40 %)</td>
</tr>
</tbody>
</table>

Table 3.13: Increasing the number of clusters to detect intra-interaction intents. Dense UI indicates the number of clusters with more than 5 different users. Expertise indicates the number of clusters with both types of users (beginners and experts).

3.4 Chapter conclusions

Our first contribution in this thesis is an approach for identifying coherent intents of BI users with various expertises, querying datasources by means of keyword-based analytical queries. Our approach relies on the identification of discriminative features for characterizing BI interactions and on the learning of a similarity measure based on these features. We have shown through user tests that our approach is effective in practice and could benefit beginner analysts whose interests match those of expert users. Overall, our results show that keyword-based interaction systems provide semantically rich user traces well adapted to the detection of coherent BI user interest.

This work is presented as a long paper [Drushku et al., 2017] in the International Conference on Advanced Information Systems Engineering (CAISE). Furthermore, the identification of user intents in a BI context is now a patent of SAP, delivered in June 2018. Here is the patent reference of this application in a SAP BI system: us 20180157734A1: "Business Intelligence System Dataset Navigation Based on User Interests Clustering".
Chapter 4

Intent-based recommender system

4.1 Introduction

To illustrate the practical use of our user intents elicitation approach, in this chapter, we present IbR (Interest-based Recommender), a simple recommender system specifically designed to exploit the clusters that represent user intents. IbR builds upon previous approaches proposed to predict or recommend OLAP queries.

First, inspired by the collaborative recommender system described in [Aligon et al., 2015], IbR recommends a sequence of queries representing the sequence of moves that is expected to best complete the beginning of an interaction. As remarked by [Aligon et al., 2015], it is expected that users, especially non-expert ones, benefit from a sequence of recommended queries, in that it gives them a compound and synergic view of a phenomenon, carries more information than a single query or set of queries by modeling the potential expert user’s behavior after seeing the result of the former query.

Second, we borrowed from the work of [Sapia, 2000] and the work of [Aufaure et al., 2013] the idea of using an order-1 Markov model to probabilistically represent user behaviors. Like in the latter [Aufaure et al., 2013], the states of the Markov model are clusters constructed from a set of past interactions, with the notable difference that observations are used in our case, instead of queries, and that the clusters represent user intents. IbR can be seen as a model that guides the user’s next moves based on the probabilities of moving between discovered user intents.

By construction, we expect our recommender system to have two types of benefits: the first one is sharing expertise between different users, and the second one is recommending queries that are diversified in terms of intents.

4.2 Item-based Recommender System

The principle of the recommender system follows the same two-step approach as that of [Aufaure et al., 2013]. The first step is off-line and consists in clustering the observations to detect user intents, as detailed in the previous sections. The second step consists in treating these clusters as states of a Markov chain model and in computing the probabilities
of the most likely next state as explained below. The only on-line phase of the recommender is when a new interaction begins, each observation of the interaction is used to compute the most likely query in the sense of the Markov model. An exhaustive calculation is needed for the first observation to define its current state by comparing it with every observation of every state. For the rest of observations, an exhaustive search of its current state is not needed since the recommendation only derives from the previous recommendations, i.e., the last state calculated by the Markov model.

4.2.1 Learning the Markov model

The creation of the Markov model is done as follows. Let \( C_I \) be the set of clusters expressing user intents. The states of the Markov model are the clusters of \( C_I \). The transition probability distribution is given by:

\[
\Pr(X_{n+1} = x \mid X_n = y) = \frac{n_{xy}}{n_y}
\]

where \( x \) and \( y \) are clusters in \( C_I \), \( n_y \) is the size (the number of observations) of cluster \( y \) and \( n_{xy} \) is the number of interactions that contain two adjacent observations \( o_i, o_{i+1} \) such that \( o_i \) is in cluster \( y \) and \( o_{i+1} \) is in cluster \( x \). We use a special state to represent the end of interactions, which is used to obtain the probability of ending the recommendation.

4.2.2 The prediction algorithm

Given an observation, called the current observation (whose chosen query is called the current query) from now on, we identify the user intent (i.e., the cluster) from which this observation is the closest to by computing the average similarity between the current observation and all the observations of each cluster. Noticeably, instead of all pairwise distance calculations, a possible optimization would be to directly compute the distance between the current observation and the representative of each cluster. However, as this optimization is not guaranteed to lead to the same result as that obtained with all pairwise calculations, especially in the presence of overlapping clusters, we deliberately chose not to implement it.

Once we have identified the cluster, the Markov model gives the most likely next state. By construction, since states coincide with user intents, it is expected that the most likely next state is the current one. To distinguish between the two types of benefits our recommender can have, we devised two strategies for generating the recommended sequence, reflected in the two modes our recommender can operate. Mode 1, named IbR1, tries to benefit from the expertise coming from this next probable cluster only. Conversely, Mode 2, named IbR2, tries to anticipate when users change their focus and, for instance, address other business questions in their explorations. IbR2 fully combines the Markov model with intent detection, with a two-fold purpose, which is: i) to anticipate the users’ change of focus, and ii) to propose recommendations diversified in terms of user intents. We now describe these two modes precisely.
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

IbR1 forces the recommender to choose the queries for the recommended observations in the next probable state only. In other words, IbR1 does not use the Markov model but uses only intent identification. The chosen queries are ordered by decreasing similarity to the current query. The length of the recommended sequence is ruled by a similarity threshold that ends the sequence if the similarity between two consecutive queries is considered too small.

The second mode, named IbR2, fully uses the Markov model based on user intents. In other words, IbR2 acknowledges the fact that user interactions may span across different intents and composes the recommended sequence of queries as follows:

1. the first query of the sequence is the chosen query of the observation that is the most similar to the current observation;
2. this observation is used as the new current observation for which the next intent is identified with a random draw using the Markov model, which means that the probability to reach another intent is low, not null, which is different from IbR1;
3. the most similar observation of the next probable state, according to the Markov Model, that has not been yet recommended, is identified and is added to the sequence;
4. this algorithm iterates until the final state of the Markov model is reached.

4.3 Evaluating the Intent-based Recommender System

We experiment with the two recommendation modes that use the Markov model over the clusters (introduced in Section 4.1), respectively called IbR1 and IbR2, and we compare them with two state-of-the-art query recommender systems: the one proposed by [Aligon et al., 2015] and one of the recommenders proposed by [Eirinaki et al., 2014], adapted to recommend a sequence of queries instead of a set of queries (Section 4.3.1.3).

We first present our protocol (Section 4.3.1), and then the experimentation results (Section 4.3.2).

4.3.1 Experimental protocol

4.3.1.1 Recommender system construction and analysis

Our recommender system is built as a Markov model over the interactions from which observations have been clustered to identify user intents. Consistent with the protocol proposed in [Aufaure et al., 2013], we remove from the set of interactions the ones consisting of only one observation. For the constructed Markov model, we report the transition probabilities between clusters and check if, as expected for consistent user intent, the highest probabilities are for transitions leading from one cluster to itself.
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

4.3.1.2 Evaluation of recommendation results

To evaluate our approach, we rely on the literature on recommender systems [Herlocker et al., 2004, Baeza-Yates et Ribeiro-Neto, 2011, Gunawardana et Shani, 2015] as well as on a recent protocol specially conceived for comparing recommendations of query sequences [Aligon et al., 2015]. We measure two of the most commonly employed criteria to judge the recommendation quality to assess whether our recommender is able to achieve a good balance between the ability to recommend and the quality of its recommendations, namely:

- accuracy, i.e., the degree to which recommendations correspond to what is expected in terms of queries, and
- coverage, i.e., the degree to which recommendations can indeed be generated.

We use evaluation methods from the state-of-the-art and we enrich this set of measures with the following criteria to understand whether our recommender system favors expertise sharing between users and intent diversity:

- expected diversity, i.e., the degree to which recommendations correspond to what is expected in terms of user intents,
- expected user, i.e., the degree to which the current user is retrieved in the recommendations,
- expertise, i.e., the degree to which recommendations come from experts, and
- expertise benefit, i.e., the degree to which beginners can benefit from expert recommendations.

Regarding accuracy, the protocol acknowledges the fact that finding the exact next query of an interaction is very unlikely, as our set of interactions consists of mostly unique observations. The protocol therefore implements extended versions of precision and recall measures to incorporate similarity between interactions.

For our tests, we use the similarity between sessions defined and proposed in [Aligon et al., 2014b], that is itself based on the similarity between queries. In our adaptation we assimilate interactions to sessions and we use the similarity between queries to compare the chosen queries of interactions to recommended queries. We use this similarity measure because it is independent from our proposal and can fit any recommender system under testing, contrary to ours, which needs proper interactions to work.

Definitions. The underlying idea is, given the beginning of an interaction, to compare the recommended queries with the unseen queries (that actually continued the interaction) and calculate the precision, recall and F-measure. In what follows, depending on the criteria considered, we use \( \sim \) to denote a similarity function between sequences of queries, which is instantiated differently in each measure (accuracy, expertise, expected diversity, expected user).

Let \( I \) be a set of interactions and \( I_C \) be a set of current interactions for which recommendations are to be computed. Given an interaction \( i \in I_C \), let \( f_i \) be its actual future
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

(i.e., the sequence of queries the user would have formulated after the last query of \(i\) if they had not been given any recommendation) and \(r_i\) be a recommended future. Recommendation \(r_i\) is considered to be correct when \(r_i \sim f_i\), i.e., when it is similar to the actual future of \(i\).

Let \(FI = \{f_i | i \in I_C\}\) and \(RI = \{r_i | i \in I_C\}\). The set of true positives is then defined by

\[
TP = \{r_i | i \in I_C ; r_i \sim f_i\}
\] (4.2)

i.e., the set of recommended futures similar to their actual counterparts. The set of false positives is \(FP = RI \setminus TP\) and the set of false negatives is \(FN = FI \setminus TP\). Then:

\[
\text{Recall} = \frac{|TP|}{|TP| + |FN|} = \frac{|TP|}{|FI|}
\] (4.3)

\[
\text{Precision} = \frac{|TP|}{|TP| + |FP|} = \frac{|TP|}{|RI|}
\] (4.4)

\[
F\text{-measure} = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\] (4.5)

Let \(RQ\) be the set of queries in \(RI\), \(FQ\) be the set of queries in \(FI\), \(TQ\) be the set of queries in \(TP\), \(I_E\) be the set of interactions written by expert users, \(Q_E\) be the set of queries of \(I_E\), \(Q_{Uj}\) be the set of queries of interactions of interest \(U_j\) for \(j \in \{1, c\}\), where \(c\) is the number of discovered intents, and \(Q_{user_k}\) be the set of queries of interactions of user \(k\), for \(k \in \{1, |\tau_u|\}\), where \(\tau_u\) is the set of users.

Our measures are defined as follows:

- **accuracy** is measured using the F-measure, where \(\sim\) is the similarity between sessions proposed in [Aligon et al., 2014b],

- **coverage** is the number of recommendations divided by the number of interactions:

\[
\frac{|RI|}{|I_C|}
\] (4.6)

- **expected diversity** is measured using F-measure, where \(\sim\) is the Jaccard similarity on interests of interactions, defined by:

\[
r_i \sim f_i = \frac{|\{Q_{U_j} | \exists q \in Q_{U_j} \cap r_i\} \cap \{Q_{U_j} | \exists q \in Q_{U_j} \cap f_i\}|}{|\{Q_{U_j} | \exists q \in Q_{U_j} \cap r_i\} \cup \{Q_{U_j} | \exists q \in Q_{U_j} \cap f_i\}|}
\] (4.7)

- **expected user** is measured using the F-measure, where \(\sim\) is the Jaccard similarity of users participating in interactions. Note that observations in \(f_i\) correspond to a unique user (the actual user), while recommended observations in \(r_i\) may come from several users.

\[
r_i \sim f_i = \frac{|\{Q_{user_k} | \exists q \in Q_{user_k} \cap r_i\} \cap \{Q_{user_k} | \exists q \in Q_{user_k} \cap f_i\}|}{|\{Q_{user_k} | \exists q \in Q_{user_k} \cap r_i\} \cup \{Q_{user_k} | \exists q \in Q_{user_k} \cap f_i\}|}
\] (4.8)
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

- **expertise** is measured with the F-measure, but for a version of precision and recall that incorporate expert queries, as follows:

\[
\text{precision}_{\text{expertise}} = \frac{|TQ \cap Q_E|}{|TQ|} \quad (4.9)
\]

\[
\text{recall}_{\text{expertise}} = \frac{|TQ \cap Q_E|}{|Q_E|} \quad (4.10)
\]

- **expertise benefit** is measured as the probability to recommend a query made by an expert to a session made by a beginner:

\[
\text{benefit}_{\text{expertise}} = P(q \in Q_E \cap r_i | i \in I_C \setminus I_E) \quad (4.11)
\]

**Protocol** To create sets \(FI, RI\) and \(TP\), our protocol uses cross-validation as follows. We iterate over a set \(L\) of interactions with a leave-one-out approach by (i) picking one interaction \(i \in L\); (ii) taking one of its prefix \(i_n\) of size \(n\) as one current interaction of \(i\) and the remaining subsequence \(f_i\) as one actual future of \(FI\), with \(n \in \{1, |i - 1|\}\), where \(|i - 1|\) is the number of the interaction’s queries; (iii) finding a recommendation \(r_i\) for \(i_n\) using the remaining interactions, \(L \setminus \{i\}\). If such a \(r_i\) exists, it is added to \(RI\). \(r_i\) is considered correct and added to \(TP\) when \(r_i \sim f_i\) and is incorrect otherwise. For accuracy and expertise, the similarity between interactions is parametrized by a threshold varying in \([0,0.9]\). This threshold controls the extent to which two interactions should be considered similar. Hence these criteria are measured by progressively increasing this threshold ruling the minimal demanded similarity between the expected future and the recommendation. The same is done to compare systems in terms of expected diversity and expected user, with a threshold ruling the similarity of sets of recommended queries in terms of intents and users, respectively.

4.3.1.3 Comparison with state-of-the-art recommendation algorithms

In order to illustrate the added value of user intent detection, as well as to show that our approach is agnostic of the query language used, we compare it with two state-of-the-art, session-based recommendation algorithms, both based on collaborative filtering and kNN. These approaches are i) an OLAP session recommendation approach [Aligon et al., 2015, Aligon et al., 2014a], referred to as Falseto in what follows, and ii) a SQL query recommendation approach, QueRIE [Eirinaki et al., 2014]. Precisely:

1. **QueRIE**. We use the fragment-based, non-binary version of QueRIE, which handles queries (both in the log and in the current session) represented by their SQL fragments, i.e., projected attributes (levels and measures), selections, and group-by expressions, that are easily extracted from the query-parts. This representation of user sessions based on query fragments is called the signature. The recommended queries are selected from the closest session to the current one based on their respective signatures. As the number of queries to recommend is a parameter in QueRIE,
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

we tuned it and selected the value of the parameter that achieved the best accuracy, precisely 2 queries. Finally, the set of recommended queries is ranked using the similarity to the current session and arranged in a sequence as was done for IbR1. Note that this transformation of QueRIE output is necessary to ensure that it is comparable to the other recommenders and is under the same conditions.

2. Falseto. In order to use Falseto, an OLAP schema (a constellation) was reverse engineered from data sources (schemata and constraints) following DFM methodology [Golfarelli et Rizzi, 2009], pruning attributes not accessed by users (based on the query log). Interfacing with Falseto is straightforward, as we use the same format for representing queries. However, it should be noted that different from IbR1, IbR2 and QueRIE, Falseto does not directly recommend queries that are simply picked in a log file of past queries. Indeed, it picks queries from a log file and then modifies these queries to align them with the current interaction. Therefore, to measure the diversity and expertise related criteria, we disregarded that alignment and looked at the original queries picked in the log. We expect this recommender system to explore more globally the space of possible queries as it builds new queries (not necessarily existing in the logs) based on current queries.

We use our own implementation of QueRIE and Falseto.

4.3.1.4 Impact of user intents on the recommendation strategy

Finally, we aim at investigating whether leveraging user intents calls for a tailored recommendation strategy or can benefit an existing one. To this end, we give Falseto and QueRIE, which are agnostic of user intents, the chance of knowing the user intent beforehand by restricting their input to one particular user intent. We repeat this test for each discovered user intent. We report their accuracy and coverage in each restricted log and compare them to their own results on the whole log.

4.3.2 Lessons learned

In this section, we present the results of all our experiments. We measure the quality of our recommender in Section 4.3.2.2 and we perform a comparative evaluation with other works of the state-of-the-art algorithms in Section 4.3.2.3 and in Section 4.3.2.4, we study their behavior in the presence of user intents.

4.3.2.1 Recommender system construction and analysis

The Markov model at the heart of our recommender is built from a set of 24 interactions corresponding to 530 unique observations. We removed 17% of the interactions from it, which contain only one observation and do not provide any information to the Markov model.

Table 4.1 presents the transition probabilities between clusters (states), sources in rows and targets in columns. As this model is recreated several times, these are the average probabilities measured over all the models created in our tests.
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

It is easily perceived that, as expected, observations of a cluster are mainly followed by observations of the same cluster, meaning that interactions tend to remain within the same user intent.

Table 4.1: Transition probabilities for the 10 states (clusters) of the recommender’s Markov model

<table>
<thead>
<tr>
<th>State 1</th>
<th>State 2</th>
<th>State 3</th>
<th>State 4</th>
<th>State 5</th>
<th>State 6</th>
<th>State 7</th>
<th>State 8</th>
<th>State 9</th>
<th>State 10</th>
<th>Final State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.40</td>
<td>0.13</td>
<td>0.06</td>
<td>0.0</td>
<td>0.08</td>
<td>0.13</td>
<td>0.02</td>
<td>0.04</td>
<td>0.0</td>
<td>0.0</td>
<td>0.14</td>
</tr>
<tr>
<td>0.30</td>
<td>0.58</td>
<td>0.06</td>
<td>0.04</td>
<td>0.05</td>
<td>0.04</td>
<td>0.03</td>
<td>0.03</td>
<td>0.01</td>
<td>0.04</td>
<td>0.02</td>
</tr>
<tr>
<td>0.0</td>
<td>0.06</td>
<td>0.43</td>
<td>0.02</td>
<td>0.09</td>
<td>0.13</td>
<td>0.06</td>
<td>0.0</td>
<td>0.13</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>0.0</td>
<td>0.03</td>
<td>0.05</td>
<td>0.68</td>
<td>0.05</td>
<td>0.03</td>
<td>0.05</td>
<td>0.0</td>
<td>0.05</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>0.02</td>
<td>0.07</td>
<td>0.02</td>
<td>0.05</td>
<td>0.54</td>
<td>0.03</td>
<td>0.15</td>
<td>0.10</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.02</td>
<td>0.17</td>
<td>0.02</td>
<td>0.08</td>
<td>0.67</td>
<td>0.46</td>
<td>0.15</td>
<td>0.03</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.04</td>
<td>0.0</td>
<td>0.10</td>
<td>0.14</td>
<td>0.0</td>
<td>0.1</td>
<td>0.55</td>
<td>0.03</td>
<td>0.0</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>0.03</td>
<td>0.03</td>
<td>0.08</td>
<td>0.0</td>
<td>0.06</td>
<td>0.03</td>
<td>0.15</td>
<td>0.50</td>
<td>0.06</td>
<td>0.0</td>
<td>0.06</td>
</tr>
<tr>
<td>0.17</td>
<td>0.03</td>
<td>0.03</td>
<td>0.0</td>
<td>0.03</td>
<td>0.07</td>
<td>0.03</td>
<td>0.10</td>
<td>0.50</td>
<td>0.0</td>
<td>0.04</td>
</tr>
<tr>
<td>0.11</td>
<td>0.06</td>
<td>0.11</td>
<td>0.0</td>
<td>0.06</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.39</td>
<td>0.27</td>
</tr>
</tbody>
</table>

4.3.2.2 Evaluation of recommendation results

Figure 4.1, 4.2 and 4.3 report the measures of the various criteria defined to assess the quality of the recommenders. We start by discussing these measures for IbR1 and IbR2. The coverage is as expected. By design, IbR1 achieves a perfect coverage, while IbR2, using a probability for ending the session, may not recommend, particularly for a longer current session. Regarding accuracy, both recommenders perform very well, with IbR1 performing the best, when the similarity threshold is set low (0.4 or below). Below this threshold, both recommenders show the same behavior. In terms of expected diversity, as expected, IbR2 outperforms IbR1 since the latter cannot move outside a current intent. Notably, even for quite demanding similarity thresholds, IbR2 performs reasonably well in predicting intent switches. The very low scores for both recommenders in terms of the expected user is expected in that it confirms that none of them were designed to stick to the current user. Nevertheless, we note that both IbR1 and IbR2 still do better than state-of-the-art recommenders for low similarity thresholds, which can be interpreted as a side effect of user intent detection. Both recommenders perform well in terms of expertise, with IbR2 being more robust than IbR1 to the similarity threshold. This is due to IbR2 being more likely to find expert queries in clusters other than the current one. Finally, both recommenders perform fairly in recommending expert queries to beginners. We note that they were not designed to do so and good performances for this criterion would have been a side effect of clustering intents. However, extending the recommenders to favor this behavior can be done easily if expertise is recorded or can be deduced from the observations. In summary, IbR1 performs slightly better in terms of accuracy and coverage, while IbR2, with its global exploration of the user intents, is better at identifying intent switching and proposing recommendations coming from expert users.
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

![Coverage and accuracy for IbR1, IbR2, Falseto and QueRIE.](image1)

![Expected diversity and expected user for IbR1, IbR2, Falseto and QueRIE.](image2)

4.3.2.3 Comparison with state-of-the-art algorithms

We now discuss how IbR1 and IbR2 compare to two state-of-the-art recommenders.

**Comparison with QueRIE.** QueRIE achieves perfect coverage, as does IbR1, and it is similar to it in terms of accuracy for low similarity thresholds, being slightly more robust to more demanding thresholds. This similarity of behavior can be explained by the nature of both recommender systems, which are very similar as they tend to locally explore the user intent based on current queries. Interestingly, in this case, the difference in the similarities used to rank potential recommended queries in QueRIE and IbR1 (fragment-based versus feature-based) has no influence since potential queries to recommend are already issued from the same user intent. We note that QueRIE is always better than IbR1 for expected diversity, since the latter is bound to a specific intent, and is slightly better than IbR2 for very low similarity thresholds but is expectedly less robust than it to high thresholds. Finally, as expected, its results in terms of expected user, expertise or expertise benefit show that it has not been specifically designed to take these features into account.

**Comparison with Falseto.** Among all the recommenders, Falseto achieves the worst performances both in terms of coverage and accuracy for low similarity thresholds. Re-
4.3. EVALUATING THE INTENT-BASED RECOMMENDER SYSTEM

regarding coverage, it is clearly impacted by the demanding session similarity measure that Falseto internally uses to align current and past sessions to generate candidate recommendations. Indeed, when query similarity is below Falseto’s built-in threshold, no past session is found to be similar to the current one, which results in no candidate recommendations, which disables the recommendation. Remarkably, Falseto is more robust in terms of accuracy when the similarity threshold becomes more demanding. This can be explained by its fitting phase, which aligns the recommendations with the current interaction, i.e., even if the candidate recommendation picked from the log is not the one expected, the fitting phase is able to sufficiently modify it to bring it closer to the expected future. As expected, Falseto is outperformed in terms of expected diversity, expected user, and expertise, but surprisingly achieves the best expertise benefit. This can be because its candidate recommendations are sequences that are similar to others in the log and that such sequences are more likely produced by expert users.

4.3.2.4 Impact of user intents on the recommendation strategy

In this last test, we observe the behavior of Falseto and QueRIE in the presence of discovered user intents. More precisely, we force them to recommend queries inside each cluster separately and to simulate their behavior if they were not agnostic of user intent. The goal of these tests is to compare the accuracy and coverage of the recommender over a user intent, with itself on the whole log. Note that we do not intend to investigate which cluster achieves better performance, so the identification of individual curves is irrelevant. The results are reported in Figures 4.4 and 4.5 where the test is done for all of the 10 detected user intents, with the dark square curve representing the recommender over the whole log. Coverage for QueRIE is not depicted because QueRIE achieves perfect coverage in all cases.

The results show that detecting user intent may be useful for already existing recommendation strategies. This is particularly clear for QueRIE, which performs better in the majority of cases when the intent is leveraged. QueRIE benefits from targeting its recommendation in a specific context, which increases the similarity between recommendations and the actual future that is mostly contained in a single user intent. Notably, Clusters 6 and 10 are those for which QueRIE obtains its worst results. These are less homogeneous clusters (see Table 3.9), leading to the observed decrease in accuracy.
4.4. CHAPTER CONCLUSIONS

This is more contrasted for Falseto, where in three cases only leveraging user intents makes the recommendations more accurate, interestingly, for high similarity thresholds. Due to its fitting phase, Falseto is more likely to be sensitive to cluster overlapping, and thus more likely to deviate to other neighboring intents. Therefore, when only one cluster is available, Falseto will miss those sessions spanning different clusters. Similarly, restricting the past history to a single user intent can decrease the coverage of Falseto because of the size of the clusters representing each user intent and because the intra-cluster similarity is not on par with the session-based similarity used by Falseto.

4.4 Chapter conclusions

In this chapter we presented a collaborative recommendation approach that leverages user intents in modern BI systems to relieve the user from tedious explorations. The discovered user intents are treated as first-class citizens in a collaborative BI query recommender system, that suggest next moves in an exploration based on the probability for a user to switch from one intent to another.

We have shown through user tests that our approach is effective in practice and can be beneficial to analysts whose intents match those of expert users, or whose intents change
during the analysis. Overall, our results show that keyword-based interaction systems pro-
vide semantically rich user traces well adapted to the detection of coherent BI user intent
and that such intents can also be exploited successfully by state-of-the-art recommendation
strategies.

This work was published in the Information System Journal [Drushku et al., 2019],
as an extended work of the identification of user intents paper published in CAISE. A
patent is filed by SAP to protect this system that recommends based on user intents, not
published yet.
Chapter 5

Recommending Complementary Items

5.1 Introduction

As recalled in [Djedaini, 2018], a BI exploration goes through several phases, where the user devises queries to discover and learn generalities about the data she is querying, before getting more and more focused and engaged in the task. As a consequence, frequently in a BI analysis, the answer is not obtained by the execution of a single query, but by a sequence of queries that help the user build her knowledge gradually. This notion is very important as the correct answer can only be carried out by a set of queries that are complementary to each others, while sufficiently representative and diverse to cover most aspects of the available data related to the question at hand. The queries are ordered so as to bring new knowledge, in a way that involves less cognitive load from the user perspective when going from one query to the next. Finally, a modern proactive recommender system for BI should also take into account the user profile and the context to improve the effectiveness of the recommended queries and ultimately make the data exploration process less tedious.

In this chapter we propose to adapt our metric for discovering user intents, presented in Section 3.2 of Chapter 3, and to translate this constrained recommendation problem as a problem of creating a bundle of queries. As discussed in Chapter 2, this research problem is completely new in the domain of BI. However, we find in the literature many contributions that have been proposed in the domain of Web search systems or for other specialized recommendation tasks, like crowd-sourcing workers [Alsayasneh et al., 2018, Amer-Yahia et al., 2013, Amer-Yahia et al., 2014], already presented in Section 2.5. These works emphasize on building bundles of items that exhibit some of the aforementioned properties that we expect from our new recommender system for BI. The new constraints that we introduce, in addition to the ones already elaborated in these approaches, are the relevance to context and the order of queries inside the bundle.
5.2 USER INTENTS SPACE TO COMPARE ENTITIES

Our study takes place in the context of a SAP BI platform that allows users to create reports composed of several queries. Each query is paired with an adequate visualization of the result returned during its execution. One interesting mechanism of this BI software is the possibility left to the user to share with other users some of the visualized queries (or queries for short) of the report. The objective of our recommender is to proactively recommend a bundle of queries that completes a report the user is accessing, under the following constraints for the bundle of queries:

- it must contain a pre-specified number of queries, that defines what is called the size of the bundle,
- it should be on par with the context, defined by the actual report, and with a user profile, both expressed over user intents that are learned over the past history of queries for all users, which is a problem already resolved in Chapter 3.
- it should exhibit the aforementioned properties such as: all queries of the bundle should be uniform and deal with the similar user intents, but still diverse, while being ordered in such a way that minimizes the cognitive gap from one query to the next.

The contributions of this research include:

1. a first application of bundle recommendation in the BI context,
2. an application of our feature-based metric to discover user intents and use them differently in composite recommendation,
3. a method for building bundle of queries, named BundleIntent-basedRecommender (BIbR), that brings together several constraints such as context and user profile, and focuses on the completeness of the report with a specific ordering of the queries contained in the bundle,
4. a real use case on SAP BI platform with a preliminary subjective evaluation of our algorithm, paired with a more traditional objective evaluation of our proposal as a recommender system.

This chapter is organized as follows: 5.2 shows how we model and formalize our data, and how we exploit them to discover user intents in order to represent entities over the same vectorial space and to define our problem. Section 5.3 defines the constrained recommending problem and presents the constraints we introduce to the objective function. Details of algorithms that resolve our problem are presented in Section 5.4. We provide the experimental protocol and results in Section 5.5, aiming at tuning the parameters of our dependent algorithms and evaluating the recommendations offline and online, and we conclude this chapter in Section 5.6.

5.2 User Intents space to compare entities

In the context of recommendation, as several entities like items, users profiles and contexts contribute to the recommendation process, it is important to be able to represent them...
all into a single vectorial space, where they can be compared [White, 2016]. In Section 5.2.1 we explain how we identify user intents to further represent all entities over the same vectorial space of intents and we formalize our model in Section 5.2.2.

5.2.1 Identifying User Intents

The need of a single vectorial space for the comparison of different types of entities is a problem that comes out with the development of Composite Item (CI) techniques, that introduce more and more new constraints and heterogeneous items [Bota et al., 2014]. For these reasons, literature studies, in general, use topics discovered using LDA (Latent Dirichlet Allocation) [Amer-Yahia et al., 2016, Alsayasneh et al., 2018], categories defined from item types [Leroy et al., 2015] or external sources [Bennett et al., 2012].

In this work, we propose to use the discovered user intents as an elegant way to build this vectorial space in order to represent uniformly all our the entities. Noticeably, it is treated as a specific use case of the user intents framework described in Section 3.2 of Chapter 3. Figure 5.1 recaps the main steps of this process: (1) identifying a set of descriptive features for user observations, (2) selecting the most discriminant subset of features, (3) learning a metric in this feature space, and (4) applying a relational clustering algorithm to build user intents.

![Figure 5.1: User intent discovery framework as presented in Chapter 3](image)

We apply the same method of learning user intents over our logs, that consist on queries of BI reports. The choice of features is strongly related with the use case, so we need to learn new features specific to our data. We reuse the same formal definitions of query and observation as in Chapter 3. In this use case we explore a platform that permits to create queries dragging-and-dropping query parts, instead of writing queries in natural language. Figure 5.2 shows an example of a BI report.

A slight difference to the feature-based metric we presented in Chapter 3 is the choice of clustering algorithm. As we will use the discovered clusters, that represent the user intents, to projects all entities into a common vectorial space, we need a clustering algorithm that could give their weighted participation to each user intent. This could be naturally achieved by a fuzzy clustering algorithm that produces a membership matrix indicating for each data instance (here a query) its membership, i.e. a normalized score of importance to each cluster (here the user intents). In the end, we will focus on Fuzzy C-Medoids-like approaches [Krishnapuram et al., 1999, Krishnapuram et al., 2001, Nasraoui et al., 2002], to find the user intents and an appropriate projection of the queries in this space.

---

1We apologize for the blurry figure, necessary to protect the enterprise data
5.2. USER INTENTS SPACE TO COMPARE ENTITIES

Figure 5.2: An example of a BI report\(^1\), composed of the visualized results of all the queries the users asked for her analysis.

5.2.2 Data model

This section is dedicated to the modeling of the data learned to resolve the bundle recommendation problem in the context of BI. We start with a preliminary discussion about how user intents can help in representing all the entities involved in the creation of bundles in a formal point of view. Then, we define all the main entities as the queries, the report, the bundle and the user.

**User intents** represent a semantic or topic intermediate layer that allows to abstract user exploration objectives and concepts, entailed within the different queries, reports or, as an extension, a bundle and a user of our BI system. This notion is similar to what is proposed by [Alsayasneh et al., 2018], where an LDA algorithm is used to define a space of high conceptual level topics to seamlessly represent workers or tasks. Following the method presented in Chapter 3 for discovering the user intents in BI, we will explain how we learn user intents for our query bundle problem. In this chapter, we consider that there are \(N > 1\) user intents, and we define an intents space \(I\) as a tuple of user intents \(I = \langle I_1, \ldots, I_N \rangle\).

Let \(e\) be an *entity* like a query, a report, a user or a bundle of queries. An *entity profile* \(I^e = \langle I^e_1, \ldots, I^e_N \rangle\) is a vector of weights \(I^e_j \in [0, 1], \forall j \in [1, N]\), and such that \(\sum_{j=1}^{N} I^e_j = 1\), defined over the set of intents. \(I^e = \langle I^e_1, \ldots, I^e_N \rangle\) represents the relative importance of each intent \(I_j, \forall j \in [1, N]\) for a given entity \(e\).

A multidimensional query \(q \in Q\) is a set of *query parts*, where \(Q\) is the set of formal queries that can be expressed over a database schema \(D\), as defined in Chapter 3. Similarly, a query part is either a level of a hierarchy used for grouping, a measure, or a Boolean predicate involving an attribute of the database. Aligned with the definition of entity profile, a query profile is represented by a vector \(I^q = \langle I^q_1, \ldots, I^q_N \rangle\) of importance of
5.2. USER INTENTS SPACE TO COMPARE ENTITIES

each intent $I_j$, $\forall j \in [1, N]$.

Users and reports can be represented by the set of queries that, respectively, they have viewed or they contain. The aggregation of membership vectors of their queries in user intents space defines, similarly, the user and report vectors of intents as follows.

A report of length $n_r$ is a sequence of $n_r$ queries $r = <q_1, \ldots, q_{n_r}>$ that represents the results of previous analysis on the database. A report is stored in a specific folder $\phi \in \Phi$ and relates to several data sources, as its queries can originate from different sources. The folder $\phi$ related to a query is inherited from the report that contains it. The report represents the actual analysis context of the user and thus a short-term exploration.

Let $Q^r$ be the set of all the queries contained in a report $r$. Aligned with entity profiles, the report profile $I^r = (I^r_1, \ldots, I^r_N)$ is defined by a weighted vector of intent’s participation, calculated as an average of the intents scores of all queries contributing to the report as follows:

$$I^r = \frac{1}{|Q^r|} \sum_{q \in Q^r} I^q$$ (5.1)

A user $u$ is defined as a sequence of $n_u$ queries $u = <q_1, \ldots, q_{n_u}>$ that denotes the past history of the user with the BI system. Note that this is a fine granularity definition of a user, but we could have chosen instead to represent a user at a higher level of the reports she visited. These two definitions are equivalent in our model as it is always possible, knowing the queries that compose each report and the reports visited by users, to either model a user as a sequence of queries or as a sequence of reports.

The user profile is defined over the queries the user viewed or manipulated in the past. This represents a long-term activity profile. It is computed as a result of the aggregation of intents’ representations of the queries that user visited, weighted by their respective frequency as follows:

$$I^u = \sum_{q \in Q^u} I^q \times F^u(q)$$ (5.2)

where $F^u(q)$ is the frequency of each query $q$ visited by user $u$.

A bundle $b \in B$, where $B$ is the set of all possible bundles, is, in our proposal, a sequence of $n_b$ queries $b = <q_1, \ldots, q_{n_b}>$, that respects some hard and soft constraints. When a bundle fulfills all the hard constraints that apply during its definition, it is said to be valid.

Let $Q^b$ be the set of all the queries contained in a bundle $b$. Similarly to a report, the intent profile of a bundle can be denoted by $I^b = (I^b_1, \ldots, I^b_N)$ and defined as follows:

$$I^b = \frac{1}{|Q^b|} \sum_{q \in Q^b} I^q$$ (5.3)
5.3 Problem statement

We mentioned in the previous sections the validity and the quality of a bundle, that depend on some hard and soft constraints. This concepts will be described in Section 5.3.1. We define the problem of bundle recommendation in Section 5.3.2. Section 5.3.3 formalizes and gives more details about the implementation of soft constraints, and we integrate all these constraints in an objective function presented in Section 5.3.4.

5.3.1 Hard and soft constraints identification

Bundle creation is controlled by hard constraints, ensuring the validity of bundle, and soft constraints, that maximize some measures of quality, expressed as a multi-objective function. The definition of these constraints may vary, depending on the use case or the user need. A hard constraint can be decided by a budget [Roy et al., 2010], by a maximum distance between items that compose the bundle [Benouaret et Lenne, 2016], or by required qualification for users to complete tasks of the proposed bundle in [Alsayasneh et al., 2018]. In our case, the validity of bundles depends on the size of a bundle, our only hard constraint, and it is defined as follows:

Definition 4 Validity. Given a set of queries visible to the user and the size $B$ of the number of queries we can use to build a bundle $B$, we define a valid bundle as a set of queries such that $|B|=B$.

A valid and optimal bundle for us would be a group with the required size of queries, which respect some properties, like being at the same time uniform and diverse over different axis, respecting both the user short and long term intent, and ordered to minimize the cognitive gap between the existing queries of the report and the bundle recommended queries, contrary to the classical ranking based on the relevance to the user or to the context profile. All these criteria define soft constraints and ensure the bundle complementarity. These specifications motivated our choice of the following constraints:

1. **Uniformity** ensures that the queries of a bundle are close to each other. To do so, we encourage that all queries added to a bundle are close in terms of user intents and topics covered and will relate to the same aspect of the BI question at hand.

2. **Personalization** guarantees that the queries of a bundle are as close as possible to the user profile. This ensures that added bundle will be tailored for the specific interest of a particular user.

3. **Diversity** allows to retrieve diverse results based on the prior definition of some vertical feature. This vertical feature forces the choice of queries in the bundle, which should have different values for this feature.

4. **Relevance** guarantees that each query composing the bundle shares the same intents with the report it will be added in. It will avoid the creation of a uniform bundle that does not match the current context of the analyst short-term exploration.
5. **Ranking** will order the queries in such a way that it minimizes the effort from the analyst to acquire the knowledge needed between the report and the new added queries. Other said, the bundle queries should correspond to a logical and seamless continuation of the current report.

### 5.3.2 Bundle building as a constrained clustering problem

We can formulate the problem of building bundles as an optimization problem, that firstly aims at finding representative summaries of queries and secondly selecting the group of queries respecting several constraints, to ensure the complementarity and cohesion of this group. It can be seen as a problem of constrained clustering with several penalty terms, as in [Alsayasneh et al., 2018, Amer-Yahia et al., 2016]. Indeed, if each bundle is built around a cluster, then clustering queries assures the **representativeness** of the input, meaning that all input queries are assigned to a cluster and may be attached to a bundle in the end. Moreover, in the case of traditional algorithms like k-means that are defined by an explicit objective function, it is possible to modify the definition of this function to take into account all the previous soft constraints as penalty terms.

**Choice of the clustering algorithm.** We detail hereafter the properties that guide our selection of the best algorithm for the task of bundle creation. First, we need a clustering algorithm with a simple objective function that could be easily tailored to our needs. In this respect the k-means family of algorithms is certainly a good candidate as numerous work have extended its objective function. Second, if we consider a bundle as a subset of a cluster, encouraging cluster compacity will likely improve the bundle uniformity constraint. For these two reasons, we have focused on k-means-like approaches, where clusters are built around a representative center and aims at minimizing intra-cluster distance and thus compacity. Then, when building a set of bundles to recommend, we want that a very interesting query could be attached to different bundles proposed to the user. To that aim, we need a clustering algorithm that allows a data instance to belong to several clusters. For this reason, we propose to use a fuzzy c-means like algorithm [Bezdek, 1981]. Finally, contrary to the user intent creation algorithm used previously where the input data is relational, here, we are in a vectorial space where an adapted fuzzy c-means can be applied as in [Alsayasneh et al., 2018].

**Expression of representativeness via a fuzzy clustering algorithm.** Given the set of \( n_q > 1 \) queries \( Q \), a fuzzy clustering algorithm returns \( K \) centroids and produces a membership matrix \( M = \{ \mu_{ij} \}_{i \in [1,n_q], j \in [1,K]} \) and each \( \mu_{i,j} \in [0,1.0] \) represents the degree to which a query \( i \) belongs to the cluster \( j \). The fuzzy clustering guarantees the representativeness of the whole query input as each query is assigned to all clusters. It is defined as follows:

\[
\text{representativeness}(C, M) = \sum_{i=1}^{n_q} \sum_{j=1}^{K} \mu_{iq} \cdot \text{dist}(I^q, c_j) \tag{5.4}
\]
5.3. PROBLEM STATEMENT

\[ s.t. \forall i \in [1, n_q], \sum_{j=1}^{K} \mu_{ij} = 1 \]

where \( C \) is the set of \( K \) centroids, \( M \) is the membership matrix, \( q_i \) a query, \( c_j \) a centroid, \( m > 1 \) is the fuzzy exponent and \( \text{dist()} \) is the function that measures the distances between membership vectors of entities. Our goal is to minimize the score of this function, thus to minimize the distances between queries and centroids, and to increase the input representativeness. The larger the value of \( m \), the fuzzier the clustering. At the beginning the centers are initialized randomly and they are updated until they converge. In our case, we consider the traditional distance measure used in fuzzy c-means clustering, i.e. the squared Euclidean norm.

Finally, it is important to notice, that, by construction, each cluster centroid \( c_j, \forall j \in [1, K] \), has coordinates defined in the user intent space as the other entities described in Section 5.2.2.

5.3.3 Constraints formalization as clustering penalty terms

We detail hereafter how the soft constraints can be formalized as penalty terms of the objective function of Equation 5.4. Uniformity, Personalization and Diversity constraints are extensions of existing works, while Relevance and Ranking are particular for our use case and firstly introduced in this work. These constraints impact the general clustering process to build a set of qualitative bundles. As in [Alsayasneh et al., 2018], a greedy procedure is used in a second step, locally to each cluster to define the best possible bundle inside each cluster.

Notations In the following paragraphs we consider any cluster \( C_k, \forall k \in [1, K] \) whose center is denoted \( c_k \). The bundle \( B_k \) is a sequence of queries constructed over a set of queries \( Q^{B_k} \) whose membership to cluster \( C_k \) is greater than 0, \( Q^{B_k} \subseteq \{ q_i | \mu_{ik} > 0 \} \).

(1) Uniformity is measured as the average distance of queries \( q \in Q^{B_k} \) to the centroid of their representative cluster \( c_k \). This distance is to be minimized.

\[ \text{uniformity}(B_k) = \frac{1}{|Q^{B_k}|} \sum_{q \in Q^{B_k}} \text{dist}(I^q, c_k) \quad (5.5) \]

(2) Personalization ensures the proximity of the constructed bundle to the user long-term profile by minimizing the average distance of queries \( q \in Q^{B_k} \) to the user profile \( u \).

\[ \text{personalization}(B_k) = \frac{1}{|Q^{B_k}|} \sum_{q \in Q^{B_k}} \text{dist}(I^q, I^u) \]
5.3. PROBLEM STATEMENT

(3) **Diversity** computes to which extent the distribution of values on the vertical feature is uniform. Indeed, a uniform distribution among possible modalities guarantees that each of them is equally represented and that all possible modalities are represented in the bundle. To do this, we classically use an entropy measure denoted $H$ on the distribution of values for the vertical feature of the queries of the actual report augmented by the queries of the bundle. In what follows, we denote by $div()$ the function that measures the diversity between bundle’s queries. As our objective function deals with a minimization problem and that we want to maximize the diversity, we propose to express this constraint as follows:

$$diversity(B_k) = 1 - div(Q^{B_k} \cup_B Q^r)$$

where $r$ is the report to complete and $Q^r$ is the set of its queries.

In our case, we try to uniformize the distribution of the visualization types related to queries. We denote by $viz()$ the function that inputs a set of visualized queries and returns the histogram of frequencies for each type of visualization and we define the diversity function:

$$diversity(B_k) = 1 - H(viz(Q^{B_k} \cup_B Q^r))$$

where $r$ is the report to complete and $Q^r$ is the set of its queries.

To contextualize the bundle of queries to the current report the user is editing, and to order the queries we present two new constraints: the relevance to context and the ranking of the queries of the bundle.

(4) **Relevance** to context ensures that the constructed bundle is close to the actual report, that represents the context of the exploration, in terms of user intents. The relevance is expressed as the average distance between queries $q \in Q^{B_k}$ to the profile of report $r$.

$$relevance(B_k) = \frac{1}{|Q^{B_k}|} \sum_{q \in Q^{B_k}} dist(I^q, I^r)$$

(5) **Ranking** evaluates to which extent it is possible to order queries of bundle $B_k$ so that the effort of the user is minimized to acquire the new information carried by the added queries. To do so, an optimal solution would be to consider a Traveling Salesman Problem each time a candidate bundle is to be evaluated. This is not possible because of the complexity of such ordering approach, so instead we favor a pairwise greedy comparison taking into account the distance of each added query to the previous one in the sequence of queries exploration.

$$ranking(B_k) = \sum_{i \in [1, B]} dist(I^{q_{i-1}}, I^{q_i})$$

where $B$ is the size of the bundle and $q^0$ is the last query of report $r$. 
5.3. PROBLEM STATEMENT

5.3.4 Objective function

From Sections 5.3.2 and 5.3.3 it is possible to define the complete objective function of the constrained clustering algorithm to build the bundles of visualized queries. This function, denoted $J$, depends on the centroids of clusters $C$, the membership matrix $M$ to clusters and the constructed bundles $B$ as well as the long-term intents of user $u$ and the short-term user interest as represented by the actual report $r$.

$$\min J(C, M, B)_{u,r} = \alpha \text{ representativeness}(C, M) + \frac{1}{K} \sum_{k=1}^{K} \min_{B_k \in \nu(C_k)} \left( \beta \text{ uniformity}(B_k) + \gamma \text{ personalization}(B_k) + \delta \text{ diversity}(B_k) + \rho \text{ relevance}(B_k) + \omega \text{ ranking}(B_k) \right)$$

where $\nu(C_k)$ is the set of the valid bundles of size $\mathfrak{B}$, that can be constructed from a set of queries $Q$ and $\alpha, \beta, \gamma, \delta, \rho$ and $\omega$ are hyperparameters used to balance the relative importance of each criterion in the multi-criteria optimization problem.

Replacing each constraint with its implementation in our use case, the objective function is defined as below:

$$\min J(C, M, B) = \frac{\alpha}{n_q} \sum_{i=1}^{n_q} \sum_{k=1}^{K} \mu_{ik} \text{ dist}(I^q_i, c_k)(6) + \frac{1}{K} \sum_{k=1}^{K} \min_{B_k \in \nu(C_k)} \left( \beta \sum_{q \in Q^{B_k}} \text{ dist}(I^q, c_k)(1) + \gamma \sum_{q \in Q^{B_k}} \text{ dist}(I^q, I^u)(2) + \delta \left(1 - \text{ div}(Q^{B_k} \cup Q^r)\right)(3) + \rho \sum_{q \in Q^{B_k}} \text{ dist}(I^q, I^r)(4) + \omega \sum_{i \in [1, B_k]} \text{ dist}(I^{q_i-1}, I^{q_i})(5) \right)$$

subject to $\forall i \in [1, n_q], \sum_{j=1}^{K} \mu_{ij} = 1$

where numbers in bold refers to the constraint penalty terms detailed in Section 5.3.3.
5.4 Bundle building algorithm

We focus in this section on the details related to the bundle construction algorithm. As explained previously, the function and the soft constraints that we want to implement correspond to the expression of a modified fuzzy c-means enriched with:

- adapted penalty terms as presented in parts (1) to (5) of Equation 5.7
- and a new internal step that aims at building, locally to each cluster, the best subset of queries as a bundle.

This section starts by explaining how the traditional steps of membership update and cluster centers update in fuzzy c-means are impacted by the introduction of penalty terms in Section 5.4.1. Secondly, in Section 5.4.2 we propose a greedy heuristic for bundle construction inside each cluster, similarly to [Alsayasneh et al., 2018]. Finally, Section 5.4.3 details how this new constrained fuzzy c-means algorithm is used to produce candidate bundles based on an exploration of hyper-parameters $\alpha$, $\beta$, $\gamma$, $\delta$, $\rho$ and $\omega$.

5.4.1 Membership and center update

5.4.1.1 Update of membership matrix

The membership matrix is updated after each new variation of centers. Inspired from [Leroy et al., 2015], it is derived from the objective function of Equation 5.7, which is set here with the squared Euclidean distance between queries and clusters’ centers. It is important to notice that the minimization of Equation 5.7 is performed under the constraint that $\forall i \in [1, n_q], \sum_{k=1}^{K} \mu_{ik} = 1$. To do so, a classical technique in convex optimization under constraint is to use a Lagrange multiplier as follows:

$$L(C, M, B) = J(C, M, B) + \sum_{i=1}^{n_q} \lambda_i \left( 1 - \sum_{k=1}^{K} \mu_{ik} \right)$$

where $\lambda_i$ denotes the Lagrange multiplier for query $q_i$ and $n_q$ is the size of $Q$. The value of $\mu_{ij}$ that minimizes the Lagrange objective function $L(C, M, B)$ is obtained by determining the point $\mu_{ij}$ as follows:

$$\frac{\partial L(C, M, B)}{\partial \mu_{ij}} = 0$$

As none of the new terms introduced in our objective function in Equation 5.7 depends on $\mu_{ij}$ the update equation of membership at time $(t+1)$ is similar to the update used in the literature [Bezdek, 1981] and defined as follows:

$$\mu_{ij}^{(t+1)} = \left( \sum_{k=1}^{K} \left( \frac{dist(I^k, c_j^{(t)})}{dist(I^k, c_k^{(t)})} \right)^{\frac{1}{m-1}} \right)^{-1}$$

93
5.4. BUNDLE BUILDING ALGORITHM

5.4.1.2 Update of centroids

Contrary to the membership update, one of the added penalty term, the uniformity term presented in Equation 5.5 depends on the cluster center coordinates. This case is similar to the one described in [Alsayasneh et al., 2018] from which it is possible to adapt our equation for center update as follows:

\[ c_j^{(t+1)} = \frac{\alpha}{n_q} \sum_{i=1}^{n_q} (\mu_{ij}^{(t+1)})^m I^{q_i} + \frac{\beta}{|B_j|} \sum_{q \in Q^{B_j}} I^{q} \]

(5.11)

where \( Q^{B_j} \) denotes the set of all queries contained in the bundle \( B_j \) constructed inside the cluster \( C_j \) and \( |B_j| \) is the size of bundle \( B_j \).

5.4.2 Greedy heuristic for bundle construction

We build bundles of queries in all clusters in the vicinity of their centroids, at each iteration of the fuzzy clustering. Algorithm 1 presents the greedy procedure that iteratively adds a limited number of queries \( B \) to a bundle \( B_k \), adjusting the penalties related to soft constraints to minimize the score of the objective function, and producing this way valid bundles with the desirable number of queries and with a good quality.

The procedure described in Algorithm 1 creates a bundle inside a cluster \( C_k \in C \) generated by our bundle recommender system, named B1bR, and close to its center \( c_k \). It considers the user interests profile in the user intents space \( I^u \) and the short-term intents \( I^r \) as represented by the actual report. The procedure starts with an empty bundle \( B_k \) (line 1) and, at each of the \( B \) steps of the bundle construction, chooses a query \( q \) that belongs to cluster \( C_k \) and that is not already part of the bundle \( B_k \) that minimizes the \( \text{score}(q) \) (line 3) defined as follows:

\[ \forall q \in \text{Candidate}(k), \text{score}(q) = \left( \beta \sum_{q \in Q^{B_k}} \text{dist}(I^q, c_k) + \rho \sum_{q \in Q^{B_k}} \text{dist}(I^q, I^r) + \gamma \sum_{q \in Q^{B_k}} \text{dist}(I^q, I^u) + \delta(1 - H(viz(Q^{B_k} \cup B^r))) + \omega \sum_{i \in [1, B]} \text{dist}(I^{Q_{B_k}^i}, I^{Q_{B_k}^i}) \right) \]

(5.12)

where \( \text{Candidate}(k) \) is the set of candidate queries from cluster \( C_k \) that might be integrated in bundle \( B_k \):

\[ \text{Candidate}(k) = \{ q_i, \forall i \in [1, n_q] | \mu_{ik} > 0 \land q \notin B_k \} \]

(5.13)
5.4. BUNDLE BUILDING ALGORITHM

As the penalty terms of personalization, relevance, uniformity and ranking calculate the distance between the queries and other respective entities, the purpose of the objective function is to minimize these distances to provide complementary bundles. On the contrary, as the diversity is measured by the entropy $H$ of query visualization, the objective function aims at minimizing $(1-H)$. Differently from the approaches of the state of the art [Amer-Yahia et al., 2016, Alsayasneh et al., 2018, Leroy et al., 2015], that take into account only the items added in an empty bundle, we consider the existing queries of the current report as the beginning of our bundle and a short-term context, while calculating relevance, diversity and ranking.

At each step of the $B$ cycles, the queries entered in the bundle are excluded from the set of candidate queries $\text{Candidate}(k)$ for the future iterations (line 5 of Algorithm 1). The complexity of this algorithm in the worst case is $O(B \ast n_q)$ for each cluster $C_k$.

Algorithm 1: Algorithm for building a bundle inside a cluster

<table>
<thead>
<tr>
<th>Data: Cluster $C_k$, membership matrix $M$, user $I^u$, report $I^r$, size $\mathfrak{B}$, weights $(\alpha, \beta, \gamma, \delta, \omega, \rho)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result: Bundle $B_k$</td>
</tr>
<tr>
<td>1 $B_k \leftarrow \emptyset$</td>
</tr>
<tr>
<td>2 for $\mathfrak{B}$ cycles do</td>
</tr>
<tr>
<td>3  $\text{bestQ} \leftarrow \text{argmin}_{q \in \text{Candidate}(k)} \text{score}(q)$ // query that minimizes Equation 5.12;</td>
</tr>
<tr>
<td>4  $B_k \leftarrow B_k \cup \text{bestQ}$ // add this query to the bundle $B_k$;</td>
</tr>
<tr>
<td>5  $\text{Candidate}(k) \leftarrow \text{Candidate}(k) \setminus \text{bestQ}$ // update the set $\text{Candidate}(k)$;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>6 return $B_k$</td>
</tr>
</tbody>
</table>

5.4.3 Main algorithm

Now that each step of the B1bR algorithm have been detailed, we can express how this algorithm is used in a global iteration that aims at exploring different values for hyper-parameters $\alpha, \beta, \gamma, \delta, \omega$ and $\rho$.

As in [Alsayasneh et al., 2018], the hyper-parameters $(\alpha, \beta, \gamma, \delta, \omega, \rho)$ are adjusted to their target value during $\eta$ cycles. That allows to gradually take into account the constraints, starting with a traditional fuzzy c-means algorithm when $\alpha = 1$ to the desired experimental setting after the $\eta$ cycles (line 5-6). The constrained fuzzy c-means is then run taking into account the penalty terms related to the bundles as shown by the Equation 5.10. It is run several times, updating the membership matrix (line 9) and the centers in Equation 5.11 (line 11), until the convergence of centers. We admit that centers have converged when the difference of the membership vectors of two following set of centers is inferior to a threshold, or if a maximum number of iterations is reached. The choice of centroids (line 3) guarantees the representativeness of the data input, which is one of our quality constraints and a hard problem to be resolved. Bundles are constructed around
5.4. BUNDLE BUILDING ALGORITHM

Algorithm 2: Algorithm BIbR to define a set of eligible bundles as in [Alsayasneh et al., 2018].

Data: set of queries $Q$, user $I^u$, report $I^r$, size $B$, weights $(\alpha, \beta, \gamma, \delta, \omega, \rho)$, number of iterations $\eta$

Result: Set of bundles $B$

1. $B \leftarrow \emptyset$
2. $\alpha' \leftarrow \alpha + \beta + \gamma + \delta + \omega + \rho$; $\beta' \leftarrow 0$; $\gamma' \leftarrow 0$; $\delta' \leftarrow 0$; $\omega' \leftarrow 0$; $\rho' \leftarrow 0$
3. Initialize $C$ centers with a traditional Fuzzy C Means algorithm;
4. for $\eta$ cycles do
5. 
6. 
7. end
8. repeat
9. Update cluster membership weights according to Equation 5.10;
10. Recompute bundles set $B$ by generating $B_k$ for each cluster $C_k \in C$ with Algorithm 1;
11. Update cluster centers according to Equation 5.11;
12. until convergence of $C$;
13. return $B$

Each transitional centroid.

Algorithm 1 performs local search in each cluster to find the best queries that, together, decrease the most the score of the objective function (line 10). Bundles generated at each iteration are used in the Equation 5.11 to update centers and once that the centers converge, we choose the best among the bundles created in the last step, build around converged centers. It corresponds to the bundle with the smallest score of objective function.

BIbR algorithm complexity. Trying all combinations of all possible subsets of 5 queries, in order to find the one that optimizes our constraints, consists on a NP-hard problem. The time complexity of $FCM$ only is $O(FCM) = O(n_q N K^2)$, where $n_q$ is the total number of queries, $N$ is the number of discovered intents and $K$ is the number of clusters. By analyzing algorithms 1 and 2, in each iteration of $FCM$ we add the complexity of the bundle creation at each cluster. Knowing that the time complexity of our greedy algorithm is $O(Greedy) = O(n_q N B)$, and the total complexity is $O(n_q \eta (O(FCM) + O(Greedy))) = O(n_q \eta (n_q N K^2 + n_q N K B)) = O(\eta n_q N K (K + B))$, where $\eta$ is the number of cycles, $i$ is the number of iterations before centers convergence and $B$ is the size of a bundle.
5.5 Testing our bundle building approach

The process of building bundles includes two main phases: (1) projecting entities into a single space of intents and (2) grouping queries around centroids, respecting the quality constraints. The large number of parameters of our algorithms requires an extended number of experiments to tune and to evaluate each step, and to measure the quality of produced bundles and the user satisfaction. Before detailing our experiments, Section 5.5.1 introduces the data and the challenges we faced working with real usage logs. Section 5.5.2 is dedicated to the evaluation of a Linear Fuzzy C-Medoid Clustering (LFCMD), that adapts the feature-based metric described in Chapter 3 to discover user intents. Finally, we present a rich experimental protocol of the evaluation of our BIbR algorithm, that generates bundles of queries, in Section 5.5.3. We search to optimize the hyper-parameters of our algorithms and we vary the objective function with different mono and multi constraints to observe how the quality of the final bundle recommended depends on the choice and the combination of constraints weights.

5.5.1 Dataset

Our dataset consists of users’ past events of creation or modification of documents (i.e., sets of reports) in a BI Platform. These logs include details about the event time and type (Create, Edit, Modify, View), the user, the document ID and its folder ID. The dataset is enriched with the documents and reports titles, as well as their corresponding queries. We name this dataset \( D \) to distinguish it from its subsets we will use to run different evaluation tests. We explore logged activities of 328 users over 1854 documents, extracted from an Audit instance of internal SAP users, who create and work with dynamic documents to perform their everyday tasks in this platform.

**Data pre-processing.** Working with real data may reveal problems related with their incompleteness and bad quality. The Audit log system collects limited, high level information about the documents the user accessed, not specifying the reports or queries manipulated, in order to understand the real intention behind each event. Usage distribution of documents is not uniform. We could easily identify documents highly used, and a small group of users more active than the others. As we use a training instance that logs user testing activity, we identify several documents that contain non-interesting queries, related to platform maintenance or development debugging information, highly used by some users, which add noise in our data. This complicates the identification of real users, who create reports for their everyday jobs instead of only using these documents to test platform functionalities. For this reason, a data cleaning phase was added to improve the quality of the data. This includes data anonymization, uniqueness of any element identification (document, report, query) and filtering of high quality documents, based on usage statistics.

A group of 4 product experts have labeled 437 pairs of 220 distinct reports they have manipulated in past interactions, indicating whether they represent the same intent or not. We selected the most pertinent documents for these users, which correspond to the most
5.5. TESTING OUR BUNDLE BUILDING APPROACH

regularly used documents for each of them during 2014-2017. We define this subset as $D_1$, $D_1 \subset D$, and we use it to learn the user intents, as in Chapter 3.

We test the clusters provided by the adapted feature-based metric for discovering the user intents over the cleaned dataset $D_2 \subset D$, that consists of 126 selected active users and 109 documents with multiple reports (541), that have been created and manipulated during 2016-2017. This selection was due to the bad quality of the ancient documents and the abnormal, very low or high, usage. The metric learned over $D_1$ is tested over dataset $D_2$, $D_1 \cap D_2 = \emptyset$.

**User intents ground truth.** Let $d_1$, $d_2$ and $d_3$ be three documents and $\text{label}$ be the function that gives the user annotation for a pair of documents. If $\text{label}(d_1, d_2) = 1$ and $\text{label}(d_2, d_3) = 1$, then by transitivity $\text{label}(d_1, d_3) = 1$. If we consider a graph whose nodes are the documents and edges exist when the label function between documents returns 1, then the transitive closure of this graph corresponds to clusters of documents. Each of these clusters represents one user intent and together make our ground truth. In our case, we obtained 18 clusters, each representing a user intent. As there are some clusters with very few documents defining a user intent, we can reduce the initial number from 18 to 10 intents, fusing manually the small clusters to their most similar clusters. 10 discovered intents over the $D_1$ dataset seems more reasonable for our business experts.

As these clusters contain groups of documents, but on the other hand, LFCMD clusters queries to discover user intents from logs, we extend those groups of documents of the ground truth into groups of queries they contain, to be comparable with the clusters generated by LFCMD.

Finally, we name $D_3$ a subset of reports from $D_2$, $D_3 \subset D_2$, that contain more than 5 queries, necessary for the offline measurements of the quality of our bundles. This dataset is used in several evaluations: the accuracy and the robustness of $BIbR$, its comparison with the state of the art approaches, and in online evaluation. The combination of 120 reports of $D_3$ dataset documents and 46 users of $D_2$ that have viewed these reports gave 194 report-user pairs to test. All these 194 combinations were used for the offline evaluation of recommender. As for the real user test, we selected a group of 9 users and we generated bundles only for reports of $D_3$ they had edit rights. We got feedback only for 17 reports.

All the relations between these datasets and how they are used in our testing protocol are presented in Table 5.1.

5.5.2 Assessing the User Intents quality

In this chapter, as we work on a different BI platform and with a very different functional architecture, we need features dedicated to this use case. The choice of features we use in the metric definition, has an important impact on the quality of the obtained user intents. This is one of the lessons we learned from Chapter 3. LFCMD implements an adaption of the feature-based metric, that here tries to learn user intents over $D_1$. 98
5.5. TESTING OUR BUNDLE BUILDING APPROACH

Datasets | # Documents | Explanation
--- | --- | ---
$D_1$ | 220 reports (4 users) | SAP experts annotated 437 pairs of documents from this set and use the `label()` function to produce clusters of documents that make our ground truth
$D_2$ | 541 reports (126 active users) | SAP experts cleaned up version of $D$ (2016-2017) to test user intents
$D_3 \subset D_2$ | 120 reports (46 users) 17 reports (9 users) | Subset of documents containing at least 5 queries to test:
• recommender systems with offline quality measures
• recommender system with online evaluation

Table 5.1: An overview of the datasets that are used to learn user intents and to test our recommender system.

Firstly, we introduce a new group of discriminant features in Section 5.5.2.1 and then we run two tests to evaluate the user intents:

1. We vary the possible sets of features we implement, from which we choose the best subset, in Section 5.5.2.2,
2. We compare $LFCMD$ paired with different metrics, to find who is the most accurate at learning the user intents, in Section 5.5.2.3.

5.5.2.1 Features identification

The choice of features is specific to data and to the information details logged for the dataset of this particular use case. Table 5.2 presents the set of features we have identified and their associated similarity metrics: first three features are built on the queries’ metadata ($\text{sources}$, $\text{users}$, $\text{folder}$) and the last two ones use topics discovered using LDA [Blei et al., 2003] over the query parts, or report and document titles, inspired from topic identification in [Amer-Yahia et al., 2016].

We want to use query parts to evaluate bundles, as this is close to the perception of users about the report similarity. As for the labeling of the reports that build our ground truth, the users look at the composition parts of the queries: measures, levels and filters, to estimate report similarity. Thus, we can not include them in the list of features we use to learn our model. Anyway, query parts are considered in $LDAqueryParts$ feature, in the identification of LDA topics, so they impact indirectly the discovering of user intents.

As presented in Table 5.2, the first two features measure the fraction of sources and users in common between the queries, and the third feature checks if they are part of documents classified in the same folders. More formally, we note $q$ the query of a report and $\text{users}(q)$ all the users who have used this query. Similarly, we note $\text{folder}(q)$ the folder where the report of the query $q$ is located. As for the $\text{sources}$ of queries and the similarity measures between them, we use the same definition as presented already in Section 3.1.2 of Chapter 3. The last two features, $LDAqueryParts$ and $LDAtitles$, compare weighted vectors of topics identified by LDA technique, representing the participation of queries to each topic.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

<table>
<thead>
<tr>
<th>Nº</th>
<th>Feature</th>
<th>Definition</th>
<th>Interpretation</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sources</td>
<td>$\bigcup_{q \in U} sources(q)$</td>
<td>all the sources</td>
<td>MaxFrac.</td>
</tr>
<tr>
<td>2</td>
<td>users</td>
<td>$\bigcup_{q \in U} users(q)$</td>
<td>all the users</td>
<td>MacFrac.</td>
</tr>
<tr>
<td>3</td>
<td>folder</td>
<td>$\bigcup_{q \in U} folder(q)$</td>
<td>folder of the report of q</td>
<td>NormInter.</td>
</tr>
<tr>
<td>4</td>
<td>LDAqueryParts</td>
<td>$LDA^U_{qP}$</td>
<td>membership to topics</td>
<td>Cosine</td>
</tr>
<tr>
<td>5</td>
<td>LDAtitles</td>
<td>$LDA^U_t$</td>
<td>membership to topics</td>
<td>Cosine</td>
</tr>
</tbody>
</table>

Table 5.2: Features description and their associated metrics

LDA over query parts and report titles define two spaces of topics, named $LDA^U_{qP}$ and $LDA^U_t$. $LDA^U_{qP}$ and $LDA^U_t$ give, respectively, the membership vector of query $q$ to the topics discovered over the query parts and the membership to the topics discovered over titles. Let $U$ be a user intent, we denote by $LDA^U_{qP}$ the aggregation of the membership vectors of all $q \in U$ to the topic space $LDA^U_{qP}$, and similarly, $LDA^U_t$ for topic space $LDA_t$. $LDA^U_{qP}$ is calculated as an average of the membership of each query of $U$ to topics discovered over query parts or titles:

$$LDA^U_{qP} = \frac{1}{|U|} \sum_{q \in U} LDA^q_{qP}$$  \hspace{1cm} (5.14)

Similarly, we calculate $LDA^U_t$ based on the projection of all queries $q \in U$ over topics $LDA_t$, discovered using LDA over titles.

5.5.2.2 Determining the best subset of features

In this section, we search a subset of features that maximizes the quality of the discovery of user intents. We start by combining and comparing 4 groups, from GB1 to GB4, following the same protocol as in Section 3.3.1.3 of Chapter 3, that organize features described in the previous section. GB1=$\{1, 2, 3\}$ contains only the features related directly with query metadata: the sources of query, the folder where the corresponding document is positioned and the user feature, presenting the users and the frequency they have consulted the query, measuring this way the utility of this query for different users.

The features related to LDA topics, $LDAqueryParts$ and $LDAtitles$, are each added to GB1 to produce GB2=GB1$\cup\{4\}$ and GB3=GB1$\cup\{5\}$. GB4=GB1$\cup\{4, 5\}$ includes all these features together. For each of these groups, we learn feature weights using linear SVM. We estimate the accuracy of the discovered intents comparing obtained clusters of queries from $LFCMD$ to the clusters of the ground truth we created in dataset $D_1$, measuring $ARI$ and $NMI$ (Appendix B), and we select the group of features that gives the highest score. These tests are performed for both groups of 10 and 18 intents of the ground truth.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

Lessons learned. We ran a linear SVM for each candidate group GB1-GB4 to learn features’ weights in each combination. Table 5.3 shows the coherence of these features’ importance in each group. As expected, LDAqueryParts and folder are the two decisive features in each subset we test. The first can be explained by the fact that the experts have labeled the documents looking at their content. The business knowledge is hidden behind the query parts that query the databases, correlating this way sets of query parts with database areas. The impact of the second feature can be explained by the hierarchical organization of documents in folders, based on the task they are related to and how the users organize documents, according to their interests and usage. The negative sign of feature LDAqueryParts coefficient means that it tends to classify queries opposite to features of positive coefficients, which illustrates the fact that queries similar in terms of this feature are shared across sources, users, folders and titles.

<table>
<thead>
<tr>
<th>Features</th>
<th>GB1</th>
<th>GB2</th>
<th>GB3</th>
<th>GB4</th>
</tr>
</thead>
<tbody>
<tr>
<td>sources</td>
<td>0.26</td>
<td>0.26</td>
<td>0.24</td>
<td>0.24</td>
</tr>
<tr>
<td>users</td>
<td>0.45</td>
<td>0.45</td>
<td>0.38</td>
<td>0.38</td>
</tr>
<tr>
<td>folder</td>
<td><strong>0.55</strong></td>
<td><strong>0.55</strong></td>
<td><strong>0.49</strong></td>
<td><strong>0.49</strong></td>
</tr>
<tr>
<td>LDAqueryParts</td>
<td>-0.53</td>
<td>-0.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDAtitles</td>
<td>0.25</td>
<td>0.25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Weights of features for groups GB1-GB4, learned on D1 dataset with a linear SVM. The results show that folder and LDAqueryParts are the two most decisive features, independently of the number or the other features that are implemented in LFCMD.

We measure the correctness of produced clusters to the ground truth, through ARI and NMI, identifying this way the most discriminant group in learning user intents. These clusters are not balanced and contain from 2 to hundreds of documents. The results of our tests over 18 and 10 user intents are presented, respectively, in Tables 5.4 and 5.5. Testing over 18 intents shows clearly how the precision of discovered intents depends on the choice of features. GB4 seems to be the most accurate, when measuring both ARI and NMI.

We notice that for 10 clusters, the latter are more homogeneous and Table 5.5 shows no significant difference between the quality of the clusters of intents discovered using different feature groups. From the previous tests in Chapter 3, we have learned that the more features we include, the better we refine the discovered intents. Being the group that includes all the possible features and gives better scores of ARI and NMI for finer clusters, GB4 motivates our choice to use this group of features in the implementation of LFCMD in our future tests. LFCMD algorithm was tuned with a fuzziness parameter $m_f$ of 1.2 and 5 candidate medoids $p$, which remain fixed in all future experiments of LFCMD evaluation.

5.5.2.3 Choice of algorithm for LFCMD

Except the feature-based metric that we implemented to LFCMD in the previous test, we explore two other methods to discover user intents: LDA, used in [Alsayasneh et al., 2018] to discover topics, and the classic Fuzzy C-Means (FCM) that employs the same feature-based metric as LFCMD. For this test, we choose to implement LDA over the
5.5. TESTING OUR BUNDLE BUILDING APPROACH

<table>
<thead>
<tr>
<th>Group</th>
<th>ARI</th>
<th>NMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB1</td>
<td>0.24</td>
<td>0.53</td>
</tr>
<tr>
<td>GB2</td>
<td>0.24</td>
<td>0.57</td>
</tr>
<tr>
<td>GB3</td>
<td>0.18</td>
<td>0.40</td>
</tr>
<tr>
<td>GB4</td>
<td>0.31</td>
<td>0.58</td>
</tr>
</tbody>
</table>

Table 5.4: Comparison of different feature groups to discover 18 intents. Finer intents show better the impact of the choice of features in the quality of the discovered user intents. Here it is clearly visible the advantage of G4 to the other groups.

<table>
<thead>
<tr>
<th>Group</th>
<th>ARI</th>
<th>NMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB1</td>
<td>0.24</td>
<td>0.54</td>
</tr>
<tr>
<td>GB2</td>
<td>0.23</td>
<td>0.53</td>
</tr>
<tr>
<td>GB3</td>
<td>0.23</td>
<td>0.53</td>
</tr>
<tr>
<td>GB4</td>
<td>0.25</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Table 5.5: Comparison of different feature groups to discover 10 intents. The results are very similar, as the decreasing number of clusters yield more homogeneous intents, but we can identify G4 as the group of features slightly more accurate than the others.

query parts instead of LDA over titles, based on the results of Table 5.3, which shows that LDAqueryParts is a more discriminant feature than LDAtitles. We compare observations calculating the distances between their representative vectors of membership to LDA topics. The second algorithm we use to cluster logs, the classic Fuzzy C-Means, similarly to LFCMD, considers the feature-based metric. This way, we can compare two clustering algorithms that use the same metric.

We run LFCMD, FCM and LDA to discover either 18 or 10 user intents and we measure the accuracy of the produced clusters through ARI and NMI evaluation metrics (Appendix B), comparing to the corresponding ground truth of 18 or 10 groups of documents’ queries. This permits to choose the most precise technique in learning user intents, to then implement it to BIbR and to finally test the bundle recommending system.

**Lessons learned.** The results of Tables 5.6 and 5.7 show that LFCMD is the technique that provides the closest partitions of logs to the ground truth, when we search either 10 or 18 intents. ARI and NMI evaluate hard clusters, so we have to assign each query to only one intent. From the representative vector of each query to the intents space, we choose the intent corresponding to the highest score. Discovering either 10 or 18 user intents, we notice that LFCMD defines better clusters than the other approaches, clearly visible when searching finer intents, in Table 5.7. Comparing to FCM, LFCMD shows the advantage of applying a fuzzy k-medoid instead of fuzzy c-means when working with discriminant features. This difference is more significant when testing over 10 user intents, where we risk to have more overlapping clusters. Computing the median, seems to achieve a better separation of clusters, and using medoids seems to be a better estimator of the user intents, rather than a fuzzy k-means that finds intents calculating the mean value of
5.5. TESTING OUR BUNDLE BUILDING APPROACH

each query to the candidate clusters.

<table>
<thead>
<tr>
<th>Measures</th>
<th>LDA</th>
<th>FCM</th>
<th>LFCMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARI</td>
<td>0.14</td>
<td>-0.04</td>
<td><strong>0.18</strong></td>
</tr>
<tr>
<td>NMI</td>
<td>0.29</td>
<td>0.21</td>
<td><strong>0.45</strong></td>
</tr>
</tbody>
</table>

Table 5.6: Comparison of algorithms for the identification of 10 intents.

<table>
<thead>
<tr>
<th>Measures</th>
<th>LDA</th>
<th>FCM</th>
<th>LFCMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARI</td>
<td>0.17</td>
<td>0.12</td>
<td><strong>0.25</strong></td>
</tr>
<tr>
<td>NMI</td>
<td>0.40</td>
<td>0.39</td>
<td><strong>0.54</strong></td>
</tr>
</tbody>
</table>

Table 5.7: Comparison of algorithms for the identification of 18 intents.

5.5.3 Evaluation of $BIbR$

$BIbR$ is a complex algorithm with several parameters to be tuned. Beyond the accuracy of this approach, we want to measure to what extent the recommended bundles are complementary to the report a user is editing and if the user is satisfied. Our evaluation scheme is manifold:

- in Section 5.5.3.1, we first present the protocol by which we measure the accuracy of our bundle building algorithm, that is to say how we compare queries in the recommended bundle with the last queries of tested reports, which represent the expected future,
- in Section 5.5.3.2, we study the robustness of our approach, through the accuracy scores of offline measurements on real business reports,
- in Section 5.5.3.3, we run a user study by submitting bundles to a group of SAP users to understand the importance of parameters from a human standpoint,
- in Section 5.5.3.4, we measure the robustness of our algorithm related to the quality of the learned user profile,
- in Section 5.5.3.5, we study the relationship between user and context constraints and the impact of their integration in $BIbR$,
- in Section 5.5.3.6, we evaluate the importance of ranking queries of the recommended bundle,
- in Section 5.5.3.7, we study the importance of integrating diversity in addition to uniformity between bundle queries,
- in Section 5.5.3.8, we compare our algorithm to two other Composite Item recommenders from the state of the art: BOBO and CAP.

We detail these tests in each of the following subsections.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

5.5.3.1 Offline evaluation of BIbR

The offline evaluation of bundle recommender stands on the comparison between the set of bundle’s queries and the last queries of the report we want to complete. Reports are separated in two parts: the future, composed of the 5 last queries and the seed, containing the remaining queries at the beginning of the report. We run BIbR over the seed (our context) and we try to recommend queries close to the future. As our bundle size is fixed to 5 queries, our experiments are limited to a group of reports of more than 5 queries, so we can identify the seed and the future part. We test over the reports of the documents of dataset D3.

We compare the queries of the future with the queries of the bundle produced to measure the precision of our recommender. As we recommend queries from other existing reports and we know that there are a few repeated queries, we do not expect to recommend exactly the same queries as in future. Instead of a pure Jaccard between future and recommended queries, we apply the metric of [Aligón et al., 2015], that compares the queries on three levels of query parts. Thus, the similarity score between queries of future and of the recommended bundle is considered to be positive if it is superior to a threshold.

We consider two ways to compare these two sets of queries:

[1] Keeping the order of the queries of future and the order of the queries in the suggested bundle, we compare pairs of queries of the same position from each set, and we estimate the average of theirs similarities. This shows the ability of our algorithm to rank recommended queries close to existing reports.

[2] We consider all the possible rankings of bundle’s queries, and for each ranking we compare the queries of the bundle and those of the future as in [1], and at the end we keep the best matching. It corresponds to the bundle query ranking that is the most similar to future ranking. The difference between the best score kept and the score of the actual ranking of the bundle, permits measuring to which extent we are close to finding the best ranking, preserving the same order of queries as the future contained in the report.

5.5.3.2 Robustness of parameters of LFCMD and BIbR

The complete mechanism of bundle recommending relies on two fuzzy clustering algorithms: LFCMD and BIbR, that involve several hyper-parameters, that need to be tuned before any further experiments. We decide to evaluate the settings of both these algorithms based on the combination that provides in the end, the most accurate bundles in terms of recommendation.

The previous experiments showed that LFCMD algorithm, paired with the GB4 group of features, is the best solution among the ones that we have evaluated, so it will be implemented in all next tests. LFCMD depends on two important parameters: parameter $m_f$ of fuzziness and the number $p$ of candidate medoids. A value of $m_f=1$ would provide a hard clustering that is not interesting in our case. The higher the parameter $m_f$ is, the
fuzzier are the intents we obtain. On the other hand, more candidate medoids contribute to more precise clusters, but they may slow down the calculation time.

Similarly, we need to parametrize the constrained fuzzy clustering \( BIbR \), responsible for the bundle construction. We build a candidate bundle for each cluster, with a fixed number of 5 queries. In addition to its fuzziness parameter \( m \), we need to tune the weights of the constraints integrated in its objective function \((\alpha, \beta, \gamma, \delta, \rho, \omega)\). We test all these parameters together and we use the offline evaluation, described in the previous section, to measure the accuracy of produced bundles.

For the sake of simplicity, a preliminary phase is dedicated to limit the choice of values of \( m = \{1.2, 1.4, 1.6, 1.8\} \), \( m_f = \{1.2, 1.4, 1.6, 1.8\} \) and of medoids \( p = \{3, 4, 5, 6, 7, 8\} \) to then perform brute force tests. These parameters are combined with all possible weights that the constraints can take from 0.1 to 0.5 with a step of 0.1, respecting their total sum of 1. The tests are performed over vector spaces of 18 and 10 intents and for each of them we perform a 10-fold cross validation, detailed in Section 2.2.4 of the Chapter 2.

We run several tests combining all the parameters of both algorithms, \( LFCMD \) and \( BIbR \), and we evaluate the recommended bundles offline, in order to identify the parameters’ combination that gives the most precise bundles. For both clustering algorithms, the initial selection of centers is not deterministic, and may impact the quality of the produced bundles. Thus, we decided to run each test 10 times, trying to be more precise in our evaluation. These tests are run over dataset \( D^3 \), using the group GB4 of features to detect the user intents. There are three types of parameters to be fixed: (1) number of intents, (2) clustering parameters (fuzziness and number of candidate medoids), and (3) the weights of soft constraints.

**Lessons learned.**
Each of the running test of this section was used to find and to fix different hyper-parameters of our algorithms, presented below:

*Number of intents.* To decide the appropriate number of intents that we should finally implement to \( LFCMD \), we fixed several combinations of other hyper-parameters of our algorithms. We set the fuzziness scores \( m_f = 1.2 \) and \( m = 1.4 \), and the candidate medoids \( p = 4 \) for all the experiments, and we varied the constraints weights to perform several comparative tests of \( BIbR \), on queries projected over 10 and 18 intents.

The precision of bundles built on queries projected over 10 intents, evaluated offline as in 5.5.3.1, showed to be less sensitive to the choice of constraint’s weights, than when projecting these entities over 18 intents. As a consequence, we chose to implement in future tests \( BIbR \) algorithm, that implements a \( LFCMD \) that discovers 10 user intents, as suggested by SAP experts. Now we can choose the optimal clustering parameters and constraints’ weights.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

Fuzzy clustering parameters. We fixed some random combinations of constraints’ weights and we performed a full combination of candidate medoids \( p = \{3, 4, 5, 6, 7\} \) and fuzziness parameters \( m_f = \{1.2, 1.3, 1.4, 1.5, 1.6, 1.7\} \) and \( m = \{1.2, 1.3, 1.4, 1.5, 1.6, 1.7\} \). The results are fairly uniform. For example, for the combination of these penalty terms: \( \alpha = 0.2, \beta = 0.2, \gamma = 0.1, \delta = 0.2, \rho = 0.2 \) and \( \omega = 0.1 \), the accuracy of the recommended bundles range from 0.5 to 0.54. For the sake of the rapidity of calculations and to be uniform with previous tests we choose to continue experiments with \( m_f=1.2, p=4 \) and \( m = 1.4 \).

Soft constraints weights.

As the number of user intents and the fuzzy clustering parameters are fixed, now we search for the best combination of constraints’ weights \((\alpha, \beta, \gamma, \delta, \rho, \omega)\). Each of them takes all possible values between 0.1 and 0.5 with a step of 0.1, under the constraint that each weight is strictly greater than 0 and that the sum of weights should be equal to 1. Table 5.8 provides some of the best observed combinations.

<table>
<thead>
<tr>
<th>Representativeness ( \alpha )</th>
<th>Uniformity ( \beta )</th>
<th>Personalization ( \gamma )</th>
<th>Diversity ( \delta )</th>
<th>Context ( \rho )</th>
<th>Ranking ( \omega )</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.58</td>
</tr>
<tr>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.56</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
<td>0.56</td>
</tr>
<tr>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>0.54</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.44</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.39</td>
</tr>
</tbody>
</table>

Table 5.8: Some of the best combinations of constraint’s weight

However, varying constraints’ weights with a step of 0.1 is not sufficient, looking how sensitive this algorithm is to the choice of their scores. Learning constraints’ weights deserve a more extensive test with finer scores.

It is important to mention that the constraints’ weights are customized according to the use case and the user preferences. Users can prefer, for example, to have more personalized than diverse recommendations, or inversely. The results of this tests are specific for the dataset we test and they do not guarantee to be the right choice for other use cases.

As we can see from the results of Table 5.8, the combinations that achieve higher accuracy, give a higher score to personalization, diversity and context constraints. This test gives us a better idea on how studying this constraints in the future. Looking to the last records of the table, we can conclude that strongly imposing the ranking of the queries or not considering the recommending context can lead to the degradation of the recommender performance. To respond to these questions we perform some user oriented tests in the following sections.

5.5.3.3 Online evaluation of bundles

Offline evaluation only validates the accuracy of recommended bundle based on some expected output, while the bundle quality depends on several constraints related to the perception of the user in a given context, or depending on some objectives in their search,
like favoring diversification of the results. As a consequence, different users, in different contexts, do not share the same perception of a bundle quality. For this reason, we introduce in this section an online user test where each SAP user involved in the experiment, evaluates subjectively generated bundles, relatively to an exploration context as reflected by a provided seed report, and a user profile learned over the set of all documents they accessed in the past.

**Protocol**  Our test involves 9 users from SAP and 16 reports from dataset $D_3$. Each user was asked to analyze some reports, and for each pair user-report, 5 bundles were proposed following different strategies as reflected by the weights on the constraints as detailed in Table 5.9, to complete the report.

The 5 bundles construction strategies of Table 5.9 are as follows:

- **“B1”** strategy corresponds to a uniform distribution of the weights and is considered as a neutral baseline.
- **“B2”** strategy uses the constraint’s weights learned in Section 5.5.3.2. We want to evaluate to which extent it makes sense to tune the weights automatically with brute force exhaustive search methods.
- **“B3”** strategy only involves the constraints term of the state-of-the-art approach [Alsayasneh et al., 2018]. We want to use an adapted version of [Alsayasneh et al., 2018] as a baseline in our context.
- **“B4”** strategy replaces the personalization term of “B3” by a contextualization term. We want to evaluate which is more important between long-term personalization or short-term context.
- **“B5”** strategy corresponds to a learned model where the ranking constraint is removed. We expect to evaluate the impact on user of the ranking term that we introduce.

We do not expect a definitive answer about the constraints’ weights from these tests because of the relatively small number of users, but we want to dig more into what is important for what type of user.

Each time users evaluate bundles, they go through several steps that are described hereafter and presented in Algorithm 3:

1. We present to the user a report from one of the documents of $D_3$ and 5 bundles of 5 queries each, recommended by our algorithm to complete this report.

2. The user evaluates each bundle separately. She is invited to fill a survey to validate the queries’ cohesiveness, diversity, relevance to the report, personalization and their ranking in the bundle. This will allow us to adjust our report penalty terms, according to the users findings. Finally, we ask for a global score and a complementarity score judged for each bundle of queries.

3. The same evaluation is repeated for each bundle proposed for each report.
Algorithm 3: User evaluation algorithm of recommended bundles

**Data:** Set of reports $R$, set of users $U$, bundle strategies $\in \{B_1, B_2, B_3, B_4, B_5\}$

**Result:** User evaluations $UE$

1. $UE \leftarrow \emptyset$
2. for $u \in U$ do
   3. $R_u \leftarrow R$ // set the possible reports for user $u$ as $R$
   4. for $i \in 1, 5$ do
      5. $r \leftarrow \text{random}(R_u)$ // choose a report at random in $R_u$
      6. $R_u \leftarrow R_u \backslash \{r\}$
      7. for strategy $s \in \{B_1, B_2, B_3, B_4, B_5\}$ do
         8. Generate a bundle $B_{u,s,r}$ for user $u$, strategy $s$ and context $r$
         9. $ue_{u,s,r} \leftarrow \text{evaluation of (cohesiveness, diversity, relevance, personalization, ranking, complementarity, global score)}$
      10. end
   8. end
5. end
11. return $UE$

<table>
<thead>
<tr>
<th>Bundle</th>
<th>Representativeness</th>
<th>Uniformity</th>
<th>Personalization</th>
<th>Diversity</th>
<th>Context</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>$\gamma$</td>
<td>$\delta$</td>
<td>$\rho$</td>
<td>$\omega$</td>
</tr>
<tr>
<td>B1</td>
<td>0.166</td>
<td>0.166</td>
<td>0.166</td>
<td>0.166</td>
<td>0.166</td>
<td>0.166</td>
</tr>
<tr>
<td>B2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>0.25</td>
<td>0.25</td>
<td>0.0</td>
<td>0.25</td>
<td>0.25</td>
<td>0.0</td>
</tr>
<tr>
<td>B5</td>
<td>0.1</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 5.9: Constraints’ weights for each of the 5 bundles we propose to complete a user report online.

As we could not implement such a system in the production BI platform, we developed a simple interface to allow the users to evaluate the visualized bundles recommended for several reports. We chose different reports from dataset $D3$, prioritizing those created by the users or frequently edited by them, and we ask the users if the bundles of queries we recommend would help in report Editing mode.

The dedicated testing prototype and the phases of the user evaluation are included in Appendix C. We asked internal users at SAP to fill an evaluation form for each report we recommend and to rate each bundle, generated from each strategy. We asked a global score to evaluate the tuning of constraints’ weights, as well as a specific score for each constraint, with a minimal rating of 1 point and a maximal of 5 points.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

**Lessons learned.** The form we asked users to complete, allows to derive several conclusions about the user satisfaction, globally and specifically related to some particular soft constraints, such as: uniformity, diversity and personalization.

*Global evaluation.* We asked users to rate globally each bundle of queries we recommend for the same report and to evaluate their satisfaction. As shown in Figure 5.3, the global average score that users give for all the recommended bundles is 2 out of 5. In general, the parameters of bundles $B_1$, $B_3$ and $B_5$ (Table 5.9) satisfy the users, as there are several tests rated 5 out of 5. But at the same time, each of them have recommended inappropriate queries more than once, as the minimum value is set to 1 for all the bundles.

![Global evaluation of bundles](image)

Figure 5.3: Distribution of global scores for all testers and each bundle strategy.

We can notice in the Figure 5.3 that the combination of constraints used for the recommendation of the third bundle give in general recommendations slightly more precise than the others. Comparing the average precisions of the first set of bundles $B_1$ (with uniform constraints’ weights) with the second $B_2$ (based on learned weights), shows that our tests varying constraints values with a step of 0.1, were not sufficient as $B_1$ and $B_2$ have nearly the same evaluation score. The bundles that have less matched users expectations are the bundles of group $B_4$, where the personalization constraint is missing. However, as shown by the range of variation and the difference on average, it is difficult to give a definitive conclusion with so few user evaluations.

**Bundle uniformity and visualization diversity.** Figures 5.4 and 5.5 show that the users are mainly satisfied of the uniformity of the queries recommended in a bundle and the diversity of their visualization whatever the bundle construction strategy is. Uniformity is rated with an average global score of 3 out of 5 and diversity of 3.3 out of 5. These results first show that our method is able to bring together queries related to the same topic even in the presence of several constraints that could bring noise in the produced bundles.

Second, interestingly, the sets of bundles $B_2$, $B_3$ and $B_4$, that are defined by the highest weights for diversity, are perceived on average (as denoted by the cross in Figure 5.5) as diverse as the other bundle strategies. However, it can be noticed that if we consider the median (as denoted by an horizontal line in each bar in Figure 5.5), the bundle $B_2$ is considered as better than all the others with a median score of 3.5 out of 5. If we compare...
5.5. TESTING OUR BUNDLE BUILDING APPROACH

$B_2$ to $B_3$ and $B_4$, this may be due to the fact that $B_3$ does not include context constraints and similarly $B_4$ does not take into account personalization. As a possible consequence, these strategies might produce bundles that are less in line with what users expect, which makes it more difficult to be positive about diversification of the bundle for a final user. Finally, it can be noticed that all 3 diversity oriented bundles $B_2$, $B_3$ and $B_4$ exhibit a much lower variation in terms of diversity score than $B_1$ and $B_5$, showing that introducing the diversity constraint term produces bundles that are more homogeneous in terms of perceived diversity. Again, all these conclusions have to be considered cautiously because of the number of users considered in the experiment and more test should be conducted.

As for the uniformity of the queries of a bundle, fewer constraints included in the objective function increases the importance score of uniformity, reflected in higher ratings for the last bundles’ proposition ($B_3$-$B_5$).

Finally, this results confirms our choice of uniformity and diversity axis, which do not interfere and allow the recommender systems to satisfy both these constraints without decreasing the whole user experience at once.

![Figure 5.4: Distribution of uniformity scores for all testers and each bundle strategy.](image)

![Figure 5.5: Distribution of diversity scores for all testers and each bundle strategy.](image)
5.5. TESTING OUR BUNDLE BUILDING APPROACH

Personalization of bundles. Figure 5.6 shows that bundles produced by the strategy $B_4$ contain the queries that were considered as the most personalized queries, by our sample of user. This result was not expected since this strategy does not take into account the user profile, but the user context, which is reflected by the first queries of the actual report. This result may be related with the fact that in our experiments, context and user profile are close to each other. Indeed, the limited dataset and the number of participating users, narrow our choice of possible reports to test. Often, the user is familiar with the reports, for which we recommend the bundles we aim to evaluate. As these reports might have been already visited by the user in the past, the queries of their seeds that define the context, contribute also in the definition of their profile. This explains the relation and the similarity between user and context profile, and once that the user profile constraint is set to 0, the context is sufficient to represent the short and long term intent of the user. We leave as future work experiments that would differentiate gradually user profiles and context, and each time measure the most appropriate bundle strategy.

To further study how $BibR$ personalizes globally the recommended queries, we focus in Figure 5.7, on the corresponding scores of personalization given by each user for the suggested bundles of each strategy. Even though there is not the same number of evaluations given by each user, we can notice that User 2 and User 3 mostly consider the recommendations as personalized. For others, as User 6, we see that it is hard to learn their profile and to recommend close queries. Looking closer to the reports that define their profiles, we can explain this result by the fact that User 2 and User 3 have created and consumed professional documents, while User 6 has mostly worked with different documents to test the platform functionality, often with no relation between them, which adds noise in the user profile.

![Personalization scores for each bundle](image)

Figure 5.6: Distribution of personality scores that all testers gave to each bundle. This box-and-whisker plot shows as well the minimum and the maximum score attributed to them.

At the end of these user tests, we first conclude that the tuning of our hyper-parameters is not sufficient, as the combinations of scores we propose for constraints’ weights do not generates the most relevant bundles. Second, measuring the user satisfaction related to
different constraints, which represent qualities of our bundles, we resume that we have chosen the good axis to uniform and diversify bundles, which are coherent and seems to generally satisfy the user. Finally, we have observed that in some cases, because of restrictions related to our experimental protocol, the absence of user profile constraints could be balanced by context constraints when some users might share the same user profile and context. This raises the need for further tests to better understand how long and short term intents impacts the bundle quality, and how to satisfy users with well defined and narrow interests.

5.5.3.4 Robustness to quality of user profile

Our bundle building algorithm quality relies heavily on the ability to characterize some entities, like the user profile, that is one of the main proposal of our approach. It is thus of prime importance to be able to evaluate to which extent the quality of the user profile impacts the quality of the produced bundle in terms of recommendation. To do so, we first conducted some bundle recommendation evaluation tests, by providing our algorithm either with the best user profile description possible, or with degraded versions, that carries a less accurate representation of the user interests or her actual objectives.

Second, we want to evaluate to which extent our model as presented in Section 5.2.2, can generate user profiles of good quality, that is to say how do these automatically learned profiles compare with the previous ideal or degraded profiles. This question is very important as this modeling step is crucial in our recommendation process, and also because our data is very noisy.

**Protocol.** We test over reports of $D_3$ dataset documents. Each report can be split in two parts as usual: the first queries reflect an exploration context as a seed and are used to build the context, while the last 5 queries correspond to what a good bundle should propose as the expected future. In this context, we simulate:

- an ideal user profile $IU$ generated using the future queries that should be discovered in the test report; in this case, we provide the bundle algorithm with the queries it
is supposed to discover in the bundle, hence the ideal case,

- a degraded user profile, that modifies the values of the intents with a random noise whose percentage reflects how much the profile is different from $IU$,
- a “real user profile” $RU$ in the sense that this is a profile that is learned automatically by our approach as defined in Equation 5.2 based on past queries of a real user of the platform.

In the end, our experiment consists in measuring the loss of prediction precision at different thresholds of similarity, as explained in the offline evaluation in Section 5.5.3.1, when comparing respectively the ideal user profile $IU$, degraded profiles with 20% or 40% noise and a real user $RU$ profile extracted from the user’s previous query usage. For this experiment, we focus on the learning of user profile and we ignore the context and ranking constraints. We fixed the other constraints’ weights as below:

<table>
<thead>
<tr>
<th>Representativeness $\alpha$</th>
<th>Uniformity $\beta$</th>
<th>Personalization $\gamma$</th>
<th>Diversity $\delta$</th>
<th>Context $\rho$</th>
<th>Ranking $\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Lessons learned.** Figure 5.8 shows that the better the quality of the user profile fed to our algorithm, the more relevant the recommended item. The perfect user profile allows very good recommendations for low similarity threshold (that is to say, when compared queries from future and recommended bundles are most likely judged identical during evaluation), while the noise degrades the performances as expected. According to this test, comparing our real user profile with the perfect and the degraded profiles, we notice that the user profile learned automatically in our approach is close to the ideal user and gives better precision than profiles that introduce 20% or 40% of noise. We consider that this is a good result when considering the lack of information in the query log used for this experiment.

**5.5.3.5 Integration of user profile and context**

User and context profiles represent long and short-term user intents, and in this section we study the relationship between these constraints and the impact of integrating them together in our model. First of all, similarly to what is done in Section 5.5.3.4 for the user profile, our first experiment evaluates to which extent the precision of the recommender system depends on the quality of the context. To do so, we will generate an ideal context and a deteriorated one with random noise. Then, our second experiment concerns how mixing information from a user profile with a context might increase or decrease the recommendation precision. Several tests are performed to compare the combination of good/bad user profiles/contexts.

**Protocol.** We first set a test group of 13 reports that share nearly the same intents to uniformize the basis of our tests. This group is chosen by comparing representative vectors of intents of each report (calculated using Equation 5.1), and finally keeping the group
5.5. TESTING OUR BUNDLE BUILDING APPROACH

Figure 5.8: Precision for different similarity thresholds, for a simulated ideal user \( IU \), whose profile is degraded with 20% and 40% of noise, to be compared with a real user profile \( RU \).

that minimizes the distance between reports. We denote by \( C \) the profile built over all the seed queries of this report selection. Then, we define:

- two contexts: \( C_1 \) is the context that is defined automatically by our approach as defined in Equation 5.1, based on the seed queries of each tested report, while the context \( C_2 \) is \( C_1 \) deteriorated with 20% noise,

- two user profiles: \( U_1 \) is a real user profile that is the closest to the profile of \( C \) and \( U_2 \) is the most different user from \( C \) in our test batch.

We choose these two types of users, close and very different to the profile of all 13 reports we test, and we expect to observe higher bundle precision by integrating similar or different short and long term intents together, in comparison with systems that implement them separately. Degrading the context profile \( C_1 \) of tested reports to \( C_2 \), should lead to a loss of the recommendation precision, which increases when combining either good or degraded user profiles or contexts. The constraints weights for both tests, while considering user or context profile separately or integrated together, are shown below:

<table>
<thead>
<tr>
<th>Test</th>
<th>Representativeness ( \alpha )</th>
<th>Uniformity ( \beta )</th>
<th>Personalization ( \gamma )</th>
<th>Diversity ( \delta )</th>
<th>Context ( \rho )</th>
<th>Ranking ( \omega )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>User</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Context + User</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Lessons learned.**

*Context.* Learning a good context profile is important to recommend queries close to the user short-term intents and to improve the quality of the recommended bundle. As expected, degrading the profile of context \( C_1 \), decreases the accuracy of the recommender.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

The degraded context, named $C_2$, is less precise than $C_1$, as shown in Figure 5.9. It is interesting to notice that the overall precision is low compared to previous experiments. This due to the fact that all other constraint’s weights are set to 0 and the system recommends only based on the user context.

Integrating user and context profiles. Figure 5.10 presents two important results. First, the recommender system reveals a low precision while relying exclusively on the context $C_1$, compared to $U_1$ in Figure 5.10, where we only rely on user profile, and even if the observed differences are not very important. Knowing only the long-term intents helps better than knowing only the short-term intents. This result is in accordance with the user evaluation in Figure 5.3, where comparing the bundle strategies that consider only the user profile ($B_3$), or only the context ($B_4$), bundles that rely on user profile are rated higher by the users than the ones relying only on the context. In our setting this is an expected result as user profile aggregates all past activities of a user when context is limited to 5 queries. Furthermore, considering both, user and context profile, which means taking into account the long and the short-term intents achieves a better accuracy than when considering these intents separately, more significant for lower similarity thresholds.

Besides the promising results we had while integrating good user and context profiles, we wanted to test the performance of our recommender when the context profile is not appropriate in the case of the degrated context $C_2$. Figure 5.11 illustrates how the precision decreases when we integrate $C_2$ instead of $C_1$ to the user profile $U_1$. We observe that precision is still better than considering only $C_2$, or even $U_1$ for lower thresholds.

Indeed, in our experiments, context relates to report seeds which represents only a few queries that does not borrow the same variety of information as a user profile. As a consequence, it is more interesting to consider a user profile than a context, but using both allows one to compensate over the limitation of the other: user profile identifies potentially interesting queries while containing potentially too much variety while context focuses on
5.5. TESTING OUR BUNDLE BUILDING APPROACH

5.5.3.6 Benefit of introducing the ranking constraint

In this experimentation, we show the importance of adding the penalty of ranking queries, testing with the first mode of Offline Evaluation (Section 5.5.3.1), that takes into account the order of queries in the bundle.

Protocol. We follow the same protocol defined in Section 5.5.3.5 and we continue testing over the user profile $U_1$ close to the context. Differently from all the previous tests where
we compare bundles based on their accuracy scores, here we take into account the order of queries in the computation of precision scores. More precisely, we only compare queries of the same rank between the future and the bundle to evaluate the precision.

Below are the constraints’ weights given to user profile and ranking during our tests:

<table>
<thead>
<tr>
<th>Test</th>
<th>Representativeness $\alpha$</th>
<th>Uniformity $\beta$</th>
<th>Personalization $\gamma$</th>
<th>Diversity $\delta$</th>
<th>Context $\rho$</th>
<th>Ranking $\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Ranking + User</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Lesson learned.** Figure 5.12 details the precision obtained when considering the personalization constraint for user $U_1$ or the personalization constraint for $U_1$ and the ranking constraint, denoted by $U_1 + R$, with the precision that takes into account the order of queries (hence “-O” suffix). It can be seen that adding the constraint of ranking queries in $B1bR$ allows to find better bundles, that is to say an order of queries in the bundle that is closer to the order of the future expected queries. It is important to notice that our ranking constraint also selects the most appropriate queries based on the one already in the bundle. Thus, this result shows that ordering do not degrade performance, and on the contrary allows for a more relevant selection of queries, at least for low similarity threshold values.

![Figure 5.12](image)

Figure 5.12: Comparing the recommended bundle with the real future, considering the order of queries suggested.

### 5.5.3.7 Benefit of introducing diversity

The objective function is built in a way that respects uniformity and diversity of bundles queries, and we need to evaluate the benefit of integrating them both. This test shows how adding diversity instead of demanding strongly unified queries improves the accuracy of recommendations, as shown in Figure 5.13.
5.5. TESTING OUR BUNDLE BUILDING APPROACH

Protocol. We test over dataset $D_3$ and Table 5.10 indicates the weights assigned to constraints when recommending only based on the uniformity of queries and when adding diversity.

<table>
<thead>
<tr>
<th>Test</th>
<th>Representativeness $\alpha$</th>
<th>Uniformity $\beta$</th>
<th>Personalization $\gamma$</th>
<th>Diversity $\delta$</th>
<th>Context $\rho$</th>
<th>Ranking $\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniformity</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Uniformity+Diversity</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 5.10: Constraints weights to evaluate the contribution of uniformity and diversity

Lessons learned. The result of this test, presented in Figure 5.13, shows how the accuracy of $B1bR$ increases when we diversify bundle queries in terms of visualization rendered. Providing bundles only under the uniformity constraint favors the collection of queries that share very similar query parts. The distance function implemented in our uniformity function (Equation 5.5) is measured over the membership vector of queries to the intents space. As $LDAqueryPart$ is one of the most decisive features in the discovery of user intents (Section 5.5.2.2), the uniformity encourages the choice of queries that share similar query parts.

Queries are limited to a set of visualization types their results can be displayed, as not every dimension or measure can be represented by the same visualization type. Diversifying the visualizations proposed to the user implies the diversification of query parts. This increases the chances for the user to find relevant queries in the recommended bundle, as showed in the Figure 5.13 comparing the results when included uniformity separately and when adding diversity constraint.

Figure 5.13: Balancing uniformity and the diversity constraints gives more precise bundles.
5.5.3.8 Comparing to state-of-the-art approaches

In this section, we compare \textit{B1bR} to two well known algorithms of the literature, that build bundles of queries around pivots, namely \textit{BOBO} [Amer-Yahia \textit{et al.}, 2014] and \textit{CPS} [Bota \textit{et al.}, 2014]. We implement two adapted versions of these algorithms over our data and we use the same feature-based metric to measure the similarity between queries and the same diversity function based on query visualization. These two algorithms are agnostic of any user model or ordering of the items.

\textbf{Protocol.} We evaluate offline the bundle produced by \textit{BOBO}, \textit{CPS} and \textit{B1bR} and we compare the accuracy of these methods for several recommendations over reports of \textit{D3} dataset. \textit{BOBO} and \textit{CPS} propose several bundles to the final user. To be comparable with our system, we choose to evaluate the best bundle they produce for each report.

We introduce two different evaluations for each algorithm as detailed in Section 5.5.3.1:

- the suffix "O" indicates that we use the evaluation that takes into account the order of each query as provided by our ranking constraint,
- the suffix "NO" indicates that the order is not taken into account and that a bundle is considered as a bag of queries. In this case, we only consider the ranking that provides the best matching found between the bundle and the expected future.

\textbf{Lession learned.} The results presented Figure 5.14 show that the accuracy of our method \textit{B1bR} overcomes the adapted state of the art approaches. For all approaches, we notice that the order of queries, as evaluated by \textit{approach-O}, is close to the expected ones as denoted by \textit{approach-NO}. This means that our ranking heuristic allows to find an ordering of queries close to the best possible arrangement of queries. Finally, in our algorithm \textit{B1bR}, our greedy heuristic also selects the candidate queries to integrate the bundles based on the ranking constraint contrary to the adapted \textit{BOBO} and \textit{CPS}. It can be seen that this selection is beneficial since our algorithm performs better than the other approaches on this test.

5.6 Chapter conclusions

In this chapter, we proposed a bundle recommender systems in BI, based on discovered user intents. First of all, we prove that the feature-based metric presented in Chapter 3 is easily adaptable for new datasets and that the identified intents can be used to define an uniform space, that permits the comparison between different types of entities.

Amongst all works of the state-of-the-art that build bundles of items, we choose to implement a CAP approach for our \textit{B1bR} algorithm. Its main advantage stands on the ability to customize the number and the type of the soft constraints we want to integrate, that define the quality of bundles. Our experiments showed that clustering data and creating bundles around centroids contribute to a higher accuracy of suggested queries, rather than choosing pivots, to which we associate other items, that each separately should fulfill
Figure 5.14: BOBO vs CPS vs $B1bR$

constraints, as uniformity or diversity. Moreover, the number and the implementation of constraints for these approaches is limited.

This first recommender system that suggests bundles of queries in BI domain, with the goal of completing a BI report, was published in the Workshop on Recommendation in Complex Scenarios (ComplexRec) of ACM Conference on Recommender Systems in October 2018 [Drushku et al., 2018].
Chapter 6

Conclusions and future work

6.1 Conclusions

The goal of this thesis is to provide collaborative recommending approaches that leverage user intents in modern BI systems to relieve the user from tedious explorations and to help them searching sequences of complementary queries. These systems combine, and sometimes extend, state-of-the-art techniques from literature in Web search, composite items and BI query recommendation.

At the heart of our recommenders is an approach for identifying coherent intents of BI users with various expertise querying data sources by means of keyword-based analytical queries. Our approach [Drushku et al., 2017] relies on the identification of discriminative features for characterizing BI interactions and on the learning of a similarity measure combining these features. Experimenting with different sources and BI platforms, we illustrate how this metric is generic and easy adaptable in several cases. Implementing clustering algorithms that use our feature-based metric permits to define a strict or a fuzzy membership of queries to each discovered intent.

The identified user intents are treated as first-class citizens in two collaborative BI query recommender systems. The IbR recommender [Drushku et al., 2019] learns the probabilities for a user to switch from one intent to another, to suggest the following steps of an exploration. We have shown through user tests that our approach is effective in practice and can be beneficial to analysts whose intents match those of expert users, or whose intents change during the analysis. BIbR [Drushku et al., 2018] exploits furthermore the intents related to both the user long and short term history, to suggest to the final user a bundle of coherent and diverse queries related to her profile and her current context. This is achieved though a uniform representation of BI queries, user profiles and current BI reports. Overall, our results show that the semantically rich user traces provided by modern BI systems help the detection of coherent BI user intents and that such intents can also be exploited successfully by state-of-the-art recommendation strategies.
We proposed an offline evaluation technique that compares suggested queries to expected ones, inspired from [Aligon et al., 2015], to measure the accuracy of the recommended sequences of queries generated by IbR and BIbR. Besides this global evaluation score, as BIbR aims to provide personalized, diverse and complementary queries to a specific user in a given context, we needed to specifically evaluate the quality constraints. The set of mono-variation tests with bundles of queries showed how important is for the bundle precision to calculate a good user and context profile and the benefit of integrating them both. Furthermore, balancing query similarity and diversity seems to contribute as well to the recommender accuracy, rather than considering only one of them separately. Building user profiles based on the queries the user manipulated in the past, represented over a intents space, provides profiles very close to a theoretic ideal one. Both algorithms, IbR and BIbR, give more accurate recommendations compared to state-of-the-art approaches.

The online evaluation of BIbR with intern users at SAP was important for the validation of our findings from the previous offline evaluation. The most important conclusions we retrieve from user experience is the appreciation of visualization diversity and the personalization of recommendations. Interestingly, we discovered a significant relation between the user long and short term intents, when testing over reports already visited by the users. In general, we had a positive feedback and users found the query recommender helpful in the completion of BI reports.

### 6.2 Future work

The recommending systems we propose are designed as a composition of several modules, each inspired from a different existing technique. Experiments showed that the algorithms we employed perform better than the state-of-the-art approaches, but several robustness tests and deeper analyses to optimize our clustering algorithms, as well as the proposal of novel, more appropriate evaluation schemes, have been left as future works.

#### Extending offline/online tests and recommender models.

The BIbR recommender is an algorithm combining a dozen of finetuned parameters, each impacting considerably its quality. We performed a lot of tests looking for the best parameter values, and managed to find stable values for e.g., the number of intents, fuzziness or candidate medoids. Soft constraints seem to be key parameters that directly impact the quality of recommendations, that is why further experiments are needed to "brute force" new combinations of such constraints with finer scores.

A short-term evolution of our work would be to integrate both our recommenders in a single system, IbR and BIbR being based on the user intent and suggesting sequences of queries. We have identified several “user situations” like starting a new session/document, being at the start/middle of her exploration or having a different perception of her own context. We want to test how these recommenders behave to different inputs and how we can automatically switch between them to use the most optimal one, depending on the
6.2. FUTURE WORK

user situation and the information provided. A deeper study is required to observe how different algorithms from the state-of-the-art and our recommenders behave in different situations. This would allow us to design a new dynamic system, that automatically can switch between recommenders, as the user conditions change.

We can further modify BIbR, as the extensibility of its greedy function implemented allows us to easily add a new penalty term: novelty. Inspired from some of the literature works that define novelty [Vargas et Castells, 2011, White, 2016], we can add a new soft constraint to add relevant items, that the user would not have found by herself, to recommended bundles.

Another objective to realize in the near future is finding and testing new and more BI logs, which will permit us to better choose the learning subset of data, by choosing qualitative documents and active users. The technique of discovering the user intents, the feature-based metric and the recommending algorithms are easily adaptable for different data sets. We are searching for bigger and richer datasets to asses the robustness and the efficiency of our approaches. Testing with new data sets imposes a parallel study of the adaption of the actual features and a research of new features, to profit from the richness of the data we dispose. Working with web data would enable to explore the external categories like ODPs', to classify keywords or query parts entered by the user, or to employ it as one of the features.

Other axes to explore are the user categories defined on their activity and habits over the BI platform. We want to expand the initial work of learning over experts to help the beginners. Other intermediate type of users may be added and a larger database can permit us to limit and to test the learning dataset to experienced users.

Broadening online evaluation and optimizing algorithms.

The extension of BIbR and the integration with IbR need additional user evaluations. A good way to ameliorate and to adapt our recommenders to user preferences is to observe the user choices following our suggestions. This feedback will help us to understand the relationship between the users, context and the items they choose to use from our recommendations, and use this knowledge for future recommendations. Our future plans include an extension of the user tests to observe a real situation with SAP customers, i.e. a large scale online experiment, to measure their satisfaction towards the new novelty constraint and the alteration of two recommenders. As a consequence, the online evaluation protocol will be broadened with new evaluation metrics related with the new recommending mechanism, and the users will judge the two recommenders separately, as well as the way they collaborate.

Optimizing our clustering algorithms is another mid-term target. To reduce the computational time of the user intents, we can simplify the feature distance measurements, select a group of active users and provide an incremental version of segmentation algorithm, as in [Guha et al., 2015]. Meanwhile, we have started the integration of the second prototype
6.2. FUTURE WORK

of Bundle Recommending Algorithm to the SAP BI platform. We aim at extracting more
details to enrich the feature-based model from another similar production database of real
users’ reports. The implementation of these optimizing schemas are necessary to scale up
our calculations for big data. Furthermore, we will work to choose the best way to visualize
the ranking of the recommended bundle.

Improving our recommender systems through user feedback.

{BIbR} is a small step towards a promising future of Composite Item recommendation.
We need more evaluating systems of complementary and online user tests. As future long-
term directions we want to go beyond the classical evaluation of the recommendations that
measures the accuracy of the result regarding to the expected future, which is not conform
to the complementary and diversity-orientated recommenders we propose. The lack of
general measurements of diversity and complementarity of items, and of user satisfaction
estimation, emerged the need of a new validation technique. Subjective interestingness
measure for BI exploration, involving the user belief modeling over past interactions, is a
next phase to enrich our evaluating methods. Our recent attempt [Chanson et al., 2019] is
a preliminary step in this direction. Following the idea of [Bie, 2013], on the long run, we
aim to measure the interestingness of the suggested bundles through the interestingness
calculated for their composing queries, based on the prior knowledge that the user has
about the data and the cost for the user to understand the query and its evaluation. A
study of the literature shows that there exists such interestingness measures in the context
of data exploration, as the discovery of interesting patterns [Bie, 2013, Bie, 2018], but not
in the context of data sources interrogation.

An important long-term goal is to yield a generic approach, easily adaptable to every
dataset and not only on the BI domain, that can deal with structured or unstructured data.
The user intent discovering model uses a supervised learning algorithm, so in the future
we preview implementing unsupervised algorithms. As the first attempts of discovering
the user intents are mostly orientated in the detection of regular patterns and association
rules, to complete these study, we would like to test a similar system that recommends
based on itemsets.
Appendix A

Business needs

Hereafter, we list the business needs that were proposed to SAP analysts during our experiments.

1. Plan what kinds of films should Spain media shops aim to sell this year and in which media format? According to your prediction, calculate a target sales revenue.

2. For each European country, detect which genres of films did not reach the expecting sales.

3. For which Medias and Genres Iceland has not reached the total sales target? What should they change in their selling politics, knowing their preferences and top retailers?

4. For each region, and for each month (from January to June), which should be the priority country for Decathlon to be supplied with goods?

5. Which retailer has the best performance (sales revenue/target sales revenue) in 2013 and what is its growth for 2014 and 2015? Which were the most successful colours in these three last years?

6. Which are the three most problematic countries running out of stocks? Which are the retailers and products they should be supplied?

7. Which airports have the most increasing number of passengers from 1990-2010? Analyze the main airports with the greatest number of passengers (2010) for each region to find out if this is related to the increasing number of destination/population.

8. Which is the country with the greatest number of airports? How many passengers have flown from this country in 2005 and in 2010? Is this in proportional with the number of destinations from each airport?

9. For at least one country of low / lower middle / upper middle income countries, analyze how the GDP has evaluated.

10. In which Income Group would you classify a candidate country with a GDP of $6 billion?
Appendix B

Metrics to evaluate the quality of discovered intents

The evaluation of discovered intents can use the classical measurements of cluster quality, as the intra/inter cluster similarity assessed by Silhouette coefficient. A higher intra-similarity over a low inter similarity assures well separated clusters. On the other hand, fuzzy clusterings and overlapped data do not guarantee well separated clusters. Thus, we consider other measures of cluster quality, as RI, ARI and NMI, which hold only on the item repartition and not in class labels. They are used to compare the calculated intents to the ground truth. We detail these metrics below.

**Silhouette Coefficient**: The clusters we obtain and the collection of all proximities between their items are the only thing we need to calculate the Silhouette Coefficient [Rousseeuw, 1987] and to validate the cluster consistency. For each item \( i \) and the cluster \( A \) to which it has been assigned, \( A(i) \) denotes the average dissimilarity of \( i \) to all other objects of \( A \). Similarly, for all other clusters \( C \) different of \( A \), \( d(i,C) \) denotes a set of average dissimilarities of \( i \) to all objects of each \( C \) respectively, and \( B(i) = \text{minimum} \{d(i,C)\} \) is the minimal distance of \( i \) to the closest cluster \( B \) of \( C \). The Silhouette Coefficient is calculated as:

\[
s(i) = \frac{B(i) - A(i)}{\max\{A(i),B(i)\}}
\]  

(B.1)

It ranges from -1 to 1, where the higher the positive value is, the better the observation is classified in its own cluster and the further it is from the other clusters. Negative values show that most of the observations of the clusters are closer to another one rather than in the cluster in which they are categorized. Noticeably, Silhouette coefficient was initially designed for k-medoids-like algorithms that produce compact clusters.

**RI**: Rand Index [Rand, 1971] evaluates the accuracy of clustered items compared to the ground truth, as in Equation B.2. It computes the percentage of pairs of objects for which both classifications, the computed and the ideal expected, agree. It considers only
the pairs of points that either both items are in the same class either they are not.

\[ RI = \frac{a + b}{n_2} \quad (B.2) \]

where \( n \) is the total number of elements classified, \( a \) is the number of pairs found in the same cluster and \( b \) is the number of pairs in different clusters for both clusterings.

**ARI**: The inconvenience of Rand index is that it takes into account only the pairs that are both found in the same cluster or not. This information is not sufficient to validate the clustering. A good RI score does not allow to say if the clustering is good or the problem is simple. Adjusted Rand Index is the Rand Index corrected using the ‘chance’ to get the same partitions randomly. The ARI takes into account the chance of overlap. It can take negative values indicating that the amount of overlap is less than expected. When there are a lot of clusters, there’s a higher chance that a pair of items in both sets are in different clusters. This is still counted as a concordant event in the RI. ARI takes the randomness into account and it is defined:

\[ ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \frac{\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}}{2} - \frac{\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}}{\binom{n}{2}}}{\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2} - \frac{\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}}{\binom{n}{2}}} \quad (B.3) \]

where \( n_{ij} \) is the number of elements that are found in both clusterings in cluster \( i \), \( a_i \) is the sum of all \( n_{ij} \) of the row \( i \) and \( b_j \) is the sum of all \( n_{ij} \) over the column \( j \) of a contingency matrix.

**NMI**: Normalized Mutual Information is a measure of the knowledge about the ground truth by knowing the clusters. It computes how far the detected clusters are from the ground-truth ones. [Tackx et al., 2017] measure how much knowing one of these partition reduces uncertainty about the other. Normalized mutual information can be information-theoretically interpreted [Manning et al., 2008].

\[ NMI(C, C) = \frac{I(C, C)}{H(C) + H(C)} \]

where \( I(C, C) \) is the mutual information. It measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are:

\[ I(C, C) = \sum_k \sum_j \frac{|c_k \cap cl_j|}{N} \log \frac{N|c_k \cap cl_j|}{|c_k||cl_j|} \]

\( H \) is the entropy:

\[ H(C) = -\sum_k \frac{|c_k|}{N} \log \frac{|c_k|}{N} \]

and \( C = \{c_1, \ldots, c_k\} \) is the set of clusters and \( I = \{cl_1, \ldots, cl_j\} \) is the set of classes of the ground truth. NMI takes a value between 0 and 1. If it is close to 0, knowing the cluster of an item gives no information about the real class he belongs to.
Appendix C

Stages of prototype testing by internal SAP users

The testing application and the stages of the user evaluation are shown in Figures C.1, C.2 and C.3. An example of the survey form is found in Figure C.4.

Figure C.1: The open page of our testing application, where the user can choose the report for which she wants to consult recommended bundles.

1We apologize for the blurry pictures, but these examples can disclose enterprise sensitive information.
Figure C.2: An example of an existing report\textsuperscript{1} that the user will complete.

Figure C.3: The user can click on 5 tabs to visit the 5 bundles generated from 5 different strategies, each composed of a list of 5 queries, that we propose for the given report.
Figure C.4: The form the user has to fill to rate each bundle with 1 to 5 points for each constraint. We asked a global score as well to evaluate the tuning of constraints’ weights.
Bibliography


[Bie, 2018] Bie, T. D. (2018). An information-theoretic framework for data exploration. from itemsets to embeddings, from interestingness to privacy. In In the Keynote presentation given at IDEA’18 @ the KDD’18 conference.


