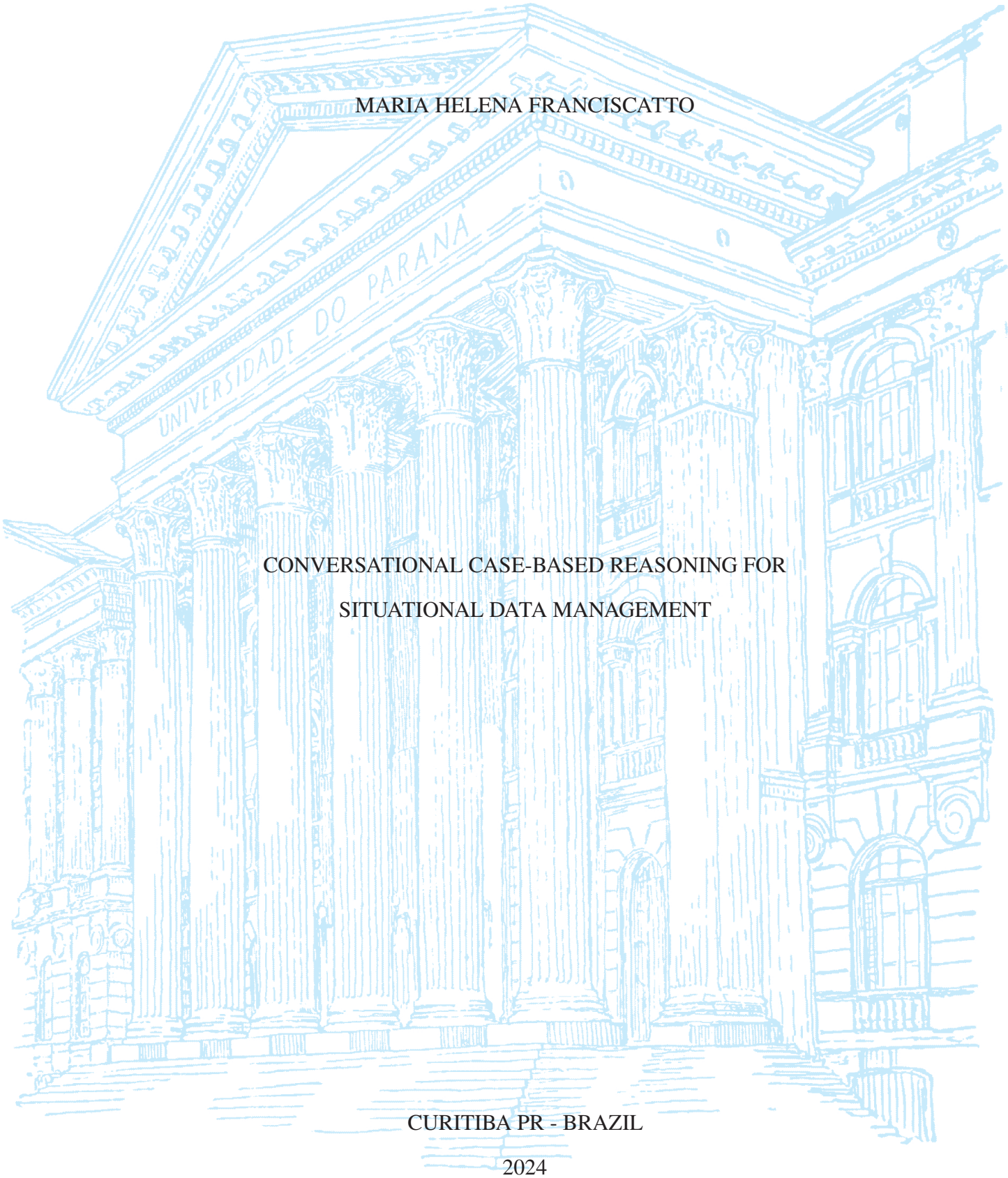
UNIVERSIDADE FEDERAL DO PARANÁ

MARIA HELENA FRANCISCATTO



CONVERSATIONAL CASE-BASED REASONING FOR SITUATIONAL DATA MANAGEMENT

CURITIBA PR - BRAZIL

2024

MARIA HELENA FRANCISCATTO

CONVERSATIONAL CASE-BASED REASONING FOR SITUATIONAL DATA MANAGEMENT

Tese apresentada como requisito parcial à obtenção do grau de Doutora em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. Luis Carlos Erpen de Bona.

Coorientadores: Prof. Dr. Celio Trois, Prof. Dr. Marcos Didonet Del Fabro.

CURITIBA PR - BRAZIL

2024

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Franciscatto, Maria Helena
Conversational case-based reasoning for situational data management / Maria Helena Franciscatto. – Curitiba, 2024.
1 recurso on-line: PDF.

Tese (Doutorado) – Universidade Federal do Paraná, Setor de Ciências da Exatas, Programa de Pós-Graduação em Informática.
Orientador: Prof. Dr. Luis Carlos Erpen de Bona
Coorientador: Prof. Dr. Celio Trois
Coorientador: Prof. Dr. Marcos Didonet Del Fabro

1. Processamento da linguagem natural (Computação) 2. Interação homem-máquina. 3. Aprendizado do computador. 4. Feedback. I. Bona, Luis Carlos Erpen de. II. Trois, Celio. III. Fabro, Marcos Didonet Del. IV. Universidade Federal do Paraná. V. Programa de Pós-Graduação em Informática. VI. Título.

Bibliotecária: Angela Pereira de Farias Mengatto CRB-9/1002

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **MARIA HELENA FRANCISCATTO** intitulada: **CONVERSATIONAL CASE ‑ BASED REASONING FOR SITUATIONAL DATA MANAGEMENT**, sob orientação do Prof. Dr. LUIS CARLOS ERPEN DE BONA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutora está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 01 de Março de 2024.

Assinatura Eletrônica
21/03/2024 15:28:12.0
LUIS CARLOS ERPEN DE BONA
Presidente da Banca Examinadora

Assinatura Eletrônica
22/03/2024 14:46:41.0
LUIZ CELSO GOMES JUNIOR
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
27/03/2024 14:39:49.0
LETICIA MARA PERES
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
22/03/2024 13:17:17.0
HEGLER CORREA TISSOT
Avaliador Externo (UNIVERSIDADE DREXEL)

Assinatura Eletrônica
27/03/2024 18:59:25.0
CELIO TROIS
Coorientador(a) (UNIVERSIDADE FEDERAL DE SANTA MARIA)

Aos meus pais, que sempre me falaram: "o conhecimento ninguém pode te tirar". Amo vocês de todo o meu coração.

ACKNOWLEDGEMENTS

Ao longo do desenvolvimento desta tese eu pude contar com o apoio e a generosidade de pessoas maravilhosas, as quais eu não poderia deixar de agradecer.

Em primeiro lugar, agradeço à minha família, minha fortaleza e meu amor maior, por serem minha referência de coragem, honestidade e dedicação. Minha eterna gratidão aos meus pais, Sandra e Rudiberto, que tanto se sacrificaram para que eu pudesse chegar onde cheguei hoje, e à minha irmã, Maria Elisa, por sempre ter sido a maior apoiadora dos meus sonhos.

Ao meu marido, Cezar, que pacientemente acompanhou todas as minhas noites e finais de semana de trabalho. Obrigada por ser o melhor companheiro que eu poderia pedir na vida e por celebrar cada conquista minha como se fosse tua.

Gostaria de agradecer ao meu orientador, Luis Bona, pelos conselhos e confiança no meu trabalho. Ao professor Marcos Didonet, que foi meu orientador nos primeiros anos do meu doutorado, obrigada por tantos direcionamentos e por me apoiar na minha mudança de país. De forma especial, agradeço ao professor Celio Trois, que me acompanha desde o mestrado em Santa Maria, por ser não apenas um professor, mas também um grande amigo.

Agradeço aos amigos da minha terra natal que sempre torceram por mim, e agradeço especialmente aos amigos que estiveram comigo em Curitiba durante minha trajetória: meus colegas de laboratório/aula/vida (Krissia, Marcelo, Luis, Cris), e amigas que dividiram casa comigo (Bruna e Edilaine). Jamais esquecerei o quanto vocês tornaram minha vida mais leve e feliz.

Por fim, agradeço a Deus por tantas graças e oportunidades recebidas. Sem Ele, nada disso seria possível.

RESUMO

Melhorar o suporte ao usuário na integração de dados exige o tratamento de informações em tempo real, um processo desafiador para sistemas de banco de dados convencionais. Muitas abordagens de integração de dados realizam a integração em tempo de consulta ("on-the-fly") para gerenciar consultas situacionais. Os métodos variam desde *mashups* de serviços até abordagens baseadas em cruzamento de dados. No entanto, lidar com a incerteza na descoberta automática de dados e garantir a relevância e a integralidade da informação continuam a ser desafios chaves. Uma forma de minimizar esses desafios é capturar o conhecimento do usuário para resolver ambiguidades e melhorar a recuperação de bases similares. A presente tese propõe uma arquitetura conversacional de Raciocínio Baseado em Casos (CBR), que visa melhorar o gerenciamento de dados situacionais, incorporando feedback do usuário no processo através de um agente conversacional. Para incluir um mecanismo de aprendizagem adaptável a feedbacks positivos e negativos, foi utilizada a metodologia de Raciocínio Baseado em Casos, a qual resolve problemas utilizando ou adaptando soluções passadas. A abordagem aproveita uma base de conhecimento histórica que é atualizada dinamicamente com base no feedback do usuário, permitindo um sistema mais responsivo e adaptável. Este feedback desempenha um papel crucial na recuperação, revisão e retenção de casos dentro do ciclo CBR, permitindo que o sistema evolua com base nas interações do usuário. Cada fase do CBR é retratada na presente tese e avaliada em três experimentos diferentes, focados na recuperação de fontes, reutilização de soluções multidimensionais em uma aplicação chatbot e aprendizagem baseada em conhecimento histórico, respectivamente. Neste último, foi realizado um estudo empírico para avaliar o impacto do feedback dos usuários nas recomendações do sistema, incluindo cenários de testes estáticos e dinâmicos, focando em aspectos como visibilidade, suporte e utilidade. Os resultados destacaram uma preferência por recomendações influenciadas pelas contribuições dos participantes, indicando a eficácia da incorporação do feedback humano no processo de tomada de decisão. Como contribuição adicional, esta tese também demonstra uma incompatibilidade terminológica envolvendo abordagens de integração de dados on-the-fly, propondo uma nova terminologia e taxonomia para o gerenciamento de dados situacionais. Com base na taxonomia proposta, a arquitetura baseada em CBR também é avaliada em relação a critérios como Recuperação de Dados, Integração On-the-fly e Entrega de Dados. No geral, a pesquisa realizada contribui para a gestão de dados situacionais, ilustrando como uma estrutura conversacional baseada em CBR pode melhorar processos de integração e descoberta, e evidenciando o potencial para um maior desenvolvimento na área.

Palavras-chave: Integração de Dados Situacionais. Sistemas Conversacionais. Feedback humano. Aprendizagem incremental. Chatbots.

ABSTRACT

Enhancing user support in data integration demands addressing real time information, a challenging process for conventional database systems. Many data integration variants integrate data "on-the-fly" for managing situational queries, i.e., queries that cover dynamic requirements. The methods range, for example, from service mashups to traversal-based approaches; however, the uncertainty in automatic data discovery and ensuring the relevance and completeness of information remain as key challenges. One way to minimize these challenges is by capturing user feedback, since it can help solving ambiguities and improving matching tasks. This thesis introduces a conversational Case-Based Reasoning (CBR) architecture, aimed at improving situational data management by incorporating user feedback into the process. The core of the architecture is a "human-in-the-loop" approach implemented through a conversational agent, which facilitates interaction between the user and the system. For including a learning mechanism adaptable to both positive and negative feedback, the Case-Based Reasoning methodology was used, which solves problems by using or adapting solutions from previous cases. The CBR-based approach leverages a historical knowledge base that is dynamically updated based on user feedback, allowing for a more responsive and adaptive system. This feedback plays a crucial role for case retrieval, review, and retention within the CBR cycle, enabling the system to evolve based on user interactions. Each CBR phase is depicted in the present thesis and evaluated in three different experiments, focused on source retrieval, reuse of multidimensional solutions within a chatbot application, and incremental learning based on historical knowledge, respectively. In the latter, an empirical user study was conducted to assess the impact of user feedback on system recommendations, including both static and dynamic test scenarios, and focusing on aspects such as visibility, support, and usefulness. The results highlighted a general preference for recommendations that were influenced by user input, indicating the effectiveness of incorporating human feedback in the decision-making process. As an additional contribution, this thesis also demonstrate a terminology mismatch involving on-the-fly data integration variants, proposing a new terminology and taxonomy for managing situational data. On the light of the proposed taxonomy, entitled Built-up Integration, the CBR-based architecture is also evaluated regarding Data Retrieval, On-the-fly Integration, and Data Delivery features. Overall, the research conducted contributes to situational data management by illustrating how a conversational CBR framework can improve processes such as data integration and data discovery, and evidencing the potential for further development in this area.

Keywords: Situational Data Integration. Question Answering Systems. Human Feedback. Incremental Learning. Chatbots.

LIST OF FIGURES

1.1 CBR cycle [1]. 15
1.2 Concepts weaknesses and strengths. 16

2.1 SDI general amplitude in the surveyed studies.. 21
2.2 Timeline of SDI in QA approaches. 25

3.1 Taxonomy for Built-up Integration 35

4.1 Functional view of the Conversational CBR approach for Built-up Integration Support. 43
4.2 Case retrieval from history. 44
4.3 Source Discovery Model. 47
4.4 Term-Frequency Vector example. Adapted from [98]. 48
4.5 Illustrative example of LDA. Adapted from [16].. 50
4.6 Reuse Activity Flow. 51
4.7 Case Repairing Flow. 54

5.1 Matching results with different methods. 63
5.2 Comparison between all approaches - third evaluation round 64
5.3 Overview of the hybrid approach based on Cosine Similarity and LDA-W2V. . . 65
5.4 Source selection accuracy with blended LDA-W2V-Cosine (1st evaluation round). 67

6.1 Chatbot state flowchart for proposing a solution. 70
6.2 Proposing solutions (Chatbot execution example). 74
6.3 Overview of the chatbot evaluation by categories. 77

7.1 Chatbot state flowchart (CBR-based overview). 87
7.2 Managing attributes (Chatbot execution example - Part 1). 89
7.3 Managing attributes (Chatbot execution example - Part 2). 90
7.4 Handling complementary data (Chatbot execution example - Part 3). 91
7.5 Comparison between static (a) and dynamic (b) evaluations. 100
7.6 Comparison between linear scale questions in static and dynamic evaluations. . . 102
7.7 User feedback for Data Integration questions. 105

A.1 Ad-hoc Data Retrieval in Question Answering systems. 131
A.2 Data Management in Question Answering systems. 137
A.3 Timely Decision Support in Question Answering systems. 142

LIST OF TABLES

2.1 Situational Data Integration Features 19
2.2 Preprocessing Techniques in QA-based Approaches 22
2.3 Response Time Improvement (RTI) Techniques in QA-based Approaches 22
2.4 SDI features fully covered (✓), partially covered (❇), and not covered (✗) in QA-based approaches 30

3.1 Comparison between different types of integration. 34
3.2 Built-up Integration examples in the literature 38

4.1 Metadata from candidate sources example and matches based on value overlap. . 47

5.1 Examples of Test Questions (English Translation) 58
5.2 LDA-W2V models 61
5.3 Accuracy for different questions (Q) and sources (S). 62
5.4 Source selection accuracy in three evaluation rounds with different sources (S) and questions (Q).. 66

6.1 Chatbot Evaluation Form 75
6.2 Score Percentage for Linear Scale Questions (1/disagree - 5/agree) 76
6.3 Participants Suggestions for the Chatbot 78

7.1 Initialization of static history and dynamic storic (snippet). 82
7.2 Example of history file management.. 89
7.3 Chatbot Evaluation Form (Review and Retain steps) 93
7.4 Dynamic log file (snippet). 96
7.5 Dynamic history (snippet). 97
7.6 Static log file (snippet). 99
7.7 Score Percentage and Average for Linear Scale Questions (1/disagree - 5/agree) . 101
7.8 General feedback for the chatbot prototype (highlights) 104

A.1 SDI's Ad-hoc Data Retrieval Requirements 130
A.2 SDI's Data Management Requirements. 135
A.3 SDI's Timely Decision Support Requirements 141

LIST OF ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
BI	Business Intelligence
BIOD	Blended Integrated Open Data
CBR	Case-Based Reasoning
CQA	Community Question Answering
CSV	Comma-Separated Value
d'	Dimensions chosen by the user based on **RD** set
DBMS	DataBase Management System
DW	Data Warehouse
ETL	Extract-Transform-Load
GPT	Generative Pre-trained Transformer
KB	Knowledge Base
KG	Knowledge Graph
LDA	Latent Dirichlet Allocation
LLM	Large Language Model
LOD	Linked Open Data
m'	Metrics chosen by the user based on **RM** set
ML	Machine Learning
NL	Natural Language
NLP	Natural Language Processing
QA	Question Answering
RD	Related Dimensions (CBR's Reuse phase) or Recommended Dimensions (CBR's Review phase)
RM	Related Metrics (CBR's Reuse phase) or Recommended Metrics (CBR's Review phase)
SDI	Situational Data Integration
VQA	Visual Question Answering
W2V	Word2Vec ("Word to Vector" embeddings)

CONTENTS

1 INTRODUCTION 13
1.1 RESEARCH HYPOTHESIS AND OBJECTIVES 15
1.2 CONTRIBUTIONS 17
1.3 THESIS STRUCTURE 17
2 SITUATIONAL DATA INTEGRATION IN QUESTION ANSWERING SYSTEMS: A SURVEY OVER TWO DECADES 18
2.1 SITUATIONAL DATA INTEGRATION (SDI) 18
2.1.1 Visualizing SDI: A Practical Example in QA Domain 20
2.1.2 Searching for SDI in QA Systems 20
2.2 TRENDS AND PATTERNS OF SITUATIONAL DATA INTEGRATION 21
2.2.1 Timeline Evolution 23
2.3 RESEARCH GAPS AND OPPORTUNITIES 26
2.3.1 SDI Features in QA 26
2.3.2 The Recent Years 27
2.3.3 A Terminology Mismatch 28
3 BUILT-UP INTEGRATION: A NEW TERMINOLOGY AND TAXONOMY FOR MANAGING INFORMATION ON-THE-FLY 31
3.1 BEYOND SDI: OVERVIEW OF OTHER DATA INTEGRATION VARIANTS . 31
3.1.1 Data Mashups 31
3.1.2 Traversal-based Integration 32
3.1.3 Pay-as-you-go Integration (Dataspaces) 33
3.1.4 Compiling Concepts 33
3.2 REORGANIZING THE KNOWLEDGE: BUILT-UP INTEGRATION 34
3.3 WHERE WE CAN FIND BUILT-UP INTEGRATION? 37
3.3.1 The Main Features 37
3.4 CHAPTER REMARKS 40
4 CONVERSATIONAL CBR ARCHITECTURE FOR SITUATIONAL DATA MANAGEMENT 42
4.1 RETRIEVAL: HANDLING A NEW CASE 44
4.1.1 General Knowledge (History of Previous Cases) 44
4.1.2 Source Discovery 46
4.2 REUSE: SOLUTIONS BASED ON DIMENSIONAL DATA 51
4.2.1 Differentiating and Recommending Data Attributes 51
4.2.2 Question Answering Prototype: Chatbot Application 52

4.3 REVIEW AND RETAIN: REPAIRING A CASE BY USER ACTION 54
4.3.1 Case Repairing Flow 54
4.3.2 When the Answer is Not Enough 55
4.3.3 The Learning Step. 56
5 EVALUATING THE DISCOVERY OF INFORMATION 57
5.1 IMPLEMENTATION DETAILS 57
5.1.1 Candidate Sources and Competency Questions 57
5.1.2 Methods Setting. 59
5.2 RESULTS 61
5.2.1 Cosine Similarity, LDA-W2V, and U-Sem Test: Comparative Evaluation 62
5.2.2 Blending Topic-Based Embeddings and Cosine Similarity. 64
5.3 CHAPTER REMARKS. 67
6 EVALUATING THE REUSE OF SOLUTIONS. 69
6.1 IMPLEMENTATION DETAILS 69
6.1.1 Chatbot Conversation Flow 69
6.1.2 Database Characterization 71
6.1.3 Natural Language Interaction and Error Handling 72
6.1.4 Chatbot Running Example 73
6.2 RESULTS 74
6.3 CHAPTER REMARKS. 77
7 EVALUATING THE REVIEW AND LEARNING OF A CASE. 80
7.1 IMPLEMENTATION DETAILS 80
7.1.1 Handling the History Retrieval 81
7.1.2 Source Discovery and Attributes Recommendation 84
7.1.3 Chatbot Conversation Flow 85
7.1.4 Chatbot Running Example (Attributes Management) 88
7.1.5 Chatbot Running Example (Handling Complementary Data) 89
7.2 RESULTS 91
7.2.1 Dynamic History Evaluations. 94
7.2.2 Static History Evaluations 98
7.2.3 Analyzing Questions by Category 100
7.2.4 Built-up Integration and General Feedback. 103
7.3 CHAPTER REMARKS. 105
8 CONCLUSIONS. 107
REFERENCES 109

APPENDIX A – SITUATIONAL DATA INTEGRATION IN QA SYSTEMS 130
A.1 AD-HOC DATA RETRIEVAL IN QA 130
A.1.1 Ad-hoc questions 130
A.1.2 Source Discovery 131
A.1.3 Ad-hoc questions & Source Discovery 133
A.2 DATA MANAGEMENT IN QA 135
A.2.1 Unstructured Data Processing. 135
A.2.2 Situational Source Inclusion 137
A.2.3 Unstructured Data Preprocessing & Situational Source Inclusion 139
A.3 TIMELY DECISION SUPPORT IN QA 141
A.3.1 Decision-Making Support. 141
A.3.2 User Guidance 142
A.3.3 User Guidance & Decision-Making Support 143
A.3.4 Response Time Improvement 143
A.3.5 Decision-Making Support & Response Time Improvement 144
A.3.6 User Guidance & Response Time Improvement 145
A.3.7 User Guidance, Decision-Making Support & Response Time Improvement. . . . 146
APPENDIX B – PUBLICATIONS 147

1 INTRODUCTION

The last years have marked a paradigm shift in access to information, where the *Big Data* concept raised the need to deal with large data volumes in various formats made available on the web [51, 186]. Exploiting such large amounts of data makes the search for relevant information a truly complex and time-expensive task: the difficulties lie at different levels including data capture, storage, sharing, analysis, management, and visualization [186, 21].

To extract the full value of Big Data, solutions involving information retrieval, Natural Language Processing (NLP), and Artificial Intelligence (AI) have been investigated for capturing useful information and use it in diverse domains [203]. There is also a concern with the user's standpoint and experience, as the captured information should become valuable knowledge to satisfy their specific needs [79]. Obtaining such valuable insights and actionable knowledge is not a trivial task, as it requires cross-analysis of data coming typically from multiple sources [122]. In other words, satisfying user's needs often demands some form of data integration [59].

Data integration aims at joining data from different sources and providing users with a unified view [142]. In fact, useful insights can be extracted from integrated data, allowing to improve services or make smarter decisions in several areas [35]. In the area of Business Intelligence (BI), for example, integrated information can be used for obtaining competitive advantages and leverage intraorganizational collaboration [123], while in Open Data, it favors governmental transparency [170]. Usually, data integration involves some structured local data and different ETL (Extract-Transform-Load) flows, which are periodically executed for adding external information of interest [123].

However, applications are no longer limiting their analysis to structured databases, but they are demanding actionable information for attending specific needs and improving decision making [243]. In other words, when decisions must be made based on real-time analyzes, it may be impractical to wait for information only when ETL flows are executed. In such cases, the integration should be performed **on-the-fly**, i.e., with data discovered, extracted, and merged *at query time*, so the information can become actionable [180]. Assuming that data can scattered over multiple sources, on-the-fly integration makes it possible to provide the right information, at the right time [182].

One umbrella concept related to on-the-fly integration is the **Situational Data Integration** (SDI), which is managed at query time in order to deal with specific and dynamic query requirements that are challenging for traditional database management systems and search engines [154, 99, 244]. Specifically, SDI integrates up-to-date external data with local data for inferring solutions that enable accurate decisions [29, 244, 178]. This means that if a question cannot only be answered with the current data, a SDI system might *discover* relevant sources that contain the data needed. These data are called *situational data*, i.e., data that are not owned by the user and have the role of providing a complete answer to a *specific* problem or need [244].

Situational challenges can be handled by many ways, e.g., by using service mashup tools which combine data from different data sources into a single joint place [213, 248], or through traversal-based approaches, which handle the source discovery process [103]. However, an existing challenge in these situational applications is the uncertainty when discovering data with automatic matchers [147, 81]. Purely automated methods cannot fully address this issue, and beyond that, they infer the best way to integrate data sources based on the features of these sources, which can output several errors [59]. A second challenge refers to ensuring that information acquired from discovered and integrated data sources is in fact useful and complete to the user, i.e,

once situational data are retrieved and returned, the user should be capable of deciding whether they are suitable for the task at hand [3]. If they are not, the user should be provided with a mechanism that learns from mistakes and improve future recommendations.

One way to minimize errors and uncertainties when manipulating information from several data sources is by *capturing user feedback*, since integrating human knowledge in the loop helps solving ambiguities, improving matching tasks, and even generating high-quality rules for a system operation [142]. In cases where a system makes a recommendation, the user feedback can help refine the results and fine-tune data models. This can be exemplified trough many studies in the literature that use human feedback [228, 187, 273]. But despite the benefits that human inclusion can bring to a system's operation, when we are seeking appropriate solutions for situational problems such as the ones mentioned before, two main questions arise: How to collect user knowledge? And most importantly, how to use the collected knowledge to continuously improve the use (discovery, integration, or recommendation) of data coming from multiples sources?

For addressing the first question, conversational interfaces[1] such as **Question Answering** (QA) systems can be applied. QA systems aim at providing a precise answer for a specific question [21, 71], and are currently integrated in virtually all personal assistants (e.g., Siri, Cortana, Alexa Echo, and Google Home) [203]. By using a QA system as interaction and mediation tool, human knowledge can be detected and used for improving many tasks within the system operation [142]. Also, besides capturing user feedback, QA systems are user-friendly solutions that could be used to access data sources and guide a data integration process, which are usually restricted to experienced computer users [188]. They act as an access point to data sources, obtaining fast results and delivering them in Natural Language (NL) [28, 243].

QA systems have proven to be good solutions for capturing human intentions [259, 193, 268, 165]. However, a QA system by itself does not address the second question on how to use the collected knowledge to continuously improve situational data integration tasks. Let us consider the source discovery challenge mentioned before in a QA scenario: if a QA system has the role of discovering sources based on the posed question, how could it leverage the user feedback to improve further recommendations? For that, it would be necessary to have a mechanism that learns from failure and success cases, i.e., negative and positive feedback. Thus, when the feedback is positive, the recommendation is reinforced, otherwise, a more suitable source is retrieved and presented next time.

One learning mechanism that suits this type of situation is the **Case-Based Reasoning** (CBR) [1], which is a paradigm of Artificial Intelligence (AI) that solves problems using or adapting solutions to old problems [200]. This methodology draws on human reasoning for problem solving, since in real life, once we have learned a solution to a problem, we often try to reuse that solution in similar problems [223]. The problem solving in the CBR methodology goes through a cycle of four activities demonstrated in Figure 1.1: *retrieving* cases that resemble the description of the problem, *reusing* an existing solution for a similar case, *reviewing* this solution in order to meet the new problem, and *retaining* this solution once it has been confirmed [253].

There are many advantages of applying CBR: the knowledge acquisition task is reduced, there is flexibility in knowledge modeling, it is possible to reason when domains are not completely understood or when data are imprecise, the cumulative experiences allow learning successes and failures over time, thus mistakes made in the past can be avoided [196]. In addition, the methodology is compatible with other AI (Artificial Intelligence) methods, so that systems that rely on CBR can take advantage of several Machine Learning (ML) techniques to improve their

[1]Conversational interfaces are those that provide the front-end to a virtual assistant, allowing the user to interact with the app using text, speech, touch, and other input methods [167].

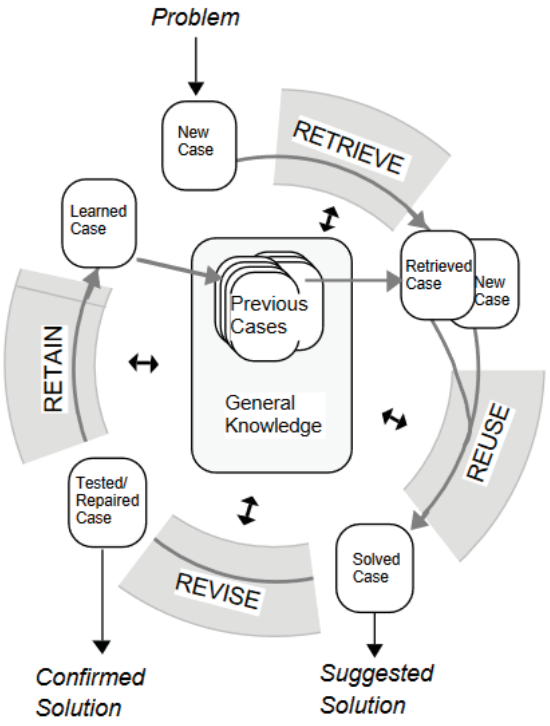


Figure 1.1: CBR cycle [1].

performance [6]. Most importantly, due to its four-activity cycle, CBR has potential to cover some gaps observed in on-the-fly and situational approaches, as illustrated in Figure 1.2.

First, as CBR covers a *review* step which mostly involves human knowledge, it can be used to attenuate difficulties when finding a suitable data source through automatic matchers. This human inclusion can be used to directly measure whether additional and/or situational data is useful (either for a personal task at hand or existing data that needs to be complemented). Also, considering that on-the-fly integration inefficacy impairs the information delivery [222], it is necessary to pursue a continuous improvement strategy, which can be provided by a learning-oriented methodology such as CBR. Finally, decision support is a common goal in situational approaches, but is still a key challenge (more details in Chapter 2). In contrast, CBR supports interactive explanations to the users, including both the proposed solution and the reasoning process [85].

Conversational interfaces are also compatible with CBR, as they are user-friendly solutions that favor explainability. In fact, several conversational CBR approaches have been proposed in the past, as they allow the user to inform problem descriptions step by step, facilitating the use of the methodology for problem solving [108, 23]. The explainability is also linked to the type of reply provided by conversational interfaces (e.g., QA systems), since the user is usually looking for a concise and fast information. Thus, explaining the solution and the strategy to the user, in a simple way, can help in decision processes. These characteristics make a conversational CBR approach a promising combination for handling situational tasks, which are dependent of the user immediate needs.

1.1 RESEARCH HYPOTHESIS AND OBJECTIVES

Considering the challenges raised in the previous section and the strengths observed in the mentioned concepts, the central hypothesis raised in this thesis is: *A conversational system that*

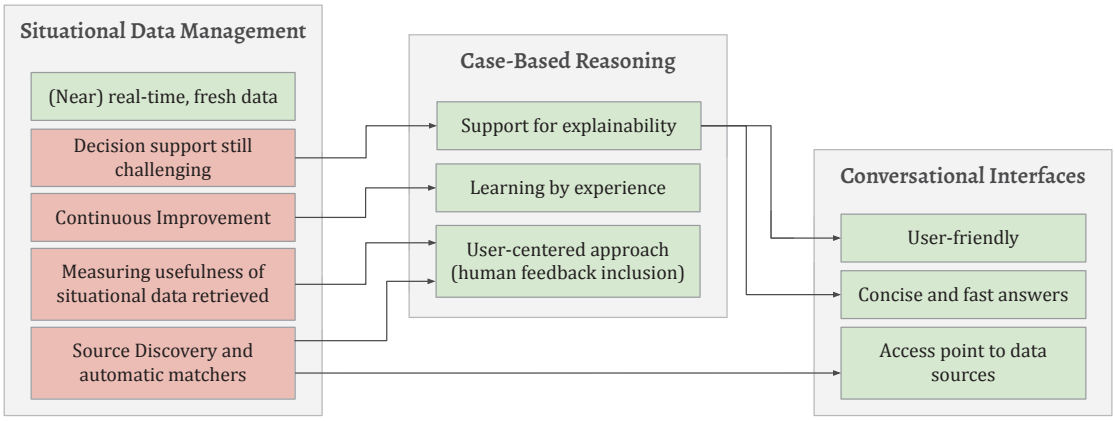


Figure 1.2: Concepts weaknesses and strengths.

captures user feedback and executes Case-Based Reasoning as feedback-based methodology for learning can improve data source management in situational contexts and provide a better user support.

This hypothesis considers that the concepts have correlated features that, when executing together, could potentially benefit the end user when it comes to querying situational data. In a particular way, it is based on the belief that human knowledge can be used within the CBR cycle for incremental learning, i.e., if we can detect the knowledge from the human, then we can use the knowledge to optimize data integration tasks [142]. The inclusion of human knowledge in a system's operation is a hot topic, especially considering the emergence of robust technologies such as *ChatGPT*[2], which answers complex questions with human-level performance [145]. Although these approaches already exist, the focus of the present work is to gather human knowledge by means of a CBR methodology, focusing on how situational data is managed. Overtime, the improvement enabled by user feedback can to compensate a possible lack of quality when retrieving situational data, inherent in the use of automation [37, 81].

Following the aforementioned motivations, this thesis has the main goal of modeling and evaluating a conversational approach based on CBR for improving data source management and recommendation. To achieve this objective, a set of specific activities were established:

- Investigate how situational integration (i.e., SDI) is handled in the related literature, specially in Question Answering domain, in order to identify main features, challenges, and opportunities;
- Model an architecture of conversational system which executes a Case-Based Reasoning cycle and tackles some of the challenges identified in the first step;
- Propose and model novel activities that should be executed in each of the four phases of the CBR cycle (Retrieve, Reuse, Revise, and Retain);
- Develop a QA-based prototype that bases on the architecture previously defined, and interacts with the user to gather his/her feedback, which should be later used for learning within CBR's *Retain* step;
- Evaluate the QA-based prototype respecting the recommendation of solutions to the user (*Reuse* step);

[2]Available at: <https://openai.com/chatgpt>.

- Evaluate the QA-based prototype respecting the use of human feedback for continuous improvement (particularly in data source management context);
- Confirm the research hypothesis previously established, which is based on the joint use of so far independent concepts;
- Publish reliable results in the scientific community.

1.2 CONTRIBUTIONS

As main contributions, this thesis:

- Presents a survey of situational integration in QA systems which covers two decades of related literature;
- Proposes a taxonomy that addresses the lack of consistency regarding situational integration and other types of on-the-fly integration;
- Proposes a conversational CBR-based architecture which uses human knowledge for a continuous learning process, and the multidimensional data model as strategy for situational data management within a conversational interface;
- Presents a QA prototype that executes the architecture steps, discovering and recommending data sources in situational contexts;
- Evaluates the CBR-based architecture through three experiments, covering source discovery tasks, the reuse of multidimensional solutions, and the impact of user feedback for future recommendations;
- Details evaluation tasks supported by real users and covering qualitative criteria observed in consolidated studies from the literature;
- Demonstrates positive results in evaluations shaped by user feedback, evidencing the value of the proposed CBR architecture, as well as the potential for expanding research and development in the area;
- Shares the knowledge generated along its development, through scientific publications mentioned in Appendix B.

1.3 THESIS STRUCTURE

The structure of this thesis is presented as follows. In the next chapter, a survey on Situational Data Integration in Question Answering systems is presented, showing the connection between the concepts and the lack of situational data management in more recent approaches. In Chapter 3, a novel terminology and taxonomy is proposed, aiming at organizing similar characteristics observed in data integration variants named differently. Based on the gaps and opportunities previously mentioned, the Chapter 4 proposes a conversational approach based on the CBR methodology, for promoting incremental learning and improve situational tasks. The architecture is evaluated through three distinct experiments, which are detailed and discussed in chapters 5, 6, and 7, respectively. The final considerations of this thesis are presented in Chapter 8.

2 SITUATIONAL DATA INTEGRATION IN QUESTION ANSWERING SYSTEMS: A SURVEY OVER TWO DECADES

Question Answering (QA) is a well known research field in computer science, with the first QA systems designed in the 1960s [51]. At that time, these systems were limited to consulting databases within a restricted domain [257, 90] and the analysis patterns were built manually, so the questions could be translated into a structured query form needed to interrogate the databases [132]. In contrast, the current years have shown a huge increase in the amount of information available on the Internet, so that QA systems are no longer limiting their analysis to structured databases, but they are demanding *actionable* information for attending specific needs [243]. Advances in NLP and Artificial Intelligence have powered these systems with more comprehensive interactions and effective analysis, driving QA-based proposals in many areas, such as Business Intelligence [70], healthcare [28], and Open Data [254].

The fact that QA systems have presented an increasingly interactive nature (e.g., conversational settings, user feedback, and interpretability) and need to flexibly integrate components to fulfill specific tasks [203, 221] makes situational data eligible for providing timely and complete answers. In other words, in cases where a single source is not enough to answer the question, a situational integration (SDI) could be triggered on-the-fly to obtain and validate answers from multiple sources.

Despite that, situational and on-the-fly data integration is a big challenge in the literature [180, 244, 235, 161], which may be an indication that its inclusion remains a gap in QA-based systems. Most importantly, considering the main goal of this thesis (improving situational data management by means of human knowledge collected and used for continuous learning), the understanding of the current scenario and the relation between the QA domain and situational approaches is highly desirable. By systematically investigating the association between the concepts, it is possible to identify consolidated features and improvement points that can make way for the conversational CBR architecture targeted in this research work.

Following the aforementioned motivations, this chapter surveys Situational Data Integration in Question Answering systems, covering two decades of research, aiming to establish the state-of-art of data retrieval, data management, and decision support in these conversational systems. As survey results, a timeline was constructed for highlighting the most prominent SDI features in each time range. The survey also clarifies that some features are moving towards consolidation in the QA domain (as they are often explored in the literature), while most of them remain challenging in QA approaches.

2.1 SITUATIONAL DATA INTEGRATION (SDI)

Situational Data Integration (SDI) is an integration that uses data sources discovered on-the-fly for dealing with specific and immediate queries and their dynamic requirements [99, 247]. The term "*Situational*" in SDI comes from the concept *Situation-Awareness* (SA), which is the perception of events related to an entity (i.e., the user) and the understanding of what is going on around, allowing to make accurate decisions [126]. By following this concept, a Situational Data Integration may be understood as an integration oriented by *situations of interest* to the user, providing information that are at the basis of decision-making.

Although SA is a long-established concept [60], situational integration only gained focus years later, in the *Business Intelligence* (BI) domain, where SDI was initially investigated

due to its impact on operational decisions [154]. It came as an alternative to the traditional integration settings[1], aiming to analyze and combine large data sets (comprising both structured and unstructured data) for dealing with dynamic requirements and data-intensive flows [154, 123].

The growing interest in obtaining and using real-time information motivated several BI-related studies to present the characteristics of SDI [154, 244, 29, 99, 3, 247, 123, 7]. Based on the related literature, SDI is composed by three main features: *Ad-hoc Data Retrieval, Data Management*, and *Timely Decision Support*. Each feature, in turn, is associated with subfeatures, i.e., specific tasks that are involved in SDI, as demonstrated in Table 2.1.

Table 2.1: Situational Data Integration Features

SDI Main Features	**Subfeatures**
Ad-hoc Data Retrieval	Ad-hoc Questions and Source Discovery
Data Management	Unstructured Data Preprocessing and Situational Source Inclusion
Timely Decision Support	User Guidance, Decision-making Support, and Response Time Improvement

Ad-hoc Data Retrieval refers to the abilities to retrieve suitable sources of information for dealing with query requirements. It relies on *Ad-hoc Questions* and *Source Discovery*. Frequently, ad-hoc questions involve situational data, which have a narrow focus on a specific domain problem or a unique set of needs [3, 154]. Situational data are usually external to an organization control and hence without guaranteed access [123]. Thus, in Source Discovery, the goal is to find a provider for data not available internally, as well as systematically analyze the different potential sources of information at hand for attending specific and dynamic requirements [247, 3]. When data integration requests are changed or new requests are submitted on-the-fly, a new round of integration should be quickly performed to meet the new requirements [99]. Thus, we assume that a situational integration involves dynamic data, coming from changeable sources.

Another main feature of SDI is **Data Management**. This feature refers to how an SDI system deals with unstructured and heterogeneous information from a discovered source, which is an essential step to ensure an effective situational integration. Two subfeatures are involved, named *Unstructured Data Preprocessing* and *Situational Source Inclusion*. Situational data are often scattered across heterogeneous and unstructured sources available on the Web [3, 244, 7], so, the integration has to tackle the fact that data sources usually contain erroneous, out-of-date, or conflicting data [176]. *Unstructured Data Preprocessing* deals with these challenges, making the integration of the extracted information feasible. After the processing, situational data needs to be integrated with the current information in order to extend the information previously available and completely satisfy users' specific needs [244, 154, 3, 19, 178, 56]. This often requires a correlation analysis process for assessing whether the situational source is suitable for the task at hand [154].

The last main feature of SDI is **Timely Decision Support**. A situational picture is necessary to go beyond the simple perception of the elements in the environment, supporting the overall comprehension of the current situation and the user's decision making process [19, 56]. Thus, a situational data integration system provides *Decision-making Support* if it provides information for alerting the decision makers of situations that can potentially affect their activities. The support can also be achieved, for example, if the system leads the user through the sequence of steps he has to apply, subsequently providing explanation of how resolutions were made, or why a certain operator was selected [214]. Situational projects also require just-in-time delivery

[1]Based on the survey presented in [123], we refer to "traditional integration settings" as those where data sources and query requirements are static, demanding a regular update of the data repository rather than real-time data flows.

of good-enough solutions that are not addressed by traditional offerings [7], so *Response Time Improvement* is a highly desirable feature. Useful techniques involve, e.g., provide new compute nodes as needed, reduce data complexity for processing, or distribute computing tasks through parallel computing [154]. By providing timely insight into situations, an appropriate action can be taken in real time [29]. Concerning *User Guidance* subfeature, situational applications may involve active human participation when solving specific needs [99, 154]. With human feedback, for instance, the system is able to optimize its processes and responses.

After presenting the main features of SDI, the next subsections present a practical example of SDI in QA domain and the methodology applied for conducting the review, respectively.

2.1.1 Visualizing SDI: A Practical Example in QA Domain

We can reason about SDI features in QA taking as example a scenario involving speech therapy, since this domain has been explored in situational contexts [75]. Regarding Ad-hoc Data Retrieval, suppose that a speech-language pathologist wants to know whether the government investment in education affects children's speech abilities. The professional owns a stationary database that contains information about the patients (including their age, schooling, region), but this database does not include any educational data associated with the regions, which would be necessary to address the initial information need. So, he/she poses an ad-hoc question as input to a QA system (either by natural language text or voice), and the system triggers a discovery process in Web sources. Then, the system finds several open databases in official government portals that provide data related to education, e.g., data on schools performance in national exams and educational indicators. By verifying patterns in the question that match each candidate source, the system selects the one that best fits the information need: a government open data source that describes investments in basic education, which could provide a complete answer to the professional's question.

Regarding Data Management subfeatures, consider that the selected source contains raw situational data that are difficult to process and analyze. So, a preprocessing step is performed in order to obtain a smaller set of data that are relevant and applicable for the query. After this process, the next step is to combine the situational data with the stationary data (i.e., the patients data owned by the professional), by using correlation techniques that semantically map both databases attributes. The result should be an augmented set of data that allows to identify all data patterns between educational and clinical data.

Concluding the speech therapy example with Timely Decision Support, the QA system now has a specialized data set where relations between educational data and patients data can be extracted. Hence, the system can use ML and NLP techniques to rapidly infer these relations and present some suggestions to the professional through a graphical interface, where the user can also correct eventual wrong matches. During the interaction, the system could highlight a relevant pattern found in the augmented data set, allowing the professional to infer that "most patients with lower pronunciation performances are from regions with lower educational investments". This answer makes the speech-language pathologist better informed while saving time, and it can be used to guide the therapeutic planning. In other words, the SDI+QA system allowed the therapist to have a broader view on the patient's case and make appropriate decisions based on current context.

2.1.2 Searching for SDI in QA Systems

For adequately investigating the state-of-art of SDI in QA systems, keywords "Question Answering", "data integration", and "multiple data sources" were used to selected QA-based studies.

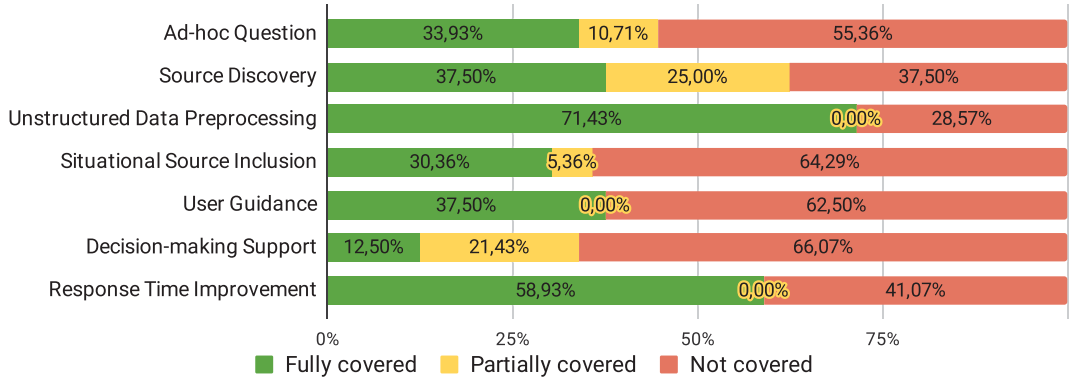


Figure 2.1: SDI general amplitude in the surveyed studies.

Based on these keywords, studies from the last two decades were covered[2], thus enabling to track trends in time ranges (in terms of methods and data sources), as well as to raise hypotheses about future developments. Scopus search engine[3] and Google Scholar[4] were used as data sources, since they include other digital libraries such as ACM Digital Library[5], IEEE Xplore[6], and Science Direct[7].

For assessing the presence of SDI in the surveyed studies, the features shown in Table 2.1 were taken as basis. Specifically, given the particularities of the QA approaches, the survey presents how each SDI feature is present in the studies, assessing whether its coverage occurs in a *complete or partial way*. More than 50 studies in QA domain were surveyed, with respect to SDI features. **All papers and features coverage aspects are detailed in Appendix A**. The analyzes and discussion are provided in the next section.

2.2 TRENDS AND PATTERNS OF SITUATIONAL DATA INTEGRATION

This subsection presents analysis regarding SDI in QA domain. In a first moment, the overview of SDI features in the area and methodological aspects are presented. Next, a timeline of SDI in QA is shown, which organizes approaches and features overs two decades. Based on the timeline, we discuss the evolution of SDI and trends in each period of time. The results are summarized in Figure 2.1, and a detail description of all SDI features covered in each QA approach is given in Table 2.4.

The features coverage in Figure 2.1 shows that *Unstructured Data Preprocessing* and *Response Time Improvement* were addressed by most of the studies (71,43% and 58,93%, respectively). Preprocessing is desirable in a SDI-based system, since situational data are often spread over heterogeneous and unstructured sources and need to be accessible for efficiently supporting the user. A summary of Preprocessing techniques in the surveyed papers is presented in Table 2.2. Among the studies that presented *Unstructured Data Preprocessing*, most deal with duplications and ambiguities in the data. Resolution of entities - as well as relations between them - are also handled by a considerable number of studies. As regards *Response Time Improvement*, almost 60% of the studies cover this feature to speed up the processing and delivery of responses to the user. As shown in Table 2.3, indexing methods are the most used ones for data retrieval

[2]Priority was given to studies published between 2010 and 2020.
[3]Available at <https://www.scopus.com/>.
[4]Available at: <https://scholar.google.com.br/>.
[5]Available at: <https://dl.acm.org/>.
[6]Available at: <http://ieeexplore.ieee.org/Xplore/home.jsp/>.
[7]Available at: <https://www.sciencedirect.com/>.

and, in this category, it is interesting to highlight that *Lucene* tool [166] is applied in more than half of QA approaches using indexing.

Table 2.2: Preprocessing Techniques in QA-based Approaches

Preprocessing Techniques	**Application**
Data conversions	[70, 245, 237, 158]
Filtering	[73, 38, 40]
Normalization	[70, 163, 66]
Stopwords removal	[252, 246, 63, 156, 24]
Noise removal/cleaning	[28, 152, 171, 239, 230] [158]
Resolution/exclusion of duplications and ambiguities	[177, 33, 28, 129, 107] [209, 152, 189, 96, 160] [77, 42, 24, 267, 230] [271, 119]
POS tagging	[155, 189, 171, 264]
Resolution of entities and relations	[155, 36, 73, 27] [189, 156, 264]
Tokenization	[246, 107, 27, 171]
Exclusion/pruning of irrelevant data	[246, 88, 119]
Completeness verification	[93, 152]
Semantic annotations	[177, 9, 96, 120] [38]
Stemming and lemmatization	[27, 63]

Table 2.3: Response Time Improvement (RTI) Techniques in QA-based Approaches

RTI Techniques	**Application**
Search indexing	[70, 73, 53, 189] [232, 63, 96, 264, 239] [120, 27, 267, 8, 158] [229, 107, 152]
Constrained data retrieval	[70, 129, 77, 72] [88]
Parallel processing	[218, 129, 12, 38] [119]
Query rewriting	[12]
Graph-based optimizers	[36, 229]
Cache-based optimizers	[160, 12]
Complexity reduction	[177, 28]
Training acceleration (ML-based approaches)	[33, 229]
Redundancy removal	[129, 209]
Mathematical optimizer	[271]

Another prominent feature is *Source Discovery*, which represents the system' ability to find a provider for needed data on-the-fly, and process the data source according to new requests. As shown in Figure 2.1, more than 35% of the studies *fully* cover this feature, and 25% of the total *partially* includes it. This means that the majority of QA-based approaches (i.e., more than 60% of them) performs Source Discovery in some way. The use of a non-predefined source is an important requirement for Source Discovery, thus in the papers where the *partial* feature was covered, the sources were not predefined but they had a dynamic nature (e.g. they were updated, exchanged, or "switched off" based on the posed query). Almost all discovered sources are Web-based (including Websites, publicly available ontologies, and linked datasets), while Knowledge Bases (KBs) such as DBpedia, Wikipedia, and Freebase are also widely applied. The considerable amount of studies that partially covers *Source Discovery* indicates that this SDI feature is still missing in QA systems, but is being increasingly explored in a way that could become consolidated in future research.

All other SDI features shown in Figure 2.1, i.e., *Ad-hoc Question, User Guidance, Situational Source Inclusion*, and *Decision Support*, are integrated less often in the studies. The

main challenges lie in the last two mentioned, which means that QA systems do not usually include a situational source for augmenting the data available internally, and are not able to support the user in his actions or decisions. These characteristics allow us to establish some research opportunities with respect to SDI, which will be addressed in Subsection 2.3.

Considering the main features of SDI and how they are distributed in the reviewed papers, some characteristics stand out. When considering *Ad-hoc Data Retrieval*, the fact that most studies that cover *Ad-hoc Questions* also cover *Source Discovery* (Figure A.1) reinforces the idea that searching for a situational source is often triggered by a specific and momentary need, even though it might also occur in order to answer a conventional query (e.g., a benchmark query). With respect to *Data Management* (Figure A.2), besides the evidence of high coverage of *Unstructured Data Preprocessing*, we can observe that this subfeature is also covered by more than half of the studies that presented *Situational Source Inclusion*, demonstrating a concern with the treatment of the data before their effective use. Furthermore, in what concerns *Timely Decision Support* (Figure A.3), there is a strong correlation between *Decision-making Support* and *User Guidance*. This is because these subfeatures are more applied together than in isolation, indicating that authors concerned with developing decision support approaches will often take into account the orientation of the end user, e.g., which interventions he finds interesting or how the system can improve its assistance. In addition, *Decision-making Support* is mostly covered in the partial way, indicating that it is still explored at an early level, e.g., without explicit and accurate support.

Only two studies covered Situational Data Integration in a complete way (see Table 2.4): the framework for enriching Data Warehouses (DWs) with QA data by [70] and the integration-oriented ontology in [177]. Analyzing these works, we can acknowledge some similarities, such as the use of web-based external sources. The framework described in [70] uses QA for retrieving data from blogs and social networks, whereas [177] retrieves information from Web sources in the form of REST APIs (Application Programming Interfaces). All external sources contain unstructured data that complement the information of a structured source, i.e., a dataset [70] or an ontology [177]. Both approaches performed ontological mappings as correlation method when including the situational source. Regarding the query language, natural language queries are considered in [70], whereas [177] manipulates SPARQL queries.

2.2.1 Timeline Evolution

For investigating the evolution of SDI throughout the years, a timeline was constructed, exposing features that have been intensely researched in each time range. The timeline is shown in Figure 2.2. For constructing it, the most relevant papers were prioritized, i.e., those with more SDI features covered, besides their overall influence (based on number of citations). For space reasons, 40 studies from the total of surveyed papers were chosen for composing the timeline, inspired by the feature weighting approach in [231]. For each paper, a relevance score was calculated based on its coverage of SDI features and overall influence, as demonstrated in equations 2.1, 2.2 and 2.3.

$$MFScore(p,m) = \sum_{i=1}^{m_n} \left(\frac{p_{c_i}}{n}\right) \tag{2.1}$$

$$CScore(p) = \frac{p_{tc}}{(y+1) - p_{pub}} \tag{2.2}$$

$$TScore(p) = w_c \cdot CScore(p) + w_M \sum_{m=1}^{M} MFScore(p,m) \tag{2.3}$$

The Equation 2.1 calculates the main feature score $MFScore$ for a paper p, considering a given main feature m. m_n is the total number of subfeatures $\in m$ and p_{c_i} is the paper coverage of each subfeature, which may range from 0 to 10 where 10 means "fully covered", 5 is "partially covered", and 0 stands for "not covered". Not only SDI main features were considered for papers ranking, but also their relevance by means of citations per paper age. So, in Equation 2.2, the citation score $CScore$ is calculated for a paper p, where p_{tc} is the total citations of the paper[8], y is the current year (i.e. 2020), and p_{pub} is the paper publication year. Finally, Equation 2.3 determines the total score ($TScore$) of a paper p; it considers the $CScore$ weighted by w_c, added with the summing $MFScore$ for all main features M, considering w_M as its weight. The weight assigned for w_M was 0.3 and the assigned weight for citations w_c received 0.1. The top 40 papers with highest $TScore$ were chosen to compose the SDI timeline.

The timeline shows the most relevant QA studies along with *highlights*, i.e., SDI features that were frequent in each time range. From 2002 to 2005, the majority of the studies incorporated *Data Management*. This means that both preprocessing and situational sources were present in the proposals. Situational sources were included by merging components performing syntactic and semantic matchings [107, 9, 127], axiom-based theories [9, 245], contextual and lexical correlations [264], and pattern matching [40]. Situational sources are exploited again in 2010, and included using ensembles of matching algorithms and evidence-gathering techniques [73], as well as semantic mappings [237].

The first occurrence of *Timely Decision Support* was in 2007, with MedQA system [267]. This feature had the highest occurrence in 2011, and interestingly, all papers that covered it in that year and in 2007 were restricted-domain ones, specifically medicine [27, 28, 267] and biomedicine [120]. This fact demonstrates that decision support is being explored not only in areas that involves human action, but also where such human action is decisive (i.e., with low tolerance for errors), in a way that an assistive system can greatly impact the efficiency of the offered services.

In 2010 and 2011 there were several new approaches exploring SDI features, and also in these years, *Ad-hoc Data Retrieval* was present (at least partially) in almost all approaches. In this period, knowledge bases and ontologies stood out either as internal sources [73, 224, 237, 171, 232, 72, 28] or external sources [73, 237, 93, 171, 120, 27, 42, 28], and correlation methods were mostly based on semantic and ontological mappings in these studies. In addition, natural language and SPARQL queries were predominant in all studies excepting [120], where the search is performed using keywords.

Unstructured Data Preprocessing was predominant in all time ranges until 2019 and the most consolidated SDI feature, as in every time range there was a concern about handling noisy and raw data to obtain better system performance. The methods are diverse (consider Table 2.2): in the previous years, they were less present and more focused on filtering [40, 38], pruning resources [88, 119] and removing data duplications [129, 24, 119, 107]; as of 2007, NLP and semantic-based methods (such as POS tagging, tokenization, entity resolution, stopwords and noise removal) have gained focus.

Finally, Situational Data Integration was completely covered for the first time in 2016, in the framework proposed by [70]. It also occurred in 2017 [36] and 2019 [177]. However, there is a trend starting in 2010, since other studies which covered most part of SDI features - not necessarily the mandatory ones - were published after 2009 [73, 152, 27, 96, 237, 120, 158]. Of these studies, the greater part (i.e., $5/7$) was published between 2010 and 2012. In this time range, *Situational Source Inclusion* was still a challenging feature, whereas *Timely Decision Support* starts to become more frequent in the proposals. In this context, and considering the beginning

[8]The citations number exhibited in August 2020, on Google Scholar, was considered for each paper.

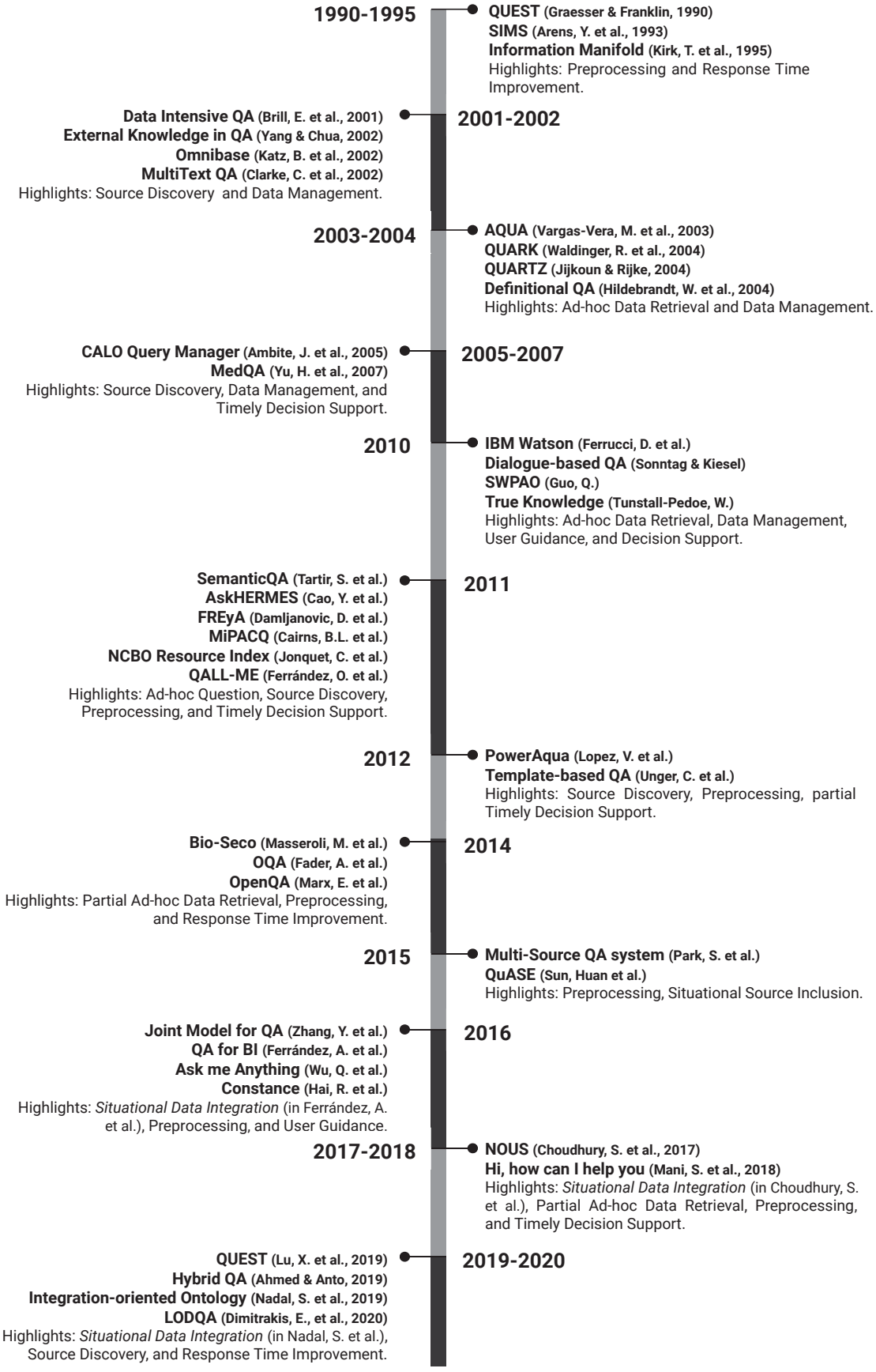


Figure 2.2: Timeline of SDI in QA approaches.

of SDI in the timeline, it is possible to infer that existing needs since 2010 were gradually being addressed as strategic decisions in several areas became more urgent.

2.3 RESEARCH GAPS AND OPPORTUNITIES

This section discusses some research opportunities identified with the survey analyzes. First, the SDI features that represent the greatest opportunities for development in QA domain are addressed, based on the surveyed studies. In a second moment, hot topics are discussed, considering studies published after 2020. Finally, a terminology mismatch is disclosed, involving the term Situational Data Integration and other on-the-fly integration approaches, demanding a proper organization of the related literature.

2.3.1 SDI Features in QA

Based on the previous analysis, this subsection exposes a set of characteristics that represent directions for further research. As stated in Section 2.2, some SDI features are moving towards consolidation in QA systems (such as *Unstructured Data Preprocessing, Response Time Improvement*, and *Source Discovery*), even though they still involve many aspects to be explored. Despite this fact, all other features are rarely covered (see Figure 2.1), e.g., *Situational Source Inclusion, User Guidance, Ad-hoc Question*, and *Decision Support*, which are discussed in the following paragraphs.

When analyzing *Ad-hoc Question* in Figure 2.1, we see that it is covered by more than 30% of the studies, but still this represents a low inclusion rate. In fact, a considerable part of the approaches includes benchmarks of complex questions, which are quite valuable for evaluation purposes, but since the questions are stored and reused, they do not reflect specific and momentary needs the same way as ad-hoc questions. It is essential that this momentary need is captured and managed in order to deliver dynamic and timely responses, as opposed to results that may suffer depreciation (i.e., lose meaning or value) over time. Also, we identified that, among the question benchmarks used by the reviewed proposals, there was no specific collection to evaluate SDI in QA. With recent advances in NLP and deep learning, it is necessary to introduce new datasets for testing the ability of language models to make inferences that rely on situation awareness [135]. Thus, comparative results involving SDI features in QA domain could be promoted.

With respect to *User Guidance* and *Decision Support*, although they are independent features, the former has great impact on the latter's efficiency. Consider, for example, the application of SDI in a clinical QA system (see Subsection 2.1.1): after integrating situational data with data owned by the therapist, the system can identify patterns and infer in which therapeutic activities these patterns impact. Knowing this information, the therapist can make appropriate decisions, considering the patient context. Although in this scenario *Decision Support* is beneficial, this support can be optimized with *User Guidance*, e.g., if the physician define portions of data that are of his interest, by including constraints or assigning weights to sentences involved in the query. Following the example of user feedback applied to source discovery and integration, here the guidance may also be a feedback from the professional: the manual correction of mistaken correlations made by the system, or a manifestation about the usefulness of the results. In both cases (instructions or feedbacks), *Decision Support* can be leveraged, seeing that past experiences can be used to train the system to give better responses.

Another way to leverage *Decision Support* could be investing more on user experience, through interfaces where the user can interact with results and recommendations. QA systems also favor this interactivity. Some reviewed studies [120, 163] expressed future goals which we

have taken as clues on how to strengthen the adoption of this feature: the former aims to improve the user interface, whereas the latter aims to guide the exploration of available data resources towards the one that are more appropriate for his needs. Both tasks can potentially improve decision support. Similarly, [252] wants to explore a daily report for generating informative sentences in a short period, which favors decision support in a timely manner.

Almost 65% of the studies did not cover *Situational Source Inclusion*. In the context of SDI, it is strongly connected to *Source Discovery* since the situational source is not a source known beforehand, but usually the result of a search for specific data. In other words, the core of SDI is a discovered and dynamic situational source, meaning that situational data loses its value if it is not retrieved when needed. Traversal-based query approaches might be investigated to deal with this task, since they traverse data links on the Web of Linked Data, allowing to discover data from initially unknown sources [103].

Another opportunity regarding discovery and inclusion of situational data is the exploration of semi-automatic approaches for source matching. The semi-automated procedures could involve, e.g., the integration of *human knowledge* in the system activities, since there are some errors that are obvious to humans, but still not obvious for a machine. So, when the system is searching for the "best" data source to be integrated, a manual intervention could identify the presence of errors that cannot be identified by a completely automated procedure [81]. This idea is the core of the current thesis, which considers conversational systems as propitious environments for human feedback to be captured.

It is important to recall that the optimization of one feature can positively impact other features. E.g., if Source Discovery is improved in the sense that relevant knowledge is mined and summarized, the amount of time to provide answer could decrease, thus bettering the coverage of Response Time Improvement. Likewise, promoting the inclusion of situational data sources can leverage decision support, and especially when combined with QA systems, many areas such as healthcare, education, e-Tourism, and BI can be benefited. On the other hand, trade-offs among the features are also a possibility, e.g., focusing on achieving the finest Source Discovery method, with great accuracy, might slow down the response time. Thus, investigating positive and negative effects when working on SDI features, as well as assessing an ideal fine-tune, are interesting research directions.

2.3.2 The Recent Years

Along this chapter a survey covering studies published over two decades (1990-2020) was presented, as a way to track the SDI inclusion in these years. Considering this time range assumed as requirement for the analyzes, papers published after 2020 were not considered, however, they represent opportunities for future investigation.

Visual Question Answering (VQA) is a hot topic, specially in the medical area, where the systems provide support (for both doctors and patients) during the treatment. The authors in [55] point out that most of the existing medical VQA methods rely on external data for transfer learning. In fact, there is a new trend to solve VQA by introducing external knowledge, which has already achieved promising results [261], so that we can infer the SDI usefulness for providing targeted external data to overcome eventual data limitation issues (either for training a model or complementing a knowledge base). Recent VQA approaches seem to present SDI features by, e.g., performing information fusion with different sources [272] or including human collaboration [250].

Besides VQA, knowledge Graphs (KGs) are still on focus in recent QA studies, supported by the continuous growth of Linked Open Data (LOD). The authors in [20] present a very interesting approach that, instead of relying on data retrieval from *static* knowledge graphs,

generates contextually relevant knowledge, which is needed for the integration but not often available in current KBs. Graph completion for QA is also being widely studied in recent years, as a way to better represent the knowledge [30, 148]. Situational sources could be investigated in this context, in order to complete missing facts in KGs.

A recent survey [202] has proposed a taxonomy for classifying QA skills along five dimensions: inference, retrieval, input interpretation/manipulation, world modeling, and multi-step. SDI could be particularly interesting for the "world modeling" dimension, where knowledge sources often need to be combined with spatial, temporal, causal and motivational elements. When considering these elements for situational integration, features such as *Source Discovery* and *Decision Support* could be potentially improved.

Besides the topics mentioned above, extensive advances involving deep learning models must be evidenced [61, 2, 217, 145]. In fact, we have been witnessing the "explosion" of Large Language Models (LLMs) such as GPTs (Generative Pre-trained Transformers), which can understand and generate content according to the current context and specific situations [187]. Since LLMs are trained on vast amounts of data (including situational data), they have been applied in several QA-based solutions that leverage context and memory to generate a specialized conversation flow [204, 220, 22, 92]. An outstanding example is the ChatGPT[9], a powerful chatbot based on the GPT architecture that answers complex questions with human-level performance.

A system like ChatGPT differs from a QA system designed for situational integration since it does not have direct access to databases, but operates based on the data it has been trained on [234]. Also, it aims at simulating a human-like conversation through text generation, rather than facilitating information retrieval or data manipulation (as occurs in a QA system focused on situational tasks). Still, these models can be used to assist SDI in many innovative ways. E.g., they can be an effective source of situational data due to API integration capabilities, retrieving real-time information from heterogeneous sources, specially when data augmentation is needed [114]. Also, LLMs can be customized for operating in specific scenarios, and their ability of context interpretation represents a notable opportunity for enhancing *Decision Support* for users. In clinical domain, for example, the use of ChatGPT as QA tool has proven valuable for assisting the professional in diagnosis and treatment planning [67, 149]. Besides medicine, there is a wide range of applications involving large language models that should be leveraged by future studies for building QA interfaces with situational capabilities [95].

While the rapid evolution of NLP undoubtedly warrants attention, it is important to emphasize that the focus on Situational Data Integration presented in this chapter serves a distinct purpose that complements, rather than competes with, the advancements in QA facilitated by LLMs. SDI addresses the complex challenge of integrating and synthesizing heterogeneous data sources to provide contextually relevant responses, which can be assisted by a natural language interface such as a QA system. While LLMs play a crucial role in enhancing question answering capabilities, the current survey contributes to the broader NLP landscape by offering insights into the practical application of SDI features within conversational systems. Future researchers can take advantage of the challenging topics mentioned in this subsection, towards the development of more dynamic QA systems that can make room for situational data inclusion in several applications areas, with user support as ultimate goal.

2.3.3 A Terminology Mismatch

During the development of this thesis (and specifically the survey described in this chapter), it was observed that, over the years, the term Situational Data Integration has been deprecated in

[9]Available at: <https://openai.com/chatgpt>.

the literature. The concept culminated around the last decade, and gradually, other terminologies started to be applied to talk about integration in situational scenarios [99, 32]. One example is the study in [247], where the authors develop a data mashup process that allows the recommendation of operators for performing situational integration.

Besides the term deprecation, it was observed that SDI is not the only concept in the literature that covers the *on-the-fly* mode, i.e., the idea of data discovered, extracted, and merged at query time. In other words, a terminology mismatch was identified in the literature involving on-the-fly integration: core activities linked to the concept can be found in the literature under many different names. E.g., when data integration is motivated by a specific context or situation, it is often called SDI, so data is integrated on-the-fly to deal with ad-hoc requirements and provide decision support [99, 29, 244]. In other cases, the integration targets a cost-effective initialization, i.e., data is integrated as needed, postponing the addition of labor-intensive data, so it is called *pay-as-you-go integration* [41, 14]. Following this principle, the system incrementally understands and integrates the data over time by asking users to confirm matches on-the-fly, i.e., as the system runs [118].

On-the-fly integration also appears in the literature connected with *mashup applications*, which integrate data on-the-fly to provide a unique service for addressing immediate need in near real-time [213, 147, 172, 43]. Moreover, the idea of discovering and joining data on-the-fly is often seen in the so-called *traversal-based approaches*, where the information is obtained by looking up data links related to a query, and intertwining them with the query result construction [103, 238, 104]. There is no unified nomenclature for embracing these concepts and other similar ones. Although each has its own particularities, they address highly connected (or even the same) activities: selecting and integrating data sources at query time, based on specific requirements, aiming to deliver good results to the end user.

Possibly the biggest issue concerning this lack of standardization is the disorientation it may cause to researchers of the area. E.g., if a researcher is interested in investigating one concept and is not aware of very similar concepts, his/her work could be significantly impacted. Furthermore, if different nomenclatures are frequently assigned to a common set of goals and tasks, eventually we will obtain numerous branches related to the same concept, hampering the search for features and their analysis in a comprehensive way. As a consequence, the development in this domain may be impaired. Improving knowledge organization in this area is, therefore, desirable.

Following this motivation, the next chapter of this thesis discusses the above-mentioned concepts (*mashups, pay-as-you-go integration*, and *traversal-based integration*), similarities and differences among them, and their relation with SDI. Based on this existing connection, the chapter argues for the need to organize knowledge in the area by means of a novel taxonomy. The taxonomy, in its turn, introduce a set of features that are latter used to evaluate the CBR-based architecture proposed in thesis.

Table 2.4: SDI features fully covered (✓), partially covered (❊), and not covered (✗) in QA-based approaches

Approach	Ad-hoc Data Retrieval		Data Management		Timely Decision Support		
	Ad-hoc Question	*Source Discovery*	*Unstructured Data Preprocessing*	*Situational Source Inclusion*	*User Guidance*	*Decision Support*	*Response Time Improvement*
[70]	✓	✓	✓	✓	✓	✓	✓
[252]	✗	✗	✓	✗	✓	❊	✗
[218]	✗	✗	✗	✗	✗	✗	✓
[212]	✗	✗	✗	✗	✓	✓	✗
[50]	✓	✗	✗	✓	✗	❊	✗
[163]	❊	❊	✓	✗	✓	❊	✗
[224]	✓	✓	✗	✓	✗	✗	✗
[155]	✗	✓	✓	✗	✗	✗	✓
[246]	✗	✗	✓	✗	✗	❊	✗
[36]	❊	✓	✓	✓	✗	❊	✓
[93]	✓	✗	✓	✗	✓	✓	✗
[177]	✓	✓	✓	✓	✓	❊	✓
[232]	❊	✓	✗	✓	✗	✗	✓
[153]	✗	❊	✗	✓	✗	❊	✗
[33]	✗	✗	✓	✗	✗	❊	✓
[258]	✗	✗	✗	✓	✓	✗	✗
[256]	❊	✗	✗	✗	✓	✗	✗
[28]	✗	✗	✓	✗	✓	✓	✓
[127]	✓	✓	✗	✗	✗	✗	✗
[129]	❊	✓	✓	✗	✗	✗	✓
[240]	✓	✓	✗	❊	✗	✗	✗
[107]	✗	✗	✓	✓	✗	✗	✓
[73]	✓	✓	✓	✓	✗	✗	✓
[209]	✗	❊	✓	✗	✗	✗	✓
[152]	✓	✓	✓	✗	✓	✗	✓
[27]	✓	✗	✓	✗	✓	✓	✓
[53]	✓	✓	✗	✗	✗	✗	✓
[189]	✓	❊	✓	✗	✗	✗	✓
[9]	✗	❊	✓	✓	✗	✗	✗
[63]	✗	✗	✓	✗	✗	✗	✓
[171]	✗	❊	✓	✗	✓	✗	✗
[245]	✓	❊	✓	✓	✗	✗	✗
[96]	✓	✓	✓	✗	✓	✗	✓
[160]	✓	❊	✓	✗	✗	✗	✓
[156]	✗	✗	✓	✗	✗	✗	✗
[66]	✗	✗	✓	❊	✗	✓	✗
[237]	✓	❊	✓	✓	✓	❊	✗
[264]	✗	✓	✓	❊	✗	✗	✓
[77]	✗	✗	✓	✗	✗	✗	✓
[239]	✗	✓	✓	✗	✗	❊	✓
[42]	✓	✗	✓	✗	✓	✗	✗
[12]	✓	❊	✗	✗	✗	✗	✓
[72]	✗	✓	✗	✗	✓	✗	✓
[24]	✗	✓	✓	✗	✗	✗	✗
[88]	✗	✗	✓	✗	✓	✗	✓
[120]	❊	❊	✓	✗	✓	❊	✓
[267]	✗	✓	✓	✗	✓	❊	✓
[38]	✗	✗	✓	✗	✗	✗	✓
[8]	✗	✓	✗	✓	✗	✗	✓
[207]	✗	✓	✗	✓	✗	✗	✗
[230]	✗	✗	✓	✓	✗	✗	✗
[158]	✓	❊	✓	✗	✓	✓	✓
[229]	✗	❊	✗	✗	✗	✗	✓
[271]	✗	✗	✓	✗	✓	✗	✓
[119]	✗	✓	✓	✗	✗	✗	✓
[40]	✗	❊	✓	✓	✗	✗	✗

3 BUILT-UP INTEGRATION: A NEW TERMINOLOGY AND TAXONOMY FOR MANAGING INFORMATION ON-THE-FLY

The investigation of Situational Data Integration features (discussed in the previous chapter) allowed the identification of similar concepts in the area, where tasks related to data retrieval, on-the-fly integration, and data delivery, despite being often used together, are recognized in the literature under different names. This chapter addresses the lack of consistency regarding some terminologies in related literature, and proposes a new term, *Built-up Integration*, as a knowledge regulation approach. Beyond a formal definition, a taxonomy is proposed for organizing similar characteristics found in the literature through features and subfeatures, which follow a unified nomenclature.

The new term and taxonomy presented in this chapter is essential for the proposal of this thesis, as it is used as a foundation and support for the system architecture described in the subsequent chapters. Thus, as initial guidance to the reader, the next section presents an overview of three data integration approaches that, just like SDI, manage data sources on-the-fly. The existing connection between the terms is the main motivation for proposing a new terminology.

3.1 BEYOND SDI: OVERVIEW OF OTHER DATA INTEGRATION VARIANTS

This section presents an overview of three data integration concepts: *mashups, pay-as-you-go integration*, and *traversal-based integration*. Besides being related to SDI and coping with data augmented in an on-demand way, these concepts were chosen as they cover important tasks included in this thesis proposal, such as data sources selection and user support.

3.1.1 Data Mashups

Data Mashups are usually Web-based applications that integrate data from multiple and heterogeneous sources, in order to provide a unique service [188]. They reuse and combine data sources (encapsulated as data services) on the Web, being developed in a rapid and ad-hoc manner to automate processes and mix information [89]. Unlike applications aimed at expert users, mashups aim to move control over data closer to regular users, allowing to create applications through the merge of several existing data sources [235]. In website mashups, for example, the web page can be changed by removing elements, adding additional widgets, and changing their appearances [89].

Mashups are not restricted to web applications, but they also support the development of situational applications for providing solutions to specific problems [188]. IoT-based mashups, e.g., support on-the-fly integration of contextual information such as real-time sensory data and historical data, so the decision-making process can be executed based on the integration [34]. Another example comes from in [32]: the authors describe a highway emergency scenario, where the emergency staff should transfer wounded people to different kinds of hospital according to the injuries. In this case, the dispatcher needs to know estimated time of each ambulance to the target hospital to implement the proper scheduling. Based on this motivating scenario, the study proposes an approach for end users to discover data services and arrange them in a logical order, to finally create data service mashup plans automatically. Data service discovery and selection are performed among several candidate data services, e.g., *getInjuredInfo, getPersonInfo, getHospitalInfo*, and so on.

The source selection from the above example also allows us to highlight the dynamic and auxiliary nature of mashups when interconnecting several data sources: the user can control the services integration, in a way that he can use any service he wants, putting away the ones that he does not use anymore [138]. With respect to finding sources in mashups, users can mostly perform text-based searches, although context-specific suggestions can provide the needed elements to the user without requiring a search [89]. Indeed, many approaches in the related literature focus on discovering and providing services with minimal manual settings [213]. We can mention, e.g., the studies in [140, 139], which propose algorithms to automate the discovery and composition of Web APIs, and the situational mashup in [113], in which the user context such as location and schedule determines the configuration of accessible widgets.

3.1.2 Traversal-based Integration

Traversal-based integration is a kind of virtual integration[1] that executes queries over Linked Data, traversing data links and merging up-to-date information from initially unknown sources [104, 176]. The exploration of data links is performed at query execution time: first, it searches for URIs (Uniform Resource Identifiers) informed in the query body or as additional parameters. Secondly, it searches for more URIs that can possibly *enrich* the query results. The most relevant URIs and datasets for answering the query are selected, and finally, the answers from the sources are combined to return a final answer [176].

An example of traversal-based integration is given in [102]: the authors consider a SPARQL query that asks for people who authored a paper about ontology engineering at some conference. This query cannot be answered from a single dataset, but requires data from the conference corpus, the names of the paper topics and the authors names. Thus, the traversal based query execution starts with some data retrieved from the conference corpus, by dereferencing the URI that identifies the proceedings. This data contains a set of RDF triples that match one of the triple patterns of the query, and results in Linked Data about published papers, including their topics. In the newly retrieved data, the query engine finds matching triples for the *publication* binding, so that solution mappings can be augmented with bindings for *topic*. Since the topics are also denoted by URIs, additional data can be retrieved to generate bindings for the topic label (e.g., ontology engineering). Following this strategy, it is possible to determine mappings that cover the whole query pattern and get to an integrated solution.

As well as situational integration and data mashups, traversal-based approaches offer up-to-date results, which are at the basis of user support [238]. That is because during the query execution, query links are traversed to expand the set of data already discovered, so further augmentations can be computed for partial solutions [103]. Concerning this expansion, the interesting part is that query execution (i.e., the link traversal) can start without a prior knowledge of available data sources. This *zero-knowledge* method is in line with the dynamic nature of the Web, motivating decentralization and dispensing with the use of data providers to setup costly endpoints [64].

Many traversal-based approaches for integration can be found in the literature. The authors in [162] present a knowledge hypergraph-based approach, able to virtually integrate heterogeneous data from multiple sources and enhance the query answering process in terms of completeness. The SQUIN system [101] discovers relevant data sources within RDF triples during the query execution, integrating the traversal of data links into the result construction. The system may be used either as a Java library that can be integrated in Web applications or as a Web interface. The proposal in [100] also supports on-the-fly integration, specifying the traversal

[1] A virtual integration assumes virtual repositories and the need for near real time data [116].

method in rules. The approach sends requests to specific URIs, discovering links from where new data can be retrieved. Also, a software interface allows to poll the current state of resources at specific time intervals and react to updates, easing the transition from static to dynamic sources.

3.1.3 Pay-as-you-go Integration (Dataspaces)

Providing a coherent view of data is a classical challenge for data integration: although automatic approaches can bring together lots of correct, valuable information, they also may present a fair amount of misleading data [191]. Another classical challenge concerns the high cost of integration initialization, which demands the automatic inference of schema matches and semantic mappings [161]. These tasks are able to produce highly accurate results, but usually involve a delayed start-up time.

To overcome these drawbacks and provide a cost-effective data integration, the variant *pay-as-you-go integration* was proposed, also known as *dataspaces* [97]. The idea of this variant is to distribute the costs of data integration creation to other stages of the integration process, by starting the initialization of dataspaces at the earliest opportunity, and also gathering feedback from the user to improve the integration [14]. In this approach, the assumption is that some application contexts do not require full integration in order to provide useful services, so data is integrated on an "as-needed" basis, with the labor-intensive aspects of data integration postponed until they are required, and when tighter semantic integration is required, it can be achieved in an incremental "pay-as-you-go" way [41, 46]. The user feedback is also gathered in a continuously manner, throughout the entire lifespan of the dataspace, so a better quality in the integration is achieved with lower upfront-cost.

Similar to other integration approaches such as SDI or mashups, pay-as-you-go integration aims to support the user by meeting his requirements. Hence, it is necessary to identify and select data sources that can effectively provide complete answers or results [14]. The pay-as-you-go integration is specially well suited to unstable query requirements and sources that may change rapidly, as it consumes data at an on-demand recombination perspective [80]. Consider, e.g., an e-commerce company that wants to compare price among competitors; relevant sources come and go frequently, and both format and contents change regularly. A classical integration that produces perfect results would not be practical nor effective to integrate the relevant sources and support well-informed decisions [190].

The literature shows several studies that address pay-as-you-go integration and data management. E.g., the study in [106] proposes to query data using on-the-fly mappings, which support a pay-as-you-go paradigm where data is embedded into the search process. Also, the authors in [215] propose to quantify the quality of an integration: given a set of mappings and a set of workers of unknown trustworthiness, feedback instances are collected in the extents of the mappings that characterize the integration.

3.1.4 Compiling Concepts

As observed in the previous subsections, some data integration approaches share many functions and goals, slightly differing from each other as some have a greater focus on one task than others. This is shown on Table 3.1, which summarizes characteristics that best differentiate the integration approaches, such as the type of input query, type of user support, and the level of human involvement. These characteristics are mentioned in the related papers discussed in the previous subsections. Besides these, the comparative table also shows dominant features (i.e., the main focus of each integration approach) and keywords (for which goal they are usually applied). We can see that some concepts are more focused on the data discovery process (e.g.,

Table 3.1: Comparison between different types of integration

Approach	Dominant Features	Input query	Type of Support	Human Involvement	Keywords
Situational Data Integration	Source discovery, Data augmentation, User support	Necessarily situational, various formats	Fresh data for decision-making	Mostly feedback	Business Intelligence, decision-making
Traversal-based Integration	Source discovery, Data augmentation	Mostly conjunctive queries	Data augmentation	Not required	URI exploration, zero-knowledge execution
Data Mashup	Source selection, Data augmentation, Human involvement	Mostly situational and GUI-based	Single service building, interactive interface	System operation (services composition)	User experience, control of the integration
Pay-as-you-go Integration	Data augmentation, Human involvement	Mostly conjunctive queries	Fresh data for decision-making, dynamic adaptation	Feedback and training	Low cost execution, User Feedback, Gradual improvement

traversal-based approaches) or the decision-making support (e.g., SDI). Also, in some of them (such as Data Mashups and Pay-as-you-go integration), the user role is more significant.

But most importantly, with regard to the *similarities* between the approaches, they generally perform source discovery and/or selection, some kind of data augmentation made at query time, and have the common goal of supporting the end user (either by helping him to make a decision, or by providing valuable visualization from the integrated data). All of them may involve multiple and heterogeneous data sources in the integration process. In addition, it is possible to find in the literature studies that mention more than one concept, i.e., SDI and mashups [247, 34], mashups and pay-as-you-go integration [233, 78, 109], or even traversal-based approaches and mashups [101, 164].

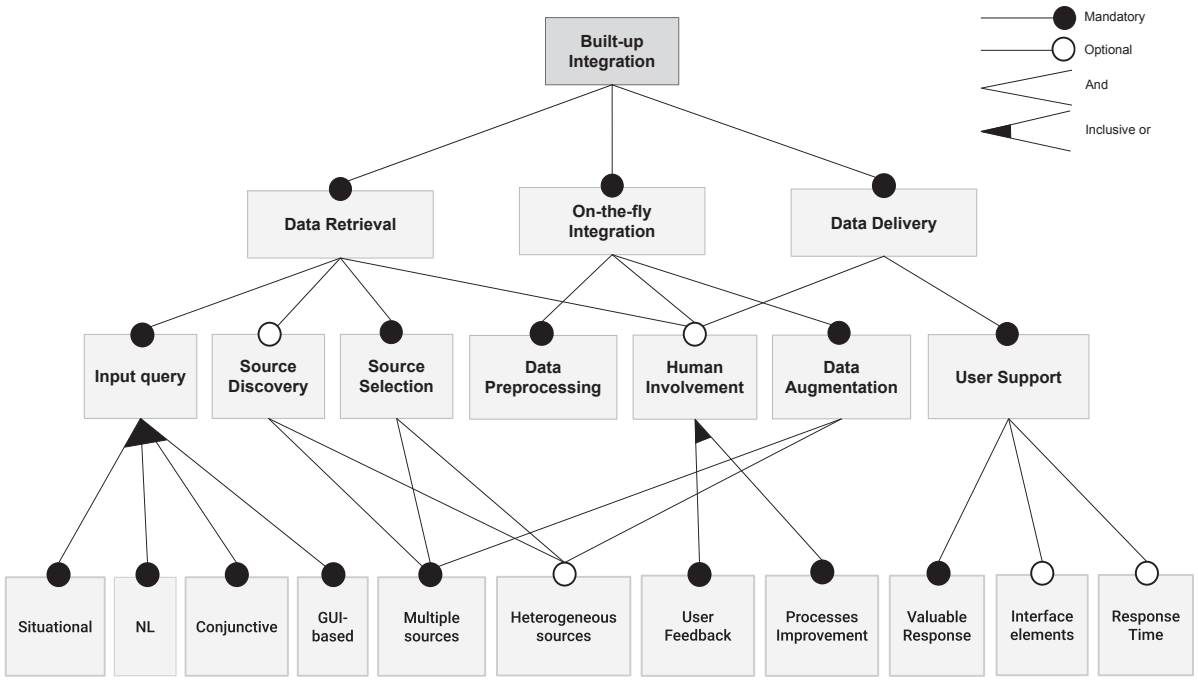
Despite the similarities found, there is still a lack of a nomenclature for unifying all associated tasks. The problem with this gap is the confusion it generates for researchers that may be interested on a concept often addressed in the literature under another name. Let us suppose, for example, that a researcher wants to systematically review all studies in the literature that address the term “Situational Data Integration” (SDI), to check how the discovery of data sources takes place in a certain period of time. In this case, several studies that present the same idea as SDI under a completely different name could be ignored, since the researcher is not aware of the similarities and differences among the existing concepts. As a result, this “non-awareness” would certainly impact on the reliability of the research produced. Thus, a taxonomy for properly organizing the knowledge in this area is needed.

In the next section a novel terminology named **Built-up integration** is defined, aiming to regulate similar aspects observed in several integration approaches. The section also presents a taxonomy for Built-up integration, covering a set of features related to source selection and discovery, data integration and information delivery.

3.2 REORGANIZING THE KNOWLEDGE: BUILT-UP INTEGRATION

Based on common features found in different integration methods (see Section 3.1), this section proposes a novel term named **Built-up Integration**, which follows the following definition:

Figure 3.1: Taxonomy for Built-up Integration



Built-up Integration selects and manages data sources on-the-fly for dealing with specific query requirements, resulting in an augmented data set for supporting the user.

The term is proposed to assign a common term to data management characteristics that are found together in several integration approaches, such as the ones mentioned in Section 3.1. In these approaches, the information value is often transient, so that applications consume data on-the-fly to perform specific and/or additional analysis tasks demanded by user requirements. We believe the term *Built-up Integration* is appropriate to accommodate such approaches, since it expresses the idea of data being combined gradually and systematically, until reaching a set of unified data, complete enough to support the end user. At this basis, a Built-up integration system must systematically analyze potential data sources at hand and select the one(s) that can meet the users needs. The selected sources must be reconciled and combined towards the delivery of up-to-dated solutions, which cannot be addressed by loading data repositories from time to time. The user has a central role in the integration, since data management occurs targeting a timely support and, desirably, a positive impact in his decisions. In addition, the user can participate actively in all tasks involved in Built-up integration, by deciding which information are relevant or giving feedback on the responses received.

Beyond a formal definition of Built-up integration, this section also presents a taxonomy that unifies a set of features that are found in related literature. The taxonomy (shown in Figure 3.1) was created as a feature diagram using the Feature-Oriented Domain Analysis method [124]. A feature diagram is a hierarchically arranged set of features, where relationships between a parent feature and its child features may be categorized as: *and* – all sub-features must be selected, *alternative* – only one subfeature can be selected, *inclusive or* – one or more can be selected, *mandatory* – features that are required, and *optional* – features that are optional [17].

For building the taxonomy from related studies (see Section 3.1), we first searched for integration-based approaches on Google Scholar search engine[2], using keywords such as *situational integration, mashup systems, on-the-fly integration, linked data, traversal-based query, pay-as-you-go integration*, alternately and merging the keywords for filtering results. The Connected Papers online tool[3] was also used for verifying related and derivative papers.

Next, a set of characteristics was extracted from the related studies, grouping them by similarity. A generic nomenclature was assigned for each group, so that similar characteristics in a group were represented in an unified way. By following this process, the Built-up integration taxonomy was built, composed by three main features: **Data Retrieval**, **On-the-fly Integration**, and **Data Delivery**, which represent important steps executed by several recent data integration systems. These features are detailed as follows.

The first main feature, Data Retrieval, covers the ability of retrieving useful sources of information for dealing with specific domain problems. For doing this, it relies on two mandatory subfeatures, named `Input Query` and `Source Selection`. Through the information contained in the input query, a system can analyze different candidate sources of information and select the ones that are most likely to provide a reliable answer. A user query can be expressed through many ways: by using a particular language such as SQL or SPARQL (which are types of conjunctive queries), through natural language, by interacting with GUI-based elements in an interface, and so on.

Based on an input query, the integration system can perform `Source Discovery`. This feature assumes that data sources must be discovered on-the-fly for dealing with users requirements [4]. The discovery may be explored through table relatedness measures [270], similarity joins between data collections [262], or general similarity methods that determine how related the source is according to the query. `Human Involvement` may be present during Data Retrieval, if the system allows the user to solve ambiguities in data sources attributes, or choose one source among several good matches.

As the second main feature of the taxonomy (see Figure 3.1), there is On-the-fly Integration. This feature refers to the ability of analyzing and combining, *at query time*, heterogeneous data from the previously discovered sources. Hence, it should also consider possible changes in users requests, adapting data sources and/or processes accordingly [123]. Due to these characteristics, an integration performed on-the-fly allows to retrieve situational data, from which we can derive valuable insights that enable accurate decisions [244]. Regarding the taxonomy, On-the-fly Integration has `Data Preprocessing`, `Data Augmentation`, and `Human Involvement` as subfeatures.

`Data Preprocessing` is a step that aims to deal with heterogeneous data sources that usually contain out-of-date, noisy, or conflicting data. In order to perform a proper integration, the information extracted from a source needs to be accessible through data cleansing methods, which should resolve unique entities or fill up missing information [154]. Also, data integration often requires some sort of correlation, by means of obtaining the data schema from the discovered sources, estimating available dimensions, facts, and measurements, performing similarity calculation, and so on [180].

After cleaning the data and assessing how they can be used to meet the users needs, the selected data sources are finally merged, causing an "extension" of the information previously available. We call this feature `Data Augmentation`, which can occur by combining stationary data (e.g., a DW) with external data [244], or simply merging external resources [104]. The

[2]Available at: <https://scholar.google.com>.

[3]Available at: <https://www.connectedpapers.com/>.

resulting augmented set is then used for answering requests that could not be properly answered, due to lack or insufficiency of the current data [3].

During the integration, `Human Involvement` may be present to help the system to improve the quality of data integration. When involved, the user can check the presence of errors, decide which is the best action or option among the available ones, and even create quality rules for the system [81]. Most importantly, the user feedback can be collected and used as training tool, so the system can make better choices in the future, such as the recommendation of a service based on personal preferences. This participation is not only valuable during the data integration, but also in the Source Discovery step, where human knowledge can help in the identification of relevant data sources to be retrieved.

Data Delivery is the last feature of the Built-up Integration taxonomy, and it refers to the just-in-time delivery of valuable solutions to the user after completing the integration. Particularly, Data Delivery demands `User Support`, i.e., making the user aware of situations that can potentially affect his activities. This can be done by the system simply producing a valuable response (e.g., providing some explanation of how calculations took place), or making a recommendation (e.g., which sequence of steps the user should apply) [214]. This kind of support can be assisted by a graphical interface, favoring the user interaction and feedback collection. Also, another way to support the user is through methodologies for improving the system's response time by, e.g., reducing the processing complexity, implementing multi-thread execution, and so on [154]. Finally, as well as the other main features, Data Delivery can also have `Human Involvement`, where user feedback about the responses can refine the results and fine-tune data models.

Several approaches in the literature perform Built-up integration, since they have mechanisms to jointly execute Data Retrieval, On-the-fly integration, and Data Delivery (the main features of the taxonomy). The next subsection presents examples of them, indicating how Built-up integration can be found in data integration studies.

3.3 WHERE WE CAN FIND BUILT-UP INTEGRATION?

This section aims to correlate Built-up Integration with the types of data integration discussed in Section 3.1, in order to provide an overview of how the proposed features can be identified in the literature.

3.3.1 The Main Features

As mentioned in Section 3.2, three main features are often shared between integration approaches, which are *Data Retrieval, On-the-fly integration*, and *Data Delivery*. The Table 3.2 exemplifies where these features can be found, taking as a basis a set of related studies mentioned in the Section 3.1. Besides the table, the main features identification is discussed in details next.

3.3.1.1 Data Retrieval

As mentioned in Section 3.2, Data Retrieval requires `Input Queries` and `Source Selection`. Concerning the former, we assume that users needs may be expressed in many different ways. Conjunctive queries (such as SQL and SPARQL) are the most common class of queries used in database systems, and are used in traversal-based/pay-as-you-go approaches [11, 103, 105]. Source Selection in mashup approaches often relies on a graphical interface, so the user can interact with components, and overload models until a satisfactory

Table 3.2: Built-up Integration examples in the literature

Approach	Type of Integration	Data Retrieval Features	On-the-fly Integration Features	Data Delivery Features
[103]	Traversal-based	Conjunctive queries, discovery and selection of URIs.	Link exploration made at query time, the discovered URIs are used to enrich the results.	Provide a complete answer to a question, achieved with data augmentation.
[238]	Traversal-based	Conjunctive queries, prototype with a source selector that decides which query and URIs should be dereferenced, and which links should be followed.	Query evaluation strategy that discovers additional sources on-the-fly and integrate data during query-answering.	Offers the potential to get fresh answers when dynamic information is involved.
[70]	SDI	Situational and NL query, selection of external sources (obtained with a Question-Answering system) to be integrated in a Data Warehouse system.	Full integration of unstructured and structured information, allowing to compare data instantaneously through a dashboard.	Data delivery allows the user to take quick strategic decisions based on richer data.
[177]	SDI	Situational and conjunctive queries. Situational data is achieved by means of RESTful APIs.	The proposed Big Data Integration ontology semi-automatically adapts to situational data acquired under a schema evolution process.	The proposal aims to evolve decision making and exploits end-user feedback to improve the quality of experience.
[247]	SDI/mashup	Situational and GUI-based queries, selection of several data sources by means of web services.	Retrieved data can be accessed and combined, and finally published as a composite data service.	The user is involved and supported in the data mashup process, as the system provides interactive recommendation of composition operators for performing situational integration.
[41]	Pay-as-you-go	Conjunctive queries, dataspace query service for real-time data streams that enables unified queries across live streams, historical data, and entities.	Real-time query service which preprocesses the streams on-the-fly instead of storing them, complementing older views already achieved.	Validation of the Real-time Linked Dataspace proposed within five real-world smart environments pilot deployments to build real-time analytics, decisions support, and smart apps for smart energy and water management.
[59]	Pay-as-you-go	Conjunctive queries, interface-based discovery and selection of RDF data sets. Human feedback collected during the interaction is used to reject links and discover new links similar to the ones approved by the user.	Discovered links are automatically incorporated in the query processing, aiming to achieve a complete output/answer.	Data delivery provides more complete answers to the user.
[34]	Mashup/SDI	GUI-based queries to select the different types of data sources, such as sensory data, local files, relational databases, and Web service resources.	The situational IoT services mashup approach supports on-the-fly integration of the different data services.	Decision-making processes can be executed based on real-time sensed data. The user is also supported in the composition of situational applications.
[140]	Mashup	GUI-based queries, automatic discovery and selection of Web APIs.	Graph-based composition algorithm for the integration of Web APIs, where a composition is gradually generated by a backward chaining of APIs. At each step, suitable APIs are automatically added to the composition.	Users can obtain immediate composition results visually, and iteratively refine their goals to achieve improved results.
[233]	Mashup/Pay-as-you-go	GUI-based queries, source discovery and selection based on automated schema matching. The system lets the user refine the results interactively.	The system helps a user to improve the results from sources integration, at a query time, until he is satisfied (along with the "pay-as-you-go" principle).	Query results can be visualized in an interface, where the user can also give feedbacks. If the result is satisfactory, it can be saved in several ways, dynamic spreadsheets or a new web service.
[101]	Traversal-based/Mashup	Conjunctive queries. The SQUIN proposal discovers and retrieves data that might be relevant for answering a query during the query execution process itself. A mashup application queries the Web of Linked Data using SQUIN.	Incremental construction of query results with the traversal of data links. Suitable for an "on-demand" live querying scenario.	Data delivery of fresh answers.

solution is found [213]. Mashups can be also built with programming languages such as EMML (Enterprise Mashup Markup Language), Orc, and YQL (Yahoo Query Language) [188].

Regardless of query format, in SDI the input query is *situational*, i.e., it focuses on a particular problem and cannot be defined in advance [3, 247]. Mashup queries can also be situational, since a mashup is usually developed for rapidly address an immediate need or a specific situation [138]. In pay-as-you-go integration, the querying service can be made through keyword search, structured queries (assuming that the user understands the underlying data schema), browsing of available datasets, and even through question-answering (which focus on natural language interaction) [41].

The most challenging task related to source selection in Built-up integration is `Source Discovery`. In some cases, users spend more time searching for relevant information than analyzing it, and for facing this issue, Source Discovery automatically identifies and retrieves one or more data sources suitable to a user query. This taxonomy feature can be found in several integration systems. In SDI, a common assumption is the existence of a local database, although it usually cannot provide an actionable information to the user by itself. Thus, a source discovery engine should discover external information to be further integrated with the local data [244]. The data sources are determined according to particular query requirements, or to previous integration results [247, 154]. Regarding traversal-based integration, existing approaches can discover initially unknown data sources at runtime, so they start querying without first having to populate a repository of data [104]. In data mashups, services can be discovered by, e.g., text-based searches and browsing of services' structural properties, whereas the content is mostly collected with the help of APIs (Application Programming Interfaces) [89, 213].

In pay-as-you-go approaches, new sources are included in the dataspace automatically. In this kind of integration, the semantic relationships derived may be approximate, but the inclusion of user feedback (`Human Involvement`) assists in gathering information about the selected sources and dealing with data uncertainty [14]. In fact, `Human Involvement` in Data Retrieval can also be observed in SDI approaches, because when situational data are retrieved and returned, the user may decide that they are not suitable for the task at hand [3].

3.3.1.2 On-the-fly Integration

On-the-fly integration combines sources at query time, aiming to satisfy a situational need (i.e., a specific and ad-hoc requirement). This feature covers `Data Preprocessing` and `Data Augmentation` as important subfeatures within Built-up integration, and it can be observed in derived approaches.

In SDI, the integration joins situational/external data with an information previously available, generating an augmented set of data, which is used to provide useful insights [244]. The situational approaches described in [122, 177, 70] exemplify data preprocessing and augmentation covered by Built-up integration. Pay-as-you-go integration implies that resource-intensive data integration should be performed at much lower cost, thereby it occurs on demand, starting with a lower data quality. As a result, the approaches make use of techniques that infer relationships between resources and refine these relationships in the light of user feedback [105]. In relation to Data Augmentation, pay-as-you-go approaches execute a *data fusion* step, where the dataspace instances are transformed into a single and consistent representation instance, which will be later available for user viewing [161].

Proposals that cover traversal-based integration also present some sort of Data Augmentation. In link traversal, for example, data links may be traversed during the query execution to expand discovered data, i.e., to augment a dataset [103]. Such augmentations can also be found in graph-based approaches that execute traversal algorithms for integration [133, 192]. With

respect to Mashups, data from different sources are merged into a single joint place. The result from this augmentation can be visualized as a web page, a web application, or a service, which is able to fulfill users requirements [213].

3.3.1.3 Data Delivery

After data integration, the user needs to receive and visualize problem solution in an effective way. In the taxonomy, this feature is called *Data Delivery*, which covers `User Support` and `Human Involvement`.

User support is a key feature for SDI: as it bases on Situation-Awareness, the integration focuses on providing decision-making in complex and dynamic situations. The support can be achieved by means of predictions, alerts, or recommendations given to the user [19]. The general idea is that data integration can make users aware of the current situation, and hence they have the opportunity to take immediate action [29]. Pay-as-you-go integration also focuses on supporting decision-making in highly dynamic environments. In this setting, a practical and fast integration is better than a perfect integration, since the user can revise the results and gradually improve the process [161].

In Data Mashups, user support occurs mostly through an effective visualization of results, i.e., graphical interfaces that allow the user to combine services and see solutions that meet the initial requirements [213]. User support can also mean an improvement of user experience. Traversal-based integration, for instance, covers query optimization techniques and settings for URI lookups that aim to reduce the response time [104].

In the proposed taxonomy of Built-up Integration, Human Involvement is not mandatory. Some integration approaches (such as the traversal-based ones) do not require user guidance or feedback, whereas for others, this kind of human participation is essential. E.g., in pay-as-you-go approaches, human feedback is highly important to indicate the correctness of the received answers, constantly improving the integration [59]. Similarly, in SDI, fused data may be approved by the user, who can either confirm the results or propose alternatives [3]. Also, data mashups can be executed in a semi-automated way, with user guidance during data discovery and integration [188].

3.4 CHAPTER REMARKS

Built-up Integration was proposed for systematically grouping and labeling similar tasks found in integration-based approaches. It is important to recall that the term does not come to fully replace the terms already defined in the literature, since each one has its particularities (see Table 3.1), especially considering the research context in which they were defined. In this sense, it would be correct to affirm, e.g., that SDI is a type of Built-up Integration (as it contains the characteristics defined in the proposed taxonomy), but keeping distinct characteristics, such as its application in strategic decision processes and the premise of stationary data. Most importantly, the Built-up Integration terminology highlights the similarities between the approaches already available in the literature, and makes way to future approaches to be better categorized.

For demonstration purposes, Built-up Integration was only analyzed related to four integration concepts existing in the literature, which were chosen due to the many common tasks identified, and whose connection can be even more strengthened if we consider the joint mentions in past publications (see Table 3.2). However, besides these concepts, other types of integration executed on-demand (or considering situational problems) could also be considered and classified within the taxonomy features. The taxonomy is generic, meaning that it can be extended in the

future to cover more detailed aspects such as data sources types, adaptation methods for source selection, or preprocessing techniques. In this case, the level of detail should consider the need to specify optional edges, i.e., a feature optional on one side and mandatory on another [210].

Regardless of the taxonomy granularity, we consider that Built-up Integration features are still very challenging. Considering that Built-up Integration represents an umbrella term for several kinds of integration, some of the existing challenges involve how to execute its steps at query time, considering the highly heterogeneous structure of datasets. More than that, the user experience should be leveraged throughout the execution steps, targeting not only the delivery of fast answers, but the improvement of tasks that favor smarter decisions. With the decision-making goal in mind, discovering and merging information at query time through Built-up Integration are steps that should be explored towards a proper delivery of services to the user.

However, as mentioned in Section 1, using a completely automated method to discover a data source might result in ambiguities and errors within the retrieved data [147, 81], consequently affecting analysis and user support. So, we raise the hypothesis that human knowledge can be collected and used to solve ambiguities during data retrieval and, most importantly, to systematically train a system on useful data to recommend [115]. Particularly, when considering conversational interfaces as mediation tool (as proposed in this thesis), the agent is able to learn from human evaluative feedback and demonstrations, using them to build or modify internal execution rules [146]. As a consequence, Built-up Integration tasks such as source discovery and on-the-fly integration could become more efficient.

Based on the opportunities exposed in this Chapter and research gaps previously discussed in chapters 1 and 2, the next chapter presents a conversational Case-Based Reasoning architecture for Situational Data Management. The system architecture covers a conversational agent for enabling user interaction, and applies Case-Based Reasoning as learning methodology, since it embraces a *Review* component that leverages human feedback. Being situational data an essential component of the Built-up Integration concept, the proposal aims to support Data Retrieval, On-the-fly integration, and Data Delivery tasks, using human knowledge as basis for incremental learning. Thus, the importance of the current Chapter is evidenced in its closing, relying not only on the organization proposed through the new taxonomy, but on the identification of features that compose the thesis proposal.

4 CONVERSATIONAL CBR ARCHITECTURE FOR SITUATIONAL DATA MANAGEMENT

As mentioned in the previous chapter, promoting Built-up Integration involves several challenges, such as discovering relevant data sources and supporting the user (either through a good interactive experience or the delivery of useful insights). This chapter proposes a conversational CBR architecture to alleviate these challenges, through a "human-in-the-loop" approach. Specifically, it is based on the hypothesis that human feedback can be used for reinforcement learning (i.e., for guiding a system behavior), thus improving the delivered answer or result. To assess this hypothesis, a *conversational agent* was chosen as a mediation tool to interact with the user and collect feedback, and *Case-Based Reasoning* was chosen as reinforcement learning strategy.

The use of a conversational agent is inspired by a recent trend in computer science, where conversational interfaces are investigated for democratizing data access and reducing querying complexity, especially regarding Open Data [74, 174]. Thus, a conversational agent was included in the proposed architecture to recognize conversation topics, use them to identify data sets, recommend suitable attributes, and present the information to the user. In the identification of datasets, the Multidimensional Data Model [111] was taken as reference, which assumes that data is explicitly modeled with facts structured as *data cubes*. A cube, in turn, is defined with respect to several *dimensions* and includes a number of measures, or *metrics*, to hold the measurable aspects of facts[1] [62, 241]. In short, the conversational agent uses database metrics, dimensions, and filters to communicate with the user and gather the proper feedback. It is also responsible for executing a Case-Based Reasoning methodology in order to retrieve a consistent solution, reuse and adapt it according to the human review, and retain it for the system's learning (see Figure 1.1).

The use of CBR methodology can be justified in many ways. First, it has been successfully applied for a long time in recommendation systems, and also recommendation dialogs, through conversational interfaces [137, 23, 108, 45]. Second, it favors explainability to users, which is a highly desirable feature when considering that machine learning techniques are frequently complex and opaque (e.g., deep neural networks) [211, 85]. Also, it allows to efficiently use the human knowledge to solve problems, which is particularly interesting for Built-up Integration tasks: considering that humans cannot always remember relevant information for problem solving, a CBR system augments the person's memory by providing cases that can be used as guidelines for decision making [131]. Thus, it is possible that the user makes better decisions because the augmentation provides more cases (and perhaps better ones) than would be available without the machine. As discussed in chapters 2 and 3, user support is a key feature for situational approaches and Built-up Integration.

The conversational CBR architecture is presented in Figure 4.1, and has the main goal of proposing suitable databases and attributes to be queried according to an input case. The cycle execution is detailed as follows: First, a user queries a conversational system (let us assume a chatbot) by posing an initial question, which is considered as a **New Case** inside the CBR Cycle. When searching for a good database match for the input question, the chatbot maintains a historic knowledge base containing previous cases, i.e., a set of triples <*dbName, dbMetrics, dbDimensions*> from which we can extract the triple that best suits the new case. The best match is supposed to satisfactorily answer the user question, and it is verified through a similarity score

[1] A cube containing demographic facts, for instance, could be defined by dimensions "region" and "population". Metrics could address, e.g., the total of citizens in a given city or the average number of cities per region.

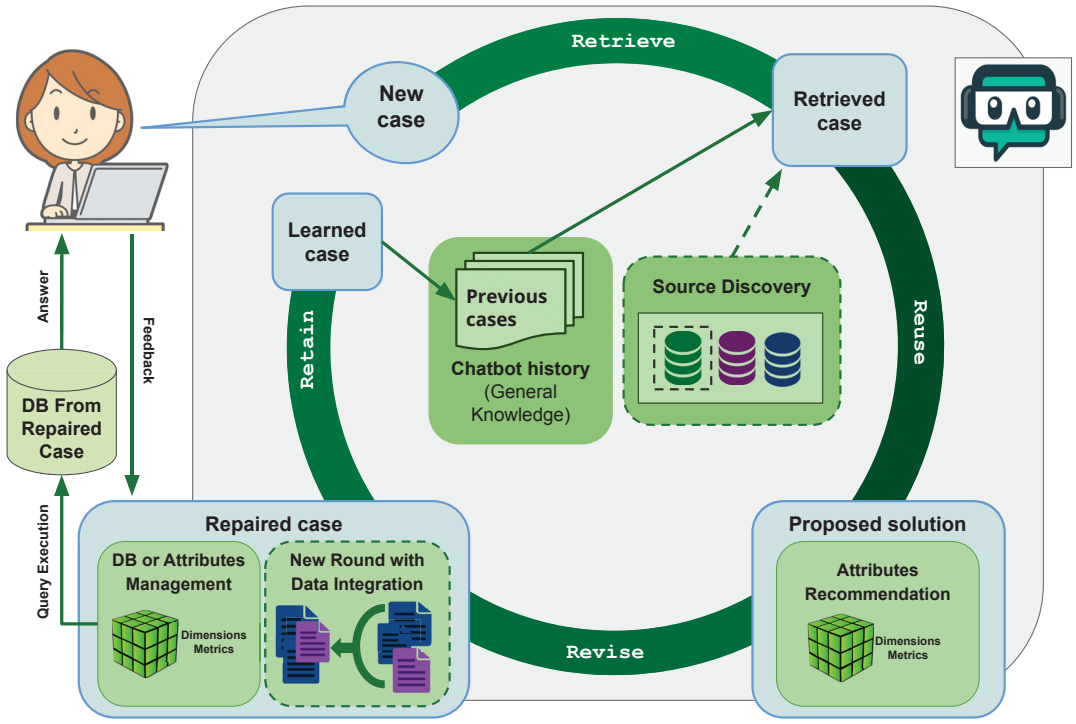


Figure 4.1: Functional view of the Conversational CBR approach for Built-up Integration Support.

associated with each triple: the higher the score, the more likely the triple matches the input case. The selected triple represents the **Retrieved Case**. When there is no similar case in the historic knowledge base, the system triggers a Source Discovery process, aiming at finding an appropriate data source to answer the question. In this case, the selected/discovered source should be preprocessed to extract relevant values for *dbMetrics* and *dbDimensions*, which then represent the Retrieved Case in the cycle.

From the Retrieved Case, the conversational system proposes a solution to the user, i.e., the database found as the most relevant one, along with metrics and dimensions that could answer the New Case. Next, in a Revision step, the user has the opportunity to manage the attributes suggested (e.g., replace them or add potentially useful attributes) and use them to build a definitive query that should retrieve the final answer. In this context, the database and attributes become a **Repaired Case**. Another repair possibility is when the answer contains only a part of the information the user wants, so he/she could choose to complement it with a second source (either discovered or selected among the available options), meaning a second execution of the "Reuse" and "Revise" steps. After the second execution, the user has two partial answers that could be integrated and visualized together in the chat. Thus, the integration of answers also represent a Repaired Case, dependent on the user feedback. The Repaired Case, after all the user checks, turns into a **Learned Case**, i.e., it is stored in the historic knowledge base to be considered by the system every time a new case is received. It is important to highlight that, in the architecture overview (Figure 4.1), the components bordered with a dashed line are secondary within the CBR Cycle, meaning that they will be executed when previous cases are not enough (Source Discovery), or the answer received is not complete, demanding a second round of the cycle (Data Integration).

The next subsections focus on the isolated CBR steps shown in Figure 4.1, and how their activities are executed in consonance with Built-up Integration support.

4.1 RETRIEVAL: HANDLING A NEW CASE

Case retrieval in the CBR cycle involves measuring the similarity between the new case and previous/historical cases. In the CBR architecture proposed in this thesis (Figure 4.1), all the cycle relies on the Multidimensional Model, i.e., metrics and dimensions are used as basis for retrieving cases, recommending solutions, and repairing a case that will be further retained.

The Retrieval step of the proposed CBR architecture relies on two activities: (1) retrieving cases from a *historic source*, and (2) *discovering and retrieving a new source* to serve as new reference, when the history itself is not enough. These activities will be addressed next.

4.1.1 General Knowledge (History of Previous Cases)

According to the CBR methodology, the use of memory (i.e., historical cases) is a good way to quickly retrieve relevant knowledge to solve a new situation [157]. In the proposed architecture, each previous case *PCn* in the history is identified by a set of tokens T_{PCn}, stored from past questions. Each *PCn* is also represented by a *database name* and its most significant *metrics* and *dimensions* according to T_{PCn}. Thus, given a new case Q, the history retrieval determines the most similar *PCn* to be reused throughout the cycle. The retrieval process is represented in Figure 4.2.

Let us consider, for example, that the user asked the following question Q: "`How many graduate students are enrolled in public schools?`", which is assumed as a new case in the CBR cycle. The question is preprocessed (more details in Section 4.1.2.1) in order to output a set of tokens T_Q: `<graduate, student, enroll, public, school>`. Next, the tokens T_Q are compared to the content of the the historical source in order to return the best match. Two scores are considered in this phase: *Similarity score* and *Usefulness score*.

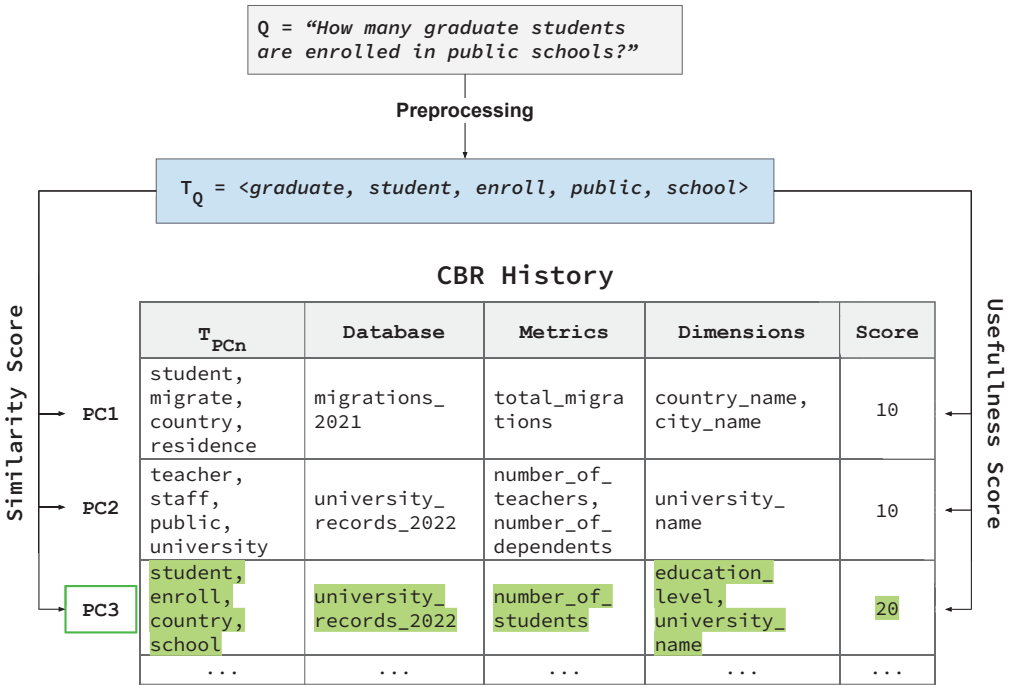


Figure 4.2: Case retrieval from history.

The first one, *Similarity Score*, considers the similarity between T_Q and all T_{PCn}. Considering the example in Figure 4.2, and assuming a similarity based on common attributes,

the previous case PC3 would be retrieved, since 3 out of 4 tokens were found in T_Q. The *Usefulness Score*, on the other hand, determines how much each case was retrieved and reused in the past. Considering, for example, that each case is initialized with a $UsefulnessScore = 10$ when first included in the history, and $UsefulnessScore+ = 10$ for each time the case is reused[2], PC3 would be chosen again, as its score is higher than the other previous cases. This two-step verification (through different types of retrieval scores) aims at a more reliable recommendation to the user, and execute a "tiebreaker" if there are two or more cases with the same similarity score to the input question. In addition, as the history receives more cases, it is possible to have two or more previous cases share the same Similarity Score and the same Usefulness Score. Then, the temporal proximity becomes the tiebreaker criterion, and the last case inserted is retrieved.

Regarding *text preprocessing* within the Retrieval step, it is valid to highlight that the input data must be consistent enough for searching a similar case in the history or external sources. Online texts, for example, usually contain a lot of noise and uninformative parts (such as HTML tags), which have low significance for the text analysis [168]. Similarly, user replies to a conversational interface can be full of stop words and informal text that can not be effectively analyzed. By removing irrelevant elements and cleaning the words during the Retrieval step, performance and speed can be improved [65, 185].

Although the related literature discusses several approaches for text preprocessing [48, 168, 125], we highlight a small set of them, since they were chosen for composing this thesis' methodology:

- Special characters removal: This technique aims at removing uninformative characters from a text, such as tags, punctuation, extra white-space characters, and special characters in general (such as $, %, or &) [48].

- Lowercasing: The existence of uppercase in a word usually does not affect its meaning, so as the words "Door", "door", or "DoOr" should refer to the same object. Thus, lowercasing sentences avoids differentiation based on character and ensures consistency when, e.g., sentences are being compared.

- Stopwords removal: Stopwords are words that recur very frequently in documents, as they are used to join words together [125]. It is the case of "and", "are", "this", and others. These words do not contribute to the text content, and most importantly, they may confuse a retrieval system or classification algorithm. Their removal is therefore highly recommended.

- Tokenization: Tokenization is the process of breaking a sentence into words, phrases, symbols, or other elements called *tokens* [125]. This segmentation allows the lexical exploration of the words in a sentence, and especially the extraction of meaningful keywords for further processing. With tokenization, a new case "what are the colors of the Brazilian flag?" generates the tokens "colors", "Brazilian", and "flag" (considering that stopwords removal was also applied in a previous step).

- Stemming: This technique removes inflectional endings or suffixes from a word [168]. The word "stripes", for example, would return "strip" when stemmed. This technique is useful to normalize sentences when searching for a match in a target source.

[2]The number 10 is only for example purposes. The choice of a number for initialization and increment can be based on a specific methodology or motivation.

- Named Entity Recognition (NER): It aims to recognize mentions or proper names from text, belonging to predefined semantic types such as person, location, and organization [143]. If a user asks, for example, for "the gasoline price in Brazil", *Brazil* will be recognized as as entity. In data retrieval contexts, NER can be highly useful to capture entities and link them to a class of information being searched.

Preprocessing the user question is needed for the CBR Retrieval step, in which a relevant match is returned from the historic source. There may be situations, however, where no case in the history is good enough to match the new case, either due to Similarity and Usefulness scores below the expected (in case a threshold has been defined), or because the user was not satisfied with the system recommendation. Thus, the CBR Retrieval should consider a *Source Discovery* mechanism, which will be discussed next.

4.1.2 Source Discovery

A data discovery problem occurs when users spend more time looking for relevant data than analyzing it [68]. So, Source Discovery is a process that aims to mitigate this obstacle, finding one or more relevant data sources (among many possible sources) suitable to a user query [76]. As shown in Figure 3.1, a Built-up Integration involves the selection of an appropriate data source, and in many cases, the discovery of the "best match" for answering situational questions. This subsection addresses the Source Discovery component, whose purpose is to achieve the most adequate data source among a set of candidates. The component bases on *Metadata Matching*, i.e., the similarity between an input question and the sources attributes, and it is demonstrated in Figure 4.3.

First, it receives the input question Q (also called new case) that expresses the user's intention, and a set of candidates sources ($[CS_1, ..., CSn]$) that are eligible to provide an answer. The question, as mentioned in the previous subsection, is preprocessed to obtain the tokens T_Q, whereas the candidate sources are first given as input to a Metadata Extraction function that extracts the attributes description, which are latter preprocessed. The result from the candidate sources preprocessing are sets of tokens $[T_{CS1}, ..., T_{CSn}]$. Next, the sets go through a *Source Selection* step, that systematically assesses how similar Q is to each T_{CSn}, returning the best match.

To demonstrate the Source Discovery component based on metadata, let us consider the tokens T_Q from the previous example (<graduate, student, enroll, public, school>), which will be used to look for elements in the candidate sources schemas that indicate potential to provide answers. Let us also consider three candidate sources, whose metadata are represented in Table 4.1. When analyzing T_Q, we see a stronger connection to some attributes from T_{CS2}, with respect to value overlap. Based on this assumption, if we use all sets of tokens ($T_Q, T_{CS1}, T_{CS2}, T_{CS3}$) as input to a source selection method, the highest similarity observed in T_{CS2} would result in $CS2$ as the selected source.

Obviously, measuring similarity based on value overlap does not consider *semantic similarity*, e.g., a token "school" is highly connected to "university", although they have completely different nomenclatures. This would mean that a source metadata set similar to the input question could be eventually skipped. One way to avoid this kind of misinterpretations within the Source Discovery component would be including some method that performs semantic similarity between the tokens, besides the syntactic similarity [69, 134, 181].

The next subsection describes some syntactic and semantic methods investigated in this thesis for source selection.

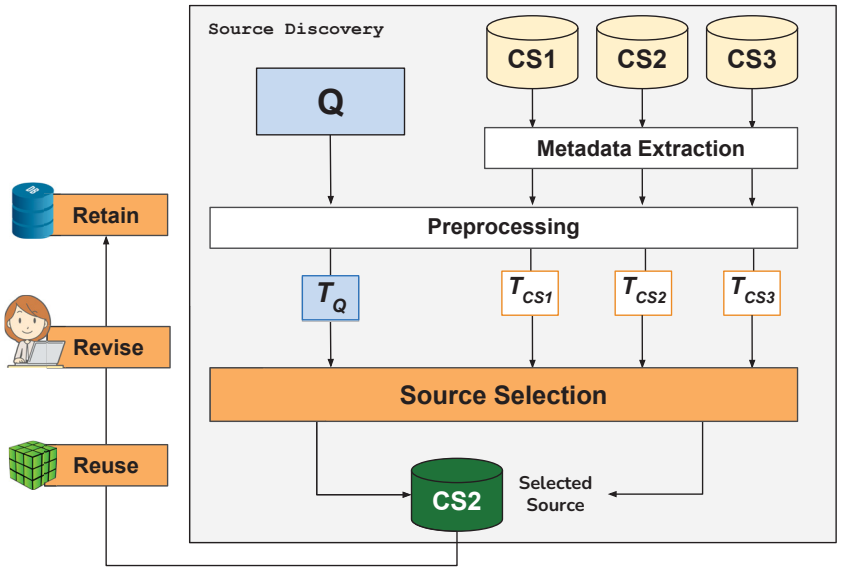


Figure 4.3: Source Discovery Model.

Table 4.1: Metadata from candidate sources example and matches based on value overlap.

T_{CS1}	report_date, region, county, school_name, **school_code, school_type, positive_students**, positive_teachers, positive_staff, **school_administered_tests**, **number_of_students**, number_of_teachers, number_of_staff
T_{CS2}	year, county_code, county_name, sector, **school_type, school_level_code, school_level**, legal_name, **undergraduate_fulltime_total, undergraduate_ft_enrolled_program, undergraduate_ft_not_enrolled_program, undergraduate_parttime, undergraduate_pt_enrolled_program, undergraduate_pt_not_enrolled_program, graduate_fulltime, graduate_parttime, all_students_fulltime, all_students_parttime**
T_{CS3}	american_community_survey_period, borough, borough_community_district_code, community_district_name, community_district_population, total_lep_population, percentage_population_proficient

4.1.2.1 Source Selection Methods

Accessing external data (e.g., Open Data) often represents an obstacle for regular users, due to the amount of data made available, its format and diversity. This subsection discusses three different methods that can act as source selection methods within the CBR's Source Discovery component: Cosine Similarity, Semantic Unionability Test, and LDA-W2V. Cosine Similarity is a popular and simple method for measuring similarity between two sets of data, as it only requires term-frequency vectors from the sets being compared. The Semantic Unionability Test [181] was proposed to determine the unionability between two sets of attribute metadata, which can be particularly interesting for measuring how close a data source is from a user question, based on schema or source description. Finally, the LDA-W2V algorithm is discussed, which performs topic modeling and extend the resulting topics with pre-trained word embeddings, allowing to add semantic meaning to the inference. The methods are detailed next.

Cosine Similarity. Documents can be represented by thousands of attributes that record the frequency of a particular word in the document. Thus, a document is an object that can be represented by a *term-frequency vector* [98]. Given two vectors, one can measure similarity

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

Figure 4.4: Term-Frequency Vector example. Adapted from [98].

between them by using Euclidean distance, angular separation, correlation, and others [216]. Among the existing methods, the Cosine Similarity is frequently used.

Cosine Similarity measures similarity as the angle between two vectors being compared, assuming that each word in a text or document corresponds to a dimension in a multidimensional space [87]. When measuring the angle of the documents, smaller the angle, higher the similarity. So, considering that cosine of 0° is 1, two vectors are said to be similar when cosine similarity is 1 [91]. Cosine similarity calculation is represented in Equation 4.1. For exemplifying, suppose that $\vec{A}$ and $\vec{B}$ are the first two term-frequency vectors from Figure 4.4, so that $\vec{A}$ = (5,0,3,0,2,0,0,2,0,0) and $\vec{B}$ = (3,0,2,0,1,1,0,1,0,1). By applying Equation 4.1, the cosine similarity between the two vectors is 0.94.

$$\cos(A, B) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| \cdot |\vec{B}|} \tag{4.1}$$

Cosine similarity has been widely studied in the past years and, besides Information Retrieval, it reaches several other application domains, such as medical diagnosis (e.g., for finding a probable disease based on a set of symptoms [195]), object tracking [175], and pattern prediction [205].

Semantic Unionability Test. The *Semantic Unionability test* (or *Sem-unionability*) was proposed by [181] and determines the probability of attributes being semantically *unionable*, i.e., belonging to the same domain. The proposal is applied in Open Data domain, aiming to find a related table, given a query table.

Based on the mentioned study, suppose we want to determine if two sets of attribute values, A and B, are belonging to the same semantic domain D. Thus, A and B, their metadata sets, and the intersection size between these sets are used to calculate Sem-unionability. The calculation follows a *hypergeometric distribution* [199]: suppose the semantic domain D contains the values of A as success values, and n_a samples are drawn from D without replacement. If a draw from D is in the intersection of A and B, it represents a *successful draw*. In the hypergeometric test, the number of s successful draws is used to calculate the *probability that A and B belong to the same domain*, being the intersection size $|A \cap B|$ the maximum value of s. Thus, if A is in D and B is drawn from D, the cumulative distribution of a hypergeometric distribution, F, using the actual intersection $t = |A \cap B|$, is defined as:

$$F(t|A, B) = \sum_{0 \leq s \leq t} \frac{C(n_a, s)C(n_D - n_a, n_b - s)}{C(n_D, n_b)} \tag{4.2}$$

where $C(m,n)$ is the combination of n items out of m items. Now, assuming $\hat{A}$ and $\hat{B}$ as metadata sets from A and B, respectively, and D as the disjoint union of $\hat{A}$ and $\hat{B}$, the Sem-unionability U_{sem} of attributes A and B is defined as:

$$U_{sem}(A,B) = F(\hat{t} \mid \hat{A}, \hat{B}) \tag{4.3}$$

where $\hat{t}$ is the intersection size of $\hat{A}$ and $\hat{B}$.

Example[3]: Consider two data tables, *universities* and *institutions*, from two different datasets, and *universities* = [Yale, Trinity, College of New Jersey, Butler University], *institutions* = [Bryant, University of Portland, Harvard, University of Delaware, Princeton, Columbia]. Suppose their metadata are $\widehat{universities}$ = [east-universities, west-universities, north-universities, midwest-universities], and $\widehat{institutions}$ = [north-universities, west-universities, east-universities, south-universities]. Considering $\hat{D} = \widehat{universities} \cup \widehat{institutions}$, $U_{sem}(\hat{t}= 3|\widehat{universities}, \widehat{institutions}, \hat{D})$ is 0.9857.

When querying open data tables, U_{sem} is a good alternative for determining the likelihood of their attributes being related. Considering the Source Discovery model and the goal of recommending a suitable data source based on the new case, the method can be investigated for measuring the closeness of databases metadata to the user question.

LDA-W2V Hybrid Algorithm. Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a corpus, based on the idea that documents are represented as random mixtures over latent topics, and each topic is characterized by a distribution over words [18]. Thus, given a document, paragraph, or sentence, LDA performs *topic modeling*, allowing to predict the main topics (e.g., subjects) covered by the text by assigning their probabilistic mixture.

A common LDA task is demonstrated in Figure 4.5: The document narrative is summarized as a mixture of three topics, whose words receive weights (proportions) according to their relevance, or frequency, in the narrative. We can observe that, in the example, LDA has determined the most relevant topics, since the words in topic 1 appear more often in the narrative compared to the words related to other topics [16]. The LDA model has clear internal structure that allows efficient inference, and it is independent of the training documents number, thus being suitable for handling large scale corpus [150]. In contrast, it presents a lower performance when applied to short texts, and a frequent lack of definite meaning caused by the isolated analysis of topic impact [251, 249].

For handling such drawbacks present in the classic LDA method, the authors in [117] propose a hybrid model called **LDA-W2V**. Specifically, the model joins LDA with Word2Vec algorithm [86], considering that word embeddings allow to capture semantics when processing vast amounts of linguistic data. First, it assumes a set of documents, which are preprocessed for obtaining a list of representative words or *tokens*. Each document set of tokens is given as input to the LDA algorithm, which predicts the topics (i.e., words and their proportions) that best describe the document content. For exemplifying, a document containing information on universities could be represented by a topic containing the following words and proportions: *0.048*"high" + 0.047*"education" + 0.033*"students" + 0.033*"university" + 0.032*"course" + 0.032*"public" + 0.032*"federal" + 0.031*"private" + 0.031*"administrative."*

The next step of the approach performs a Word2Vec (W2V) Extension, which aims to extend words in sources topics by using similar words acquired from Word2Vec model. The similarity is measured by Cosine Similarity (see Subsection 4.1.2.1): Supposing a topic word W_n, the W2V model is traversed to find similar words $[w2vWord_1, ..., w2vWord_n]$. Then, the

[3]Adapted from [181].

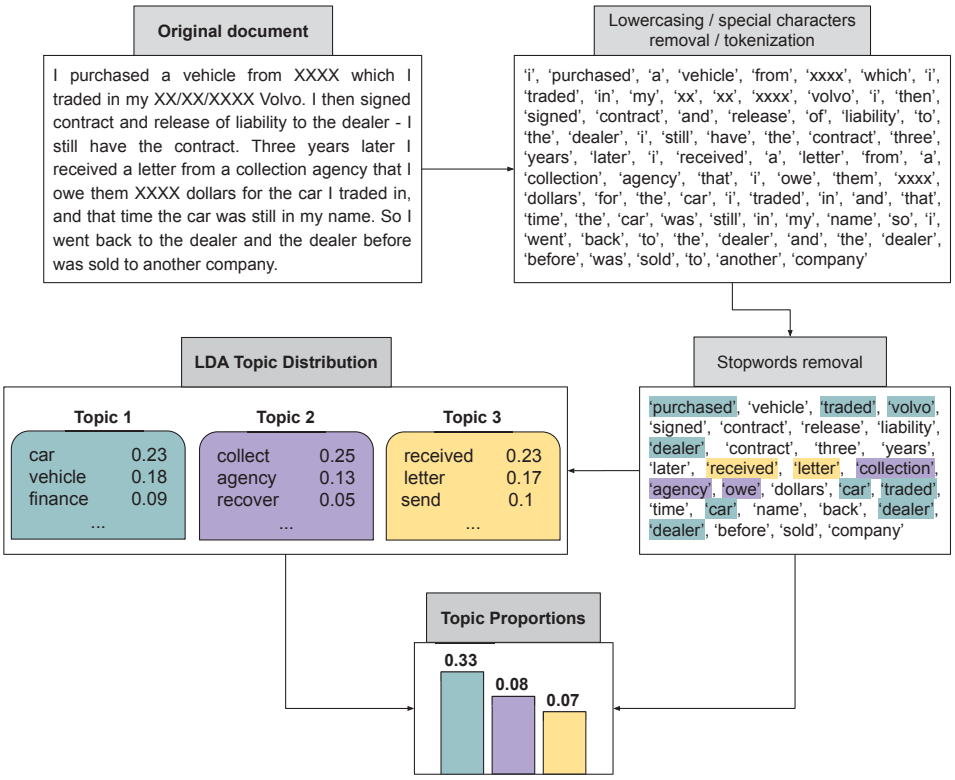


Figure 4.5: Illustrative example of LDA. Adapted from [16].

similarity score between a pair $[W_n, w2vWord_n]$ is multiplied by the LDA proportion score for W_n, obtaining a derived proportion DP_n. Each similar word found represents an extended word EW_n. Following the previous topic example, the W2V model retrieves the word *institution* as similar to the topic word *university*, with a similarity score 0.71. This score is multiplied by LDA proportion score for *university*, 0.033, thus obtaining DP_n=0.023. *institution* becomes an extended word EW_n, and generates a tuple $[W_n, DP_n, EW_n]$, e.g., [*university, 0.023, institution*]. After word extension, the approach tries to classify an input sentence (the test set) into a topic, based on the probabilities (DP_n) within all possible tuples. In short, the input sentence is assigned to the topic that contains the highest probabilities for each sentence word. As the extension with pre-trained word embeddings allows to add semantic meaning to the inference, LDA-W2V can be an accurate method in Source Discovery tasks.

For closing this subsection, we recall that Source Discovery is a well known research challenge [227, 263], which has been tackled by several studies. To mention a few, the Aurum system [68] flexibly finds relevant data through properties of the datasets and syntactic relationships between them; The DataMed approach [31] includes a Source Discovery task for finding relevant biomedical datasets from heterogeneous sources, making them searchable through a web-based interface. Recent approaches have also incorporated human knowledge, such as feedback, to improve discovery tasks [179, 269, 260]. This thesis aims to leverage human knowledge to improve situational data management. Based on the CBR architecture, a retrieved source should be used recommended data attributes gather proper feedback.

The next section addresses the *Reuse* phase of the architecture, explaining how the multidimensional data model is used to make recommendations, and how they are visualized by the user through a conversational interface.

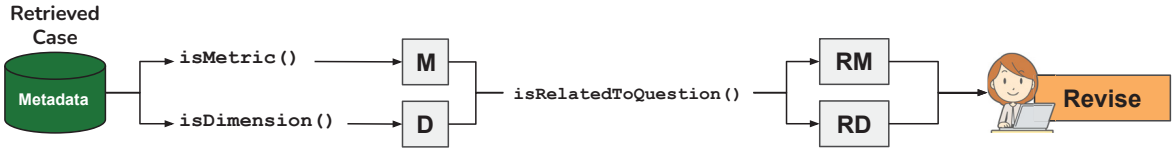


Figure 4.6: Reuse Activity Flow.

4.2 REUSE: SOLUTIONS BASED ON DIMENSIONAL DATA

In the Reuse phase, the case previously retrieved (either a database from the history or a database discovered on-the-fly) should be used to build a proper solution and recommend it to the user. This recommendation is based on Dimensional Data, i.e., the most suitable metrics and dimensions within the retrieved source are used for executing a query and returning an answer.

The idea is that, given the new case Q, the conversational system makes suggestions such as "*This is the metric most similar to your question*" or "*I found these dimensions as the most relevant ones*". For achieving this, the first step is to *differentiate* metrics and dimensions from a single set of attributes retrieved, as discussed next.

4.2.1 Differentiating and Recommending Data Attributes

Considering a case retrieved from the system history, and considering that the history maintains a structure such as the one in Figure 4.2, we can easily determine the solution to be recommended, since the sets of metrics and dimensions are already discriminated. However, considering that *Source Discovery* was executed in the Retrieval step, and not all databases are made available with a data dictionary, we may not have the same discrimination. Following the example from Subsection 4.1.2, if the candidate source CS2 was the retrieved one, it would be necessary to deal with the whole set of attributes T_{CS2} (i.e., the metadata, as shown in Table 4.1), by differentiating the sets of metrics and dimensions. This process is illustrated in Figure 4.6.

With the similar case retrieved and the corresponding database schema at hand (let us suppose a schema $\boldsymbol{S}$), we first execute a searching procedure for obtaining the sets of metrics $\boldsymbol{M}$ and dimensions $\boldsymbol{D}$. The sets $\boldsymbol{M}$ and $\boldsymbol{D}$ are read from $\boldsymbol{S}$ as shown in Equations 4.4 and 4.5. The function *is_metric* verifies whenever database scheme contains metrics (sum, min, max, average) while the function *is_dimension* returns dimension information. Next, it is necessary to build the set of meaningful attributes to be recommended to the user, gather his/her approval and proceed with the query execution (see Figure 4.1). The recommendation process is based on the subset of metrics and dimensions that are closer to the user input.

$$M \subset S \mid M = \sum_{j=0}^{S} is\_metric(j) \forall j \in S \tag{4.4}$$

$$D \subset S \mid D = \sum_{i=0}^{S} is\_dimension(i) \forall i \in S \tag{4.5}$$

Considering the user input from the previous examples, after preprocessing it we obtain a set of tokens T_Q. For recommending the most relevant dimensions and metrics, the metrics set $\boldsymbol{M}$ is traversed for finding those related to T_Q, resulting in a list of related metrics $\boldsymbol{RM}$ (see Equation 4.6). Similarly, the dimensions set $\boldsymbol{D}$ is matched with the T_Q, producing a set of related

dimensions ***RD***, as demonstrated in (Equation 4.7). At this point, the conversational system should return ***RM*** and ***RD*** to the user, which should evaluate their usefulness.

$$RM = \sum_{w=0}^{TQ} \sum_{m=0}^{M} w \subseteq m \forall w \in TQ, m \in M \tag{4.6}$$

$$RD = \sum_{w=0}^{TQ} \sum_{d=0}^{D} w \subseteq d \forall w \in TQ, d \in D \tag{4.7}$$

To exemplify, let us consider the previous T_Q <`graduate, student, enroll, public, school`>) and the selected source CS2, whose attributes are described in Table 4.1. Based on the T_{CS2}, we have ***M*** = <*undergraduate_fulltime_total, undergraduate_ft_enrolled_program, undergraduate_ft_not_enrolled_program, undergraduate_parttime, undergraduate_pt_enrolled_program, undergraduate_pt_not_enrolled_program, graduate_fulltime, graduate_parttime, all_students_fulltime, all_students_parttime*> and ***D*** = <*year, county_code, county_name, sector, school_type, school_level_code, school_level, legal_name*>. Based on the equations 4.7 and 4.6, ***RM = <graduate_fulltime, graduate_parttime>*** and ***RD = <school_type, school_level_code, school_level>***.

With ***RM*** and ***RD*** defined, the recommendation can take place. At this point, a conversational agent that reproduces a conversation with the user is needed, both to deliver information and to receive feedback. Thus, the next subsection presents the Question Answering prototype used in the CBR architecture, and features of the development framework for translating user inputs into intents.

4.2.2 Question Answering Prototype: Chatbot Application

For conducting the proposed CBR cycle, a Question Answering prototype was developed for handling all the interactions from the input question (the new case) to the retained case. The implementation covers a *chatbot application*.

The term *chatbot* indicates a software that simulates and reproduces an intelligent conversation with a user, by capturing requests and replying accordingly [49]. A chatbot can integrate a QA mechanism, but it has additional capabilities for exploring user experience and improving usability, learning, and memory retention [130, 82]. Indeed, many QA systems interact with the user in a way not yet compatible with how humans engage in conversations, especially if the search result presented to the user is a lengthy text that demands locating the needed answer [144]. On the other hand, chatbot interactions favor the incremental understanding of the user need through explanations and feedback gathering. Also, unlike information retrieval systems, chatbots do not need the user to have domain knowledge for formulating questions [151], which encourages their application for reducing querying complexity and answering queries related to information that might be unreachable [136].

For conducting a conversation, chatbots perform NLU tasks for extracting *intentions*, *entities*, and *context* from user inputs. An **intention**, or **intent**, represents a mapping between what a user says and what action should be taken by the chatbot [5]. An intention can be detected through **training sentences**, i.e., different ways a user could express the same goal. For instance, a "Greeting" intention can be recognized from training sentences such as "Hi", "Hello", or "Good morning" [44]. Matched intentions usually carry parameters referring to **entities**, which can be a city name, a date, or a location. Intentions also carry a **context** for storing current topic information and maintaining past intentions that guide the conversation [58, 5].

Typically, chatbots are extensively applied in businesses domains, for approximating companies and users, as they present the ability to solve users problems by creating pleasant conversations. However, their application has reached several other domains, including personal assistants [203], education [255], and public administration [151]. Most recently, we have witnessed the arrival and evolution of *ChatGPT*, a powerful chatbot based on the GPT architecture, which uses a neural network to process natural language and generate responses based on the input context [259, 145].

The chatbot prototype in this thesis was based on the *Xatkit* framework. Xatkit is an open-source chatbot development framework[4] that embeds a platform-independent modeling language to specify user intentions, computable actions, and callable services [44]. Specifically, the Xatkit modeling language is split up into two different Domain Specific Languages (DSLs) named *Intent* and *Execution* packages, which are used by the chatbot designer to specify intentions and actions. The Intent package is used to recognize the intention from training sentences, whereas in the Execution package, the bot reactions to user intents are defined using the state-machine formalism, comprising states and transitions. In the chatbot context, states can be understood as functions containing the bot logic (which are triggered depending on the captured intention), and transitions are used to navigate from one state to another.

Listing 4.1: Intention Definition within Xatkit.

```
1 val educationalData = intent("EducationalData")
2    .trainingSentence("I want to see schools in CITY")
3    .trainingSentence("Show me all schools in CITY")
4    .trainingSentence("How many students attended the ENEM exam in CITY?")
5    .trainingSentence("What is the literacy rate in CITY?")
6    .parameter("city-name").fromFragment("CITY").entity(city());
```

Xatkit has a mechanism for translating user inputs into intents, which is called *Intent Recognition Provider* [44]. Specifically, this provider is an abstraction layer that represents any NLP/NLU library that can be used with Xatkit for understanding the meaning of a question. In our case, an Intention Recognition Provider was implemented based on regular expressions for fast testing of the QA prototype, as demonstrated in Listing 4.1. So, if the user poses a question similar to any of the above training sentences, the intention **educationalData** is recognized. Also, when the recognition occurs, the Intention Recognition Provider triggers a corresponding *execution rule* for handling the intention. This could be, e.g., a state that receives the `CITY` parameter detected from the sentence and executes a query on a candidate source.

Considering the conversational approach architecture, the *Intent* and *Execution* packages embedded in Xatkit assume an important mediating role in the Source Discovery model, since an intention recognized from the Q sentence can call a state, or a sequence of states, for performing schema reading of candidate sources and metadata matching. Thus, the chatbot application built as QA prototype can execute all necessary procedures while maintaining a natural language conversation with the user. In addition to Xatkit being suitable for these purposes, it can be used with several NLP tools, ensuring its application in specific languages and vocabularies, such as Brazilian Portuguese. With respect to open data mining foreseen in the conversational approach, Xatkit is extensible, facilitating the integration of any external services as data sources, including open data APIs [57, 58].

The chatbot prototype for conducting the CBR cycle should manipulate dimensional data during the interaction, i.e, proposing metrics and dimensions of a retrieved database, based on a new case inside the cycle. In the related literature, several Question-Answering approaches explore data cubes for leveraging multiple aspects of data. The authors in [111] introduce the

[4]Available at: `https://github.com/xatkit-bot-platform/xatkit`

CubeQA approach, which converts a NL question into a SPARQL query and generates answers by indexing target datasets. Similarly, the study in [13] presents QA^3, a QA system based on tagging and a regex-like pattern language to describe query templates. So, the solution is able to choose the dataset that better "covers" the user question. The authors in [194] explore conversational interfaces for Business Intelligence applications by modeling the cube definition. This modeling approach provides important information on metrics and dimensions, allowing to establish query patterns used in dialog structures.

The CBR approach described in this thesis relates to the above-mentioned ones as it focuses on multidimensional data, also relying on a chatbot as conversational interface. Especially, the prototype serves as an access point to candidate databases, using their metadata to propose solutions to the user. After the recommendations, the next step in the CBR cycle is **Review**, where the user receives ***RD*** and ***RM***, repairing (if necessary) the proposed solution.

4.3 REVIEW AND RETAIN: REPAIRING A CASE BY USER ACTION

Human involvement is an important aspect of Case-Based Reasoning and Built-up Integration. For example, in SDI (a branch of Built-up terminology), once situational data are retrieved, the user may decide if they are suitable or not for the task at hand [3]. In the proposed architecture, when the related metrics and dimensions (***RM*** and ***RD***, respectively) are suggested through the conversational system, the user can make several case repairs following a conversation flow, until the expected result is retrieved.

4.3.1 Case Repairing Flow

Occasionally, metrics and dimensions recommended for query execution might not retrieve the answer the user is expecting. Besides, even before the query execution, the user might realize the suggestions are not adequate, either because the retrieved database is not correct, or because the attributes have divergent semantic respecting the input case.

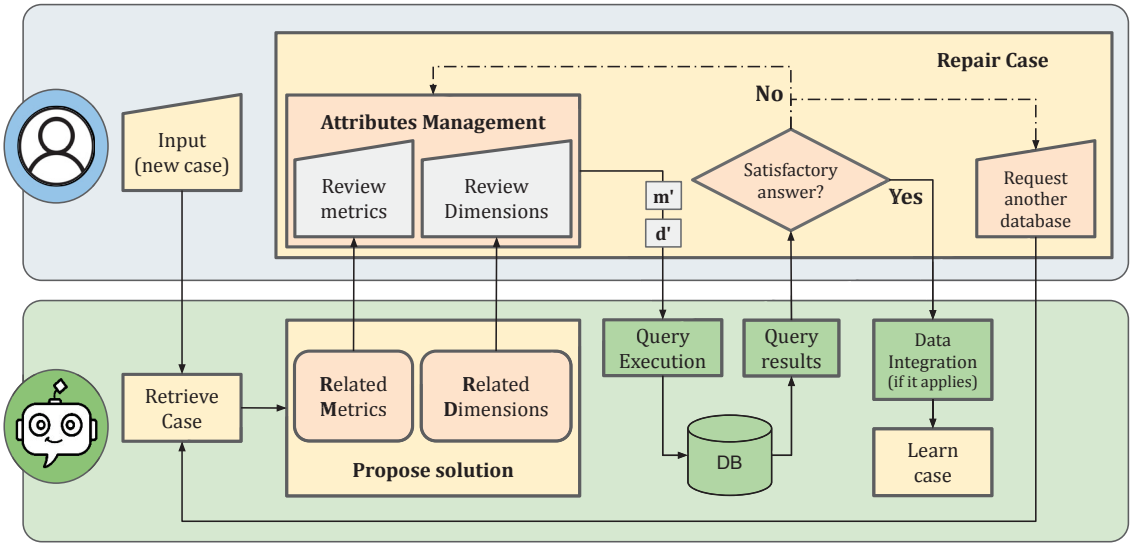


Figure 4.7: Case Repairing Flow.

In such situations, it is possible to repair the proposed solution, as demonstrated in the case repairing flow from Figure 4.7. The flow shows the CBR steps (identified as yellow boxes) assigned to two roles: the user role and the conversational system role. The main repairing

activity assigned to the user role refers to *Attributes Management*, where the user examines the sets of dimensions and metrics suggested (***RD*** and ***RM***), and chooses the subsets ***d'*** and ***m'*** of his interest. The reviewed subsets are then sent to the *Query Execution* process, where they are used as query parameters.

Taking as example the initial question/case "`How many graduate students are enrolled in public schools?`", and the proposed sets ***RM = <graduate_fulltime, graduate_parttime>*** and ***RD = <school_type, school_level_code, school_level>***, the user could choose ***m' = <graduate_fulltime>*** and ***d' = <school_type>*** as the most related attributes. Filters can be also chosen (e.g., *school_type = 'public'*, so, after collecting these data as query parameters, the query is built with them and executed in the database. Following the example, a possible SQL query with ***m'*** and ***d'*** could be *SELECT graduate_fulltime FROM selected_source WHERE school_type ='public'*, being *selected_source* the one reused after the Retrieval phase, from where ***RM*** and ***RD*** are derived.

After the Query Execution, the user receives the query results through the chat and evaluates their usefulness. This evaluation is decisive for the proposed CBR approach, since it determines whether the solution should be retained or discarded for future input cases. A solution (i.e., the attributes and database linked to the input case) is only retained when a positive feedback has been given, otherwise, if the answer was not satisfactory, the user can manage the suggested attributes again, choosing another subset ***m'*** and ***d'*** to compose a new query.

Another possibility, when the answer is not the expected one, is requesting *another database* to be queried, as demonstrated in Figure 4.7. Changing the database is a repairing activity that aims to propose a new solution, when the Attributes Management is no longer useful or viable. So, the user should be able to select another database manually, or choose the *Source Discovery* mechanism for automatically retrieving a relevant source.

Requesting another database to be queried not only covers cases in which the query results are wrong, but also cases in which the answer is *incomplete*. This topic will be addressed in the next subsection.

4.3.2 When the Answer is Not Enough

The answer to a user question may be spread over several databases [54]. This is also an assumption in Built-up Integration, which comprises, in addition to on-the-fly integration, an integration built from *complementary data*.

Following the example from the previous subsections, the SQL query *SELECT graduate_fulltime FROM selected_source WHERE school_type ='public'*, assumes that attributes of interest come from a single database. However, it is important to consider that information on "graduate students" might be available in one source, whereas information on "public schools" might be in another one. At this basis, during the case repairing flow (Figure 4.7), the user can state that the answer received is *partial* (i.e., incomplete), meaning that only a part of what was asked was indeed answered.

This action triggers a new round of the CBR cycle in order to search for the "missing part" of the answer in *another database* (which can be retrieved through Source Discovery or the history itself). The answer considered as *partial* is temporarily stored, and another round `Retrieve` → `Reuse` → `Revise` → `Query Execution` is performed, bringing fresh additional data. Several rounds can be performed in the Repair step, until a complete answer is found. When the additional round(s) is(are) finished, the parts of information from different sources are combined and sent to the user, and the case is ready to be *Retained*.

4.3.3 The Learning Step

When a case is solved, it is desirable to retain the plan in order to use it later for a similar problem. This if the task of improving the experience of the system [1, 226]. According to [226], the plan retention involves the decision of what to use as source of learning, e.g., by storing the whole case or an abstraction version of the case. The Retention phase proposed in this thesis bases on the last technique, *abstraction*, and performs a memory arrangement so the system can become a better recommender.

The Learning step relies on the history model shown in Figure 4.2. When a case is solved, i.e., when the user explicitly state that the received answer was good enough, three types of information are considered to rearrange the history: (1) the **input case** (i.e., the initial user question), (2) the **database** used to retrieve the answer, and (3) the **repaired solution** (metrics and dimensions) used in the Query Execution. When including this type of information in the history, the system is able to recommend them in the future, when an input case is similar to the one stored.

Based on the types of information mentioned above, two possibilities of memory arrangement can be performed:

- **Case Transformation:** When (1), (2), and (3) are already in the history. In this case, we assume the history was effectively used to propose a solution and no repair has been made by the user. Thus, the Usefulness Score of the historic case should be incremented, representing how much this case was reused in the past (see Subsection 4.1.1).

- **Case Addition:** When at least one of the information (1), (2), or (3) are not in the history yet. In this case, we assume that either Source Discovery was used to retrieve a solution, and/or the solution was repaired by the user (e.g., by changing the attributes to be queried). Thus, a new line is inserted in the history, containing a newly initialized Usefulness Score.

It is important to state that, when a new round of the CBR cycle is performed to repair an incomplete answer (see Subsection 4.3.2), Case Transformation and Case Addition may occur more than once. In other words, if we have, for the same user question, a partial answer from the *history* complemented with additional information from *Source Discovery*, the historic case would be *transformed*, whereas the external solution would be *added* to the case base. However, triggering a new round in the cycle is an optional step, meaning that should occur less often.

Concluding the Retain step of the cycle, for both types of memory arrangement (transformation and addition), the user question is abstracted in the case, meaning that the questions *tokens* are considered, rather than the raw question. This abstraction aims at refrain the lack of significance in stopwords, and the ambiguity they might cause when traversing the historic knowledge base.

Along this chapter, a CBR-based architecture for situational data management was presented and detailed regarding all its components. In the subsequent chapters, the experiments of this thesis are presented, involving each step of the CBR cycle. The next chapter the *Retrieval* step of the cycle, specifically the Source Discovery task.

5 EVALUATING THE DISCOVERY OF INFORMATION

A Source Discovery strategy must be evaluated with respect to choosing, among several candidate datasets, the most likely one to answer a situational question. This subsection describes the experiments from the Source Discovery component shown in Figure 4.3, which is part of the *Retrieval* step of the CBR architecture.

The objective of the experiment is comparing different similarity methods performing Source Discovery in a matching context, where a new case in the cycle should be assigned to the most similar dataset among the candidates given as input. For this, the methods Cosine Similarity, Semantic Unionability Test, and LDA-W2V (presented in Subsection 4.1.2.1) are set up, and the desired output is having an overview of their weaknesses and strengths. With this examination, it is possible to latter choose which method should compose the Source Discovery strategy in the conversational prototype.

This chapter is organized as follows: Implementation details such as candidate sources for performing Source Discovery, the methods configuration, and test questions are described in Subsection 5.1. The results are demonstrated in Subsection 5.2, first comparing the methods (Subsection 5.2.1) and then blending two techniques (Subsection 5.2.2). Finally, Subsection 5.3 discusses the outcomes from the discovery tasks.

5.1 IMPLEMENTATION DETAILS

This subsection discusses all elements needed for performing Source Discovery tasks, i.e., the candidate sources involved, the configuration of methods used for discovery, and the questions elaborated to test the methods performance.

5.1.1 Candidate Sources and Competency Questions

For performing experiments involving the Source Discovery Model from the proposed approach, eight candidate data sources were considered: FIES[1], INEP[2], PROUNI[3], CadUnico[4], School Census[5], IBGE[6], DataSUS[7], and Ibama[8], all of them extracted from Brazilian Open Data portals. This choice was made considering the many research opportunities involving Brazilian Open Data, particularly the need of providing solutions that can quickly retrieve decisive information [242, 184, 25].

Half of the data sources contain educational data from different contexts: **INEP** contains data on public and private higher education in Brazil, which comprises students, courses, institutions (e.g., universities) and their professionals. **FIES** and **PROUNI** store data on funding and scholarships for higher education students, respectively. The **School Census** dataset stores

[1]Available at: <http://dadosabertos.mec.gov.br/fies>.

[2]Available at: <https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/censo-da-educacao-superior>.

[3]Available at: <https://dados.gov.br/dataset/mec-prouni>.

[4]Available at: <https://dados.gov.br/dataset/microdados-amostrais-do-cadastro-unico>.

[5]Available at: <https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/censo-escolar>.

[6]Available at: <https://www.ibge.gov.br/>.

[7]Available at: <https://opendatasus.saude.gov.br/dataset>.

[8]Available at: <http://www.ibama.gov.br/dados-abertos>.

Table 5.1: Examples of Test Questions (English Translation)

Query	**Source**
How many rural schools have computers?	School Census
What is the average investment in education per city?	INEP
How many gypsy families in CadUnico attended a college course per year?	CadUnico
How many FIES beneficiaries are there for the Medicine course?	FIES
How many students that benefited from racial quotas dropped out of a course per year?	INEP

data on basic education (i.e., Elementary School), containing information about schools and their infrastructure, classes, enrollments, students and teachers. The other four datasets are of different subjects: **CadUnico** contains socioeconomic information on low-income citizens and families included in a government social program called *Cadastro Único*. The dataset covers information such as family/citizen income, education level, type of employment, region, among others. **IBGE** aggregates mainly social, economic and environmental indicators from Brazilian cities and states, including data from the demographic census, being one of the largest data sources on Brazil. **DataSUS** manages health information such as health indicators, epidemiological and morbidity information, healthcare networks, and service providers in Brazilian localities. Finally, **Ibama** dataset contains information related to environment actions: environmental quality control, use of natural resources, guidelines, environmental monitoring, prevention and control of deforestation, and so on.

For enabling the experiments, each candidate source schema description was manually extracted either from a CSV (Comma-Separated Value) file or a data dictionary in its respective open data portal. The extracted information was placed in an auxiliar CSV file that summarizes metadata from all sources, so that each row contains information from a different source. For simplicity reasons, along this chapter we will refer to this file as `sourcesMetadata.csv`. Each source content within this file was preprocessed (see Figure 4.3) through lowercasing, removal of special characters, stemming, and removal of stopwords, resulting in a set of meaningful tokens (represented by T_{CSn} in the Figure). All Source Discovery methods were implemented in Python, taking `sourcesMetadata.csv` as input, along with *competency questions*.

Based on the sources content, the competency questions were elaborated for assessing the model effectiveness. Two test sets were considered, containing 48 and 74 questions, respectively, from which a source should be inferred. The test questions and their correct answers (i.e., sources names) were defined manually, based on indicators available on data monitoring platforms SIMOPE[9] and LDE[10].

For composing the sets of competency questions, inclusion and exclusion criteria were considered. As **inclusion criteria**, **(I1)** the question content should be related to at least one of the candidate sources; **(I2)** the question should contain at least two entities (e.g., city, school); and **(I3)** the question should cover at least one metric (e.g., counting, average, maximum, and so on). As **exclusion criterion**, **(E1)** the question should not specify values for dimensions or entities, e.g., the question could ask about universities in general, but not about the Federal University of Paraná, specifically. This criterion was established to ensure an evaluation essentially based on *metadata* rather than data. Examples of the competency/test questions are shown in Table 5.1. As demonstrated in Figure 4.3, just as the candidate sources are preprocessed, resulting in sets of tokens T_{CSn}, the competency questions are also preprocessed, resulting in sets of tokens T_Q.

[9]Available at: <https://seppirhomologa.c3sl.ufpr.br/>.

[10]Available at: <https://dadoseducacionais.c3sl.ufpr.br>.

The Source Discovery component expects as input the tokens T_Q from the competency questions along with the candidate sources tokens T_{CSn}, and outputs the source with highest similarity respecting the input question. Next, we present how the methods were configured to deal with the discovery tasks.

5.1.2 Methods Setting

For detailing methodological aspects of this thesis, the current subsection presents how the methods Cosine Similarity, Semantic Unionability Test (U_{sem}), and LDA-W2V were implemented and adapted for their operation in the context of the CBR architecture.

5.1.2.1 Semantic Unionability Test

As explained in Subsection 4.1.2.1, the Semantic Unionability Test, or U_{sem} measure [181], aims to find a related data table given a query table A. Specifically, $U_{sem}(A, B)$ calculates the unionability between data tables A and B, based on their metadata sets and intersection size.

Algorithm 1 Flexible Intersection Algorithm.

Input data:
Set A (question tokens T_Q)
Set B (source metadata tokens T_{CSn})
1: *flexibleIntersection* = 0;
2: **for each** tokenA **in** A **do**
3: **for each** tokenB **in** B **do**
4: **if** tokenB.contains(tokenA)
5: *flexibleIntersection*++;
6: **end if**
7: **end for**
8: **end for**

In this experiment, rather than determining how close two data tables are, we wanted to determine how similar is a source to a query. So, we adapted the approach as a query-based unionability test, considering $A = T_Q$, and $B = T_{CSn}$, as metadata represent one or more data tables within a source. In short, each source information from `sourcesMetadata.csv` corresponds to a B set, and each test question corresponds to the A set. For exemplifying, suppose the query "`How many rural schools have computer`", which results, after preprocessing, in a set $A = [rural, school, computer]$. We also retrieve, from the candidate source content, a list of its metadata as B set. If this source is, e.g., the School Census, B could contain attributes such as *student_name, school_name, school_region, number_of_classrooms, number_of_computers*, and so on.

As stated in Subsection 4.1.2.1, the U_{sem} measure calculates the probability of A and B belong to the same semantic domain D, by following the hypergeometric distribution. Based on the study, the domain D was considered as the disjoint union between A and B. For the hypergeometric calculation, the approach considers value overlap (i.e., intersection size) between A and B, as demonstrated in equations 4.2 and 4.3. We realized that there might be *no intersection values* between the sets, even though the values are semantically related. Considering our example sets A and B (or simply T_Q and T_{CSn}), no intersection value would be returned, despite the similarities between $school$ in A and $school\_name$ in B, for example.

For handling this situation, a *flexible intersection* was performed, assuming that terms from A might be *contained* in terms from B and vice versa, so that intersection does not need to

contain exactly the same values in both sets. The process of finding the flexible intersection is demonstrated in Algorithm 1, which basically searches for each A token within a B token, and every time it is found, the flexible intersection is incremented.

At the end of the function, Sem-unionability can be calculated using *flexibleIntersection* as t value in Equation 4.3. For exemplifying, if we consider A=[*rural, school, computer*], B = [*student_name, school_name, school_region, number_of_classrooms, number_of_computers*], and $D = A \cup B$, U_{sem} = 0.8214. The Sem-unionability implementation is demonstrated in Algorithm 2. It is worth recalling that U_{sem} should be calculated for all candidate sources in the Source Discovery component, allowing to determine the most appropriate candidate source within the Retrieval step.

Algorithm 2 Sem-unionability Algorithm.

Input data:
Set A (questions tokens T_Q)
Set B (source metadata tokens T_{CS})
Flexible intersection $flexibleIntersection$
1: aux = 0;
2: U_{sem} = 0;
3: $domainSet$ = A + B;
4: **while** aux <= $flexibleIntersection$ **do**
5: U_{sem} += hypergeometricDistribution($domainSet$, A, B, aux);
6: aux++;
7: **end while**
8: **return** U_{sem};

5.1.2.2 LDA-W2V Algorithm

As shown in Subsection 4.1.2.1, the LDA-W2V approach joins LDA and word embeddings for performing text matching. We based on this approach for assigning an input query to one of the candidate sources in a Source Discovery task. First, each source information in `sourcesMetadata.csv` is sent as a corpus to the LDA model (*Gensim* implementation[11] was used). Multiple runs of the LDA were performed alternating the `num_topics` parameter in the model, which determines the number of latent topics to be extracted from each corpus. The parameter value ranged from 8 to 10, based on the coherence measure[12] for each source, which evaluates how coherent the produced topics are, by capturing their semantic interpretability on the LDA distribution.

Since all candidate sources are from Brazilian open data portals, the W2V Extension (see Figure 5.3) was implemented with a Portuguese pre-trained model from *FastText*[13]. The model receives the LDA topic distribution (words and proportions) and searches for similar words for the extension. For measuring the similarity between a topic word W_n and a model word, the Word2Vec's `similar_by_vector` method was used for finding the top-N similar words given a word vector. In the W2V Extension experiments, N ranged from 30 to 60, and the retrieval of model words similar to topic words was based on a similarity threshold, represented as K, which assumed value 0.45. Only similarity scores above K were multiplied with the W_n proportion score to derive the proportion DP_n (see Subsection 4.1.2.1). All LDA-W2V models and respective parameters are shown in Table 5.2.

[11] Available at: <https://radimrehurek.com/gensim/models/ldamulticore.html>.
[12] Available at: <https://radimrehurek.com/gensim/models/coherencemodel.html>.
[13] Available at: <https://fasttext.cc/>.

Table 5.2: LDA-W2V models

Parameter	**LW1**	**LW2**	**LW3**	**LW4**	**LW5**
num_topics	8	9	10	8, 10	8, 9, 10
top-N	30	30	60	50	50
K	0.45	0.45	0.45	0.45	0.45

After W2V Extension, we obtain tuples $[W_n, DP_n, EW_n]$ for each candidate source. So, the Source Selection step of the implementation receives and traverses all tuples, aiming to find the best correlations for an input sentence, i.e., a query. In other words, the query tokens T_Q (extracted after preprocessing) are used to search for equivalent W_n or EW_n with higher DP_n. When a match is found, an array for each candidate source is created, containing one DP_n for each query token T_Q. Otherwise, if a query token is not found in the extension, a default probability (0.000001) is inserted in the array.

For example, suppose the query tokens [*rural, school, computer*]. If all these tokens are found in the extension for the source **School Census**, the array for this source could be, e.g., [0.0024, 0.032, 0.0096]. The same query tokens for source **IBGE** could originate an array [0.0012, 0.0018, 0.000001], considering that "computer" token was not found in the extension. Thus, each source array will contain different probabilities, one for each query token. After all arrays are arranged, the probabilities average is calculated for each source, so the source with highest average is chosen as the most likely one to meet the input query.

5.1.2.3 Cosine Similarity

As stated in Subsection 4.1.2.1, Cosine Similarity $cos(A, B)$ measures similarity as the angle between two feature vectors A and B. As we wanted to measure the similarity between a query and a candidate source from `sourcesMetadata.csv`, we extracted the term-frequency vectors for each query and source.

First, the query and the source content were preprocessed, obtaining a query list and a source list. Then, Python Counter tool[14] was applied in the lists, allowing to map elements as dictionary keys and their counts as dictionary values, thus obtaining the term-frequency vectors. The vectors (query and source) are sent to the Cosine function, which performs the calculation shown in Equation 4.1. The calculation is performed for all candidate sources, so the source with highest similarity score is chosen as the discovered one.

In the next section, the results from Source Discovery evaluation are presented, addressing the methods discussed and the candidate sources from Brazilian Open Data.

5.2 RESULTS

The next subsections describe two analyzes that were conducted based on the methods previously characterized: the first one examines the methods separately, from where we extract advantages and challenging points of each method. Based on this comparative evaluation, the second analysis is based on a blended approach executing under the same setting conditions.

[14]Available at: <`https://docs.python.org/3/library/collections.html`>.

Table 5.3: Accuracy for different questions (Q) and sources (S)

Q/S	U_{sem}	**Cosine**	**LW1**	**LW2**	**LW3**	**LW4**	**LW5**
48/5	**83.33**	**85.42**	77.08	75.00	**81.25**	**81.25**	77.08
48/8	**83.33**	**85.42**	72.92	75.00	77.08	**81.25**	75.00
74/8	**86.49**	**85.14**	71.62	71.62	75.68	**81.08**	72.97

5.2.1 Cosine Similarity, LDA-W2V, and U-Sem Test: Comparative Evaluation

This section describes the results obtained with Cosine Similarity, Unionability Test and LDA-W2V performing Source Discovery tasks. The evaluation considered the competency questions described in Subsection 5.1.1, and consisted of three parts: First, we evaluated the test questions with a smaller set of candidate datasets; next, we added the remaining datasets, maintaining the set of test questions; finally, we added more questions to the test set. The variations aimed to investigate possible changes in matching results.

Thus, in the first moment, the evaluation was conducted with **Prouni, School Census, FIES, INEP**, and **CadUnico** sources, being the latter the only source that does not contain education data. The objective was to verify whether the accuracy would be satisfactory with data sources containing similar metadata. Concerning the Semantic Unionability Test (U_{sem}), the ideal source was chosen for 83.33% of the test questions. For Cosine similarity, the accuracy rate reached 85.42%. For LDA-W2V models (see Table 5.2), the highest accuracy was 81.25%, achieved by LW3 and LW4. The accuracy values obtained in the first evaluation are shown in the first line of Table 5.3, with the highest values highlighted in bold. The results by source are summarized in Figure 5.1.

In general, considering five data sources and a test set containing 48 questions, only one question received the wrong match by all approaches. For six questions, most of the approaches (including all LDA-W2V models) made the wrong match. However, for the remaining questions (41), the correct match was given by the majority. It is interesting to highlight that four out of five datasets contained information on Education, representing a more challenging matching task compared to predicting distinct datasets; yet, the accuracy rate was over 81% for the best models.

Due to this content similarity, a second round of evaluations was conducted, adding the remaining data sources (**IBGE, Ibama**, and **DataSus**) to investigate whether the inclusion of sources, distinct from each other and from the first five, could impact the models accuracy. For this execution round, the accuracy rates had no marginal loss: 72.92% for LW1, 77.08% for LW3, and 75% for LW5. The other models had no score changes, and the best models were U_{sem}, Cosine, and LW4, with accuracy values 83.33%, 85.42%, and 81.25%, respectively. In this second run, 2 questions were incorrectly assigned by all approaches and 7 questions by the majority. The remaining 39 questions were correctly assigned by the majority. Concerning the best three models, 33 questions were correctly assigned by all of them, and 7 were correctly assigned by at least two. This means that, even in cases where the right answer is not given by all approaches, a Source Discovery task could be highly effective by applying a voting mechanism such as Majority Rule [173], which determines the final result as the one selected by most methods in a voting group.

In the third evaluation round, more questions were added to the test set, aiming to verify the models consistency. In total, 74 questions were given as input to the models, considering all eight candidate sources. The best models remained the same (see Figure 5.2): U_{sem} (86.49%), Cosine (85.14%), and LW4 (81.08%), as demonstrated in the last line of Table 5.3. Two questions

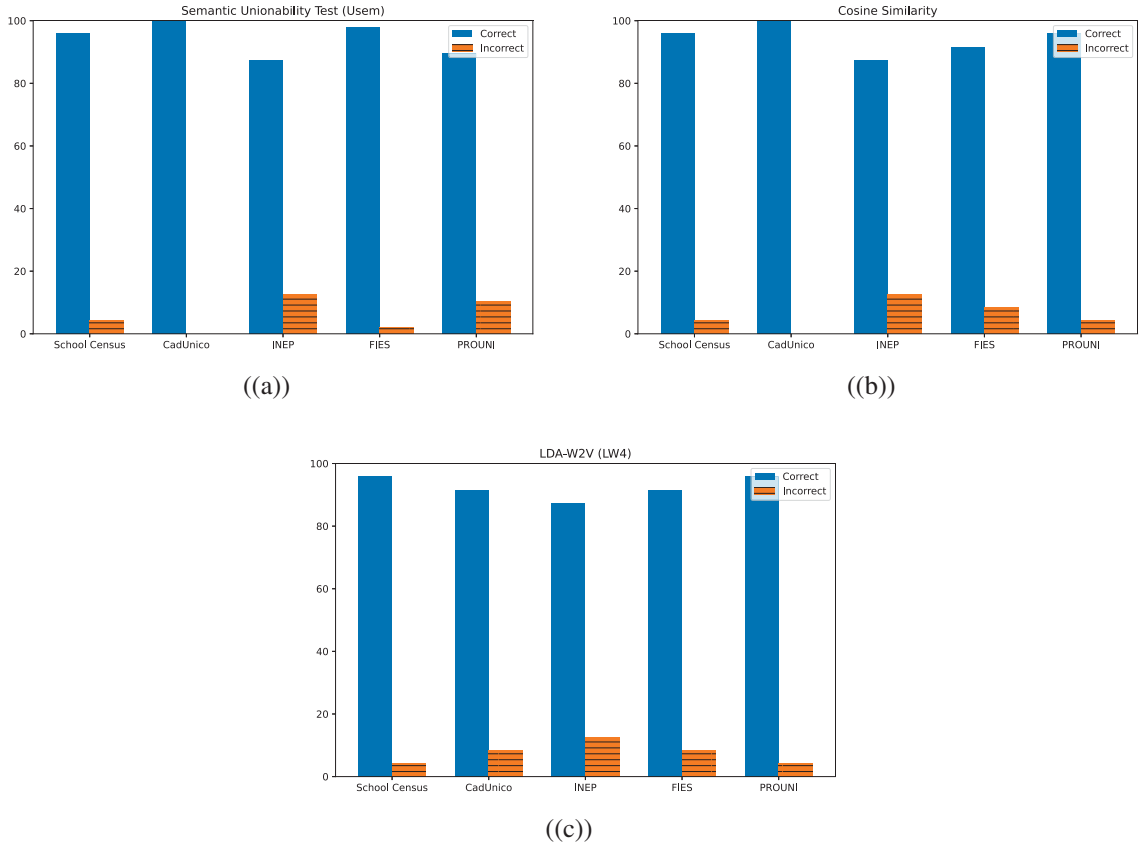


Figure 5.1: Matching results with different methods.

were incorrectly assigned by all approaches. Considering only the best three models, 51 questions were correctly assigned by all of them, and 12 questions were correctly assigned by at least two.

In general, most of the erroneous matches occurred for the same questions in the test set, despite being resulting from different forms of processing. In addition, in some cases, different datasets were predicted for the same question, allowing us to observe the particularities of each method and infer their strengths in the Source Discovery context. Let us consider, e.g. the input question "*Which educational institutions offer full* `Prouni` *scholarship?*". All methods (including all LDA-W2V models) made the correct match, i.e., **Prouni**, except for U_{sem}, where **INEP** was the chosen source with similarity score of 100%.

In fact, $U_{sem}(A, B)$ is strongly influenced by the intersection size between the sets being compared, as well as B size (which, in this experiment, assumed the source metadata size). The intersection set found between the preprocessed question (set A) and INEP metadata ([*education, institution, offer, full, prouni, scholarship*]) was larger than the intersection found between the question and Prouni. Considering the hypergeometric calculation and the number of successful draws that determine relatedness in U_{sem} (see Subsection 4.1.2.1), INEP was chosen primarily. Another example of U_{sem} miscomprehension was observed in question "*Number of master's and doctoral courses offered by city*", whose ideal source was **INEP**. In this case, U_{sem} returned **School Census** as the right source. Although the intersection set size was the same between the question and INEP, and between the question and School Census, the metadata size of the latter was smaller, which influenced in the probability calculation. These observations may indicate that U_{sem} measure is more assertive when A and B sets have similar sizes. Despite that, the adaptation showed promising results, especially in the last round of tests, where it obtained the highest accuracy.

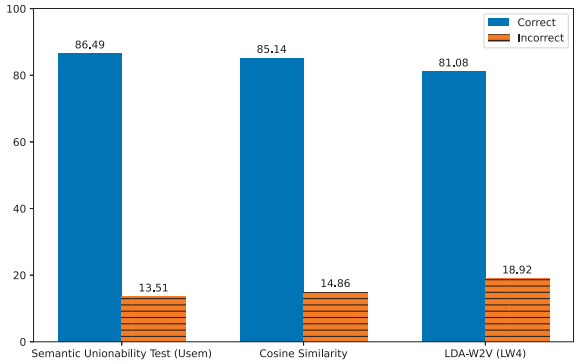


Figure 5.2: Comparison between all approaches - third evaluation round

Cosine similarity was the most assertive method, performing correct matchings in more than 85% of the cases. Although its results were satisfactory, we observed that most of erroneous matches involved the **INEP** dataset (see Figure 5.1). Besides that, we observed that most of mispredictions were quite understandable: for questions where INEP was chosen, the correct source was actually very similar to INEP with respect to question content. E.g., the question "*What is the average monthly fee of funded courses?*" has its answer in FIES dataset; Cosine has chosen INEP for this query, which also contains information about funded courses. In contrast, for that same question, LDA-W2V models choose Cadunico, which is a data source not related to education.

LDA-W2V was the method with the lowest accuracy (around 81% for the best model). When observing the erroneous matches, we realized that predictions made with this method can be highly variable depending on the probabilities assigned in W2V Extension step (see Subsection 5.1.2.2). Let us consider the preprocessed question [*course, master, doctoral, offer, city*], whose predicted source was **FIES**, and ideal source was **INEP**. When searching for the query tokens within INEP extension, the probability array average for this source was smaller than the average for **FIES**. Specifically, FIES array contained higher probabilities corresponding to similar words *student* and *town*, whereas smaller probabilities were found INEP extension for similar words *class* and *location*, thus causing an erroneous match. As a consequence of this behavior (observed for most error cases), we realized that the presence or absence of a single word in a test question, and its respective probability value obtained in the W2V extension, could significantly change the resulting array average and cause an unexpected match.

Despite the encouraging results with the methods (where half of the data sources used were related to Education), it was observed that LDA-W2V performance might be improved if its probability arrays were more informative. Similarly, Cosine Similarity might present better results when considering context information. Thus, the next subsection presents a hybrid approach that adapts LDA-W2V and Cosine methods, combining them for performing source selection.

5.2.2 Blending Topic-Based Embeddings and Cosine Similarity

In the previous subsection, it was observed that Cosine similarity presented good coherence, but it may fail in determining similarity when context information is not available. For LDA-W2V method, we observed a large variation in the assignment of probabilities for similar words in the sources, causing erroneous matches. These factors have motivated an additional investigation, by combining LDA-W2V and Cosine Similarity to leverage both syntactic and semantic capabilities

for discovering data sources that match a situational question. The approach is demonstrated in Figure 5.3.

The approach takes as input a set of candidate sources and the new case (the user question) Q. Each candidate source content is preprocessed (see the methods in Subsection 4.1.1), resulting in a set of tokens for each source (T_{CSn}). Q is also preprocessed, resulting in a set of tokens T_Q. T_{CSn} and T_Q are sent to a Vectorizer and a Cosine Similarity module, responsible for the tasks described in Subsection 5.1.2.3. The output from the Vectorizer are term-frequency vectors V_Q and V_{CSn}, so the Cosine Similarity cos_{CSn} is the distance measured between these vectors. The output from this module is the Cosine Similarity for each candidate source.

While this process occurs, the sources tokens T_{CSn} are given as input to the Topic Detection task, which outputs several topics for each source, represented by words (W_n) and their proportions (P_n). The component LDA-W2V receives the topics and performs Word2Vec extension as described in Subsection 5.1.2.2. Next, in the Matching step, T_Q tokens are used to search for equivalent W_n or EW_n with higher DP_n, As already mentioned, whenever a match is found, a probability array (represented by PA_{CSn} in Figure 5.3) is created for each candidate source.

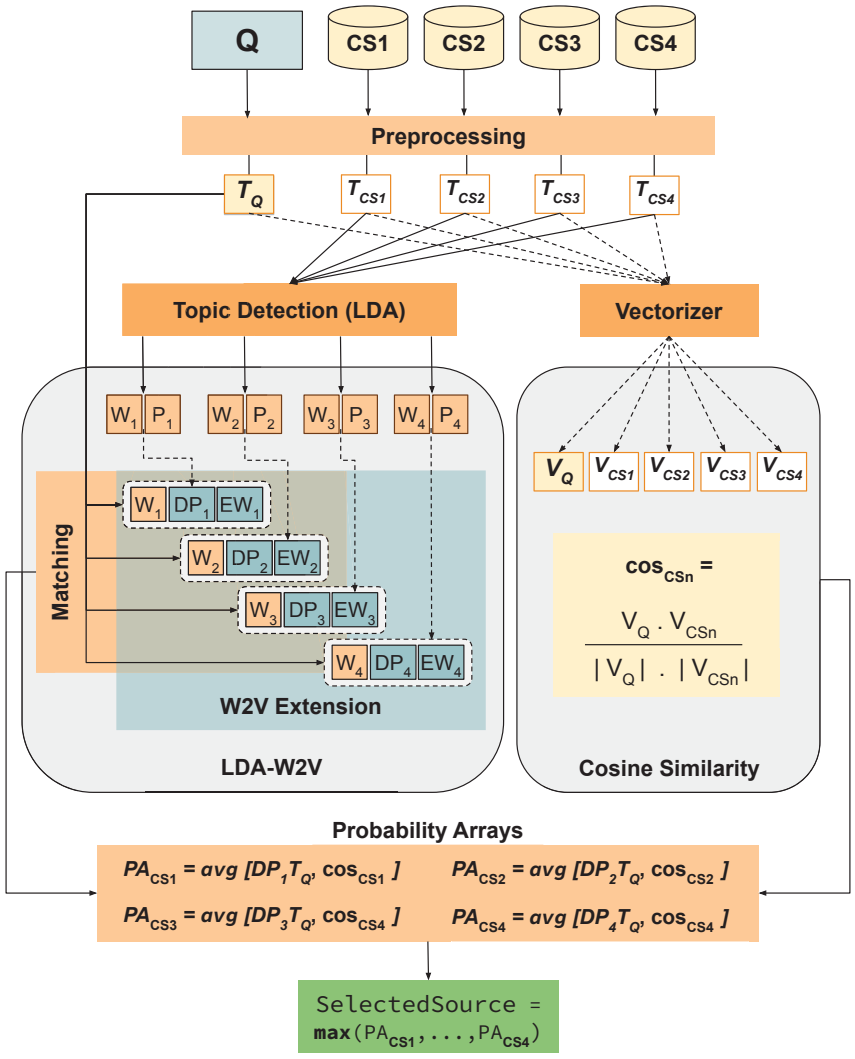


Figure 5.3: Overview of the hybrid approach based on Cosine Similarity and LDA-W2V.

In the LDA-W2V setting described in Subsection 5.1.2.2, an array PA_{CSn} contains one derived probability DP_n for each query token in T_Q. As the objective of the present experiment

Table 5.4: Source selection accuracy in three evaluation rounds with different sources (S) and questions (Q).

S	Q	Cosine Sim.	LW4 (LDA-W2V)	**LDA-W2V + Cosine**
5	48	85.42	81.25	**93.75**
8	48	85.42	81.25	**93.75**
8	74	85.14	81.08	**87.74**

was making the array more syntactically and semantically informative, the Cosine Similarity for each source cos_{CSn} is also inserted in the array, and next, average probability for each array is calculated. Finally, the source(s) most likely to meet Q is/are selected by considering maximum average(s).

For exemplifying, let us extend the example from Subsection 5.1.2.2, with T_Q = [*rural, school, computer*] and candidate sources CS1 (**School Census**) and CS2 (**IBGE**). After W2V Extension and T_Q matching, the sources are represented by probability arrays PA_{CS1} = $[0.0024, 0.032, 0.0096]$ and PA_{CS2} = $[0.0012, 0.0018, 0.000001]$, respectively. After Cosine Similarity calculation, cos_{CS1} is *0.85* for CS1, whereas cos_{CS2} is *0.7* for CS2. Both values are included in their respective probability arrays, resulting in PA_{CS1} = $[0.0024, 0.032, 0.0096, 0.85]$ and PA_{CS2} = $[0.0012, 0.0018, 0.000001, 0.7]$. After the arrays are arranged, the probabilities average is calculated for each source, so the source with highest average is chosen as the most likely one to meet the input query. In this example, CS1 would be the selected one.

For hybrid approach (LDA-W2V + Cosine Similarity) the eight candidate data sources and competency questions from Subsection 5.1.1 were considered. For this experiment, only LW4 model was considered for LDA-W2V, since it presented superior results (see Table 5.3). Three evaluation rounds were also performed, alternating between the test sets (48 and 74 questions) for investigating possible changes in matching results. The blended LDA-W2V-Cosine was compared with Cosine and LDA-W2V individually for observing accuracy variation; the results are summarized in Table 5.4.

In the first evaluation round (with sources Prouni, School Census, FIES, INEP, and Cadunico), Cosine and LDA-W2V reached 85.42% and 81.25% of accuracy, respectively, whereas the blended model reached 93.75%. The result was promising, since four out of five datasets contained information on Education, representing a more complex matching task compared to inferring datasets with very distinct information. Moreover, as we can see in Figure 5.4, there was a significant prediction improvement for each of the five datasets in comparison to Cosine and LDA-W2V predictions, especially for School Census and Cadunico datasets, where no incorrect match occurred.

In the second round, we conducted the evaluation with all data sources and the test set containing 48 questions. No changes were observed; the blended model achieved the best result, with 93.75% of correct matches. In the third round, using the test set containing 74 questions and the same eight open data sources, a small decrease was observed in all models. LDA-W2V reached 81.08% and Cosine accuracy reached 85.14%, while the blended model accuracy reached 87.74%. This decrease may be caused by the structure of some test questions, with more general content. E.g., if a question is mostly composed by words that are common in many candidate sources, an incorrect match might occur. The accuracy reduction can also be due to the stemming method applied in preprocessing stage, since the reduction of some Portuguese words may originate ambiguous tokens, thus confusing the algorithm. Despite that, the blended approach had superior results than the isolated methods.

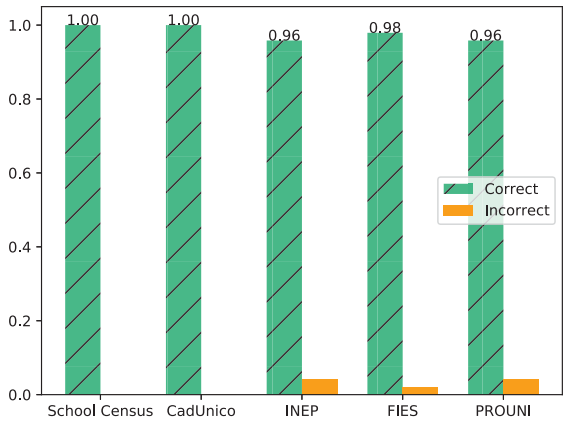


Figure 5.4: Source selection accuracy with blended LDA-W2V-Cosine (1st evaluation round).

5.3 CHAPTER REMARKS

Source Discovery tasks can enhance the access to information, and most importantly, provide support to the user by inferring the data set most likely to answer a query. Regarding the access to Open Data, in a particular way, many challenges arise involving data format, size, and visualization. Discovering the right source is a way to decrease users manual work and the complexity of finding the right information.

As mentioned in the beginning of Section 5, the objective of Source Discovery experiment was comparing different similarity methods performing source selection in the context of the proposed CBR architecture, where a new case in the cycle should be assigned to the most similar dataset among the candidates. Thus, a comparative study was conducted throughout this section, involving three different approaches (Semantic Unionability Test or U_{sem}, Cosine Similarity, and LDA-W2V) and evaluating their performance when inferring data sources suitable to a question. Each method received the metadata of eight candidate datasets extracted from open data portals, along with different test sets containing competency questions. Three evaluation rounds were performed with each method by alternating the sources and test sets, which resulted in accuracy values over 81%. Then, as an additional investigation, we presented a Source Discovery approach that joined topic-based embeddings from the LDA-W2V algorithm and Cosine Similarity for inferring a data source (based on the same eight candidate datasets and same test questions). By blending both measures, the accuracy was above 93%.

By comparing the execution of the three methods applied, some characteristics were observed. First, considering a pair (candidate_source, question), U_{sem} measure seems highly influenced by the intersection size between the sets, as well as the size of source content. Cosine similarity, besides good accuracy, presented good coherence: in cases where it did not predicted the right source, it inferred a source of very similar content, unlike most LDA-W2V models. For LDA-W2V, we observed a large variation in the assignment of probabilities for similar words in the sources, causing some incorrect matches. Respecting the hybrid model proposed (LDA-W2V + Cosine), the accuracy rate was considerably superior than LDA-W2V and Cosine evaluated separately, which demonstrates a good potential towards data transparency and user support.

In general, all methods showed good results: considering that half of the open datasets had very similar content, and data itself were not used for prediction, the accuracy rates demonstrate high value for Source Discovery tasks. Thus, they could be applied as discovery strategies within

the CBR architecture, while observing development requirements and/or restrictions that may exist in a prototype (e.g., computational cost). Taking as example the third evaluation round in the comparative study (see Subsection 5.2.1), we observed that most of the questions were correctly assigned by all of the best three models, whereas some questions were correctly assigned by at least two models. As already mentioned, applying a voting method such as Majority Rule [173] could solve eventual "disagreements" between the methods, by selecting the most voted source.

However, the results showed a few cases in which the wrong source was inferred by most of the methods being analyzed. In these cases, applying a voting mechanism would be ineffective, since the most voted source would be the undesirable one. In contrast, if we consider a Source Discovery activity running in a conversational CBR solution, the user feedback could be collected in the *Review* step and fix the inference. In other words, the user could state that the recommended source is not the expected one, and indicate the adequate option, which would be used for improving the system.

Thus, based on the proposed architecture, the next section discusses the implementation of a conversational interface to interact with the user and propose solutions, so that human feedback can be obtained aimed at incremental learning.

6 EVALUATING THE REUSE OF SOLUTIONS

This section presents experiments related to the *Reuse* step of the conversational architecture presented in Section 4. Reuse relies on the recommendation of solutions to the user, which should be reviewed in the latter step, *Review*. For this to be possible, a proper mechanism for interaction is necessary. Thus, the experiment involves a chatbot conversation flow that was implemented in the conversational prototype (see Subsection 4.2.2) for interacting with the user and making recommendations.

The objective of the second experiment is investigating how metrics and dimensions can be proposed to the user within the *Reuse* step of the proposed CBR architecture, after a source has been selected. As the chatbot prototype is used for proposing these solutions, a subgoal of the experiment is also assess the user acceptance regarding visibility, support, usefulness, and simplicity during the interaction. Thus, the experiment also presents an empirical user study conducted for the chatbot evaluation, which addresses the target features in a systematic way.

Along this Chapter, implementation details such as the conversation flow implemented within the chatbot prototype (which covers intention recognition functions, query building, and additional requirements for the interaction) are described in Subsection 6.1. Besides the prototype, the *Reuse* step evaluation is supported by real participants and thoroughly addressed in Subsection 6.2. The subsection details how the evaluation took place, what qualitative criteria were taken as basis, as well as statistics about the participants answers. The results are discussed in Subsection 6.3.

6.1 IMPLEMENTATION DETAILS

The interaction between users and the chatbot prototype presented in Subsection 4.2.2 is based on multidimensional querying. Specifically, the chatbot captures user intentions and implements a conversation flow for linking these intentions to multidimensional metadata. For this task, the bot should recognize terms informed by the user that match the retrieved source's metadata, using these correspondences for guiding the user along all interaction and for building query parameters.

This subsection presents the implementation details involved in the second experiment, such as the conversation flow that the chatbot executes for guiding the user through query formulation, the database accessed for retrieving and suggesting attributes, and aspects of the interaction regarding natural language.

6.1.1 Chatbot Conversation Flow

The user interaction with the chatbot is demonstrated in the flowchart in Figure 6.1, where the bot actions are defined by *states and transitions*. States represent the chatbot methods triggered according to the captured intention, whereas transitions are the interactions needed to create dimensional queries and navigate from one state to another.

The conversation starts with a query sentence (i.e., an input), from which we recognize the user intention. Assuming that the input sentence may contain terms related to database attributes, it is preprocessed for finding all meaningful keywords, which will be used in the **Reads Database Schema** state. This state searches the keywords in the database schema, aiming to find related metadata. In this step, dimensions and metrics of interest are returned to the bot, so it

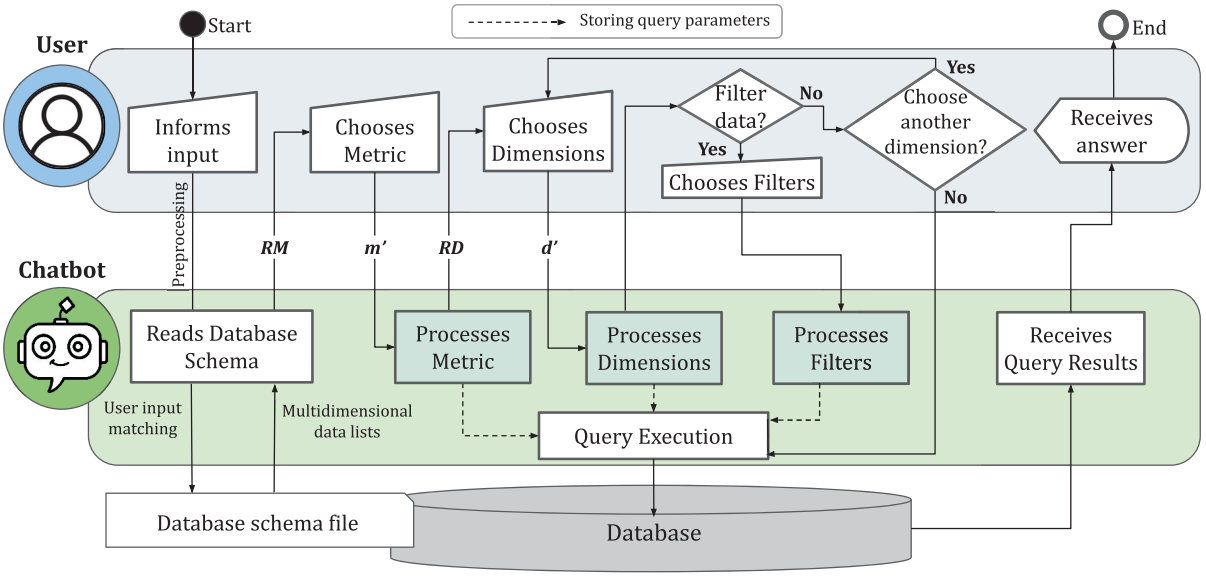


Figure 6.1: Chatbot state flowchart for proposing a solution.

can make appropriate suggestions to the user. These attributes are represented by the sets ***RM*** and ***RD***, which are built as discussed in subsection 4.2.1.

The metrics set ***RM*** is the first one suggested to the user in the conversation. So, following the flowchart sequence, the user chooses one metric of interest from this set (***m'***, as explained in Subsection 4.3.1). The bot processes the chosen metric in **Processes Metric** state, temporarily storing it as query parameter. Next, it asks for a dimension for complementing the chosen metric, which is also processed as ***d'*** in **Processes Dimensions** state, and stored for later use. After choosing a dimension, the user is asked whether he wants to filter data, i.e, search for specific values within the dimension. When the user specifies a filter, the chatbot processes it in **Processes Filters** state. Otherwise, if no filter was specified, the chatbot will consider all possible dimension values in its search. After deciding on a filter, the user also decides about including more dimensions in the search; thus, if an additional dimension is chosen, the bot returns to the state **Processes Dimensions**. When all dimensions have already been chosen, the state **Query Execution** is called in the sequence, which is responsible for effectively accessing the database. In this state, the chatbot has information on a metric, dimension(s), and filter(s) (if specified), i.e., the query parameters. After query execution with the parameters, the chatbot receives the results, finally presenting them to the user.

Practical Example: Suppose that a user inputs the query "*show me schools per cities and states*". After database schema reading performed from the query keywords, the chatbot presents a list of related metrics (***RM***) such as `schools count, schools average`, and so on. The user chooses `schools count` (***m'***). The bot then suggests a list of related dimensions (***RD***), from which the user chooses `city name` (***d'***). The user does not want to filter by city, but wants to include another dimension (`state name`) in the query, this time filtering by the American state `Texas`. The bot then stores all dimensional information as query parameters, executing it and returning an answer about schools count in each city of Texas.

With the steps described, the user is able to access multidimensional data through an interface, exempting specific knowledge about the database content or data format.

6.1.1.1 Intention Recognition and Query Building

As observed in Figure 6.1, the query execution state demands to process dimensions, filters, and metrics informed by the user, storing them as parameters for query execution. In the conversational interface (i.e., the chatbot), this task is assisted by an *Intent Recognition Provider* [44] that translates user inputs into intents, which are then used for triggering actions. Every time the user informs a dimension, metric, or filter, the Intent Recognition Provider uses regular expressions to match these inputs to an intent by means of training sentences (see Subsection 4.2.2). Then, assuming a matched intent, existing dimensional values in the input are recognized as entities within the intent, so the bot is able to use these entities as query parameters.

Practical Example: Following the previous practical example, suppose that after the user had chosen the `state name` dimension, the bot asked for filters related to it, to which the user answered "*filter by* `Texas`". Suppose, in addition, an intention called *FiltersForQuery*, whose training sentence is "*filter by* `FilterValue`". The user reply should match this training sentence, and the value "`Texas`" fits in the parameter `FilterValue`. Hence, the intention *FiltersForQuery* is recognized, triggering the action state *Processes Filters* (see Figure 6.1), where the entity "Texas" is stored as a parameter for Query Building.

The Query Building depends on multidimensional information such as dimensions, metrics, and filters. So, after collecting these data as query parameters, the query is built with them and executed in the database. It is worth mentioning that if no filter was recognized in the user's intentions, the query returns an answer considering all possible values within the chosen dimension(s). In the practical example, the lack of a filter means that the chosen metric (e.g., "schools count") would be shown for every possible state value within the database.

With the query execution, the flowchart represented in Figure 6.1 is completed, and the user is able to visualize the query answer on chat. Next, the case study and execution examples are presented.

6.1.2 Database Characterization

The experiment comprised an integrated repository called BIOD (Blended Integrated Open Data), which contains Brazilian open data from two open databases, named SIMMCTIC[1] and LDE[2]. The project was created by C3SL Research Group from Federal University of Paraná (Brazil) for making open data more accessible, and it is publicly available as a microservice[3]. It integrates and makes available more than 300Gb of data containing more than 1700 attributes (dimensions and metrics) and about 2.5 billions of records from originally disconnected data sets. The repository data is made available through *BlenDB*[4], a tool that allows the repository to be accessed through a RESTful API, by informing query parameters in the URL [206]. The queries in the API consider the multidimensional data model, thus assuming the following format:

$$\begin{aligned}&\text{http://biod.c3sl.ufpr.br/api/v1/data?metrics=}\ METRIC_1,METRIC_N\&\\ &\text{dimensions=}\ DIMENSION_1,DIMENSION_N\&\text{filters=}\ FILTER_1,FILTER_N\end{aligned}$$

Due to this access format and data structure comprising a clear database schema file, BIOD was chosen as a suitable repository for metadata experiments. The database schema is contained in a YAML file containing all available relations, metrics, and dimensions described in

[1]Available at: `https://simmctic.c3sl.ufpr.br/`
[2]Available at: `https://dadoseducacionais.c3sl.ufpr.br/`
[3]Available at: `https://biod.c3sl.ufpr.br/index_en.html`
[4]Available at: `https://gitlab.c3sl.ufpr.br/c3sl/blendb`

Brazilian Portuguese [47]. An example of this YAML file is shown in Listing 6.1. For simplicity reasons, it only shows information related to *city* dimension.

Listing 6.1: Snippet of YAML file containing the database schema (English translation).

```
1
2  alias: "view:city"
3      dimensions:
4          - "dim:city:id"
5          - "dim:city:name"
6          - "dim:state:id"
7      metrics:
8          - "met:count:city:id"
9  dimensions:
10     -
11         name: "dim:city:id"
12         dataType: "integer"
13          description: "Id of Brazilian city"
14     -
15         name: "dim:city:name"
16         dataType: "string"
17          description: "Name of Brazilian city"
18      ...
19 metrics:
20     -
21         name: "met:count:city:id"
22         dataType: "integer"
23          aggregation: "count"
24          description: "Count of cities"
25      ...
```

The YAML file containing the database schema description is used by the chatbot's *Reads Database Schema* state for accessing BIOD data through the RESTful API, thus retrieving useful information to the user. All states demonstrated in the chatbot flowchart were implemented considering the Xatkit environment and a natural language conversation.

6.1.3 Natural Language Interaction and Error Handling

This subsection presents how the chatbot prototype uses natural language for interacting with the user and establishing query parameters. As demonstrated in Listing 6.1, the database schema file has a separation of dimensions and metrics related to every data table. So, when iterating over the file, we search for keys `dimensions` and `metrics`, and from them, we extract the sets ***RD*** and ***RM***, which are used for suggesting information of interest in the chat.

However, the names of metrics and dimensions within the database schema were not adequate to interact with the user, who expects a natural language conversation instead of, e.g., dimensions suggested in a "dim:table:column" format. Thus, to overcome this, the description in the YAML file is used for describing both dimensions and metrics. By using descriptions, a dimension "dim:city:name" is suggested in the chat as "Name of Brazilian city", i.e., a more understandable sentence.

For achieving this, we first had to identify if a description string was referred to a dimension or a metric, so, we examined the YAML file line by line, looking for all **name** keys. Each time a **name** key was found, its value was stored in a temporary variable, and immediately looked for the closest **description** key in the sequence. Thus, the value (metric or dimension) within the temporary variable was put together with its respective description in a map structure, used by the bot to conduct a natural language conversation.

After the iteration, the map structure should contain, e.g., [dim:city:id = Id of Brazilian city, met:count:city:id = Total of cities, ...] and so on, according to the attributes contained in the schema file. This allows dimensions and metrics to be suggested and chosen through *descriptions*

in the chat, while their real names (needed to compose the query parameters) are manipulated within the bot states.

Regarding error handling, the chatbot states contain specific programming logic for handling unexpected inputs. E.g., when the bot suggests a set of metrics to the user and receives a metric that is not contained in the suggested set, it replies that the information was not recognized, so the user is able to rewrite his choice. The same occurs for dimensions suggestions. Also, during the conversation, the bot is instructed to inform the user about some of its actions, i.e., about the metric/dimension it has received, information it has found, or process it is executing. This kind of feedback is important for the interaction, so the user knows what is happening and feels he can trust in what the chatbot is doing.

We assumed that, when formulating a multidimensional query, the user might feel confused about choosing metrics and dimensions, so additional "help states" were implemented in the chatbot for providing orientation messages. It means that, anytime the user replies "`help`", "`I did not get it`", "`what is that?`", or similar sentences, a help state is called for giving information about the attribute (metric, dimension, or filter) that is being mentioned in the conversation at that moment.

In addition, for simplifying the user replies to the bot, the *DialogFlow* NLP engine[5] was integrated with Xatkit[6], as it bases on machine learning for translating user inputs into intents. DialogFlow is an additional intent recognition provider that makes the conversation more flexible, meaning that some syntax errors, missing words, or capital letters that might occur in entries are ignored, thus improving the intention recognition process.

In the next section, the chatbot execution is demonstrated based on the flowchart and tasks of intention recognition, schema reading, and query building.

6.1.4 Chatbot Running Example

This subsection demonstrates the execution of the implemented chatbot, considering BIOD database, Xatkit Framework, and the procedures presented in the previous subsections.

As mentioned before, the YAML file demonstrated in Listing 6.1 contains a schema description. So, this file is given as input to the *Reads Database Schema* state in order to retrieve dimensional data sets. In a next step, user choices based on these sets are used to build the database query. The chatbot execution is demonstrated in Figure 6.2[7], in accordance with the state flowchart (Figure 6.1).

In the chat, the user informs an input aiming to find information about `cities per state`. The chatbot, through database schema reading, returns the ***RM*** set with all metrics that relate to the input. Based on this set, the user chooses `count of cities`, which is stored as the first query parameter. Next, the chatbot suggests the ***RD*** set, to which the user replies the dimension `Brazilian state`, also specifying the filter `Acre` (name of a Brazilian state). All dimensional data chosen are added in a map of parameters for the query building. As mentioned in Subsection 6.1.3, the interaction performed on the chat is based on *descriptions* of metrics and dimensions, while their real names are manipulated within the bot states to compose the query. Thus, the query resulting from the user choices is corresponding to:

[5]Available at: <`https://dialogflow.cloud.google.com`>.

[6]Available at: <`https://github.com/xatkit-bot-platform/xatkit/wiki/Integrating-DialogFlow`>.

[7]The interaction with the chatbot was conducted in Portuguese, which is the language used by BIOD database. For facilitating the reading of the running example, the Figure 6.2 shows all messages in English, with translation performed through the browser developer tool.

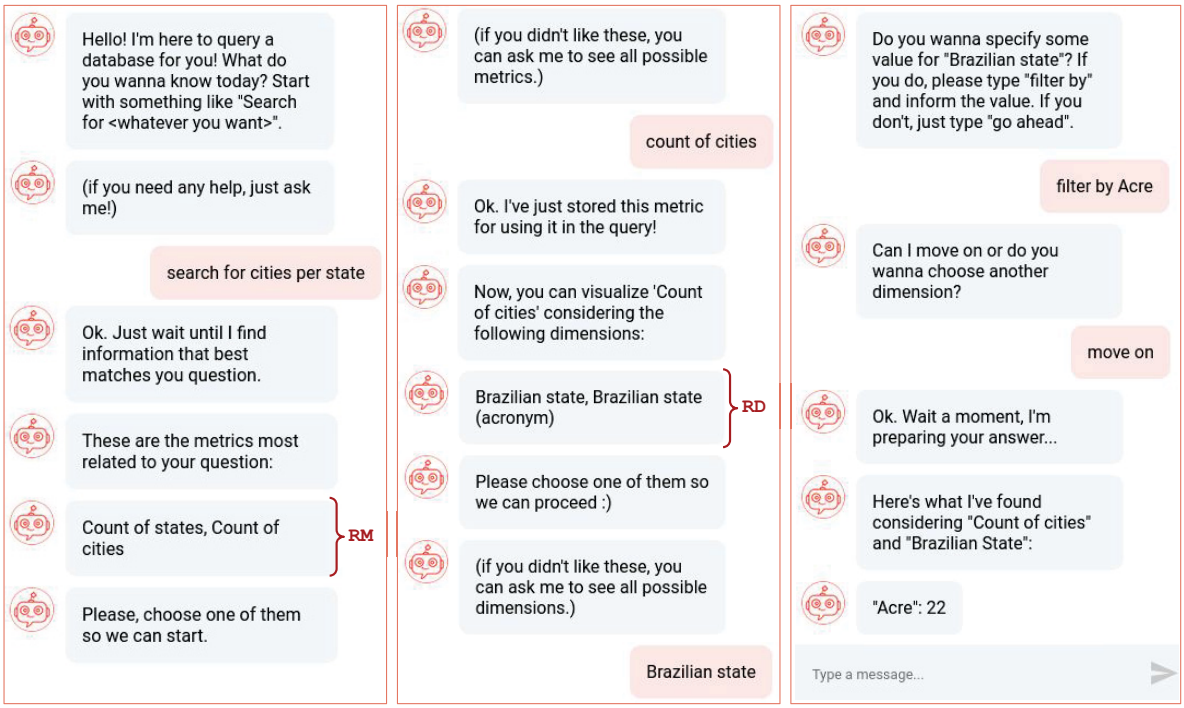


Figure 6.2: Proposing solutions (Chatbot execution example).

http://biod.c3sl.ufpr.br/api/v1/data?metrics=
met:count:city:id&dimensions=dim:state:name&
filters=dim:state:name==Acre

The result from this query is shown to the user in the chat: considering the state name `Acre`, 22 cities were found. It is important to mention that it demonstrates a case where the user specifies only one dimension and one metric from a set of more than 1000 dimensions and 700 metrics available. Indeed, for conducting an intention-guided conversation, the bot suggests ***RM*** and ***RD*** sets to the user at first. However, the whole list of dimensions retrieved from the database schema could be presented to the user during the interaction and used for querying. Since all dimensional lists suggested by the bot during the execution are extracted from the schema file, they are dependent on the nomenclature used for the database attributes. So, intention's training sentences described in the bot code are corresponding to the schema vocabulary.

This version of the chatbot is available online[8]. For evaluating its usefulness, the next subsection presents an empirical evaluation performed with a group of participants that interacted with the chatbot for querying the database.

6.2 RESULTS

This subsection details the empirical user study conducted for evaluation the chatbot prototype, which performs the *Reuse* step of the CBR architecture. The study was conducted entirely online with a heterogeneous group of 21 participants, performing a set of search tasks during the virtual interaction.

The participants evaluated the chatbot in two different moments. In a first moment, they received a document briefly stating the purpose of the assessment and providing initial

[8]Available at: <`https://gitlab.c3sl.ufpr.br/simmctic/biod/chatbot`>.

Table 6.1: Chatbot Evaluation Form

Criteria	**Form Question**	**Question Type**
User profile	Q1- What is your familiarity level with Informatics?	Multiple choice
Visibility	Q2- The information in the chat was shown in a well-structured way. Q3- The instructions displayed by the bot were visible and understandable. Q4- The bot communicated with me to inform status such as "looking for information".	Linear Scale: 1 = disagree 5 = agree
Support	Q5- The bot guided me through the conversation so I knew what to do. Q6- The bot alerted when I typed something unexpected, and instructed me on how to proceed. Q7- When I needed help during the conversation, the bot provided instructions. Q8- The bot was able to recognize what I was talking about.	
Usefulness	Q9- I was able to perform the query I wanted by selecting dimensions and metrics. Q10- The answers the bot gave me were helpful and reliable. Q11- The bot gave me fast answers. Q12- The answers provided were quicker than if I had to search on the internet on my own.	
Simplicity	Q13- It was easy to make a query using the bot. Q14- It was fun using the bot to query a database. Q15- I would use the chatbot again to search for information in a database.	
General feedback	Q16- The features I liked the most about the chatbot were: Q17- If you could change something in the chatbot, what would it be?	Selection boxes Free text

instructions. In this document, each participant also received five different query examples for starting a conversation with the bot: *(1) Counting of states per region; (2) Sum of research expenses per state; (3) Average of investments by institutions; (4) Counting of students who received funding in a given year; (5) Counting of students by course and type of institution.* All questions allowed filter values freely chosen by the user, e.g., a region name or course name. The query examples were elaborated and suggested to the users in order to guide them on the types of information contained in the database and hence possible to be consulted. After reading the instructions, each participant accessed the bot through an url to start the conversation.

In a second moment of the evaluation, after interacting with the bot, each participant was requested to fill in a form about the queries he/she performed on the chat. The evaluation form (demonstrated in Table 6.1) was designed based on previous studies [39, 121] and shows evaluation questions that cover four distinct categories: *Visibility, Support, Usefulness*, and *Simplicity*. *Visibility* criteria (questions Q2 to Q4) determine whether the information displayed in the chat follows an adequate structure; *Support* criteria (questions Q5 to Q8) evaluates how assertive is the bot when providing guidance; *Usefulness* (questions Q9 to Q12) measures user satisfaction with the received answers; and *Simplicity* (questions Q13 to Q15) indicates how intuitive is the interaction with the chatbot.

The answers to questions in these categories followed a linear scale, meaning that each participant had to choose one option from a 1 to 5 scale that represents how much she/he agrees with the current statement; the closer to the value 5, the greater the agreement. As shown in the bottom part of Table 6.1, the form also covered two questions for gathering *general feedback* about the chatbot (questions Q16 and Q17), thus allowing us to analyze features that can be improved. The results from this empirical user study are demonstrated in Table 6.2 and Figure 6.3, and are discussed as follows.

Starting by user profile (form question Q1), ten (10) participants (47.6%) declared to have only basic computing knowledge, i.e., they use a computer mostly for composing text documents and browsing the Web. Eight (8) participants (38.1%) have intermediate knowledge, as they have some background on programming and/or interface design, but neither work nor study in the Computer Science area. Only three (3) participants (14.3%) declared to have proficient knowledge, by working and/or studying computing.

Concerning *Visibility* aspects, more than 66% of the participants scored 4 and 5 for Q2, and more than 70% scored 4 and 5 for Q3. For question Q4, more than 85% have marked the maximum score (5). From these answers, it is possible to infer that information shown in the

Table 6.2: Score Percentage for Linear Scale Questions
(1/disagree - 5/agree)

Question	Score 1 (%)	Score 2 (%)	Score 3 (%)	Score 4 (%)	Score 5 (%)
Q2	0	4.8	28.6	33.3	33.3
Q3	4.8	4.8	19	38.1	33.3
Q4	4.8	0	4.8	4.8	85.7
Q5	4.8	0	19	42.9	33.3
Q6	14.3	9.5	23.8	23.8	28.6
Q7	0	14.3	28.6	28.6	28.6
Q8	4.8	4.8	33.3	4.8	52.4
Q9	4.8	0	19	19	57.1
Q10	4.8	0	4.8	28.6	61.9
Q11	0	0	14.3	14.3	71.4
Q12	4.8	4.8	9.5	42.9	38.1
Q13	4.8	14.3	23.8	42.9	14.3
Q14	0	4.8	9.5	42.9	42.9
Q15	0	14.3	9.5	28.6	47.6

chat was understandable, as well as the bot status communication. However, the structure in which the information was presented could be improved, since a considerable part (28.6%) of the participants scored 3 for question Q2.

The *Support* criteria were evaluated through questions Q5 to Q8. The highest scores (4 and 5) were more informed in question Q5, meaning that most of users were able to follow the conversation based on the bot orientation. The remaining questions ranged between scores 3 and 5, which indicates that the bot lacks some flexibility when receiving unexpected entries and recognizing the user intention. The proportional percentage of scores 3, 4, and 5 for the questions also indicates improvement needs in the bot assistance actions. However, it is possible to notice that the recognition performed by the bot (question Q8) was adequate for most of participants (52.4%).

Usefulness aspects were evaluated in questions Q9 to Q12, in which the maximum score of agreement was the most informed among the participants for 3 of 4 questions. In addition, Table 6.2 shows that more than 75% of the participants informed scores 4 and 5 for each Usefulness question. Considering these particular answers, Usefulness was the most covered aspect of the chatbot evaluation, indicating that users, besides being able to compose a dimensional query, received timely and helpful answers to their questions.

Questions Q13 to Q15 referred to *Simplicity* criteria of the evaluation. As we can observe in Table 6.2, the scores for Q13 were mostly around 3 and 4, meaning that, despite its usefulness, the chatbot is significantly complex for some users. For Q14 and Q15, most participants agreed that interacting with the bot was fun, and the assistant deserves to be used when searching for information, mostly because it prevents the users from searching the Web on their own, as observed in the scores for question Q12.

In the *General Feedback* form section, the users pointed out what they liked the most about the chatbot, and made some suggestions for improvement. For Q16, the participants could choose more than an option considering: *(1) Fast answers; (2) Help in finding information; (3) Fun interaction; (4) Recognition of my intentions during the conversation.* These options received, respectively, 18, 13, 10, and 6 votes. There was also an "Other" option, where the participant could inform a feature he liked, in case it was not on the options list. This option was not filled in by any participant.

Finally, for question Q17, the participants could express in free text what they would change in the chatbot if they could. This task was optional in the form and all suggestions

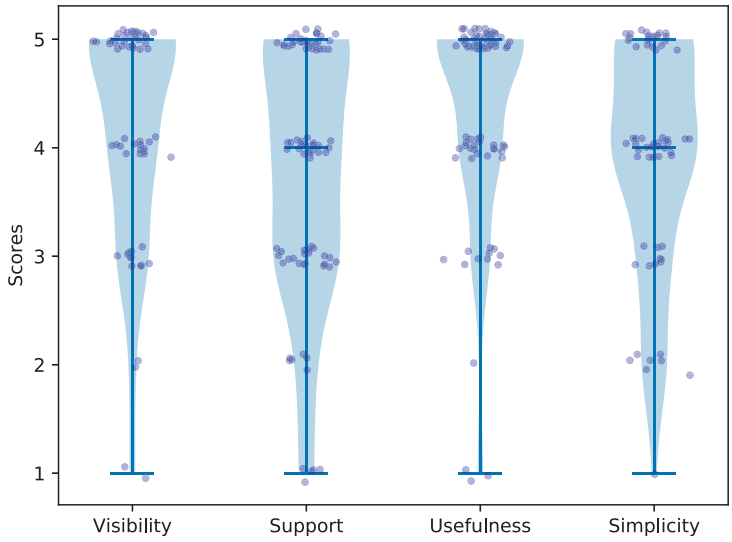


Figure 6.3: Overview of the chatbot evaluation by categories.

received are shown in Table 6.3. The participants mentioned changes in some visualization aspects such as letter formatting, arrangement of metrics and dimensions, and way of selecting options in the chat. Others expressed that the bot could be improved through more flexibility and dynamism, e.g., allowing to go back to some steps or access other databases through a broader set of questions.

In the next subsection, the chatbot experiment is discussed in the CBR context and some limitations found during the evaluation are presented.

6.3 CHAPTER REMARKS

The experiment described along this section aimed at investigating how dimensional-based solutions such as metrics and dimensions of a database could be proposed to the user within the CBR's *Reuse*. For this task, a chatbot prototype was implemented for accessing an integrated repository called BIOD, which contains open data from two independent databases. By gathering information along the conversation, the bot recommends attributes, builds the query parameters, and accesses the database data through its API, retrieving a response to the user. With the implemented prototype, a subgoal of the experiment was assessing the acceptance of the bot recommendations through four criteria (*Visibility, Support, Usefulness*, and *Simplicity*) captured during the user interaction. Thus, the chatbot was assessed in a user study conducted with a set of search tasks performed by 21 participants, and after interacting with the bot, they filled out an evaluation form covering the qualitative criteria.

The answers collected from the participants showed that the chatbot was useful for querying the open database, as it facilitates query formulation and retrieves a fast answer, which would hardly happen in a manual search. In general, it exempts the user from knowing database metadata and query languages (e.g., SQL), thus it can be adapted to improve data clarity. Regarding the criteria *Visibility, Support, Usefulness*, and *Simplicity*, an overview of their coverage in the chatbot is demonstrated in Figure 6.3, where we can observe a predominance of the highest scores for all evaluated questions. In general, the empirical study showed that

Table 6.3: Participants Suggestions for the Chatbot

Participant profile	Suggestion
Basic	"*Metrics and filters that could be selected with one click.*"
Intermediate	"*Buttons to select options would be better than typing information.*"
Advanced	"*I would include a 'Return' button to be used in all interaction levels, e.g., go back and make another question after I choose a metric. I would also list one metric or dimension per line in the chat to improve the visualization.*"
Intermediate	"*Include a better explanation of what I need to do.*"
Intermediate	"*I would add more flexibility when recognizing metrics and dimensions, when there are writing errors.*"
Basic	"*I would better highlighted the metrics and dimensions. Maybe enumerated the options.*"
Basic	"*Change the term 'dimension' for a more comprehensible one. I would like to see a more complete answer too, e.g., 'the average value for x was y in the year 2019'.*"
Basic	"*Expand possibilities for questions and databases.*"
Basic	"*More dynamism.*"
Advanced	"*Letter formatting could be lighter, more spaced and organized.*"
Basic	"*I would arrange the metrics more clearly, not between commas.*"

Visibility and *Usefulness* where the most effective features in the chatbot, whereas *Support* and *Simplicity* represented major opportunities.

During the evaluation, however, some limitations were identified regarding the interaction between participants and the chatbot prototype. The first limitation was the elaboration of query examples (i.e., search tasks) that should be accomplished by the participants through the chat (see Section 6.2). We opted for this limited number of queries rather than allowing an entirely free querying, due to the large number of dimensional data available in BIOD database, which could have added additional complexity to the user evaluation. In fact, users could take a long time to obtain an answer, if they continually searched for unavailable metrics or dimensions. We also chose to include queries up to 2 dimensions, as the evaluation focus was not on the query size, but on monitoring users interaction with the bot and patterns they apply when following the chatbot state flowchart (Figure 6.1) to compose dimensional queries.

Also, during the chatbot development, some challenges were faced when presenting attributes in the chat, since a large volume of information could be eventually retrieved. So, regarding the attributes visualization in the chat, we chose to separate the suggested metrics and dimensions by a comma, and send the whole suggestion set in a single chatbot message (rather than one metric or dimension per message), to prevent the user from taking too long to receive a reply. In user evaluation, some participants suggested another way to present these sets (see Table 6.3)[9].

Despite the limitations, useful insights were derived from the experiment. Overall, the participants were able to use the conversational interface for performing dimensional queries in an open database. Most importantly, the experiment demonstrates the value of a CBR's *Reuse* process based on multidimensional data: the user intention triggers a state for reading the database schema and finding related metadata. Through the metadata discovered, the bot guides the conversation with the users and recommends information on dimensions, metrics, and filters that are needed for building a database query. Since metrics and dimensions are are intertwined with situational data, we can infer that the chatbot could be used as support tool for situational data management.

Although the *Reuse* step was the core of the experiment, we can also see *Retrieval* characteristics in the prototype, represented by the database schema reading and metadata

[9]Some visualization issues were handled in a third experiment (presented in the following chapter), which covered a improved version of the chatbot prototype.

retrieved. This reinforces the relevance of joining a conversational interface with a CBR cycle and makes way to include a Source Discovery strategy and a historic knowledge base in the approach, as a way to fully address the *Retrieval* presented in the architecture (see Figure 4.1). Finally, the evaluation allowed us to visualize how users interact with a virtual assistant, and how the bot acts in response to the entries. Based on the positive results achieved, a chatbot modification for collecting human feedback and using it for continuous learning is encouraged.

Thus, the next chapter addresses the last experiment of this thesis, which encompasses the *Review* and *Retain* steps of the CBR architecture. In these steps, the user has active participation, as his feedback is used to boost future recommendations.

7 EVALUATING THE REVIEW AND LEARNING OF A CASE

Unlike the previous section, which focused on proposing solutions based on multidimensional data, the present section targets the *Review* and *Retain* steps of the thesis proposal. As illustrated in Figure 4.7, in the Review step the user is able to repair the proposed case, by managing the database to be queried and its respective attributes. After being repaired, the case is consistent enough to be retained, i.e., to be used for system learning.

The objective of the current experiment is to investigate how human knowledge, collected through the conversational prototype and following the CBR cycle, can impact the recommendations given (especially regarding Built-up Integration tasks) after the reviewed solutions are accommodated in the history. So, the experiment is essentially based on the user *action*, particularly his/her *feedback* about the recommendations given, and how it affects the system execution. At this point, it is important to highlight that repairing and learning a case, in the context of the proposed architecture, cannot be seen as isolated activities, but necessarily dependent on retrieving cases and proposing solutions. That is because, in case of negative feedback, the user should be allowed to modify database attributes or even trigger a Source Discovery mechanism to incorporate another case base in the system.

This Chapter starts by describing implementation details (Subsection 7.1) such as the history file building, the databases involved, preprocessing methods, and states implemented within the chatbot prototype for enabling *Review* and *Retain* activities. For assessing the impact of user feedback collected during the CBR cycle, two evaluations with real participants were performed and described in Subsection 7.2, being one based on a dynamic history (Subsection 7.2.1) and another based on a static history (Subsection 7.2.2). The evaluation results were also analyzed according to qualitative criteria and Built-up Integration features in subsections 7.2.3 and 7.2.4, respectively. Discussions about the experiment are presented in Subsection 7.3.

7.1 IMPLEMENTATION DETAILS

Considering the experiment goals mentioned above, and for allowing the user to express preferences (i.e., manage database and attributes, needed for properly assessing the Review and Retain activities), the chatbot prototype applied in the previous experiment was extended as follows:

- **Inclusion of history file:** For proposing solutions based on retrieval of *previous cases*, several functions for managing historic data were included in the prototype, including the history itself;
- **Inclusion of a Source Discovery method:** For proposing solutions based on retrieval of external cases, a Source Discovery method was included in the prototype, aiming to choose the most relevant database among the candidates;
- **Improvement of interaction elements:** Based on general feedback collected from the first user evaluation (see Subsection 6.2), the chatbot application was modified to present more intuitive elements and options to the user;
- **Several changes in the chatbot states and transitions:** For allowing the Case Repairing Flow illustrated in Figure 4.7, the conversation flow exposed in the previous experiment

(see Figure 6.1) was significantly extended, with states and transitions either removed, added, or modified to allow the proper revision or even the request of another database.

In the next subsections, these upgrades are minutely addressed, along with chatbot running examples that demonstrate the closing of the proposed CBR cycle.

7.1.1 Handling the History Retrieval

As previously mentioned, the chatbot version implemented in the current experiment includes a series of functions to propose solutions to the users based on previous cases. The previous cases should be retrieved from a *history file*, which was initialized as demonstrated in Table 7.1.

The history file is based on the structure shown in Figure 4.2, i.e., it contains the tokens of previous input cases (T_{PC}), the database used for retrieving a solution, the attributes (dimensions and metrics) used for building the query, and the Usefulness Score. The databases listed in the history correspond to a set of data tables from United States Open Data portals[1], which were considered as candidate databases for assessing the recommendation of a data source similar to the user question. The databases are detailed as follows:

- **USA Enrollments (Fall 2020)**[2]: Contains metrics about undergraduate students enrollment, residence, and migration, grouped by state os jurisdiction;
- **New York Enrollments (Fall 2010-Fall 2020)**[3]: The data table includes fall degree credit enrollment counts reported by institutions of higher education in New York State;
- **New York COVID-19 (2020-2021 School Report)**[4]: The dataset includes information on school reported COVID-19 testing and case positive data from the 2020-2021 academic year;
- **Limited English Proficiency Speakers in New York**[5]: This dataset is derived from the Census Bureau's American Community Survey (ACS), and includes information on limited English proficient (LEP) New York residents, grouped by community districts.

For performing an evaluation less restrictive in terms of language, the BIOD database characterized in Subsection 6.1.2 and the databases described in Subsection 5.1.1 were not included in this experiment, as they are Portuguese data sets. In fact, NLP tools can be significantly more challenging when the data to be analyzed are not in English, mostly due to the lower quality of processing tools for other languages [201, 236]. So, the current experiment involves open databases available in English and following the CSV format. As observed in the list above, the candidate databases mostly involve open data on New York city, and one of them on USA country. This closeness is part of the experiment, in order to evaluate the eventual integration of answers from different databases (which can occur in a Built-up Integration context).

[1]NYSED (New York State Education Department), NCES (National Center for Education Statistics), New York State Health Data, and NYC Open Data.

[2]Available at: <https://nces.ed.gov/ipeds/Search?query=residence&query2=residence&resultType=all&page=1&sortBy=date_desc&overlayTableId=29451>.

[3]Available at: <https://www.nysed.gov/information-reporting-services/higher-education-reports>.

[4]Available at: <https://health.data.ny.gov/Health/New-York-State-Statewide-School-COVID-19-Report-Ca/kaan-rxnd>.

[5]Available at: <https://data.cityofnewyork.us/City-Government/Population-of-the-Limited-English-Proficient-LEP-S/9ji4-nien>.

Table 7.1: Initialization of static history and dynamic storic (snippet).

PC	T_{PC}	**Database**	**Dimensions**	**Metrics**	**Score**
1	student	usa enrollment2020	state jurisdiction	students migration into state	10
2	student, enrolled, city	new york enrollments	legal name	all students fulltime, all students parttime	10
3	enrollment	new york enrollments	year	all students fulltime, all students parttime	10
4	enrollment, public, school	new york enrollments	county name	all students fulltime, all students parttime	10
5	enrollment, school	new york enrollments	school type	all students parttime,all students fulltime	10
6	covid, cas, school	ny covid19 school 2020 2021	School Type	positive students,positive teachers,positive staff	10
7	positive, covid, cas, student	ny covid19 school 2020 2021	school name	positive teachers	10
8	covid, cas, public, school	ny covid19 school 2020 2021	report date	positive students,positive teachers,positive staff	10
9	fulltime, enrollment	ny covid19 school 2020 2022	region	number of students	10
10	undergraduate, enrollment, public, school	new york enrollments	school type	graduate fulltime, graduate parttime	10
11	migration, enrollment	usa enrollment2020	state jurisdiction	institutions enrollment	10
12	covid, test, applied	ny covid19 school 2020 2021	school name	school administered tests	10
13	school, covid, report	ny covid19 school 2020 2021	region, report date		10

As mentioned in Subsection 4.1.1, retrieving a similar case from the historic knowledge base requires comparing the new case Q with previous cases PC_n, and then retrieving the most relevant solution. The comparison is based on the new case tokens T_Q and the tokens of each previous case in the history (T_{PCn}), allowing to retrieve the best match based on *Similarity* and *Usefulness* scores. Thus, considering an input question Q as a new case within the CBR cycle, for deriving the tokens T_Q necessary for the further steps, an intent `InputSearch` was implemented within the chatbot code (see Listing 7.1), which captures Q along with its metrics and dimensions parameters ("*QuestionMets*" and "*QuestionDims*", respectively).

Listing 7.1: Obtaining the input question using Xatkit intents.

```
val inputSearch = intent("inputSearch")
   .trainingSentence("show me how many METRIC there is DIMENSION")
   .trainingSentence("Show me all schools in CITY")
   .trainingSentence("show me the count of METRIC in DIMENSION")
   .trainingSentence("show me the total of METRIC from DIMENSION")
   .parameter("QuestionMets").fromFragment("METRIC").entity(any())
   .parameter("QuestionDims").fromFragment("DIMENSION").entity(any());
```

Next, the following preprocessing techniques and Java functions[6] were applied to Q:

- Special characters removal: The new case was lowercased by using the Java function *toLowerCase()*;
- Stopwords removal: Stopwords were removed from Q by using a file containing all possible stopwords as reference, and then replacing the eventual ocurrence of words with an empty space in the sentence. The function *replaceAll()* was responsible for the replacement;
- Tokenization: The Java function *split()* was applied to tokenize the input question;

[6]The preprocessing functions were implemented in Java, since it is the base language of Xatkit (the framework used for developing the chatbot prototype).

- Stemming: A function in Java was implemented from scratch to remove word suffixes that could interfere in the match with databases metadata;
- Named Entity Recognition (NER): For the experiment described in this Section, differently from the previous experiment, a NER method was implemented for allowing the user to inform an entity still in the beginning of the conversation, allowing greater flexibility. The Stanford Named Entity Recognizer [159][7] was used[8], which is written in Java, and hence suitable for the Xatkit environment.

In addition to the techniques above, an augmentation of the entities recognized with NER was implemented as shown in Algorithm 3. Basically, for each class recognized within the sentence (e.g., location, organization, or date), similar terms are included in the list of recognized mentions. The implementation was included considering the Source Discovery model and metadata search, since the returned list of entities is later used to search for matches based on value overlap (see Table 4.1). Taking as example an input question "how many positive cases of covid there were on 2021?", the probability of finding a metadata match for “2021” should be higher if [date, year, day, month] is considered in the Source Discovery task than simply using [date]. This augmentation would ensure, for example, that database attributes such as *report_year* or *referenceDay* are also returned as matches for a question containing “2021”[9]. Finally, as result from the preprocessing, T_Q is derived from Q and can be compared with each T_{PCn} from the history.

Algorithm 3 Entities Augmentation Algorithm.

Input data: List of recognized entities *entities*

```
1: for each entity in entities do
2:     classesList = getEntityClass(entity);
3: end for
4: augmentedClassesList = newList();
5: for each class in classesList do
6:     if class.equals("LOCATION"):
7:         augmentedClassesList.AddAll("location", "place", "address", "state", "country", "county");
8:     else if class.equals("ORGANIZATION"):
9:         augmentedClassesList.AddAll("organization", "department", "university", "org", "school");
10:    else if class.equals("DATE"):
11:        augmentedClassesList.AddAll("date", "year", "month", "day");
12:    end if
13: end for
14: return augmentedClassesList;
```

Next, basing on the history file represented in Table 7.1 , the history retrieval process occurs as described in the Algorithm 4. The algorithm traverses each PC, verifying whether the column T_{PC} contains T_Q (line 8). If so, a disjunction function is used[10] to get all differences between the sets of tokens (line 9). The objective is to retrieve the previous case with the smallest difference in terms of tokens, and thus with highest similarity score. Thus, the Similarity Score

[7]Available at: <https://nlp.stanford.edu/software/CRF-NER.shtml>.

[8]Besides adding flexibility to the interaction, the choice of including a NER method was also due to the candidate databases considered for the experiment. As they are available fully in English, a wider range of methods could be included as support, covering a more robust documentation than NLP methods for the Portuguese language.

[9]Semantic techniques such as Word2Vec could be used for metadata matching, however, the NER algorithm was implemented as a simpler alternative, as an optimal retrieval mechanism is not the objective of the present thesis.

[10]Available at: <https://commons.apache.org/proper/commons-collections/apidocs/org/apache/commons/collections4/CollectionUtils.html>.

is initialized with a high number ("10", as shown in line 1), and every time a smaller disjunction size is found, the Similarity Score is updated with its value (lines 10 to 12).

Algorithm 4 History Search Algorithm.

```
Input data: History file history
Input data: User preprocessed question T_Q
1: similarityScore = 10;
2: usefulnessScore = 0;
3: questionIndex = 0;
4: scoreIndex = 4;
5: bestMatch = "";
6: for each PC in history do
7: T_PC = PC[questionIndex];
8:       if T_PC.contains(T_Q):
9:           disjunctionList = CollectionUtils.disjunction(T_Q, T_PC);
10:          scoreInCurrentCase = disjunctionList.size();
11:          if scoreInCurrentCase <= similarityScore:
12:              similarityScore = scoreInCurrentCase;
13:              if PC[scoreIndex] >= usefulnessScore:
14:                  usefulnessScore = PC[scoreIndex];
15:                  bestMatch = PC;
16:              end if
17:          end if
18:      end if
19: end for
20: return bestMatch;
```

Besides the Similarity Score, the algorithm also includes the Usefulness Score, which indicates how many times the previous case *PC* was reused in the past as a solution. Thus, a third condition is added as part of the previous conditions (lines 13 to 16), assessing if the *PC*'s score is higher or equal to the current threshold (initialized as 0 in the line 2). Only after traversing the whole history file and getting the highest scores (Similarity and Usefulness), the *PC* identified as best match is returned as solution to be reused. In cases where there is more than one *PC* with the same "winning" scores, the most recent *PC* is proposed to the user, who is able to give a proper feedback and modify the history accordingly.

7.1.2 Source Discovery and Attributes Recommendation

As Source Discovery method, Sem-unionability (U_{sem}) was chosen (see Subsection 4.1.2.1). Although it was not the most accurate method in the Source Discovery experiments (see Section 5), it was chosen for the current experiment as it is a simple measure that works well within the Java-based Xatkit environment. U_{sem} was configured as discussed in Subsection 5.1.2.1, using the Java class *CombinatoricsUtils*[11] an the method *binomialCoefficientLog* as a basis for U_{sem}'s hypergeometric calculation.

After the data source is selected, the sets of metrics and dimensions need to be properly defined for the interaction, as discussed in Subsection 4.2.1. This definition is exemplified in Algorithm 5: given as input the data source in CSV format, we can extract the columns (*attributesList*) and the first line of data (*firstRecord*). Then, for each value non numeric in *firstRecord*, its respective column is added to the dimensions set. There might be cases in which a numeric value is actually a dimension (such as "Year" or "Code"), so we can go through the

[11] Available at: <https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/util/CombinatoricsUtils.html>.

attributesList and search for specific keywords, adding the appropriate values in the dimensions set. After that, from the whole set *attributesList*, we can extract the dimensions, so the remaining values can be considered as the metrics set. Finally, with the sets *dimensions* and *metrics* defined, a proper recommendation can be given to the user through the conversational system.

Algorithm 5 Defining Metrics and Dimension.

Input data:
Source (CSV) metadata: *attributesList*
Source (CSV) first data record: *firstRecord*
1: *DBmetrics* = [];
2: *DBdimensions* = [];
3: **for each** index, value **in** *firstRecord* **do**
4: **if** isNumeric(value) == false
5: *DBdimensions*.add(*attributesList*[index]);
6: **end if**
7: **end for**
8: **for each** index, attr **in** *attributesList* **do**
9: **if** attr.containsAny([dimensionKeywords])
10: *DBdimensions*.add(*attributesList*[index]);
11: **end if**
12: **end for**
13: *DBmetrics* ← *attributesList*.removeAll(*DBdimensions*);

Next, it is necessary to define the sets of the closest attributes to T_Q (i.e, ***RM*** and ***RD***) from the lists *DBmetrics* and *DBdimensions*. For this task, we use the parameters values *QuestionMets* and *QuestionDims* from *Q* (see Listing 7.1), comparing them with *DBmetrics* and *DBdimensions*, respectively, and searching for any overlaps. Thus, ***RM*** stores the overlapping values between *DBmetrics* and *QuestionMets*, whereas ***RD*** stores the overlapping values between *DBdimensions* and *QuestionDims*. It is important to recall that ***RM*** and ***RD*** are important parts of the chatbot conversation flow, since they represent the sets managed by the user during the interaction. The conversation flow and chatbot states implemented for the experiment are addressed in the next subsection.

7.1.3 Chatbot Conversation Flow

The main goal of the current experiment is to assess how the user actions and feedback can be used to repair a case and support the system learning. Thus, the interaction options along the chatbot conversation flow were moved from the states shown in the Figure 6.1 (i.e., more focused on proposing multidimensional solutions) to the ones illustrated in the Figure 4.7, which cover attributes management and data integration. Thus, besides choosing a metric or dimension among a set of options, the user is able to state their usefulness, based on the answer received.

For enabling the conversation flow discussed in Subsection 4.3.1 (and particularly the attributes management), the initial question given as input in the chat triggers a set of chained states, or functions:

1. `handleInputSearch`: It is mainly responsible for preprocessing the sentence and search for it within the history. After preprocessing, the bot tries to retrieve a similar case from the history, following the procedure explained in Figure 4.2. If a similar case is found in the history, its related metrics and dimensions ***RM*** and ***RD*** are proposed to the user. Otherwise, the state `handleDBquery` is called.

2. `handleDBquery`: It is responsible for triggering the *Source Discovery* mechanism (described in the previous subsection) when the history has not been useful. After the source selection, ***RM*** and ***RD*** are suggested to the user.

3. `handleRelatedAttributes`: When ***RM*** and ***RD*** are suggested, the user can choose among two options in the chat: 'Proceed with the matches' or 'See all the options'.

   (a) *Proceed with the matches*: When choosing this option, the user is stating that the suggestions seem good, so the subsets of interest ***m'*** and ***d'*** (see Subsection 4.3.1) assume the ***RM*** and ***RD*** values. Next, the state `handleSQLquery` is called.

   (b) *See all the options*: When choosing this option, the user is stating that he/she is not sure about the suggestions adequacy and wants to see all attributes options within the selected database. In this case, the list of available attributes are presented to the user, and the state `handleCaseRepair` is called.

4. `handleSQLquery`: Responsible for building a SQL query using the retrieved source, and ***m'*** and ***d'*** as parameters. The query can assume, e.g., formats such as <SELECT ***m'*** FROM ***source*** WHERE ***d'***=filter[12]> or <SELECT ***d'*** FROM ***source***>. The query built is submitted to a DBMS (Data Base Management System), and the state returns the proper answer. PostgreSQL was chosen as DBMS, which contains the candidate sources that are part of the experiment.

5. `handleCaseRepair`: When this state is called, the user can choose among the following options:

   (a) *Proceed with the matches*: see Item 3(a).

   (b) *Find another database to query*: which calls the state `handleDBquery` (Item 2).

   (c) *Choose another database*: In this case the user is presented to the candidate sources available, being able to choose one option to query. When selecting a source, ***RM*** and ***RD*** are built and the state `handleRelatedAttributes` is called (Item 3).

   (d) *Substitute attributes in the query*: When choosing this option, the user can select one metric or dimension among the available attributes, which will replace either the ***RM*** set (in case of choosing a metric) or ***RD*** (in case of choosing a dimension). The output of the state is the ***m'*** and ***d'***, which are used in `handleSQLquery` state (Item 4).

   (e) *Add attributes in the query*: As in (d), this option also allows the selection of one metric or dimension, however, these will be *added* to the suggested attributes rather than replacing them. This addition prepares ***m'*** and ***d'*** to be used in the query execution. Additionally, when a dimension is chosen, the user can inform a filter so the query is more specific. Filter are also allowed when (d) is chosen.

   (f) *Clear all matches*: In this case, ***RM*** and ***RD*** should be reset, so the user can compose a query with attributes selected from scratch.

6. `handleUserFeedback`: This state is called right after `handleSQLquery`. At this point, the user should indicate the usefulness of the answer received (as illustrated in the Repair Case component in Figure 4.7). The feedback can be given through the following chat options:

[12]The filter could be, e.g., an entity recognized during the NER task.

(a) *Satisfactory answer*: Here, the user gives a positive feedback about the query result, and the case is retained as discussed in Subsection 4.3.3.

(b) *Incorrect or incomplete answer*: In contrast, stating an incorrect or incomplete answer calls a personalized `handleCaseRepair` state, composed by the options 5(c-e) and the following additional options:

i. *Change the filters*: The user can try repeating the same query using a different dimension filter, e.g., a different entity.

ii. *Complement the answer with another database*: This option ensures that the answer received is not fully discarded, but complemented with a second database. This concept is part of Built-up Integration and was implemented in the state `handleComplementaryAnswer`.

7. `handleComplementaryAnswer`: As discussed in Subsection 4.3.2, the answer to a user question may be scattered over two or more databases. So, in Built-up Integration, data integration is executed towards the retrieval of *complementary data*. The current state aims at covering this particular feature of the concept, considering the recently received answer as a *partial answer* and triggering another round Retrieve → Reuse → Revise → Query Execution. When the additional round is finalized (Item 4) and the option 6(a) is chosen, the chatbot engine searches for the partial answer(s) previously saved and integrates[13] them in the chat for the user review. A positive feedback in this state results in the case being retained with a memory arrangement (Case Transformation and/or Case Addition), as stated in Subsection 4.3.3.

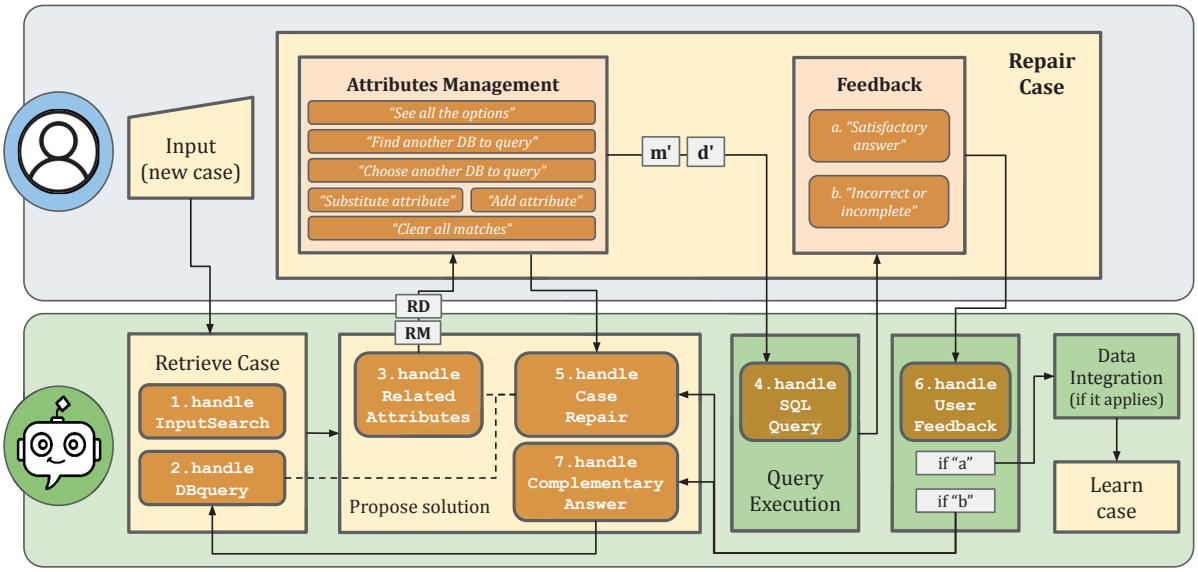


Figure 7.1: Chatbot state flowchart (CBR-based overview).

Finally, considering the CBR cycle and its four activities, the states described above can be linked with the activities *Retrieve, Reuse, Revise*, and *Retain* as demonstrated in Figure 7.1. It

[13]In the prototype implementation, the integration does not mean, e.g., aggregating two metrics and presenting a single value to the user, but presenting the answers from different sources together, so the user can reason about their connection and use them to support personal activities, if applicable. Aside from this particularity, metrics aggregation can be covered in a future work.

is valid to mention that the states described in this subsection and illustrated in the figure are not the totality of states implemented in the chatbot prototype, but an abstraction of the actual flow, which contains much more states that handle all the particularities of the interaction[14]. Hence, some transitions (i.e., state calls from another state) were also abstracted in Figure 7.1. Following this context, the next subsection exemplifies the execution of the chatbot states, considering the candidate sources presented and the Xatkit Framework.

7.1.4 Chatbot Running Example (Attributes Management)

This subsection demonstrates the execution of the chatbot prototype, considering the candidate databases and the chained stated discussed in the previous subsection. The chatbot execution is demonstrated in Figure 7.2.

Through the chat, the user informs his initial intent ("*Show me how many positive cases of covid there were on 2021*"), which triggers the state `handleInputSearch`. This is the preprocessing stage needed for case retrieval (either from history or source discovery, as shown in figures 4.2 and 4.3). Based on the question tokens derived from preprocessing, a case similar to the input case is found in the history file, associated with the New York COVID-19 School Report dataset. According to the retrieved case, the chatbot presents both metrics and dimensions that could be related to the user question, ***RM*** and ***RD***, being ***RM*** = [positive_students] and ***RD*** = [school_type], and asks for a confirmation. The user realizes that [school_type] does not match with the requested year "2021", so he chooses to see all the database attributes before proceeding with the query, which is corresponding to the state `handleRelatedAttributes (b)`. After all attribute options are shown in the chat, the state `handleCaseRepair` is called, whose options are displayed right below the question *What should I do now?*. The user chooses to *replace* attributes in the query, as he wants to substitute the recommended [school_type] with the dimension option *report_date*, which matches "2021".

The interaction continues as illustrated in Figure 7.3. After replacing the recommended dimension, ***d'*** = [report_date] and ***m'*** = [positive_students], so the state `handleSQLquery` is called. With ***d'***, ***m'***, and the entity "2021", the SQL query produced is corresponding to <SELECT *positive_students* FROM *ny_covid19_school_report* WHERE *report_date* ilike '%2021%', which outputs the result "1931". However, the user only obtained information about positive covid cases involving *students*, whereas the database also contains similar metrics related to teachers and staff. Thus, he chooses the modify the query attributes again, by stating that the answer received is not right or complete (state `handleUserFeedback (b)`). This time, he *adds* the metric [positive_teachers] to the query, which modifies the SQL query to <SELECT **SUM** *(positive_students+ positive_teachers)* FROM *ny_covid19_school_report* WHERE *report_date* ilike '%2021%'. With the new answer received (i.e., 2263), the user gives a positive feedback, so the history can now be modified.

Considering the initially recommended attributes (***RM*** = [positive_students] and ***RD*** = [school_type]) and the attributes chosen by the user after review (***m'*** = [positive_students, positive_teachers] and ***d'*** = [report_date]), the history would be organized according to the Table 7.2. The first line shows the possibly retrieved case containing ***RM*** and ***RD***, and the third line shows the recently added case containing ***m'*** and ***d'***. It is important to recall that the third line represents a repaired and learned case, as the case retrieved from the history has proved to be unsuitable for the initial question.

[14] All chatbot states and the complete implementation are available here.

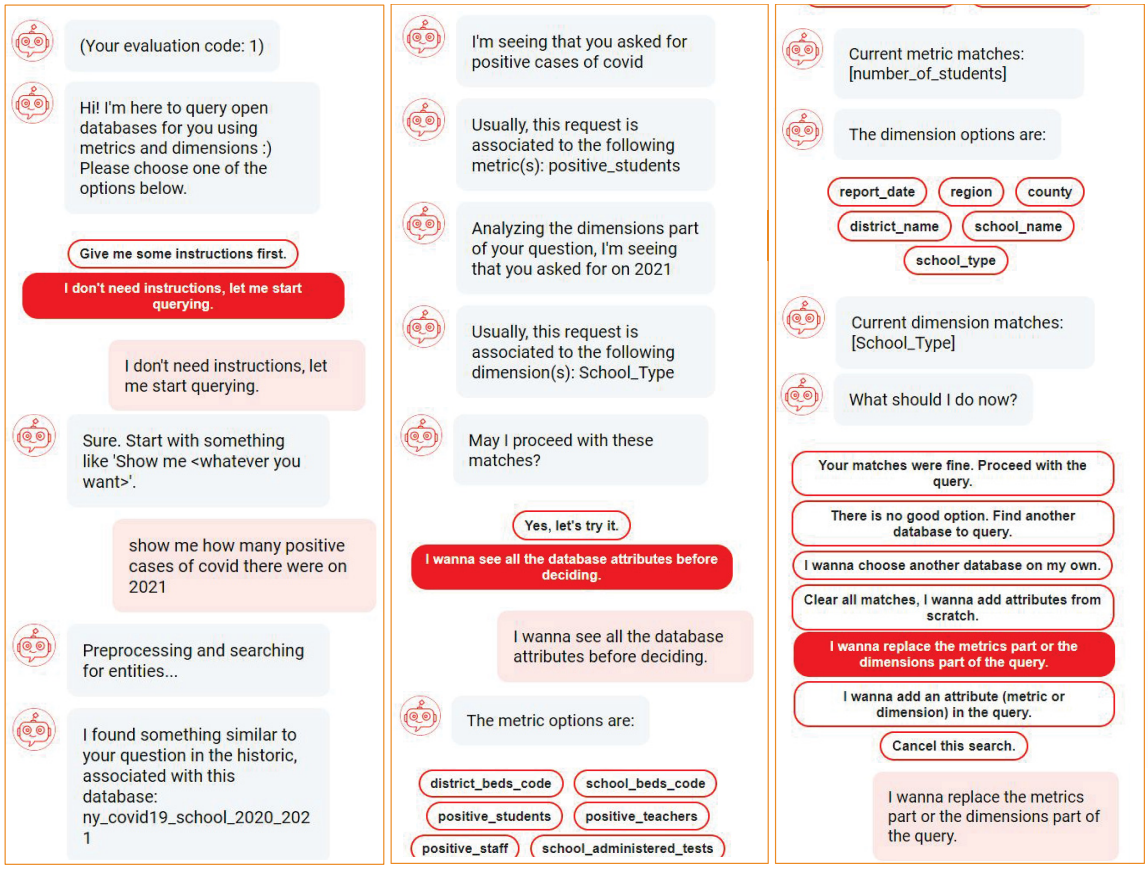


Figure 7.2: Managing attributes (Chatbot execution example - Part 1).

Table 7.2: Example of history file management.

	Question tokens	Database	Metrics	Dimensions	Score
1	positive, covid, school	NY_covid19_schoolR	positive_students	school_type	10
2	count, enrollment	NY_enrollments	all_students_fulltime	legal_name	20
3	positive, case, covid	NY_covid19_schoolR	positive_students, positive_teachers	report_date	10
4	count, enrollment	usa_enrollments	institutions_enrollment	state_jurisdiction	10

7.1.5 Chatbot Running Example (Handling Complementary Data)

This subsection presents a second execution example with the chatbot, which addresses a scenario where the complete answer to a question is stored in more than one dataset. As discussed in Subsection 4.3.2), this characteristic is observed in Built-up Integration approaches and demands the search for complementary data.

The example is illustrated in Figure 7.4 and considers the initial question "*show me the count of enrollments in New York and Orange County Community College*", where information about enrollments in the Orange County college is stored in a dataset (1)[15] whereas information about New York enrollments is in dataset (2)[16]. When the question is informed in the chat, the

[15]Available at: <https://www.nysed.gov/information-reporting-services/higher-education-reports>.

[16]Available at: <https://nces.ed.gov/ipeds/Search?query=residence&query2=residence&resultType=all&page=1&sortBy=date_desc&overlayTableId=29451>.

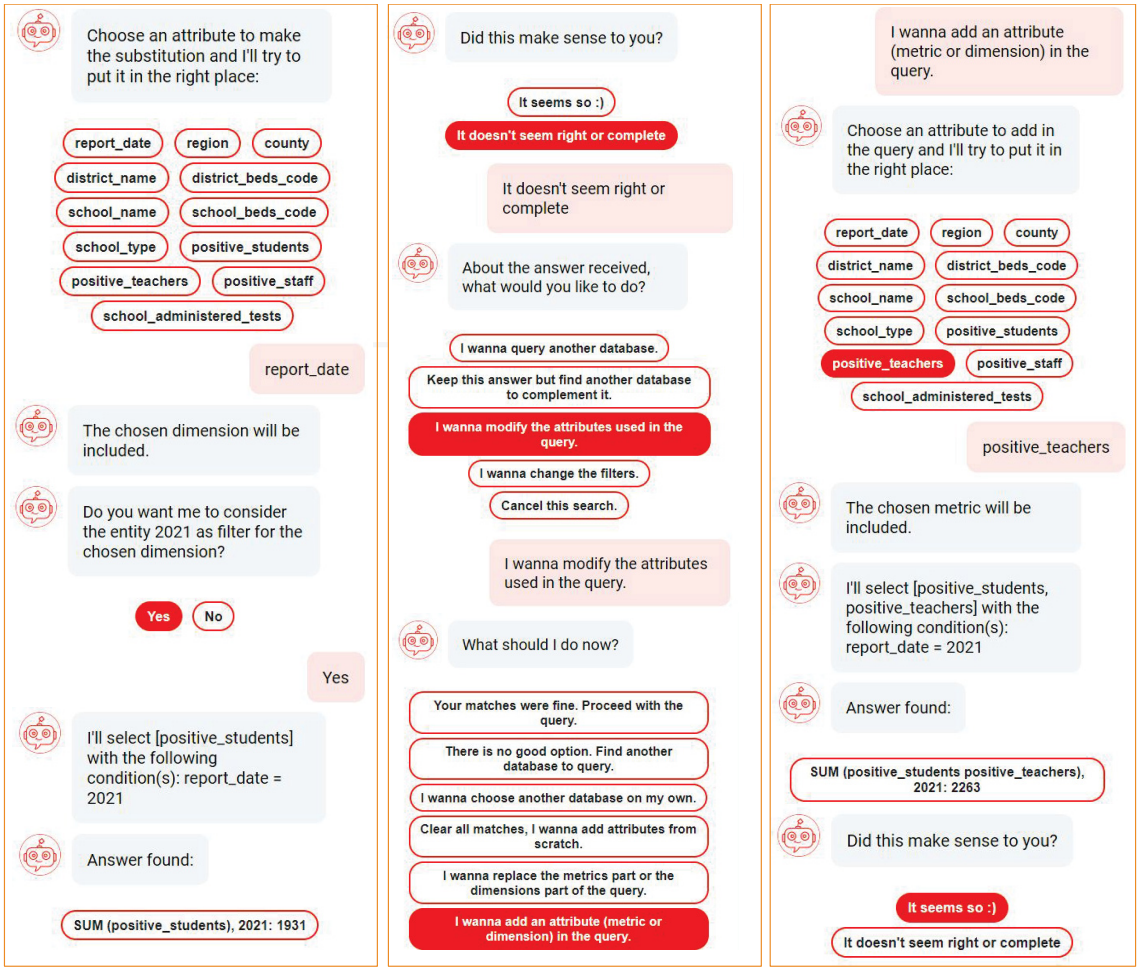


Figure 7.3: Managing attributes (Chatbot execution example - Part 2).

initial recommendation of the chatbot covers the dataset (1), ***RM*** = [all_students_fulltime] and ***RD*** = [legal_name], as shown in the top-left part of Figure 7.4.

However, the answer returned on the chat is incomplete, as it only contains information about the college enrollments and lacks information related to New York. Thus, the user chooses the option "*Keep this answer but find another database to complement it*", which leads to the state `handleComplementaryAnswer`. Within this state, the answer received is temporarily stored as *partial answer* and the chatbot searches for its complement. A possible complement is found in dataset (2) (represented in the Figure as *usa_enrollment2020)*, along with ***RM*** = [institutions_enrollment] and ***RD*** = [state_jurisdiction]. After seeing that the entity *New York* has been found in ***RD***, the user agrees with the proposed solution and the respective SQL query is built. The answer for "New York enrollments" is displayed in the chat (i.e., 177.012) and a positive feedback is given (`handleUserFeedback (a)`). As mentioned in Subsection 7.1.3, it is now possible to visualize a complete answer in the chat, composed by two answers from different databases.

The history modification is exemplified in lines 2 and 4 from Table 7.2: the second line represents the first retrieved case recommended by the chatbot, which had an increment in its score, due to its usefulness in the final answer. The fourth line is resulting from the execution of the second round of the CBR cycle, which returned information about New York enrollments. The modifications are equivalent to Case Transformation and Case Addition, respectively, as

Figure 7.4: Handling complementary data (Chatbot execution example - Part 3).

discussed in Section 4.3.3. Next, the prototype evaluation with a group of participants will be presented and discussed.

7.2 RESULTS

This subsection details the user study conducted with the chatbot prototype previously described. Differently from the evaluation discussed in Chapter 6, which evaluated *Reuse* aspects through the suggestion of multidimensional solutions, in the current experiment the objective was assessing the impact of *human feedback* in the chatbot actions and the proper modification of the historic knowledge, thus covering the *Revise* and *Retain* steps of the proposed CBR architecture (see Figure 4.1).

The user study was conducted entirely online with 22 participants, recruited through social networks and Amazon Mechanical Turk[17]. They received a detailed description of the study and its objective, being informed that their feedback about the chatbot answers would be collected during the interaction. For accomplishing the experiment, they received five specific questions that they should ask, which included: *(1) The number of students enrolled in New York city; (2) The number of positive covid cases in Saint Ambrose School students; (3) The total of enrollments in Bronx, in 2019; (4) The count of students migration in Alabama;* and *(5) The number of full time enrollments and covid tests administered in Albany schools.* Each question had an expected answer that should be visualized at some point by the participants. They were

[17]Available at: <https://www.mturk.com/>.

informed that eventually the chatbot could provide *incorrect* recommendations for the questions (respecting metrics, dimensions, or databases to be queried), which they should identify and correct (if necessary and as many times as necessary) until reaching the expected answer.

The questions were based on the candidate databases discussed in Subsection 7.1.3, and considering the fact that, for reaching the experiment purposes, the elaboration of specific tasks would be more effective than letting the users formulate questions freely. However, the participants were informed that, after completing the specific tasks, they could ask any question that could be related to the databases available, following some suggestions of attributes that were listed in the document. Some examples of how to manage recommended attributes were also illustrated in the study description, as a way to provide a better guidance to the participants.

As mentioned before, the chatbot was developed for proposing solutions to the users based on previous cases from a *history file*. Thus, for investigating the impact of user feedback in the historic case base and in the CBR context, two types of tests were organized: the first one, in which the history was **static** (i.e., the user feedback **could not** modify the history file), and the second one, **dynamic** (where the feedback **changes** the file, either by increasing the Usefulness Score or adding a new case). Specifically, the two types of history were initialized the same way, with the same previous cases as candidates for the retrieval, and each participant at the beginning of his/her interaction, was randomly assigned to a evaluation code, 1 or 2, which defined the use of a **static history** or a **dynamic history**, respectively. The users were not aware of the code meaning, ensuring an unbiased evaluation. The code assignment is demonstrated in the start of the interaction exemplified in Figure 7.2, and the initialization of the history files are demonstrated in Table 7.1.

After concluding the interaction tasks with the chatbot, each participant was required to fill in an evaluation form with questions designed on the light of previous studies [39, 121]. This means that some questions from the previous experiment evaluation form were kept or adapted (see Table 6.1), regarding *Visibility, Support, Usefulness* and *Simplicity* aspects. In addition, other categories of questions were included, in order to track the CBR goals of the current experiment (see Table 7.3): *Transparency, Justification, Relevance*, and *Data Integration*.

Three of the categories added are based on a consolidated study about Case-Based Reasoning goals [225] that presents a framework for explanation in CBR systems. According to the framework, the *Transparency* feature refers to explaining how the system reached the answer, which should be achieved by the chatbot when explaining, e.g., which attributes will compose the query to be executed. The *Justification* feature refers to explaining why the answer or strategy taken is a good one. In the chatbot, this was implemented in a state that correlates the proposed solutions with previous cases in the history (e.g., “According to the history file, users usually choose the metric M or dimension D”). *Relevance*, on the other hand, refers to explaining the relevance of a question for reaching a solution.

The evaluation form also contained questions related to *Data Integration*, and particularly, Built-up Integration (questions Q24 to Q26). As observed in Table 7.3, the question Q24 makes reference to an *integrated answer*, which should be achieved in the task (5). Thats is because the task asks for "the number of full time enrollments and covid tests administered in Albany schools", which involves information from two different datasets (New York Enrollments and New York COVID-19 School Report). Thus, the user should follow an interaction similar to the one shown in Figure 7.4 to complement the first partial answer received. As discussed in Section 3.2, the idea of obtaining complementary data is a key factor for Built-up Integration approaches, since an augmented data set can support particular tasks of the user and provide insights that might not be clear before, by only using one source of data.

Table 7.3: Chatbot Evaluation Form (Review and Retain steps)

Criteria	**Form Question**	**Question Type**
User Profile and Initial Checks	Q1- Your evaluation code during the interaction was: Q2- How familiar are you with Information Technology? Q3- What is the highest degree or level of school you have completed? Q4- What is your age?	Multiple Choice
	Q5- What is the difference between metrics and dimensions?	Selection boxes
	Q6- For the six specific tasks, did you get the expected answer in how many of them? Q7- How many free questions did you ask the bot?	Multiple Choice
Visibility	Q8- Information in the chat was shown in a well-structured way. Q9- The bot has communicated sometimes to inform the current status, such as “searching for information” or “preprocessing sentence”.	(1-5) Linear Scale
Visibility/ Transparency	Q10- The instructions and options were visible and easy to understand.	
Transparency	Q11- The bot explained how the answers were obtained, i.e., which attributes were used in the query.	
Support	Q12- The bot was able to follow the conversation and recognize my intention. Q13- When interacting with the bot I was able to decide what is useful and what is not. Q14- There was enough guidance (support options and sentences) throughout the conversation.	
Support/ Usefulness	Q15- The possibility to review and change query attributes was useful to get to the expected answer or improve the answer I received.	
Usefulness	Q16- I was able to make a query by selecting dimensions and metrics. Q17- The answers I received were compatilble with the dimensions and metrics I chose. Q18- The answers I received were faster than if I had to search in Open Data portals on my own.	
Justification	Q19- Sometimes the bot suggested metrics and dimensions based on historical information: it seemed a good way to justify recommendations. Q20- The bot justifications were enough to me.	
Relevance	Q21- I believe the chatbot questions like "what do you think about these matches?" and "did this make sense to you?" were relevant in the search for the answer.	
Simplicity	Q22- It was easy to make a query using the chatbot. Q23- I would use the chatbot again for seaching for information in databases.	
Data Integration	Q24- For task number 5, you should use an option in the chat for searching a complement for you answer in another database, so you could get to an integrated answer. Did the integrated answer make sense? Q25- Was it interesting to visualize in the chat answers from different databases?	Multiple choice/ Short Text
	Q26- Did you let the bot choose a database for you at least once? How was the experience?	Multiple choice
Final checks	Q27- Besides changing the database and attributes, would you add any other human decision during the conversation?	Short Text
	Q28- Leave your general feedback (the chatbot features you liked the most, the features you would like to change or add, etc.)	Long Text
	Q29- Was this evaluation form adequate according to your interaction experience?	Multiple choice

Most part of the questions in the evaluation form were linear scale questions, giving the possibility to choose one option between 1 and 5 for stating the level of agreement. With respect to *Data Integration* category, the questions were multiple choice followed by a text input, allowing to justify the choices and express a point of view. Text inputs were also included in the final checks of the form for gathering general feedback about the chatbot. Finally, the evaluation results were monitored through log files built during the interactions for tracking the participants actions: the **static log** maintained information about evaluations in which the static history was assigned, whereas the **dynamic log** maintained information when the dynamic history was used. Both logs were built as CSV files, containing the columns *correct recommendation, question tokens (T_Q), recommended database (R_DB), queried database (Q_DB), recommended dimensions (**RD**), queried dimensions (**d'**), recommended metrics (**RM**)*, and *queried metrics* (***m'***), being the term "recommended" referring to the solutions proposed by the chatbot and "queried" the attributes actually queried after user review. When $\{RD = d' \text{ AND } RM = m' \text{ AND } R\_DB = Q\_DB \text{ AND } positive\_feedback = 1\}$, $correct\_recommendation = 1$ in the log file; otherwise, $correct\_recommendation = 0$. Regarding this condition, $positive\_feedback$ is 1 when the state `handleUserFeedback(a)` is called (see Subsection 7.1.3).

Lastly, a termination condition (i.e., a stopping criterion) was considered for the evaluation, based on the Usefulness Score. In other words, the interactions could be finalized if at least one previous case in the *dynamic history* reached the value "100", meaning that at least one case was incremented ten or more times (assuming the value "10" at each increment), and the recommendations were stable. The results from the static and dynamic evaluations are detailed as follows.

7.2.1 Dynamic History Evaluations

As mentioned before, during the manipulation of the dynamic history, details of the users actions were stored in a dynamic log file, which is demonstrated in Table 7.4. The file begins with question (1)[18] (assuming the five specific tasks given to the participants) and an incorrect recommendation from the chatbot. The question, after preprocessing, results in the tokens "student, enrolled, city", as shown in the first line of the log. The chatbot suggested the metrics <all_students_fulltime, all_students_parttime>, which were not modified by the participant. However, the recommended dimension <legal_name> was replaced by <county_name>, implying that the use of <legal_name> as query parameter did not retrieved the expected result for the question, so the user chose another attribute. Thus, the condition $\boldsymbol{RD \neq d'}$ results in an incorrect recommendation. As observed in the snippet of the dynamic history (Table 7.5), the recommendation referred to the previous case (PC) 2, which contains the replaced dimension <legal_name>. Since the repaired case <student, enrolled, city; new_york_enrollments; county_name; all_students_fulltime, all_students_parttime> is not yet in the history, *Case Addition* occurs (see Section 4.3.3), and PC 14 is included in the dynamic history[19].

The incorrect recommendation is followed by other two incorrect recommendations in the dynamic log, respecting the questions (2)[20] and (3)[21] from the list of participants tasks. In the second line, the recommended metric <positive_teachers> (coming from PC 7) was replaced by <positive_students> by the participant, which is supposed to return the expected answer for

[18] *"Show me the number of students enrolled in New York city".*

[19] Each repaired case is included in the dynamic history with a score initialized as "10", which is incremented according to the case usefulness. The snippet in Table 7.5 shows the last version of the history (obtained after all the interactions were concluded) with the final Usefulness Scores.

[20] *"Show me the number of positive covid cases in Saint Ambrose School students".*

[21] *"Show me the total of enrollments in Bronx, in 2019".*

the question "*number of positive cases in Saint Ambrose School students*". In the third line of the log file, the dimension <county_name> was added to the recommended dimensions <year> (from PC 3). The repaired cases are then added to the dynamic history (see PC 15 and PC 16 in Table 7.5), both with an initialized Usefulness Score.

The first correct recommendation in the dynamic log refers to question (2), in which ***RD** = **d'*** and ***RM** = **m'***. At that point, the history contained two cases with the exact tokens "student, enrolled, city", with the same Usefulness Score, but recommended the most recent case included, which was the one previously repaired by a participant (PC 15). The fifth line of the log, on the other hand, shows an incorrect recommendation with empty recommended attributes for the question (4)[22]. That is because no similar case to the input question was found in the history, so Source Discovery was triggered, retrieving the dataset "usa_enrollment2020". The participant achieved the expected answer for the question by using the discovered source and the attributes <state_jurisdiction, students_migration_into_state, students_migration_out_state>[23]. The case was added in the history as PC 17.

Next, we can observe in the dynamic log a correct recommendation associated with the tokens "student, enrolled, city" from PC 14, followed by an incorrect recommendation for the same input tokens. In the latter, we can see that that recommended database and attributes were the same as the case right above, but the participant decided to change the database and attributes (probably for testing purposes), resulting in an incorrect recommendation and a *Case Addition* in the history (PC 18). We can assume that this specific case represents a false negative, since the bot recommendation was actually the one that would result in the right answer.

The following lines in the dynamic log contain the tokens "fulltime, enrollment, covid, test, administered, school" from question (5)[24], which, as mentioned before, should be answered with information from *two* different datasets. First, the bot recommends the dataset "New York COVID-19 School Report" as no similar case is found in the history and the recommended attributes are empty. The repaired case is included as PC 19 in the history. In the next line of the log, a correct recommendation is shown, using the same database and attributes queried in the previous interaction (from the recently added case 19), instead of a complementary database, expected for question (5). Probably, during the interaction, the participant did not choose the built-up integration options in sequence (i.e., *It doesn't seem right or complete → Keep this answer but find another database to complement it*, as demonstrated in Figure 7.4), but ended the task with a positive feedback. So, starting the question (5) again, she/he was presented to PC 19 as suggestion, which eventually resulted in the increase of the score for that previous case. This time, seeking the expected integrated answer, the participant followed the built-up options in the chat and the database "New York Enrollments" was queried as complement, as demonstrated in the third log line involving the question (5). The fact that the "New York COVID-19 School Report" was the recommended dataset in the log line demonstrates that the user did not interrupt the query, but chose to complement it with another source, so the recommendation is linked to the partial answer received (PC 19). Thus, assuming that the integrated answer was visualized in the chat, the complementary answer was included in the dynamic history as PC 20.

The remaining lines of the dynamic log show a sequence of correct recommendations. This indicates that repaired cases had their Usefulness Scores increased enough times for the chatbot's recommendations to be changed. Indeed, as we can see in the bottom of the history file, some scores were incremented more than ten times, allowing the cases to be repeatedly suggested

[22] *"Show me the count of students migration in Alabama"*.

[23] The empty fields in the log's recommended attributes were not filled with attributes from the discovered source because the implementation aimed at tracking the use of the *history*, specifically.

[24] *"Show me the number of full time enrollments and covid tests administered in Albany schools"*.

Table 7.4: Dynamic log file (snippet).

Correct	T_Q	R_DB	Q_DB	RM	m'	RD	d'
0	student, enrolled, city	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	legal name	county name
0	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive teachers	positive students	school name	school name
0	enrollment	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	year	county name, year
1	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive students	positive students	school name	school name
0	student, migration	usa enrollment 2020	usa enrollment 2020		students migration into state, students migration out state		state jurisdication
1	student, enrolled, city	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	county name	county name
0	student, enrolled, city	new york enrollments	usa enrollment 2020	all students fulltime, all students parttime	residents enrolled same state, students migration into state, students migration out state, residents enrolled any state	county name	state jurisdiction
0	fulltime, enrollment, covid, test, administered, school	ny covid19 school 2020_2021	ny covid19 school 2020_2021		school administered tests		school name
1	fulltime, enrollment, covid, test, administered, school	ny covid19 school 2020_2021	ny covid19 school 2020_2021	school administered tests	school administered tests	school name	school name
0	fulltime, enrollment, covid, test, administered, school	ny covid19 school 2020_2021	new york enrollments	school administered tests	all students fulltime	school name	county name
1	student, enrolled, city	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	county name	county name
1	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive students	positive students	school name	school name
1	enrollment	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	county name, year	county name, year
1	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive students	positive students	school name	school name
1	student, migration	usa enrollment 2020	usa enrollment 2020	students migration into state, students migration out state	students migration into state, students migration out state	state jurisdication	state jurisdication
1	fulltime, enrollment, covid, test, administered, school	ny covid19 school 2020_2021	ny covid19 school 2020_2021	school administered tests	school administered tests	school name	school name
1	fulltime, enrollment, covid, test, administered, school	new york enrollments	new york enrollments	all students fulltime	all students fulltime	county name	county name

Table 7.5: Dynamic history (snippet).

PC	T_{PC}	**Database**	**Dimensions**	**Metrics**	**Score**
1	student	usa enrollment2020	state jurisdiction	students migration into state	10
2	student, enrolled, city	new york enrollments	legal name	all students fulltime, all students parttime	10
3	enrollment	new york enrollments	year	all students fulltime, all students parttime	10
4	enrollment, public, school	new york enrollments	county name	all students fulltime, all students parttime	10
5	enrollment, school	new york enrollments	school type	all students parttime,all students fulltime	10
6	covid, cas, school	ny covid19 school 2020 2021	School Type	positive students,positive teachers,positive staff	10
7	positive, covid, cas, student	ny covid19 school 2020 2021	school name	positive teachers	10
8	covid, cas, public, school	ny covid19 school 2020 2021	report date	positive students,positive teachers,positive staff	10
9	fulltime, enrollment	ny covid19 school 2020 2022	region	number of students	10
10	undergraduate, enrollment, public, school	new york enrollments	school type	graduate fulltime, graduate parttime	10
11	migration, enrollment	usa enrollment2020	state jurisdiction	institutions enrollment	10
12	covid, test, applied	ny covid19 school 2020 2021	school name	school administered tests	10
13	school, covid, report	ny covid19 school 2020 2021	region, report date		10
14	student, enrolled, city	new york enrollments	county name	all students fulltime, all students parttime	**80**
15	positive, covid, cas, student	ny covid19 school 2020 2021	school name	positive students	**160**
16	enrollment	new york enrollments	county name, year	all students parttime, all students fulltime	**110**
17	student, migration	usa enrollment2020	state jurisdiction	students migration into state, students migration out state	**100**
18	student, enrolled, city	usa enrollment2020	state jurisdiction	residents enrolled same state, students migration into state, students migration out state, residents enrolled any state	**10**
19	fulltime, enrollment, covid, test, administered, school	ny covid19 school 2020 2021	school name	school administered tests	**120**
20	fulltime, enrollment, covid, test, administered, school	new york enrollments	county name	all students fulltime	**90**
21	student, enrolled, city	new york enrollments	county name	all students fulltime	**10**

and reinforced throughout the experiment. Based on the sequence of correct recommendations and the termination condition for the evaluation (mentioned in Subsection 7.2), the experiment with the participants was finalized.

Is is important to highlight that the dynamic log file presented in Table 7.4 shows the top part of the complete log file generated during the experiment. Although the complete version was much more extensive[25], the lines not included in this thesis were not relevant to the discussion, as they only show a long sequence of correct recommendations. Similarly, the Table 7.5 does not show the entire history, but relevant cases for understanding the sequence of interactions. Some other cases were included in the snippet to illustrate how the history was built (i.e., with the purpose of giving recommendations that should be reviewed and corrected).

7.2.2 Static History Evaluations

After detailing the evaluations with code 2 and the manipulation of the dynamic history, the evaluations that were assigned with code 1 are now described, which were associated with a static history. As the static history remained unchanged during all interactions, its initialized version shown in Table 7.1 should be considered for the present discussion. Each interaction linked to a code 1 was detailed in a static log file, demonstrated in Table 7.6. The table, just like the dynamic log from the previous subsection, contains the most relevant lines for interpreting the interactions between the participants and the chatbot prototype[26].

The first six lines of the static log demonstrate recommendations from the history that were corrected by the user. The first and third lines (retrieved from PC 2 and PC 3, respectively, as shown in Table 7.1), had the *dimensions* set modified, whereas the second line (from PC 7) had the *metrics* set modified. The lines 4 to 6 of the log show interactions already observed during the dynamic evaluations, i.e., input cases with no satisfactory match in the history, so that Source Discovery was performed. The lines 5 and 6, specifically, represent the question (5) (from the list of evaluation tasks) that was answered with the content of two different databases. Considering a static history, no case was added or transformed based on the corrections.

The sequence shows a correct recommendation associated with question (1), since the condition $\{RD = d' \text{ AND } RM = m' \text{ AND } R\_DB = Q\_DB \text{ AND } positive\_feedback = 1\}$ was fulfilled. We can observe, however, that the recommendation was *the same* as the first interaction, i.e., the repaired case was not retained and used for further improvement, as expected with CBR methodology. Moving forward, the static log file shows a set of wrong recommendations, where the same pattern appears: the same set of metrics and dimensions are recommended every time, despite all corrections made. In addition, in the last five lines of the log snippet, some empty spaces appear in the recommended metric and dimension sets, for the questions that previously involved Source Discovery. These gaps were supposed to be filled with newly recommendations from the modified history, but considering a static history that bypass the CBR methodology, no improvement has been made.

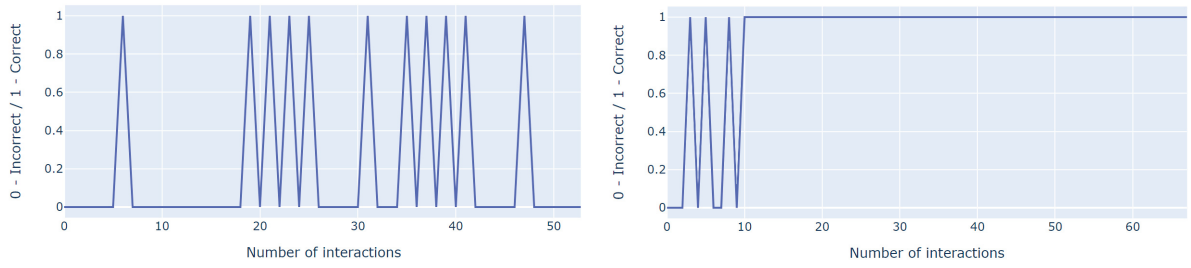
The comparison between static and dynamic evaluations is demonstrated in Figure 7.5. The x axis shows the number of interactions during the experiment (considering that each participant interacted more than one time with the chatbot, by asking different questions), wheread the y axis shows a binary 0 or 1 that represents an incorrect or correct recommendation, respectively. We can observe that, in the evaluations that used the static history (Figure 7.5(a)), the chatbot had a very unstable performance, giving incorrect recommendations most of the times

[25]The complete dynamic log is available here.

[26]A snippet was chosen to demonstrate the static log file, since there were several case repetitions in the original file that were irrelevant for the experiment discussion. The complete log file is available here.

Table 7.6: Static log file (snippet).

Correct	T_Q	**R_DB**	**Q_DB**	**RM**	**m'**	**RD**	**d'**
0	student, enrolled, city	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	legal name	county name
0	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive teachers	positive students	school name	school name
0	enrollment	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	year	county name, year
0	student, migration	usa enrollment 2020	usa enrollment 2020		students migration into state, students migration out state		state jurisdication
0	fulltime, enrollment, covid, test, administered, school	new york enrollments	new york enrollments		all students fulltime		county name
0	fulltime, enrollment, covid, test, administered, school	new york enrollments	ny covid19 school 2020_2021		school administered tests		school name
1	student, enrolled, city	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	legal name	legal name
0	student, enrolled, city	new york enrollments	usa enrollment 2020	all students fulltime, all students parttime	residents enrolled same state	legal name	state jurisdiction
0	enrollment	new york enrollments	new york enrollments	all students fulltime, all students parttime	undergraduate_pt enrolled program, undergraduate_ft enrolled program	year	county name, year
0	enrollment	new york enrollments	new york enrollments	all students fulltime, all students parttime	undergraduate_pt enrolled program, undergraduate_ft enrolled program	year	county name
0	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive teachers	positive students	school name	school name
0	student, migration	usa enrollment 2020	usa enrollment 2020		students migration into state, students migration out state		state jurisdication
0	student, enrolled, city	new york enrollments	usa enrollment 2020	all students fulltime, all students parttime	all students fulltime	legal name	legal name
0	positive, covid, cas, student	ny covid19 school 2020_2021	ny covid19 school 2020_2021	positive teachers	positive students	school name	school name
1	student, enrolled, city	new york enrollments	new york enrollments	all students fulltime, all students parttime	all students fulltime, all students parttime	legal name	legal name
0	fulltime, enrollment, covid, test, administered, school	ny covid19 school 2020_2021	ny covid19 school 2020_2021		school administered tests		school name



((a)) ((b))

Figure 7.5: Comparison between static (a) and dynamic (b) evaluations.

(79.6%). On the other hand, for the evaluations that used the dynamic history (Figure 7.5(b)), the correct recommendations were sustained after a determined number of interactions, showing the suitability of a knowledge base constantly modified by means of the CBR cycle. For these evaluations, the chatbot gave a correct recommendation in 89.7% of the times. It is important to recall that the assignment of a static or dynamic evaluation was *random* among the participants, and the experiment was concluded with 59% of static evaluations, against 41% of dynamic evaluations.

The next subsection evaluates the collected results by analyzing the categories of the questions in the evaluation form (*Visibility, Support, Usefulness, Simplicity, Transparency, Justification, Relevance*, and *Data Integration*) and the participants answers.

7.2.3 Analyzing Questions by Category

As stated in the beginning of the Section 7.2, the participants of the experiment were required to fill in an evaluation form that, in addition to containing some questions from the previous chatbot experiment, was increased with questions to measure CBR objectives, specifically [225]. The evaluation form is demostrated in Table 7.3 and the participants answers are now discussed.

Starting from user profile questions (questions Q1 to Q7), 50% of the participants declared a basic IT knowledge, through activities such as accessing the Internet and editing Word documents. 36.4% were IT students or professionals, whereas 13.5% have intermediate IT knowledge although they do not work or study in the area. Most part of them declared to own a Bachelor's Degree (59.1%), while the minority declared to own either a Master's Degree (18.2%), a Doctorate Degree (9.09%), or completed High School (13.6%). 77.3% of the participants were 25 to 45 years old, while 22.7% was younger than 25 years old.

When asked about the difference between metrics and dimensions (question Q5), approximately 88% has given the correct answer[27] (i.e., "Metrics are calculated values and dimensions are used to group the metrics" and/or "Number of schools is a metric"). For question Q6, the expected answer for *all of the specific tasks* was reached by 54.5% of the participants, *for more than a half* of the questions by 22.7% of the interviewed, and *for half or less than a half* of the questions by 22.8%. Overall, the Q6 answers were positive for 77.2% of the participants, demonstrating that they were able to gradually compose a query until reaching the expected result. Regarding question Q7, 41% declared that asked more than 4 free questions to the bot, while the remaining participants asked 0 to 4 free questions, which, as mentioned before, were additional and optional questions, apart of the specific tasks of the experiment.

[27]This question was included in the form to check the participants understanding about the experiment description and the requirements for building a query.

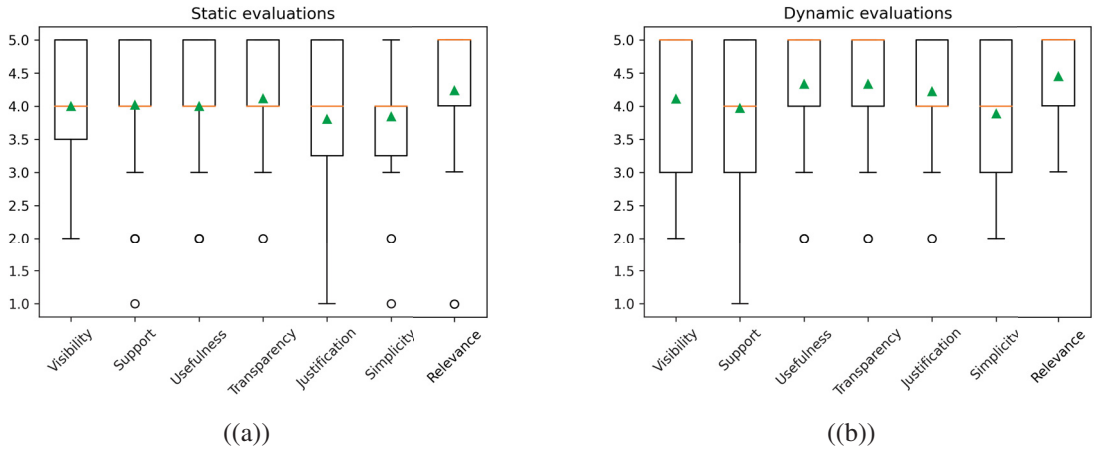
Table 7.7: Score Percentage and Average for Linear Scale Questions
(1/disagree - 5/agree)

Question-Category	Score 1 (%)	Score 2 (%)	Score 3 (%)	Score 4 (%)	Score 5 (%)	Avg Static Score	Avg Dynamic Score
Q8-VIS	0	13.6	22.7	**40.9**	22.7	3.84	3.55
Q9-VIS	0	4.55	4.55	22.7	**68.2**	4.23	**5**
Q10-VIS-TRAN	0	13.6	22.7	27.3	**36.4**	3.92	3.77
Q11-TRAN	0	0	9.09	27.3	**63.6**	4.3	**4.88**
Q12-SUP	0	4.55	18.2	31.8	**45.5**	4.15	4.22
Q13-SUP	0	13.6	9.09	**50**	27.3	**4.07**	3.66
Q14-SUP	9.09	4.55	13.6	**40.9**	31.8	3.84	3.77
Q15-SUP-USEF	0	13.6	9.09	31.8	**45.5**	4	**4.22**
Q16-USEF	0	4.55	13.6	**40.9**	**40.9**	4.3	4
Q17-USEF	0	18.2	0	36.4	**45.5**	3.84	**4.44**
Q18-USEF	0	4.55	18.2	31.8	**45.5**	3.84	**4.66**
Q19-JUS	4.55	4.55	18.2	**40.9**	31.8	3.76	**4.11**
Q20-JUS	4.55	4.55	9.09	**40.9**	**40.9**	3.84	**4.33**
Q21-REL	4.55	0	9.09	31.8	**54.5**	4.23	4.44
Q22-SIM	0	4.55	27.3	**40.9**	27.3	3.84	4
Q23-SIM	4.55	9.09	18.2	**36.4**	31.8	3.84	3.77

Respecting the linear scale questions, the results by category are summarized in Figure 7.6, and questions details (scores percentage and average) are shown in Table 7.7. Some highlights were added to the table to indicate the highest percentages and interesting differences between static and dynamic evaluations. The answers for *Visibility* category were mostly around the scores 4 and 5. While more than 80% of the participants scored 4 and 5 for question Q8, the percentage is a little smaller for Q9 and Q10 (around 63%). These results mean that, although the bot communication was satisfactory, the information in the chat was not following the proper structure. The users impressions, in this case, might have been influenced by some lengthy replies from the chatbot, which were eventually composed by many lines of text or clickable options (see Figure 7.2). For question Q9, we can notice that its average score in the static evaluations was 4.23, while it reached a 5-score in dynamic evaluations. This significant difference demonstrates that when the dynamic history was used to adapt further recommendations, the users were more satisfied with the chatbot replies.

The *Transparency* questions were Q10 and Q11 in the evaluation form. As already discussed, Q10 (which was also in the *Visibility* category) did not reached a satisfactory result, but there is a positive change for question Q11. This question referred to the bot explanations about how the answers were obtained, and reached a 5-score for 63.6% of the participants. A comparison between average scores for this question is not relevant, since the chatbot gave explanations equally for both types of evaluations. However, we observe a positive difference in the dynamic evaluations, demonstrating that users tend to be more confident in the explanations when the recommendations are adapted.

For *Support* questions (questions Q12 to Q15), the scores were mostly around 4. As expected, no significant difference between the average scores was noticed for Q12 e Q14, since the chatbot had the same actions for both static and dynamic evaluations. However, a superior average score was obtained for Q13 (which aimed to measure decision-making about the recommendations utility) in static evaluations. In fact, during the static evaluations, the participants likely had to make many more decisions than the other group of participants to correct the chatbot recommendations, which were predominantly erroneous. As a consequence, they have managed more attributes than the dynamic evaluation participants and experienced a clearer sense of decision-making. For question Q15, which measured the chatbot improvement after human correction, a slightly superior average was observed in the dynamic evaluations. For this question, the average score in static evaluations was expected to be much lower, but the explanation may be linked to question Q13. Static evaluation participants might have perceived



((a)) ((b))

Figure 7.6: Comparison between linear scale questions in static and dynamic evaluations.

some kind of improvement not because the recommendations were better, but because the chatbot retrieved the *expected answer* for the question at hand, after consecutive corrections. In any case, 75% of the *Support* questions obtained a higher average score in evaluations where the history was changed and used to adapt the conversation.

Usefulness questions covered Q15 to Q18. Q15, as already discussed, had a little increase in the average score for dynamic evaluations. For question Q16, more than 80% of the scores were around 4 and 5, and static evaluations had an average score slightly superior than dynamic evaluations. This result is closely linked to question Q13 and reinforces the hypothesis that the more manual corrections are made, the greater the users perception of their decision-making control. Consequently, we can assume that the possibility to review and change attributes was more useful to reach the right answer for static evaluations users than for the other group (which received correct recommendations most of the times). For questions Q17 and Q18, a significant improvement was observed in dynamic evaluations averages. These questions measured, respectively, the answers compatibility with the chosen attributes and the speed in receiving the final answer in comparison with a personal search in Open Data portals. The Q17 result demonstrates more compatible answer in dynamic evaluations, probably leveraged by the correctness in the chatbot suggestions. For Q18, we observe that dynamic evaluation participants received faster answers than the other group, which is the expected result considering that they did not need to make many corrections to obtain the answers.

Q19 and Q20 were categorized as *Justification* questions, covering the chatbot explanations about the strategies taken. These questions were mostly scored with 4 and 5 values, with an advantage in dynamic evaluations. In general, the users tended to be more satisfied with justifications when the bot learned from user feedback. In addition, the use of the history seemed to make more sense for participants who experienced corrected recommendations along with the explanation of the historical content. The *Justification* questions are correlated to the *Relevance* question Q21, since the latter also addresses the correlation between historical content and suggestions. For Q21, comparing the static and dynamic averages is not relevant, since the chatbot inquiries such as "*What do you think about these matches*" were present in both types of evaluations. However, it is interesting to observe that more than 85% of the participants considered the chatbot questions relevant when searching for the answer.

The last category of linear questions was *Simplicity*, with questions Q22 and Q23. Q22 asked the participants how easy was to make a query using the chatbot. The dynamic evaluations average was slightly superior for this question, but particularly, despite most of the participants

agreed with the statement with a "4" score, a considerable part of them informed "3" for the question (27.3%), indicating some kind of hesitation. We believe this particular result is linked to *Visibility* aspects, since the way in which the elements are displayed in the chat deeply impacts the sense of simplicity. Similar percentages were observed for question Q23, reinforcing the importance of a good user experience in conversational interfaces.

An overview of linear question scores is demonstrated in Figure 7.6. We can notice that, in general, the chatbot prototype was better rated in dynamic evaluations respecting *Usefulness, Transparency, Justification*, and *Relevance*. No significant changes were observed for the remaining categories (*Visibility, Support*, and *Simplicity*). Next, *Data Integration* questions will be discussed in Built-up Integration context, as well as general feedback given by the participants of the experiment.

7.2.4 Built-up Integration and General Feedback

By embracing *Review* and *Retain* activities of the CBR methodology, the proposed architecture aimed at supporting Built-up Integration features such as *Data Retrieval, On-the-fly Integration*, and *Data Delivery*, using human knowledge as learning strategy. Some criteria involved in the three main features were covered by the questions Q24 to Q26 from evaluation form (see Table 7.3). The questions had a focus on characteristics such as the use of a complementary answer (when more than one data source is considered) and Source Discovery, which represent valuable features for supporting specific user tasks (see Section 3.2). The questions Q24 to Q26 were set in multiple choice format, with the possibility to add a short comment in the first two.

For Built-up's *Data Retrieval* feature, we can reason about the related criteria (`Input queries` and `Source Selection`) included in the chatbot prototype. Regarding the former, the approach supports *situational queries*, which focus on ad-hoc needs and may involve the joint use of multiple databases. For `Source Selection`, the approach had a particular focus on Source Discovery, evaluated through question Q26. 77.3% of the participants stated that letting the chatbot choose/discover a database was interesting and/or useful, 9.09% had the opposite opinion, and 13.6% did not let the chatbot make a database choice.

The questions Q24 and W25 were related to Built-up's *On-the-fly Integration* feature, that mainly involves some kind of data augmentation made at query time. The question Q24 requested the users a personal opinion about the chatbot option "Search for a complement to your answer in another database", and specifically the significance of the integrated answer they received in the chat. 86% of the participants gave a positive feedback, stating that the integrated answer did make sense. One user (4.55% of total group) expressed a negative opinion about the integrated answer, whereas 9,09% did not used the complement option in the chat. When asked about the joint visualization of answers from different databases in the chat (Q25), 77.3% of the users found it interesting, and 22,7% had an opposite opinion. Some users also left textual comments in *Data Integration* questions that were associated with a text input, which can be seen in Table 7.8. From Q24 and Q25 comments, we can see that a complete answer was particularly interesting for a group of participants, and perhaps seen as a novelty in a QA context, although the chat visualization may have interfered in the experience. In general, considering the criteria associated with *On-the-fly Integration* feature (see Figure 3.1), the approach covered both mandatory characteristics (such as `Data Preprocessing` and `Data Augmentation`) and `Human Involvement` optional characteristic. The latter, although optional for Built-up Integration, has proven to be essential in a CBR context, where the user helps refining the selection of data sources.

For analyzing Built-up's *Data Delivery* feature, it is valid to consider the final questions of the evaluation form that addressed general feedback (Q27 to Q29). For these questions, the

Table 7.8: General feedback for the chatbot prototype (highlights)

Question	User Comment
Q24[a]	"*Yes - We were asking for two different questions and we want both solutions.*"
Q24	"*Yes - The bot showed the complete response in a simple way to understand, both information needed.*"
Q24	"*Yes - The bot provided the first answer correctly, I asked to keep it and search for other databases and then it provided the other question correctly as well.*"
Q24	"*Yes - The chatbot gave me the school administered tests first, and then the fulltime enrollments. When I asked the full complemented answer the chat gave me it in a simple, organized way.*"
Q25[b]	"*Yes - It is important to visualize answers like that because it's useful to future searches.*"
Q25	"*Yes - Although you have to explicitly request to search another database to complete the answer.*"
Q25	"*No - Too much information in a small space to visualize, it required a lot of effort to keep the focus.*"
Q25	"*Yes - It was nice to know it could get answer from different databases.*"
Q27[c]	"*The possibility to add query on an already answered query (at the end of the original query).*"
Q27	"*Choosing multiple attributes at once.*"
Q27	"*I would add an option to change the database right after it says what database it's going to query.*"
Q27	"*Perhaps I would introduce a new option that presents a short description of each database.*"
Q28[d]	"*Sharing links to knowledge base, articles, or other resources.*"
Q28	"*Wider layout for the chat when using on PC.*"
Q28	"*I liked the speed of the chatbot and the way of interaction. It would only improve the way of presenting suggestions.*"
Q28	"*I would like to have the option to add/change more than one attribute at a time. There where too many repetitions of the same questions. The bot seemed not to be learning from previous interactions.*"
Q28	"*The colors were too tiring, sometimes the reply had too much information (maybe it could stay at the start of the bot's reply and then I could only scroll down).*"
Q28	"*I think that the chatbot should give the metrics and dimensions as options before trying to give some answer.*"
Q28	"*I like how it sends short messages instead of long texts.*"
Q28	"*Very useful. Able to collect answers for specific questions that could take a longer period of time to be collected by searching/scouting on the web. I would consider adding options to select more than one attribute at a time.*"
Q28	"*The chatbot is efficient in quickly providing numbers for the given queries. It is faster than googling it. However, for real-world usage, it requires improvement in natural language understanding and it should use less technical terms.*"
Q28	"*I liked the possibility of changing metrics, visualizing the entities involved in the search, and the database. It could be better: the description of the database entities, showing less of the technical part involved in the search, for example: SUM(attribute). Overall, I really enjoyed the chat.*"
Q28	"*The color and design of the metrics and dimensions were the same as the clickable options, so sometimes it got a little confusing. The data comes fast so you have to scroll up to check everything.*"

[a] "Did the integrated answer make sense?"
[b] "Was it interesting to visualize in the chat answers from different databases?"
[c] "Besides changing the database and attributes, would you add any other human decision during the conversation?"
[d] "Leave your general feedback (features that you liked the most, features you would like to change, etc.)"

users have expressed useful comments, from which the highlights are that are shown in Table 7.8. Most part of the comments exposed weaknesses in the chat interface, such as the quantity of information in the replies and the impossibility to choose more than one attribute at a time. Such statements clarify the lower scores for *Visibility* and *Support* categories of questions, illustrated in Figure 7.6. One user also stated that "the bot seemed not to be learning from previous interactions", however, that was the expected situation, since the user received "1" as evaluation code (i.e., a static evaluation). In contrast, positive comments were left in the evaluation form, such as "very useful", "faster than googling", "I liked the possibility of changing attributes", and "I really enjoyed the chat". From the users answers and comments, we can infer that the approach was able to provide support (either by producing faster replies accessible through graphical interface and by recommending useful attributes), but not in a sufficient level, due to deficiencies observed in the prototype interface.

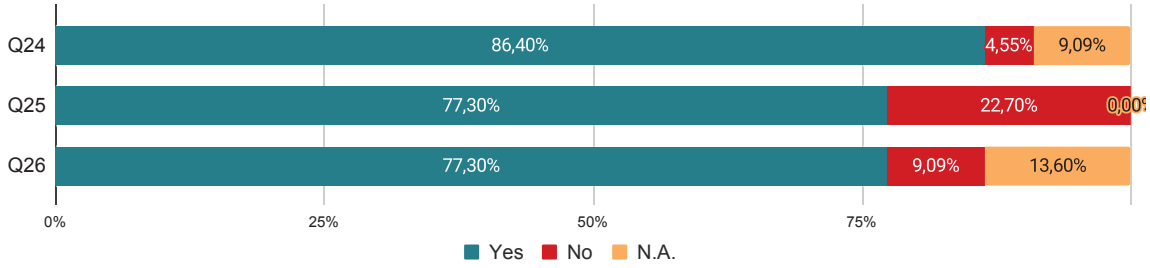


Figure 7.7: User feedback for Data Integration questions.

Overall, the results for the last category of questions in the evaluation form were positive, since the positive answers reached, in average, approximately 80% of the participants that evaluated the prototype. This allows us to establish a promising connection between Built-up Integration features and the Question Answering domain, specially regarding Source Discovery tasks that often rely on automatic methods for finding a source. In other words, user feedback collected through a conversational interface can be used to determine the relevance of a discovered source, and redirect the search, if necessary. Also, the sample of positive answers about integrating information from different sources indicates that Built-up Integration makes sense in QA tasks, i.e, a conversational interface is able bring the user closer to the integration process and therefore closer to more informed decisions.

7.3 CHAPTER REMARKS

The objective of the experiment described in this chapter was to investigate the impact of human knowledge on the recommendation of solutions, assuming a CBR context and a conversational prototype. Being the history of previous cases indispensable for learning in the CBR methodology, this chapter presented an empirical evaluation based on two types of tests that involved *static* and *dynamic* histories, respectively. The difference between them was the possibility of changing content based on user feedback, being the dynamic history the one that could improve future recommendations after the user review.

The comparison between static and dynamic evaluations showed that recommendations and user acceptance were improved when feedback was used to modify the case base. While the static evaluations were unstable regarding the correctness of the attributes recommended, the dynamic evaluations showed that, after a certain number of interactions, the correct recommendations were sustained, corresponding to almost 90% of the interactions. Also, regarding

the categories of linear questions presented in the user evaluation form, the scores were quite superior for *Usefulness, Transparency, Justification*, and *Relevance* criteria. No improvements were observed for the remaining categories (*Visibility, Support*, and *Simplicity*), mostly due to limitations in the way that information was displayed in the interface (see Table 7.8).

One of the limitations was observed through some user comments exposing that, occasionally, the chatbot presented too many options or texts in the chat, interfering with usability. Compressing several options within the same chatbot state was a choice to deal with the DialogFlow *quotas*[28], which are predefined constraints applied to limit the service load per customer. The more training sentences were defined in the prototype code, the more frequent the quota limit caused exceptions in the execution. Thus, the number of states implemented was reduced, implicating in less interactions and more sentences per interaction. Other comments left in the evaluation form concerned the colors in the chat and layout dimensions. These features are standard in the framework used for development (the Xatkit Framework), so they were not modified. An additional limitation concerned the adaptation of the interaction to follow the CBR methodology. One user mentioned, for example, that the chatbot should give metrics and dimensions options before trying to give some answer. However, for testing the *Retrieval* step of the CBR cycle, it was necessary to recommend attributes from the history or discovered source based on a similarity/usefulness score. Also, with these recommendations, the user *Review* could take place, thus allowing to fully collect human feedback.

Although the objective of the experiment (and the thesis itself) did not include the implementation of an optimal conversational interface, we believe that, with a more consistent interface, *Visibility, Support*, and *Simplicity* questions would be better rated by the participants. Despite the limitations, the experiment results were positive, demonstrating that user feedback can be used for improving systems recommendations, and the application of the CBR methodology to conversational contexts is able to support Built-up Integration tasks.

[28]For example, *requests per minute*. Documentation available at <`https://cloud.google.com/dialogflow/quotas`>.

8 CONCLUSIONS

In response to the complexities of situational data management, this thesis presented a novel conversational Case-Based Reasoning (CBR) architecture that integrates the user feedback into the CBR cycle to refine the process of identifying and utilizing situational data. The feedback was collected through a conversational interface implemented as a chatbot application.

First, in order to identify improvement points that could support the conversational architecture, a survey on Situational Data Integration in Question Answering systems (a subdomain of conversational interfaces) was conducted, covering two decades of literature studies. The survey was motivated by several benefits that the joint use of the concepts can bring to end users in terms of decision support and information completeness. Besides demonstrating SDI characteristics that remain challenging, and others that are moving towards consolidation in the QA domain, a lack of standardization involving SDI and other types of on-the-fly integration was identified, were core activities were frequently found in the literature under many different names.

Addressing this issue, the *Built-up Integration* terminology and taxonomy was proposed, as a knowledge regulation approach. The taxonomy is defined through three main features: Data Retrieval, On-the-fly Integration, and Data Delivery, which as common activities present in approaches such as SDI, pay-as-you-go integration, traversal-based integration, and mashup services. All features are challenging when considering the need for real time information and the heterogeneity of external datasets. In this context, it is necessary to leverage the user experience towards the delivery of fast answers and effective management of situational data.

Based on the opportunities exposed with the conducted survey and Built-up Integration taxonomy, the conversational CBR-based architecture was proposed based on four phases, *Retrieval, Reuse, Review,* and *Retain*. The Retrieval step receives an input question as a new case and retrieves from a historic knowledge base the source that best matches the question. The process is based on multidimensional data (i.e., metrics and dimensions) that are similar to the new case. When no similar case is found in the history, Source Discovery is performed to retrieve a source on-the-fly. With the retrieved source, multidimensional solutions are recommended to the user through a conversational interface (i.e., a chatbot), which is the Reuse step of the CBR cycle. In the next step, Revision, the user can manage the attributes suggested, from which a SQL query is executed to bring the final answer and display it in the chat. Finally, in the Retain step, the case repaired is stored in the historic knowledge base to be considered in further interactions.

The CBR-based approach was evaluated through three different experiments that assessed different phases of the cycle. The first one assessed a Source Discovery strategy to be included in the proposal. Three methods were first compared regarding source retrieval, and then blended in order to pursue a better performance. Although the matching tasks showed a higher accuracy for the hybrid approach in comparison with the methods executed separately, all methods showed good results, demonstrating its application potential to discover data within the architecture.

The second experiment (related to the Reuse step of the proposal) was based on a chatbot prototype as the architecture's conversational interface. A conversation flow that defines the chatbot actions was designed to investigate how metrics and dimensions could be proposed to the user. An empirical study was conducted for the chatbot evaluation, covering four qualitative criteria (Visibility, Support, Usefulness, and Simplicity). The experiment results showed that participants were able to use the chatbot for building dimensional queries, being Visibility and Usefulness the most scored features. Based on the results, the chatbot was considered a suitable option to interact with users, capture feedback, and execute the additional steps of the CBR cycle.

The third experiment covered the Review and Retain phases of the proposed architecture. Thus, for allowing the participants to express preferences about suggested attributes, and for testing the learning strategy based on historic knowledge, a history file containing previous cases was included in the experiment, and a set of modifications was made in the chatbot states. The evaluation with participants addressed both static and dynamic testing environments, meaning that the historic knowledge base could be transformed or not, based on the type of test assigned to each user. Besides the criteria used as basis in the second experiment, the third experiment focused on Data Integration aspects and CBR goals such as Transparency, Justification, and Relevance. The results underscored a clear preference for recommendations shaped by user input, demonstrating the importance of incorporating human knowledge in the learning process. Also, the approach covers the Built-up Integration features previously identified, by performing source discovery and selection, data augmentation made at query time, and user support through a useful visualization of integrated data.

Based on the experiments conducted, the main contribution of this thesis is the joint use of Case-Based Reasoning and a conversational interface as a valuable strategy to improve situational data management. The capture of user knowledge and its use for incremental learning within the approach sustain the fact that CBR and conversational interfaces make sense together, and should be leveraged to optimize data retrieval, integration, and delivery. In addition to the work contributions, many opportunities are linked to CBR approaches and deserve attention in future researches. Regarding the proposed architecture, specifically, one interesting direction would be adding weight-based retrieval in the cycle, so that sources could be retrieved according not only to Usefulness and Similarity scores, but also features such as computational cost, temporal evolution of the facts, and privacy. Considering LLMs as a huge trending topic, they could be explored to this end, learning how to retrieve a source based on features that are relevant for building a recommendation.

Also regarding LLMs, another opportunity for future work is investigating how Retrieval Augmented Generation (RAG) models could be used to provide situational data within a CBR architecture (specially in scenarios where real time decision making is needed). In situations where the case details are constantly changing, RAG models can adapt to the new requirements and change the search strategy to retrieve suitable sources. Also, RAG could be used to provide case descriptions when there is a lack of real-world context, facilitating Source Discovery and increasing the probability of returning relevant cases from the history.

Finally, the study conducted and described in this thesis contributes to the evolving landscape of situational data management. The integration of conversational CBR with user feedback showcases a promising direction for enhancing data integration processes. While acknowledging the need for further development, this work marks a step forward in realizing the potential of user-informed conversational systems in managing complex data environments.

REFERENCES

[1] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

[2] Zahra Abbasiantaeb and Saeedeh Momtazi. Text-based question answering from information retrieval and deep neural network perspectives: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(6):e1412, 2021.

[3] Alberto Abelló, Jérôme Darmont, Lorena Etcheverry, Matteo Golfarelli, Jose-Norberto Mazón, Felix Naumann, Torben Pedersen, Stefano Bach Rizzi, Juan Trujillo, Panos Vassiliadis, et al. Fusion cubes: Towards self-service business intelligence. *International Journal of Data Warehousing and Mining (IJDWM)*, 9(2):66–88, 2013.

[4] Alberto Abelló, Oscar Romero, Torben Bach Pedersen, Rafael Berlanga, Victoria Nebot, Maria Jose Aramburu, and Alkis Simitsis. Using semantic web technologies for exploratory olap: a survey. *IEEE transactions on knowledge and data engineering*, 27(2):571–588, 2014.

[5] Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 373–383. Springer, 2020.

[6] Mobyen Uddin Ahmed, Shahina Begum, and Peter Funk. Case studies on the clinical applications using case-based reasoning. In *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, pages 3–10. IEEE, 2012.

[7] Sabbir Ahmed and Umar Ruhi. Towards a functional taxonomy of enterprise business intelligence mashups. In *2013 Second International Conference on Informatics & Applications (ICIA)*, pages 98–103. IEEE, 2013.

[8] Waheeb Ahmed and P Babu Anto. A hybrid question answering system. *Current Journal of Applied Science and Technology*, pages 1–7, 2019.

[9] José Luis Ambite, Vinay K Chaudhri, Richard Fikes, Jessica Jenkins, Sunil Mishra, Maria Muslea, Tomas Uribe, and Guizhen Yang. Integration of heterogeneous knowledge sources in the calo query manager. *Lecture notes in computer science*, 3762:30, 2005.

[10] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[11] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. When is approximate counting for conjunctive queries tractable? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1015–1027, 2021.

[12] Yigal Arens, Chin Y Chee, Chun-Nan Hsu, and Craig A Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(02):127–158, 1993.

[13] Maurizio Atzori, Giuseppe M Mazzeo, and Carlo Zaniolo. Qa3: A natural language approach to question answering over rdf data cubes. *Semantic Web*, 10(3):587–604, 2019.

[14] Nurzety Aqtar Ahmad Azuan. *Exploring Manual Correction as a Source of User Feedback in Pay-As-You-Go Integration*. PhD thesis, The University of Manchester (United Kingdom), 2021.

[15] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440, 2015.

[16] Kaveh Bastani, Hamed Namavari, and Jeffrey Shaffer. Latent dirichlet allocation (lda) for topic modeling of the cfpb consumer complaints. *Expert Systems with Applications*, 127:256–271, 2019.

[17] Don Batory. Feature models, grammars, and propositional formulas. In *International Conference on Software Product Lines*, pages 7–20. Springer, 2005.

[18] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[19] Susanna Bonura, Giuseppe Cammarata, Rosolino Finazzo, Giuseppe Francaviglia, and Vito Morreale. A novel webgis-based situational awareness platform for trustworthy big data integration and analytics in mobility context. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 86–98. Springer, 2017.

[20] Antoine Bosselut, Ronan Le Bras, and Yejin Choi. Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. In *Proceedings of the AAAI conference on Artificial Intelligence*, pages 4923–4931, 2021.

[21] Abdelghani Bouziane, Djelloul Bouchiha, Noureddine Doumi, and Mimoun Malki. Question answering systems: survey and trends. *Procedia Computer Science*, 73:366–375, 2015.

[22] Diana Bratić, Marko Šapina, Denis Jurečić, and Jana Žiljak Gršić. Centralized database access: Transformer framework and llm/chatbot integration-based hybrid model. *Applied System Innovation*, 7(1):17, 2024.

[23] Derek Bridge, Mehmet H Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(3):315–320, 2005.

[24] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, Andrew Ng, et al. Data-intensive question answering. In *TREC*, volume 56, page 90, 2001.

[25] Kellyton Brito, Marcos Silva Costa, Vinicius Garcia, and Silvio Meira. Brazilian government open data: Implementation, challenges, and potential opportunities. In *ACM International Conference Proceeding Series*, 2014.

[26] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International semantic web conference*, pages 54–68. Springer, 2002.

[27] Brian L Cairns, Rodney D Nielsen, James J Masanz, James H Martin, Martha S Palmer, Wayne H Ward, and Guergana K Savova. The mipacq clinical question answering system. In *AMIA annual symposium proceedings*, volume 2011, page 171. American Medical Informatics Association, 2011.

[28] YongGang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J Cimino, John Ely, and Hong Yu. Askhermes: An online question answering system for complex clinical questions. *Journal of biomedical informatics*, 44(2):277–288, 2011.

[29] Malu Castellanos, Chetan Gupta, Song Wang, Umeshwar Dayal, and Miguel Durazo. A platform for situational awareness in operational bi. *Decision Support Systems*, 52(4):869–883, 2012.

[30] Xiang Chen, Ningyu Zhang, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, and Huajun Chen. Hybrid transformer with multi-level fusion for multimodal knowledge graph completion. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 904–915, 2022.

[31] Xiaoling Chen, Anupama E Gururaj, Burak Ozyurt, Ruiling Liu, Ergin Soysal, Trevor Cohen, Firat Tiryaki, Yueling Li, Nansu Zong, Min Jiang, et al. Datamed–an open source discovery index for finding biomedical datasets. *Journal of the American Medical Informatics Association*, 25(3):300–308, 2018.

[32] Xin Chen, Yanbo Han, Yan Wen, Feng Zhang, and Wei Liu. A keyword-driven data service mashup plan generation approach for ad-hoc data query. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 394–401. IEEE, 2017.

[33] Zheqian Chen, Chi Zhang, Zhou Zhao, Chengwei Yao, and Deng Cai. Question retrieval for community-based question answering via heterogeneous social influential network. *Neurocomputing*, 285:117–124, 2018.

[34] Bo Cheng, Shuai Zhao, Junyan Qian, Zhongyi Zhai, and Junliang Chen. Lightweight service mashup middleware with REST style architecture for iot applications. *IEEE Transactions on Network and Service Management*, 15(3):1063–1075, 2018.

[35] Fatma Chiheb, Fatima Boumahdi, and Hafida Bouarfa. A new model for integrating big data into phases of decision-making process. *Procedia Computer Science*, 151:636–642, 2019.

[36] Sutanay Choudhury, Khushbu Agarwal, Sumit Purohit, Baichuan Zhang, Meg Pirrung, Will Smith, and Mathew Thomas. Nous: Construction and querying of dynamic knowledge graphs. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1563–1565. IEEE, 2017.

[37] Klitos Christodoulou. *On techniques for pay-as-you-go data integration of linked data*. PhD thesis, University of Manchester, 2014.

[38] Jennifer Chu-Carroll, John Prager, Christopher Welty, Krzysztof Czuba, and David Ferrucci. A multi-strategy and multi-source approach to question answering. Technical report, IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY, 2006.

[39] Valerie Claessen, Adrian Schmidt, and Tamara Heck. Virtual assistants - a study on the usability and user perception of customer service systems for e-commerce. In *Proceedings of the 15th International Symposium of Information Science (ISI)*, pages 116–130, 2017.

[40] Charles LA Clarke, Gordon V Cormack, Graeme Kemkes, Michael Laszlo, Thomas R Lynam, Egidio L Terra, and Philip L Tilker. Statistical selection of exact answers (multitext experiments for trec 2002). In *TREC*. Citeseer, 2002.

[41] Edward Curry, Wassim Derguech, Souleiman Hasan, Christos Kouroupetroglou, and Umair ul Hassan. A real-time linked dataspace for the internet of things: enabling "pay-as-you-go" data management in smart environments. *Future Generation Computer Systems*, 90:405–422, 2019.

[42] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Freya: An interactive way of querying linked data using natural language. In *Extended semantic web conference*, pages 125–138. Springer, 2011.

[43] Florian Daniel, Maristella Matera, Elisa Quintarelli, Letizia Tanca, and Vittorio Zaccaria. Context-aware access to heterogeneous resources through on-the-fly mashups. In *International Conference on Advanced Information Systems Engineering*, pages 119–134. Springer, 2018.

[44] Gwendal Daniel, Jordi Cabot, Laurent Deruelle, and Mustapha Derras. Xatkit: a multimodal low-code chatbot development framework. *IEEE Access*, 8:15332–15346, 2020.

[45] Rajarshi Das, Ameya Godbole, Ankita Naik, Elliot Tower, Manzil Zaheer, Hannaneh Hajishirzi, Robin Jia, and Andrew McCallum. Knowledge base question answering by case-based reasoning over subgraphs. In *International conference on machine learning*, pages 4777–4793. PMLR, 2022.

[46] Anish Das Sarma, Xin Dong, and Alon Halevy. Bootstrapping pay-as-you-go data integration systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 861–874, 2008.

[47] Lucas F de Oliveira, Alessandro Elias, Fabiola Santore, Diego Pasqualin, Luis CE Bona, Marcos Sunyé, and Marcos Didonet Del Fabro. Open data analytic querying using a relation-free api. In *22nd International Conference on Enterprise Information Systems (ICEIS)*, 2020.

[48] Matthew J Denny and Arthur Spirling. Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, 26(2):168–189, 2018.

[49] Serena Di Gaetano and Pietro Diliberto. Chatbots and conversational interfaces: Three domains of use. In *Fifth International Workshop on Cultures of Participation in the Digital Age, Castiglione della Pescaia, Italy*, volume 2101, pages 62–70, 2018.

[50] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Multidimensional query reformulation with measure decomposition. *Information Systems*, 78:23–39, 2018.

[51] Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information systems*, 55(3):529–569, 2018.

[52] Dennis Diefenbach, Kamal Singh, and Pierre Maret. On the scalability of the qa system wdaqua-core1. In *Semantic Web Evaluation Challenge*, pages 76–81. Springer, 2018.

[53] Eleftherios Dimitrakis, Konstantinos Sgontzos, Michalis Mountantonakis, and Yannis Tzitzikas. Enabling efficient question answering over hundreds of linked datasets. In *International Workshop on Information Search, Integration, and Personalization*, pages 3–17. Springer, 2019.

[54] Rahma Djiroun, Kamel Boukhalfa, and Zaia Alimazighi. Designing data cubes in olap systems: a decision makers' requirements-based approach. *Cluster Computing*, 22(3), 2019.

[55] Tuong Do, Binh X Nguyen, Erman Tjiputra, Minh Tran, Quang D Tran, and Anh Nguyen. Multiple meta-model quantifying for medical visual question answering. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part V 24*, pages 64–74. Springer, 2021.

[56] Valentina Dragos and Sylvain Gatepaille. On-the-fly integration of soft and sensor data for enhanced situation assessment. *Procedia computer science*, 112:1263–1272, 2017.

[57] Hamza Ed-Douibi, Gwendal Daniel, and Jordi Cabot. Openapi bot: A chatbot to help you understand rest apis. In *International Conference on Web Engineering*, pages 538–542. Springer, 2020.

[58] Hamza Ed-douibi, Javier Luis Cánovas Izquierdo, Gwendal Daniel, and Jordi Cabot. A model-based chatbot generation approach to converse with open data sources. *arXiv preprint arXiv:2007.10503*, 2020.

[59] Ahmed El-Roby. *Web Data Integration for Non-Expert Users*. PhD thesis, University of Waterloo, 2018.

[60] Mica R Endsley. Toward a theory of situation awareness in dynamic systems. *Human factors*, 37(1):32–64, 1995.

[61] Andre Esteva, Anuprit Kale, Romain Paulus, Kazuma Hashimoto, Wenpeng Yin, Dragomir Radev, and Richard Socher. Covid-19 information retrieval with deep-learning based semantic search, question answering, and abstractive summarization. *NPJ digital medicine*, 4(1):68, 2021.

[62] Lorena Etcheverry, Alejandro Vaisman, and Esteban Zimanyi. Modeling and querying data warehouses on the semantic web using qb4olap. In *16th International Conference on Data Warehousing and Knowledge Discovery*, volume 8646, 09 2014.

[63] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165, 2014.

[64] Pavlos Fafalios and Yannis Tzitzikas. How many and what types of SPARQL queries can be answered through zero-knowledge link traversal? In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 2267–2274, 2019.

[65] Ronen Feldman, James Sanger, et al. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007.

[66] Jiayi Feng, Runtong Zhang, Donghua Chen, and Wei Zhang. Extracting meaningful correlations among heterogeneous datasets for medical question answering with domain knowledge. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 297–301. IEEE, 2018.

[67] Jannatul Ferdush, Mahbuba Begum, and Sakib Tanvir Hossain. Chatgpt and clinical decision support: Scope, application, and limitations. *Annals of Biomedical Engineering*, pages 1–6, 2023.

[68] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE, 2018.

[69] Raul Castro Fernandez, Essam Mansour, Abdulhakim A Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 989–1000. IEEE, 2018.

[70] Antonio Ferrández, Alejandro Maté, Jesús Peral, Juan Trujillo, Elisa De Gregorio, and Marie-Aude Aufaure. A framework for enriching data warehouse analysis with question answering systems. *Journal of Intelligent Information Systems*, 46(1):61–82, 2016.

[71] Antonio Ferrández and Jesús Peral. The benefits of the interaction between data warehouses and question answering. In *Proceedings of the 2010 EDBT/ICDT Workshops*, pages 1–8, 2010.

[72] Óscar Ferrández, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrández, Matteo Negri, Rubén Izquierdo, David Tomás, Constantin Orasan, Guenter Neumann, et al. The qall-me framework: A specifiable-domain multilingual question answering architecture. *Journal of Web Semantics*, 9(2):137–145, 2011.

[73] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[74] Matteo Francia, Enrico Gallinucci, and Matteo Golfarelli. Towards conversational olap. In *DOLAP*, pages 6–15, 2020.

[75] Maria Helena Franciscatto, Iara Augustin, João Carlos Damasceno Lima, and Vinícius Maran. Situation awareness in the speech therapy domain: A systematic mapping study. *Computer Speech & Language*, 53:92–120, 2019.

[76] Maria Helena Franciscatto, Marcos Didonet Del Fabro, Luis Carlos Erpen De Bona, Celio Trois, and Hegler Tissot. Blending topic-based embeddings and cosine similarity for open data discovery. In *ICEIS (1)*, pages 163–170, 2022.

[77] Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20–48, 2007.

[78] Michael Franklin, Alon Halevy, and David Maier. A first tutorial on dataspaces. *Proceedings of the VLDB Endowment*, 1(2):1516–1517, 2008.

[79] Ingo Frommholz, Haiming Liu, and Massimo Melucci. Birds-bridging the gap between information science, information retrieval and data science. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2455–2458, 2020.

[80] Tim Furche, George Gottlob, Leonid Libkin, Giorgio Orsi, and Norman Paton. Data wrangling for big data: Challenges and opportunities. In *Advances in Database Technology—EDBT 2016: Proceedings of the 19th International Conference on Extending Database Technology*, pages 473–478, 2016.

[81] Domenico Gadaleta, Anna Lombardo, Cosimo Toma, and Emilio Benfenati. A new semi-automated workflow for chemical data retrieval and quality checking for modeling applications. *Journal of cheminformatics*, 10(1):1–13, 2018.

[82] Jianfeng Gao, Michel Galley, and Lihong Li. *Neural approaches to conversational AI: Question answering, task-oriented dialogues and social chatbots*. Now Foundations and Trends, 2019.

[83] Daniel Gerber and A-C Ngonga Ngomo. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction@ ISWC*, volume 2011, 2011.

[84] Sudeep Ghimire, Fernando Luis-Ferreira, Tahereh Nodehi, and Ricardo Jardim-Goncalves. Iot based situational awareness framework for real-time project management. *International Journal of Computer Integrated Manufacturing*, 30(1):74–83, 2017.

[85] Ashok Goel and Belen Diaz-Agudo. What's hot in case-based reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[86] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[87] Wael H Gomaa, Aly A Fahmy, et al. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, 2013.

[88] Arthur C Graesser and Stanley P Franklin. Quest: A cognitive model of question answering. *Discourse processes*, 13(3):279–303, 1990.

[89] Lars Grammel and Margaret-Anne Storey. A survey of mashup development environments. In *The smart internet*, pages 137–151. Springer, 2010.

[90] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224, 1961.

[91] Dani Gunawan, CA Sembiring, and MA Budiman. The implementation of cosine similarity to calculate text relevance between two documents. *Journal of Physics: Conference Series*, 978(1):012120, 2018.

[92] Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven Hoi. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10867–10877, 2023.

[93] Qing-lin Guo. A novel approach for question answering and automatic diagnosis based on pervasive agent ontology in medicine. *International Journal of Intelligent Systems*, 25(7):655–682, 2010.

[94] Poonam Gupta and Vishal Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4), 2012.

[95] Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

[96] Rihan Hai, Sandra Geisler, and Christoph Quix. Constance: An intelligent data lake system. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2097–2100, 2016.

[97] Alon Halevy, Michael Franklin, and David Maier. Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–9, 2006.

[98] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques third edition. *The Morgan Kaufmann Series in Data Management Systems*, 5(4):83–124, 2011.

[99] Yanbo Han, Guiling Wang, Guang Ji, and Peng Zhang. Situational data integration with data services and nested table. *Service Oriented Computing and Applications*, 7(2):129–150, 2013.

[100] Andreas Harth, Craig A Knoblock, Steffen Stadtmüller, Rudi Studer, and Pedro Szekely. On-the-fly integration of static and dynamic linked data. In *Proceedings of the Fourth International Workshop on Consuming Linked Data co-located with the 12th International Semantic Web Conference*, pages 1613–0073, 2013.

[101] Olaf Hartig. SQUIN: a traversal based query execution system for the web of linked data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1081–1084, 2013.

[102] Olaf Hartig. *Linked Data Query Processing Based on Link Traversal*, pages 263–283. Linked Data Management, 05 2014.

[103] Olaf Hartig and Johann-Christoph Freytag. Foundations of traversal based query execution over linked data. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 43–52, 2012.

[104] Olaf Hartig and M Tamer Özsu. Walking without a map: Ranking-based traversal for querying linked data. In *International Semantic Web Conference*, pages 305–324. Springer, 2016.

[105] Cornelia Hedeler, Khalid Belhajjame, Alvaro AA Fernandes, Suzanne M Embury, and Norman W Paton. Dimensions of dataspaces. In *British National Conference on Databases*, pages 55–66. Springer, 2009.

[106] Daniel M Herzig and Thanh Tran. Heterogeneous web data search using relevance-based on the fly data integration. In *Proceedings of the 21st international conference on World Wide Web*, pages 141–150, 2012.

[107] Wesley Hildebrandt, Boris Katz, and Jimmy Lin. Answering definition questions with multiple knowledge sources. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 49–56, 2004.

[108] Stefanie Hillig and Romy Müller. How do conversational case-based reasoning systems interact with their users: a literature review. *Behaviour & Information Technology*, 40(14):1544–1563, 2021.

[109] Pascal Hirmer and Bernhard Mitschang. TOSCA4Mashups: enhanced method for on-demand data mashup provisioning. *Computer Science-Research and Development*, 32(3-4):291–300, 2017.

[110] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[111] Konrad Höffner, Jens Lehmann, and Ricardo Usbeck. Cubeqa—question answering on rdf data cubes. In *International Semantic Web Conference*, pages 325–340. Springer, 2016.

[112] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[113] Angus FM Huang, Shin Bo Huang, Evan YF Lee, and Stephen JH Yang. Improving end-user programming with situational mashups in web 2.0 environment. In *2008 IEEE International Symposium on Service-Oriented System Engineering*, pages 62–67. IEEE, 2008.

[114] Yizheng Huang and Jimmy X Huang. Exploring chatgpt for next-generation information retrieval: Opportunities and challenges. In *Web Intelligence*, number Preprint, pages 1–14. IOS Press, 2024.

[115] Yunhan Huang, Linan Huang, and Quanyan Zhu. Reinforcement learning for feedback-enabled cyber resilience. *Annual reviews in control*, 53:273–295, 2022.

[116] Matthias Jarke and Christoph Quix. Federated data integration in data spaces. In *Designing Data Spaces*, pages 181–194. Springer, 2022.

[117] Joanna Jedrzejowicz and Magdalena Zakrzewska. Text classification using lda-w2v hybrid algorithm. In *Intelligent Decision Technologies 2019*, pages 227–237. Springer, 2020.

[118] Shawn R Jeffery, Michael J Franklin, and Alon Y Halevy. Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 847–860, 2008.

[119] Valentin Jijkoun and Maarten De Rijke. Answer selection in a multi-stream open domain question answering system. In *European Conference on Information Retrieval*, pages 99–111. Springer, 2004.

[120] Clement Jonquet, Paea LePendu, Sean Falconer, Adrien Coulet, Natalya F Noy, Mark A Musen, and Nigam H Shah. Ncbo resource index: Ontology-based search and mining of biomedical resources. *Journal of Web Semantics*, 9(3):316–324, 2011.

[121] Chrisna Jooste, Judy Van Biljon, and Jan Mentz. Usability evaluation guidelines for business intelligence applications. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 331–340, 2013.

[122] Petar Jovanovic, Sergi Nadal, Oscar Romero, Alberto Abelló, and Besim Bilalli. Quarry: a user-centered big data integration platform. *Information Systems Frontiers*, 23(1):9–33, 2021.

[123] Petar Jovanovic, Oscar Romero, and Alberto Abelló. A unified view of data-intensive flows in business intelligence systems: a survey. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXIX*, pages 66–107. Springer, 2016.

[124] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, DTIC Document, 1990.

[125] Subbu Kannan, Vairaprakash Gurusamy, S Vijayarani, J Ilamathi, and M Nithya. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2014.

[126] Julia Kantorovitch, Ilkka Niskanen, Jarmo Kalaoja, and Toni Staykova. Designing situation awareness addressing the needs of medical emergency response. In *12th International Conference on Software Technologies, ICSOFT 2017*, pages 467–472. SciTePress, 2017.

[127] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *International Conference on Application of Natural Language to Information Systems*, pages 230–234. Springer, 2002.

[128] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[129] Thomas Kirk, Alon Y Levy, Yehoshua Sagiv, Divesh Srivastava, et al. The information manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, volume 7, pages 85–91, 1995.

[130] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 conference on designing interactive systems*, pages 555–565, 2017.

[131] Janet L Kolodneer. Improving human decision making through case-based decision aiding. *AI magazine*, 12(2):52–52, 1991.

[132] Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, 2011.

[133] Parisa Kordjamshidi, Sameer Singh, Daniel Khashabi, Christos Christodoulopoulos, Mark Summons, Saurabh Sinha, and Dan Roth. Relational learning and feature extraction by querying over heterogeneous information networks. *arXiv preprint arXiv:1707.07794*, 2017.

[134] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. Valentine: Evaluating matching techniques for dataset discovery. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 468–479. IEEE, 2021.

[135] Rudolf Laine, Alexander Meinke, and Owain Evans. Towards a situational awareness benchmark for llms. In *Socially Responsible Language Modelling Research*, 2023.

[136] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Shreya Bisen, and Vasundhara Rathod. Implementation of a chat bot system using ai and nlp. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*, 6(3):26–30, 2018.

[137] Jean-Baptiste Lamy, Boomadevi Sekar, Gilles Guezennec, Jacques Bouaud, and Brigitte Séroussi. Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. *Artificial intelligence in medicine*, 94:42–53, 2019.

[138] Rodziah Latih, Ahmed Moosajee Patel, Abdullah Mohd Zin, Tew Yiqi, and Siti Hafizah Muhammad. Whip: A framework for mashup development with block-based development approach. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–6. IEEE, 2011.

[139] Yong-Ju Lee. Semantic-based data mashups using hierarchical clustering and pattern analysis methods. *J. Inf. Sci. Eng.*, 30(5):1601–1618, 2014.

[140] Yong-Ju Lee and Jae-Soo Kim. Automatic web api composition for semantic data mashups. In *2012 Fourth International Conference on Computational Intelligence and Communication Networks*, pages 953–957. IEEE, 2012.

[141] Alon Levy, Anand Rajaraman, and Joann Ordille. Querying heterogeneous information sources using source descriptions. Technical report, Stanford InfoLab, 1996.

[142] Guoliang Li. Human-in-the-loop data integration. *Proceedings of the VLDB Endowment*, 10(12):2006–2017, 2017.

[143] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2020.

[144] Q Vera Liao, Werner Geyer, Michael Muller, and Yasaman Khazaen. Conversational interfaces for information search. In *Understanding and improving information search*, pages 267–287. Springer, 2020.

[145] Pongsakorn Limna, Tanpat Kraiwanit, Kris Jangjarat, Prapasiri Klayklung, and Piyawatjana Chocksathaporn. The use of chatgpt in the digital era: Perspectives on chatbot implementation. *Journal of Applied Learning and Teaching*, 6(1), 2023.

[146] Jinying Lin, Zhen Ma, Randy Gomez, Keisuke Nakamura, Bo He, and Guangliang Li. A review on interactive reinforcement learning from human social feedback. *IEEE Access*, 8:120757–120765, 2020.

[147] Chen Liu, Jianwu Wang, Yanbo Han, et al. Discovery of service hyperlinks with user feedbacks for situational data mashup. *International Journal of Database Theory and Application*, 8(4):71–80, 2015.

[148] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. Joint knowledge graph completion and question answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1098–1108, 2022.

[149] Siru Liu, Aileen P Wright, Barron L Patterson, Jonathan P Wanderer, Robert W Turer, Scott D Nelson, Allison B McCoy, Dean F Sittig, and Adam Wright. Using ai-generated suggestions from chatgpt to optimize clinical decision support. *Journal of the American Medical Informatics Association*, 30(7):1237–1245, 2023.

[150] Zelong Liu, Maozhen Li, Yang Liu, and Mahesh Ponraj. Performance evaluation of latent dirichlet allocation in text mining. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 4, pages 2695–2698. IEEE, 2011.

[151] Andreas Lommatzsch. A next generation chatbot-framework for the public administration. In *International Conference on Innovations for Community Services*, pages 127–141. Springer, 2018.

[152] Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic web*, 3(3):249–265, 2012.

[153] Vanessa Lopez, Pierpaolo Tommasi, Spyros Kotoulas, and Jiewen Wu. Queriodali: Question answering over dynamic and linked knowledge graphs. In *International Semantic Web Conference*, pages 363–382. Springer, 2016.

[154] Alexander Löser, Fabian Hueske, and Volker Markl. Situational business intelligence. In *International Workshop on Business Intelligence for the Real-Time Enterprise*, pages 1–11. Springer, 2008.

[155] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 105–114, 2019.

[156] Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *AAAI*, pages 8449–8456, 2020.

[157] Mary Lou Maher, M Bala Balachandran, and Dong Mei Zhang. *Case-based reasoning in design*. Psychology Press, 2014.

[158] Senthil Mani, Neelamadhav Gantayat, Rahul Aralikatte, Monika Gupta, Sampath Dechu, Anush Sankaran, Shreya Khare, Barry Mitchell, Hemamalini Subramanian, and Hema Venkatarangan. Hi, how can i help you?: Automating enterprise it support help desks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[159] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[160] Edgard Marx, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Konrad Höffner, Jens Lehmann, and Sören Auer. Towards an open question answering architecture. In *Proceedings of the 10th International Conference on Semantic Systems*, pages 57–60, 2014.

[161] Ruhaila Maskat. *Pay-As-You-Go Instance-Level Integration*. PhD thesis, The University of Manchester (United Kingdom), 2016.

[162] Maroua Masmoudi, Sana Ben Abdallah Ben Lamine, Hajer Baazaoui Zghal, Bernard Archimede, and Mohamed Hedi Karray. Knowledge hypergraph-based approach for data integration and querying: Application to earth observation. *Future Generation Computer Systems*, 115:720–740, 2021.

[163] Marco Masseroli, Matteo Picozzi, Giorgio Ghisalberti, and Stefano Ceri. Explorative search of distributed bio-data to answer complex biomedical questions. *BMC bioinformatics*, 15(S1):S3, 2014.

[164] Nikolaos Matskanis, Vassiliki Andronikou, Philippe Massonet, Kostas Mourtzoukos, and Joseph Roumier. A linked data approach for querying heterogeneous sources. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, pages 411–414, 2012.

[165] Yûki Matsuyoshi, Tetsuya Takiguchi, and Yasuo Ariki. User's intention understanding in question-answering system using attention-based lstm. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1752–1755. IEEE, 2018.

[166] Michael McCandless, Erik Hatcher, Otis Gospodnetić, and O Gospodnetić. *Lucene in action*, volume 2. Manning Greenwich, 2010.

[167] Michael F McTear. The rise of the conversational interface: A new kid on the block? In *international workshop on future and emerging trends in language technology*, pages 38–49. Springer, 2016.

[168] Mayuri Mhatre, Dakshata Phondekar, Pranali Kadam, Anushka Chawathe, and Kranti Ghag. Dimensionality reduction for sentiment analysis using pre-processing techniques. In *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, pages 16–21. IEEE, 2017.

[169] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[170] Renée J Miller. Open data integration. *Proceedings of the VLDB Endowment*, 11(12):2130–2139, 2018.

[171] Amit Mishra, Nidhi Mishra, and Anupam Agrawal. Context-aware restricted geographical domain question answering system. In *2010 international conference on computational intelligence and communication networks*, pages 548–553. IEEE, 2010.

[172] Paolo Missier, Alvaro AA Fernandes, Roald Lengu, Giovanna Guerrini, and Marco Mesiti. Data quality support to on-the-fly data integration using adaptive query processing. In *SEBD*, pages 213–220, 2009.

[173] Mohamed Mohandes, Mohamed Deriche, and Salihu O Aliyu. Classifiers combination techniques: A comprehensive review. *IEEE Access*, 6, 2018.

[174] Gonzalo Molina Gallego. Analysis, discovery and exploitation of open data for the creation of question-answering systems. *Undergraduate Thesis, Universidad de Alicante*, 2018.

[175] Driss Moujahid, Omar Elharrouss, and Hamid Tairi. Visual object tracking via the local soft cosine similarity. *Pattern Recognition Letters*, 110:79–85, 2018.

[176] Michalis Mountantonakis and Yannis Tzitzikas. Large-scale semantic integration of linked data: A survey. *ACM Computing Surveys (CSUR)*, 52(5):1–40, 2019.

[177] Sergi Nadal, Oscar Romero, Alberto Abelló, Panos Vassiliadis, and Stijn Vansummeren. An integration-oriented ontology to govern evolution in big data ecosystems. *Information systems*, 79:3–19, 2019.

[178] Mario Nadj, Stefan Morana, and Alexander Maedche. Towards a situation-awareness-driven design of operational business intelligence & analytics systems. In *At the Vanguard of Design Science: First Impressions and Early Findings from Ongoing Research Research-in-Progress Papers and Poster Presentations from the 10th International Conference, DESRIST 2015. Dublin, Ireland, 20-22 May.* DESRIST 2015, 2015.

[179] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[180] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment*, 12(12):1986–1989, 2019.

[181] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825, 2018.

[182] Daniela Nicklas, Thomas Schwarz, and Bernhard Mitschang. A schema-based approach to enable data integration on the fly. *International Journal of Cooperative Information Systems*, 26(01):1650010, 2017.

[183] Rodney D Nielsen, James Masanz, Philip Ogren, Wayne Ward, James H Martin, Guergana Savova, and Martha Palmer. An architecture for complex clinical question answering. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 395–399, 2010.

[184] Marcelo Iury S Oliveira, Hélio Rodrigues de Oliveira, Lairson Alencar Oliveira, and Bernadette Farias Lóscio. Open government data portals analysis: the brazilian case. In *Proceedings of the 17th international digital government research conference on digital government research*, pages 415–424, 2016.

[185] Nouha Othman, Rim Faiz, and Kamel Smaïli. Enhancing question retrieval in community question answering using word embeddings. *Procedia Computer Science*, 159:485–494, 2019.

[186] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, and Samir Belfkih. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4):431–448, 2018.

[187] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[188] Mario Andrés Paredes-Valverde, Giner Alor-Hernández, Alejandro Rodríguez-González, Rafael Valencia-García, and Enrique Jiménez-Domingo. A systematic review of tools, languages, and methodologies for mashup development. *Software: Practice and Experience*, 45(3):365–397, 2015.

[189] Seonyeong Park, Soonchoul Kwon, Byungsoo Kim, Sangdo Han, Hyosup Shim, and Gary Geunbae Lee. Question answering system using multiple information source and open type answer merge. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 111–115, 2015.

[190] Norman W Paton, Khalid Belhajjame, Suzanne M Embury, Alvaro AA Fernandes, and Ruhaila Maskat. Pay-as-you-go data integration: Experiences and recurring themes. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 81–92. Springer, 2016.

[191] Norman W Paton, Klitos Christodoulou, Alvaro AA Fernandes, Bijan Parsia, and Cornelia Hedeler. Pay-as-you-go data integration for linked data: opportunities, challenges and architectures. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, pages 1–8, 2012.

[192] Siyuan Qi and Yupin Luo. Object retrieval with image graph traversal-based re-ranking. *Signal Processing: Image Communication*, 41:101–114, 2016.

[193] Chen Qu, Liu Yang, Minghui Qiu, W Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. Bert with history answer embedding for conversational question answering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1133–1136, 2019.

[194] Abdul Quamar, Fatma Özcan, Dorian Miller, Robert J Moore, Rebecca Niehus, and Jeffrey Kreulen. Conversational bi: an ontology-driven conversation system for business intelligence applications. *Proceedings of the VLDB Endowment*, 13(12):3369–3381, 2020.

[195] Muhammad Rafiq, Shahzaib Ashraf, Saleem Abdullah, Tahir Mahmood, and Shakoor Muhammad. The cosine similarity measures of spherical fuzzy sets and their applications in decision making. *Journal of Intelligent & Fuzzy Systems*, 36(6):6059–6073, 2019.

[196] BKN Rao. Case-based reasoning (cbr) in condition monitoring & diagnostic engineering management (comadem): A literature survey. *International Journal of COMADEM*, 20(1):15–22, 2017.

[197] GM Rasiqul Islam Rasiq, Abdullah Al Sefat, Tanjila Hossain, Md Israt-E-Hasan Munna, Jubayeath Jahan Jisha, and Mohammad Moinul Hoque. Question answering system over linked data: A detailed survey. *ABC Research Alert*, 8(1):32–47, 2020.

[198] Mengye Ren, Ryan Kiros, and Richard Zemel. Image question answering: A visual semantic embedding model and a new dataset. *Proc. Advances in Neural Inf. Process. Syst*, 1(2):5, 2015.

[199] John A Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2006.

[200] Christopher K Riesbeck and Roger C Schank. *Inside case-based reasoning*. Erlbaum, Northvale, NJ, 1989.

[201] Gil Rocha and Henrique Lopes Cardoso. Recognizing textual entailment: challenges in the portuguese language. *Information*, 9(4):76, 2018.

[202] Anna Rogers, Matt Gardner, and Isabelle Augenstein. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *ACM Computing Surveys*, 55(10):1–45, 2023.

[203] Rishiraj Saha Roy and Avishek Anand. Question answering over curated and open web sources. *arXiv preprint arXiv:2004.11980*, 2020.

[204] Kuniaki Saito, Kihyuk Sohn, Chen-Yu Lee, and Yoshitaka Ushiku. Unsupervised llm adaptation for question answering. *arXiv preprint arXiv:2402.12170*, 2024.

[205] Lavinia Salicchi and Alessandro Lenci. Pihkers at cmcl 2021 shared task: Cosine similarity and surprisal to predict human reading patterns. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 102–107, 2021.

[206] Fabiola Santore, Lucas F Oliveira, Rafael de Paulo Dias, Henrique V Ehrenfried, Alessandro Elias, Diego Pasqualin, Luis CE de Bona, Marcos Didonet Del Fabro, and Marcos Sunyé. Blended integrated open data: dados abertos publicos integrados. *arXiv preprint arXiv:1909.00743*, 2019.

[207] Denis Savenkov and Eugene Agichtein. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 235–244, 2016.

[208] Nico Schlaefer, Petra Gieselman, and Guido Sautter. The ephyra qa system at trec 2006. In *TREC*, 2006.

[209] Nico Schlaefer, Jeongwoo Ko, Justin Betteridge, Manas A Pathak, Eric Nyberg, and Guido Sautter. Semantic extensions of the ephyra qa system for trec 2007. *TREC*, 1(1):2, 2007.

[210] Pierre-Yves Schobbens, Patrick Heymans, and Jean-Christophe Trigaux. Feature diagrams: A survey and a formal semantics. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 139–148. IEEE, 2006.

[211] Jakob M Schoenborn, Rosina O Weber, David W Aha, Jorg Cassens, and Klaus-Dieter Althoff. Explainable case-based reasoning: A survey. In *AAAI-21 Workshop Proceedings*, 2021.

[212] Marco Antônio Schwertner, Sandro José Rigo, Denis Andrei Araújo, Alan Barcelos Silva, and Bjoern Eskofier. Fostering natural language question answering over knowledge bases in oncology ehr. In *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 501–506. IEEE, 2019.

[213] Urooba Sehar, Iqra Ghazal, Hamna Mansoor, and Sehrish Saba. A comprehensive literature review on approaches, techniques & challenges of mashup development. *International Journal of Scientific & Engineering Research*, 13, 2022.

[214] Floarea Serban, Joaquin Vanschoren, Jörg-Uwe Kietz, and Abraham Bernstein. A survey of intelligent assistants for data analysis. *ACM Computing Surveys (CSUR)*, 45(3):1–35, 2013.

[215] Fernando RS Serrano, Alvaro AA Fernandes, and Klitos Christodoulou. An approach to quantify integration quality using feedback on mapping results. *International Journal of Web Information Systems*, 2018.

[216] Omid Shahmirzadi, Adam Lugowski, and Kenneth Younge. Text similarity in vector space models: a comparative study. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 659–666. IEEE, 2019.

[217] Himanshu Sharma and Anand Singh Jalal. Visual question answering model based on graph neural network and contextual attention. *Image and Vision Computing*, 110:104165, 2021.

[218] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 30:39–51, 2015.

[219] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Question answering on interlinked data. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1145–1156, 2013.

[220] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.

[221] Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, et al. Why reinvent the wheel: Let's build question answering systems together. In *Proceedings of the 2018 World Wide Web Conference*, pages 1247–1256, 2018.

[222] Anumita Sinha, Nishi Mehta, Pramod Bhakta, and Era Johri. On the fly integration of applications using context aware approach. *IJARIIT*, 5(3):435, 2019.

[223] Abir Smiti and Zied Elouedi. Dynamic maintenance case base using knowledge discovery techniques for case based reasoning systems. *Theoretical Computer Science*, pages 1–9, 2019.

[224] Daniel Sonntag and Malte Kiesel. Linked data integration for semantic dialogue and backend access. In *2010 AAAI Spring Symposium Series*, 2010.

[225] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in case-based reasoning–perspectives and goals. *Artificial Intelligence Review*, 24:109–143, 2005.

[226] Luca Spalzzi. A survey on case-based planning. *Artificial Intelligence Review*, 16:3–36, 2001.

[227] Stefan Stieglitz, Milad Mirbabaie, Björn Ross, and Christoph Neuberger. Social media analytics–challenges in topic discovery, data collection, and data preparation. *International journal of information management*, 39:156–168, 2018.

[228] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[229] Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*, 2019.

[230] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055, 2015.

[231] AE Sarabadani Tafreshi, A Sarabadani Tafreshi, and Anca L Ralescu. Ranking based on collaborative feature weighting applied to the recommendation of research papers. *en), International Journal of Artificial Intelligence & Applications*, 9:53, 2018.

[232] Samir Tartir, I Budak Arpinar, and Bobby McKnight. Semanticqa: exploiting semantic associations for cross-document question answering. In *International Symposium on Innovations in Information and Communications Technology*, pages 1–6. IEEE, 2011.

[233] Junichi Tatemura, Songting Chen, Fenglin Liao, Oliver Po, K Selcuk Candan, and Divyakant Agrawal. Uqbe: uncertain query by example for web service mashup. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1275–1280, 2008.

[234] Trifon Totlis, Konstantinos Natsis, Dimitrios Filos, Vasilios Ediaroglou, Nikolaos Mantzou, Fabrice Duparc, and Maria Piagkou. The potential role of chatgpt and artificial intelligence in anatomy education: a conversation with chatgpt. *Surgical and Radiologic Anatomy*, 45(10):1321–1329, 2023.

[235] Tuan Nhat Tran, Duy Khanh Truong, Hanh Huu Hoang, and Thanh Manh Le. Linked data mashups: A review on technologies, applications and challenges. In *Asian Conference on Intelligent Information and Database Systems*, pages 253–262. Springer, 2014.

[236] Christoph Treude, Carlos A Prolo, and Fernando Figueira Filho. Challenges in analyzing software documentation in portuguese. In *2015 29th Brazilian Symposium on Software Engineering*, pages 179–184. IEEE, 2015.

[237] William Tunstall-Pedoe. True knowledge: Open-domain question answering using structured knowledge and inference. *AI Magazine*, 31(3):80–92, 2010.

[238] Jürgen Umbrich, Aidan Hogan, Axel Polleres, and Stefan Decker. Link traversal querying for a diverse web of data. *Semantic Web*, 6(6):585–624, 2015.

[239] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648, 2012.

[240] Maria Vargas-Vera, Enrico Motta, and John Domingue. Aqua: An ontology-driven question answering system. *New Directions in Question Answering*, 8, 2003.

[241] Panos Vassiliadis, Patrick Marcel, and Stefano Rizzi. Beyond roll-up's and drill-down's: An intentional analytics model to reinvent olap. *Information Systems*, 85:68–91, 2019.

[242] Marcio Victorino, Maristela Terto de Holanda, Edison Ishikawa, Edgard Costa Oliveira, and Sammohan Chhetri. Transforming open data to linked open data using ontologies for information organization in big data environments of the brazilian government: the brazilian database government open linked data–dbgoldbr. *KO KNOWLEDGE ORGANIZATION*, 45(6):443–466, 2018.

[243] Katia Vila and Antonio Ferrández. Model-driven restricted-domain adaptation of question answering systems for business intelligence. In *Proceedings of the 2nd International Workshop on Business Intelligence and the WEB*, pages 36–43, 2011.

[244] Quoc Duy Vo, Jaya Thomas, Shinyoung Cho, Pradipta De, and Bong Jun Choi. Next generation business intelligence and analytics. In *Proceedings of the 2nd International Conference on Business and Information Management*, pages 163–168, 2018.

[245] Richard J Waldinger, Douglas E Appelt, Jennifer L Dungan, John Fry, Jerry R Hobbs, David J Israel, Peter Jarvis, David L Martin, Susanne Riehemann, Mark E Stickel, et al. Deductive question answering from multiple resources. *New Directions in Question Answering*, 2004:253–262, 2004.

[246] Yao Wan, Guandong Xu, Liang Chen, Zhou Zhao, and Jian Wu. Exploiting cross-source knowledge for warming up community question answering services. *Neurocomputing*, 320:25–34, 2018.

[247] Guiling Wang, Jun Fang, and Yanbo Han. Interactive recommendation of composition operators for situational data integration. In *2013 International Conference on Cloud and Service Computing*, pages 120–127. IEEE, 2013.

[248] Guiling Wang, Yanbo Han, Zhongmei Zhang, and Shouli Zhang. A dataflow-pattern-based recommendation framework for data service mashup. *IEEE Transactions on Services Computing*, 8(6):889–902, 2015.

[249] Jiangyao Wang, Wenhua Xu, Wenhao Yan, and Caixia Li. Text similarity calculation method based on hybrid model of lda and tf-idf. In *Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence*, pages 1–8, 2019.

[250] Tian Wang, Jiakun Li, Zhaoning Kong, Xin Liu, Hichem Snoussi, and Hongqiang Lv. Digital twin improved via visual question answering for vision-language interactive mode in human–machine collaboration. *Journal of Manufacturing Systems*, 58:261–269, 2021.

[251] Xuan Wang, Bofeng Zhang, Mingqing Huang, Furong Chang, and Zhuocheng Zhou. Improved lda model for credibility evaluation of online product reviews. *IEICE TRANSACTIONS on Information and Systems*, 102(11):2148–2158, 2019.

[252] Xueming Wang, Zechao Li, and Jinhui Tang. Multimedia news qa: Extraction and visualization integration with multiple-source information. *Image and Vision Computing*, 60:162–170, 2017.

[253] Ian Watson. Case-based reasoning is a methodology not a technology. *Knowledge-based systems*, 12(5):303–308, 1999.

[254] Matthias Wendt, Martin Gerlach, and Holger Düwiger. Linguistic modeling of linked open data for question answering. In *Extended Semantic Web Conference*, pages 102–116. Springer, 2012.

[255] Rainer Winkler and Matthias Söllner. Unleashing the potential of chatbots in education: A state-of-the-art analysis. In *Academy of Management Proceedings*, page 15903. Academy of Management Briarcliff Manor, NY 10510, 2018.

[256] Hans Friedrich Witschel, Kaspar Riesen, and Loris Grether. Kvgr: A graph-based interface for explorative sequential question answering on heterogeneous information sources. In *European Conference on Information Retrieval*, pages 760–773. Springer, 2020.

[257] William A Woods. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450, 1973.

[258] Qi Wu, Peng Wang, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4622–4630, 2016.

[259] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, 2023.

[260] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381, 2022.

[261] Yirui Wu, Yuntao Ma, and Shaohua Wan. Multi-scale relation reasoning for multi-modal visual question answering. *Signal Processing: Image Communication*, 96:116319, 2021.

[262] Pengfei Xu, Jiaheng Lu, et al. Towards a unified framework for string similarity joins. *Proceedings of the VLDB Endowment*, 2019.

[263] Chaowei Yang, Qunying Huang, Zhenlong Li, Kai Liu, and Fei Hu. Big data and cloud computing: innovation opportunities and challenges. *International Journal of Digital Earth*, 10(1):13–53, 2017.

[264] Hui Yang and Tat-Seng Chua. The integration of lexical knowledge and external resources for question answering. In *In the Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, 2002.

[265] Hui Yang and Tat-Seng Chua. Web-based list question answering. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1277–1283, 2004.

[266] Zi Yang, Ying Li, James Cai, and Eric Nyberg. Quads: question answering for decision support. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 375–384, 2014.

[267] Hong Yu, Minsuk Lee, David Kaufman, John Ely, Jerome A Osheroff, George Hripcsak, and James Cimino. Development, implementation, and a cognitive evaluation of a definitional question answering system for physicians. *Journal of biomedical informatics*, 40(3):236–251, 2007.

[268] Munazza Zaib, Wei Emma Zhang, Quan Z Sheng, Adnan Mahmood, and Yang Zhang. Conversational question answering: A survey. *Knowledge and Information Systems*, 64(12):3151–3195, 2022.

[269] Naiheng Zhang. Service discovery and selection based on dynamic qos in the internet of things. *Complexity*, 2021:1–12, 2021.

[270] Yi Zhang and Zachary G Ives. Finding related tables in data lakes for interactive data science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1951–1966, 2020.

[271] Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. A joint model for question answering over multiple knowledge bases. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[272] Wenfeng Zheng, Lirong Yin, Xiaobing Chen, Zhiyang Ma, Shan Liu, and Bo Yang. Knowledge base graph embedding module design for visual question answering model. *Pattern recognition*, 120:108153, 2021.

[273] Yu Zhou, Xinying Yang, Taolue Chen, Zhiqiu Huang, Xiaoxing Ma, and Harald Gall. Boosting api recommendation with implicit feedback. *IEEE Transactions on Software Engineering*, 48(6):2157–2172, 2021.

[274] Christian Zirpins. Situational data-analytics for the web-of-things. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, pages 1–4, 2016.

APPENDIX A – SITUATIONAL DATA INTEGRATION IN QA SYSTEMS

In this Appendix a Systematic Literature Review is presented, regarding Situational Data Integration in Question Answering Systems. The review was based on the features shown in Table 2.1.

A.1 AD-HOC DATA RETRIEVAL IN QA

Ad-hoc Data Retrieval refers to finding and accessing situational data, by discovering sources that are suitable for an information need. Its subfeatures, Ad-hoc Questions and Source Discovery, were evaluated in the surveyed QA studies as demonstrated in Table A.1. An overview of results is shown in Figure A.1.

Table A.1: SDI's Ad-hoc Data Retrieval Requirements

Ad-hoc Data Retrieval Features	**Requirement for "Complete Feature"**	**Requirement for "Partial Feature"**
Ad-hoc Questions	Situational data integration requirements are often immediate and cannot be totally defined in advance [247, 7]. We consider this feature *fully* supported if the questions or queries presented in the approach are not predefined nor recovered from databases or benchmarks.	We consider this feature *partially* supported if the question is built with the help of other techniques, e.g., auto-complete tools or parameters selection.
Source Discovery	We consider this feature *fully* supported if (i) the situational data source used in the approach is *not predefined*, but chosen at query time[1] [154, 247, 123, 99, 274] `AND` (ii) the approach includes some change in the data source(s) in order to meet query requirements [154, 99, 247, 274, 123].	The feature is *partially* considered when *predefined* sources are changed or adapted to meet query requirements.

A.1.1 Ad-hoc questions

SDI is usually motivated by ad-hoc questions, which are defined at a moment of need. They closely relate to QA area, where users interact with a system looking for precise and fast answers. Everyday questions do not come from any repository, and usually do not follow query formats unfamiliar to the user, such as SQL or SPARQL. Users questions in QA are often informed in natural language, and mainly, they are informed *at the very moment of an information need*. Situational Data Integration aims to deal with this kind of questions (i.e., the ad-hoc questions), providing support by discovering data sources that fit in the question requirements.

The study in [50] answers ad-hoc queries posed in a global schema by exploring heterogeneous definitions of indicators formulas. Ad-hoc NL questions are answered in [189] by means of a multi-source QA system. NL questions are also present in [93], which proposes a QA system and an integration method based on ontologies, and in [42], through an interactive NL Interface for querying ontologies. [27] and [183] present MiPACQ (Multi-Source Integrated

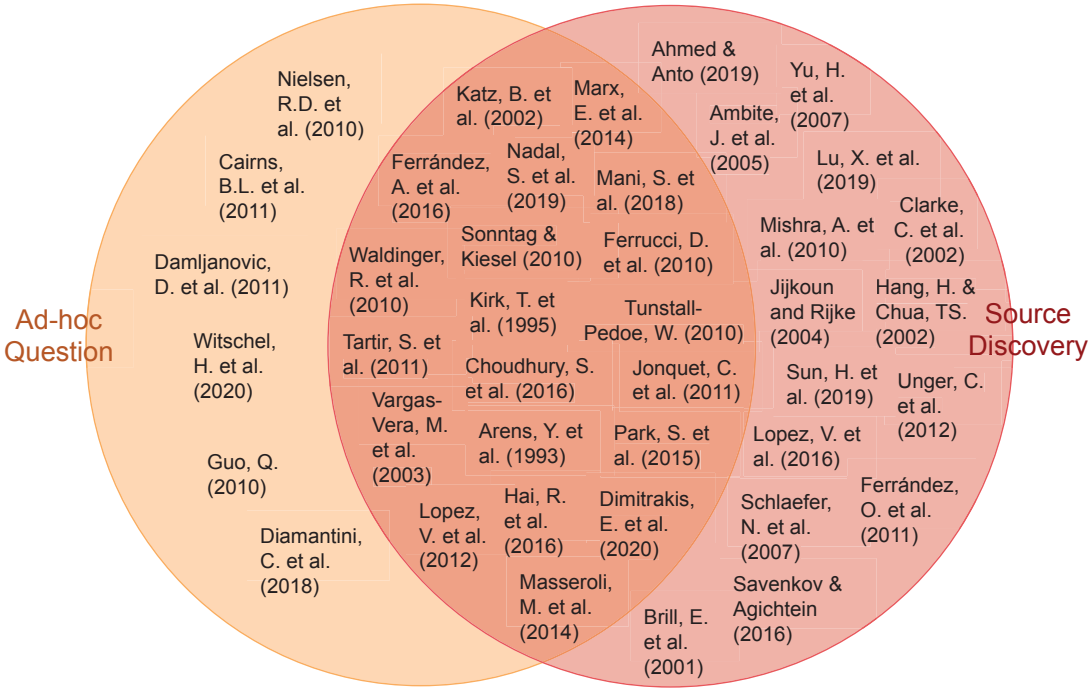


Figure A.1: Ad-hoc Data Retrieval in Question Answering systems.

Platform for Answering Clinical Questions), an integrated framework for semantic-based QA that accepts free-text clinical questions.

Ad-hoc questions may also be covered in a *partial* way if they are not asked freely, but using selection and completion methods. E.g., the authors in [256] present a novel approach for context-aware Sequential Question Answering, based on a knowledge graph (KG) that integrates information from various sources. In their experiments, each participant receives a first predefined query, and all subsequent ones require participants to select a subset of the KG nodes that were displayed or to ask questions in natural language. Similarly, in [163] the query is built by parameters selection in an interface. In [232] and [36], there are selection-based and auto-complete approaches for query formulation, respectively.

A.1.2 Source Discovery

Besides ad-hoc questions, Ad-hoc Data Retrieval in SDI includes ***Source Discovery***. Many QA systems have to deal with complex questions that can only be answered using information from more than one data source. In some cases, real time information is *decisive* for the users tasks, but cannot be provided by stationary or predefined sources. SDI's Source Discovery could fill this gap, enabling to find the right sources that contain the information.

The study in [155] presents QUEST (QUEstion answering with Steiner Trees), a method for answering complex questions from textual sources on-the-fly, by joining evidence from different documents. QUEST constructs an ad-hoc, noisy knowledge graph by dynamically retrieving many question-relevant text documents from the Web. Non-predefined and dynamic pages from the Web are also accessed in the MedQA system proposed in [267], in the hybrid QA system described in [8], and in the Quartz system [119].

The QA system proposed in [264] also presents Source Discovery as it accesses non-predefined and dynamic websites that, in combination with WordNet [169], are able to find words lexically related to the original query terms. The same occurs in [24], which proposes a system that utilizes Google search engine to find answers on the Web. A set of potential answers is

extracted from the summary text, then given a set of possible answers, an answer projection step is performed for searching supporting documents in the TREC QA document collection[2].

The authors in [239] present a novel approach that relies on a parse of the question to produce a SPARQL template representing the internal structure of the question. The query templates contain missing elements that have to be filled with URIs. For this task, a extraction is performed through a framework that uses the Linked Data Web as background knowledge and assumes a set of predicates for which equivalent NL expressions are to be detected from an arbitrary corpus (e.g., Wikipedia or the whole Web), so that labels of instances linked by these predicates determine the seed pages to be queried.

In [72] the authors present the QALL-ME framework, a reusable architecture for multi-lingual QA systems working on structured data sources. The architecture is based on Service Oriented Architecture (SOA), which uses web services for the framework components. The data sources used to retrieve the answers are non-predefined databases or XML documents, which are specified by the domain ontology. In addition, the web services accessed by the SOA-based architecture are interchangeable.

In [207], Text2KB is introduced, a system that enriches QA over a knowledge base using external text data. These data come from three sources: the whole Web, the Yahoo! Webscope L6 dataset[3] containing CQA (Community Question Answering) data, and text fragments with entity mentions. Similarly, [267, 8] and [265] also perform searches on the whole Web, by retrieving web pages according to information specified in the query.

Some studies *partially* handle Source Discovery, i.e., their data sources are dynamic, even though predefined. The authors in [153] use several predefined Linked Open Data sources including Wordnet and DBpedia[4], and these sources are dynamic in handling query requirements. The authors in [229] describe PullNet, an integrated neural framework that reasons with heterogeneous information to find the best answer for an input question. PullNet constructs a question-specific subgraph containing sentences from a corpus, and facts from a KB. Both data sources are predefined but dynamically changed: the model might use the KB when the required fact exists, and "back off" to text when the KB is missing information.

The QA system in [40] presents an "early-answering" strategy, in which a question can be answered directly from a structured collection (a database containing knowledge gathered from the Web), then a secondary corpus is used to support the answer found. In addition, if the question cannot be answered with the structured collection, publicly corpora are accessed for retrieving relevant passages. The sources are predefined, but not all of them are used for passage retrieval: passages are retrieved from at least three corpora, considering that the results in the early-answering process were insufficient. Therefore, there is source changing during the process.

Mishra A. et al. [171] present a context-aware geographical QA system that finds out the semantic relations between the question and candidate answers. Web information about various cities are collected and used to built an offline knowledge base used in the system. Although there is no discovery, source changing can be easily performed manually. The Ephyra QA system [208, 209] combines several techniques for question analysis and answer extraction, and incorporates multiple knowledge bases. The system uses predefined sources and a API for searching in dynamic sources. Similarly, the authors in [9] present a query answering system that integrates multiple knowledge sources of specific domain. The sources are predefined, including

[2]The Text REtrieval Conference (TREC) is a well-known reference for QA approaches due to Question-Answering tracks started in 1999.

[3]Available at: <https://webscope.sandbox.yahoo.com/>.

[4]Structured database of information extracted from Wikipedia. Available at <https://wiki.dbpedia.org/>.

a Mediator reasoner that extracts useful information from about two dozens of websites. The access to all knowledge sources in the proposal is performed by a module called Query Manager, which determines which knowledge source are required to produce the answers. The Query Manager also considers each knowledge source as a reasoner, so that it can be recalled to provide additional answers when needed.

A.1.3 Ad-hoc questions & Source Discovery

As we can see in Figure A.1, most of the QA-based approaches that cover Ad-hoc Questions also deal with Source Discovery. The authors in [127] present a natural language QA system that understands users' ad-hoc queries and translates them into structured queries. Also, it provides an interface to multiple non-predefined knowledge sources from the Web. Similarly, the study in [36] ingests information from web crawls and articles to continuously stay abreast of information around drones. Web sources are also external, non-predefined, and dynamic in [232], [73], and [240]. These studies also allow the user to query data sources by means of natural language questions.

The study in [53] introduces LODQA, an approach that exploits simultaneously hundreds of linked datasets by means of a suite of services. The datasets are chosen at query time, according to their similarity with the question, and they are dynamic with respect to query requirements. Although LODQA was evaluated through a collection of predefined questions, the approach allows the user to type a query in natural language or RDF format. The QA system in [158] uses a novel hybrid answering model for providing IT support. The user can interact with a platform by asking a question or uploading a screenshot of the error he is facing. Predefined and heterogeneous data sources are used for answering the questions, with an *orchestrator* module that decides which source to call under what circumstances, in order to maximize the likelihood of getting a correct answer.

Graphical interfaces for users' ad-hoc queries are also present in [12] and [120]. In the former, the SIMS (Services and Information Management for decision Systems) is proposed: the system dynamically retrieves and integrates information from external and predefined databases, which can also be changed through the interface. In the latter, an ontology-based resource index called NCBO (National Center for Biomedical Ontology) provides unified access to more than twenty heterogeneous biomedical resources. Both ontologies and resources are changing often, so NCBO tables are regularly and automatically updated. Based on the user input, the index uses an auto-complete mechanism for suggesting terms and resources. *True Knowledge* [237] is another QA platform with an interface for submitting ad-hoc queries. For the answering process, the proposal combines a structured knowledge base, a NL translation system, and an inference system. The user may also provide additional knowledge sources, and a *System Assessment* module can switch off the conflicting ones.

Ad-hoc Data Retrieval in SDI is well represented in [70]: the authors argue that BI applications should consider external data sources for achieving crucial information that help taking the right decisions, thus they integrate the DW internal structured data with external unstructured data obtained with QA techniques to answer ad-hoc questions. A question is posed through a GUI element by the user, who also identifies the sources where to search the required information, and the sources are accessed by the QA system dynamically.

In [160], the authors present *openQA*, a modular and extensible open-source QA framework that integrates other QA systems [219, 239] for accessing DBpedia endpoint. The proposal covers an answer formulation process that receives an ad-hoc query, which can be interpreted in different formats such as SPARQL and SQL. New modules can be integrated within *openQA* via a plug-in architecture, and the user can enable and disable instances used for

searching. The study in [224] presents a dialogue-based QA system that integrates several linked data sources at SPARQL endpoints. The user's ad-hoc question is analyzed and mapped to one or more suitable services found, which may answer parts of the query. With the information found in the result, additional services can be triggered for finding more information about the first result.

The work in [177] presents a Big Data Integration ontology and a query answering algorithm that converts ad-hoc queries posed over the ontology to queries over multiple sources. The approach builds upon two RDF named graphs (Global and Source graphs) and wrappers that accommodate different kinds of no-predefined data sources. The approach also deals with changes in the source schema via semi-automated transformations on the ontology. Ontologies are also present in [152] with *PowerAqua*, an ontology-based system that explores multiple and heterogeneous sources on the web. PowerAqua accesses the Semantic Web through the Watson SW Gateway, thus it only retrieves information if this has been crawled and indexed by Watson or in specified online repositories. PowerAqua provides plugins for accessing the repositories, which are loaded on demand. The system accepts users' queries expressed in NL and retrieves precise answers by dynamically selecting and combining information from the resources.

The studies in [129, 141] describe *Information Manifold* (IM), a system for browsing and querying multiple web sources. The system determines exactly the set of information sources relevant to a query, adding their descriptions to the knowledge base by means of a representation language, allowing multiple sources to be accessed. Users can formulate queries either using templates that are available for the information categories, or by combining such templates into an arbitrary conjunctive query. The Data Lake proposed in [96], named *Constance*, manages QA based on metadata. In an unified interface, the users can define their queries by using a formal structured query language or simple keywords. Constance has an Ingestion Layer, which is responsible for importing data from heterogeneous sources into the DL system, and generic extractors adapt to data source formats. Through the interface, users can import local files, databases or remote data via web services, and new extractors can be easily added using a plug-in mechanism.

The QA system proposed by [189] merges a KB-based QA (KBQA), a IR-based QA (IRQA), and a keyword QA in its architecture. Multiple information sources are used, including curated KB, raw text, and auto-generated triples. The sources are predefined but they are alternated according to the input question, e.g., when the questions is a sentence, it is sent to the sources accessed by the KBQA/IRQA system; otherwise, it is sent to the keyword QA system. Multiple information sources are also observed in [245]: the authors develop a prototype called QUARK (Question Answering through Reasoning and Knowledge), which uses several knowledge resources. Ad-hoc NL questions are submitted to the automated deduction system SNARK, a general-purpose theorem equipped with an application-domain theory, which invokes the sources when appropriate. When a new resource is introduced to the system, it is provided one or more axioms in the theory that expresses what the resource can do.

Finally, the study in [163] presents an approach to support integrated search of distributed biomedical-molecular data, aimed at answering multi-topic complex biomedical questions. A Bioinformatics Search Computing application (Bio-SeCo) is modelled and published, in which bioinformatics services are registered. Bio-SeCo provides a platform which allows the user to express requests over the multiple services registered and find answers to his/her questions. Although the sources are predefined, the user can easily inspect the sources and obtained results, select the most appropriate, expand or refine them.

Retrieved data loses its value if not refined and integrated with current data to form a complete information. Thus, SDI's *Data Management* feature is addressed as follows.

A.2 DATA MANAGEMENT IN QA

Data Management in SDI includes the system's ability to deal with heterogeneous and noisy information from situational data source, and combine this source with the stationary data, in order to support decision-making. These abilities refer to two features, named *Unstructured Data Preprocessing* and *Situational Source Inclusion* requirements are detailed in Table A.2. The feature results in QA are shown in Figure A.2.

Table A.2: SDI's Data Management Requirements

Data Management Features	**Requirement for "Complete Feature"**	**Requirement for "Partial Feature"**
Unstructured Data Preprocessing	We consider *full* coverage in approaches that apply preprocessing techniques in data sources, such as cleaning, normalization, noise removing, entity/link resolution, among others. We did not consider this feature when it occurs in the input question only.	Not applicable.
Situational Source Inclusion	Situational Source Inclusion is *fully* supported by papers that add a data source which (i) covers a targeted/specific information need AND (ii) is integrated with current data in order to provide complete insights or solutions[5] [3, 154, 7, 244, 274].	Situational Source Inclusion is *partially* considered when there is a situational source, but it does not complement the available data.

A.2.1 Unstructured Data Processing

The QA of next generation will have to take into consideration presently heterogeneous and unstructured data [94]. Although data preprocessing is already a common task in QA systems [197], handling unstructured data, in particular, is essential for successfully integrating discovered data sources. Consequently, a well-performed integration may determine the value of the QA system to the user.

With respect to Data Preprocessing, different techniques are used by QA-based systems. The novel framework named HSIN (Heterogeneous Social Influential Network) proposed in [33] integrates and simultaneously learns the questions textual contents, their related categories information and user's social interaction. The experiments were based on a large dataset from Quora service and Twitter social network, in which duplicated data were removed before composing training and testing sets. Duplicates and ambiguous data are also handled in [271], which presents a model based on Integer Linear Programming (ILP) for exploring multiple KBs in QA. The approach unites the construction of alignments among KBs, and query construction for translating a NL query into a SPARQL query. These tasks can involve ambiguous data, so the approach combines potential alignments to obtain a *Disambiguation Graph*.

Many other studies also perform disambiguation and summarization tasks for preprocessing data: The IR-based system in [189] uses a multi-source tagged text database that includes disambiguation results, types of named entities and NLP results. The system in [77] is embedded in a hybrid QA system architecture called *QUETAL*, which selects one or more information data sources to retrieve answer candidates and prepare a final answer. These processes include disambiguation of concepts and database tables, based on recognized entities. In the

Ephyra system [208, 209], answer candidates (extracted from external sources) that contain frequent keywords are favored, and additional filtering techniques are used to drop redundant and non-informative answers. Similarly, the QA model in [88] prunes candidate answers that are incompatible with the question, using a convergence mechanism and an arc-search to reach the most relevant answers. Irrelevant sources are also pruned in the Information Manifold system [129, 141], which helps to solve completeness and redundancy issues.

The Constance system [96] discovers, extracts and summarizes structural metadata from the data sources, also annotating data with semantic information to resolve ambiguities. The MedQA approach [267] includes a Text Summarization step that removes redundant sentences by clustering similarities into the same group. In the OpenQA system [160], answers may be extracted from different sources, so it can be ambiguous and redundant. To alleviate this issue, results that appear multiple times are fused by means of clustering and ranking. The FREyA system [42] also covers a desambiguation step, and in case of the system failing to automatically interpret the question, the disambiguation may occur manually, i.e., with the user interacting with a dialog box.

The study in [246] focuses on capturing the full views of user topical expertise in CQA services, by considering information from other social media websites in which they participate, e.g. the GitHub. For data preprocessing, the text from social media is tokenized, and all code snippets, stopwords and HTML tags are discarded. In fact, tokenization and stopwords removal are very frequent preprocessing methods in QA. E.g., in [27, 183], both the question and candidate answer are processed through tokenization, stemming, part of speech detection, named entity recognition and dependency parsing. In [252], the authors present a QA approach that allows precise browsing from web news data by using text and multimedia information simultaneously. The crawled news documents were parsed by using a language processing toolkit, and filtered with a stopword list.

In [24], stopwords are not permitted to appear in any potential n-gram answers when searching for candidate answers. Also, an answer tiling algorithm is applied for solving overlap in shorter n-grams. For example, “A B C” and “B C D” are tiled to “A B C D”. Answer tiling is also used in the Quartz system [119]. The system analyzes an incoming question, sends it to six streams in parallel, and each stream produces a ranked list of relevant answer candidates. The answer tiling is a subsequent process, where similar answer candidates from the six streams are identified and merged to obtain the highest confidence answer.

Noise removal is another technique that frequently appears in QA proposals. The PowerAqua’s architecture [152], e.g., deals with noisy and incomplete data throughout the retrieval process. Noise removal from audio files occurs in [158]. The clinical QA system *AskHERMES* [28] handles complex questions through the use of a structured domain-specific ontology that integrates five types of external sources. The system removes noise from texts, and merges relevant passages as parts of a potential answer, so it can retain semantic content of the data sources. In [171], the extracted documents are preprocessed through noise removal, tokenization, sentence splitting and tagging. The system architecture proposed in [239] includes a cleaner module for removing noise from crawled text. In this study, an indexer is also present.

Indexes are frequent in the analyzed proposals. The study in [38] presents PIQUANT, a modular architecture that allows for multiple answering agents to address the same question in parallel. Among the answering agents in PIQUANT, a knowledge-based agent performs predictive annotation for indexing the information with semantic classes. The QA system in [158] comprises many preprocessing techniques by running the *Ingestor Component*, responsible for converting raw data into structured knowledge. In the document ingestion phase, structural heuristics and Latent Semantic Indexing [112] are used to induce formatting into the documents.

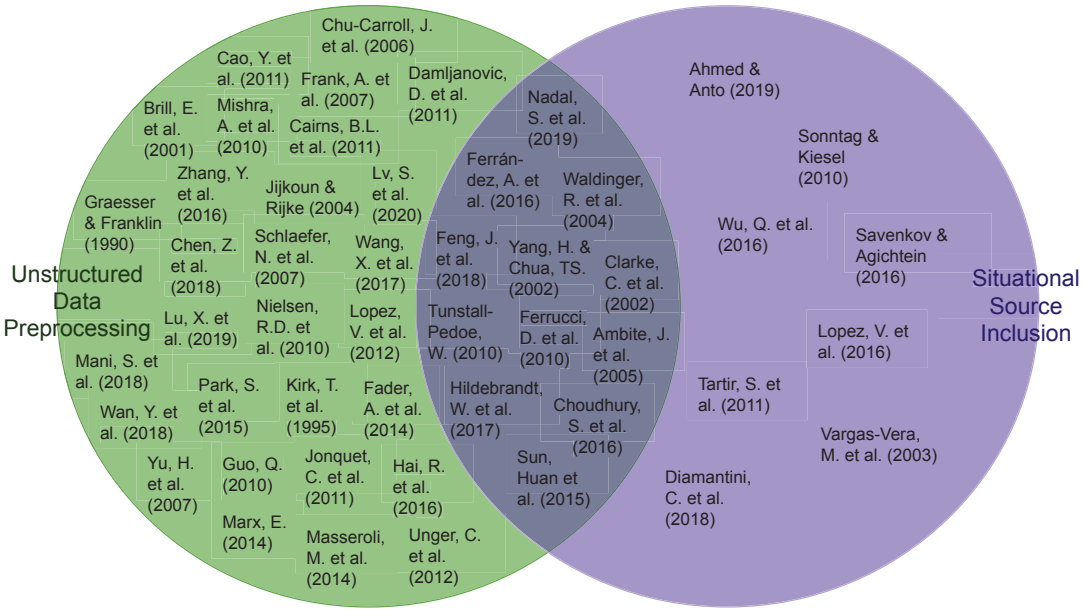


Figure A.2: Data Management in Question Answering systems.

In the NCBO Resource Index [120], after retrieving the resource elements, there are steps of concept recognition, resolution of references, data annotations, as well as semantic expansion for making the data elements more informative.

In the QA system proposed in [93], the Pervasive Agent Ontology is constructed by means of concept extraction methods and analysis methods, used for discovering internal relations between concepts and making the ontology more complete and consistent. The authors in [156] propose to extract evidence from heterogeneous knowledge sources, and use it for question answering. Preprocessing was performed on the information from the two sources, by means of identification of entities, Semantic Role Labeling (SRL) and stopwords removal.

In the Bio-Seco application proposed by [163], attributes of the connected services were normalized in order to join their values semantically. In the QUEST system presented in [155], a knowledge graph was constructed by retrieving many question-relevant text documents from the Web. These documents were preprocessed through POS tagging, named entity recognition and lightweight coreference resolution for linking pronouns to entities. The resulting graph is treated as the knowledge source for answering questions. Finally, the authors in [63] present an Open Question Answering (OQA) approach that leverages both curated and extracted knowledge bases. The union of these KBs forms a single resource containing a billion noisy, redundant, and inconsistent assertions. The approach includes lemmatization and stopwords removal before computing cosine similarity.

A.2.2 Situational Source Inclusion

Other feature in SDI's Data Management is ***Situational Source Inclusion***. This step is what makes Source Discovery effective, since it includes situational data into the answering process. The capability of incorporating situational data could make the QA system aware of current situations, thus being able to respond complex and specific questions more precisely.

In this sense, the dialogue-based QA system presented in [224] implements an interface to linked data sources, which relies on mapping between a given data source to corresponding Web resources. The dialogue platform is based on ontology structures for processing dialogue grammars and a external service connector. When a query is informed, the appropriate Web

resources are retrieved dynamically through semantic search. These resources cover specific information needs, which are combined with the stationary data (the ontology) for providing complete answers.

In [232], the questions are answered using semantic knowledge stored in ontology schemas, by means of query triples mapped to ontology elements. The approach covers Situational Source Inclusion since external information are retrieved when there is a need for complementing the current data: if the question cannot be answered solely from the ontology, SemanticQA detects the failing parts and send them to a document web search engine, for extracting answers from snippets of web documents. After the extraction, they are matched against ontology instances.

The hybrid QA system in [8] aims to combine textual and structured knowledge base data for question answering. The system consists of three main modules: a Knowledge base, an online module, and a Text-To-KB transformer. The NL question is taken as input, and the knowledge base (e.g., DBpedia) is used to retrieve the answer. As there is no context information for KB-based QA modules, a Web search is performed in order to fill this gap. In its turn, the online module searches text to find answers, and the candidate answers from the Web are merged with those in the KB. This process is concluded by the Text-To-KB transformer, which detects KB triples in both snippets and documents and then store them in the KB.

The study in [207] also presents a hybrid QA system, named Text2KB, that extends the information in an existing knowledge base. The KB used as baseline corpus is *Aqqu* KBQA system [15], which accesses Freebase data. However, some tasks such as question interpretation, candidate answer generation and candidate ranking are challenging for a KBQA system due to lack of information for solving redundancies or composing a complete answer. Thus, external web-based sources are employed, providing additional edges in the KB and allowing to produce a final answer.

The approach in [50] is based on a query reformulation procedure that correlates and adapts the global query to the schema of the local sources by using the materialized views of the local data mart. The materialized views are accessed by exploiting a state space that integrates a dimensional lattice with a formula graph. The formula graph is an additional source generated automatically, containing specific data that complement the dimensional lattice in order to answer specific queries.

In the QuerioDALI QA system proposed by [153], Linked Open Data and Knowledge Graphs (KGs) are exploited to answer complex NL queries. Given the query, Predicate Argument Structures (PAS) are extracted for finding entities and links that matter. Relevant graphs are selected based on their coverage for a given PAS and the candidate URIs are found based on this correlation. From these candidates, PAS triples are translated into Graph Patterns (GPs), which convey into a formal query executed against the KGs. If the query can only be answered by merging across graphs, the GPs from the same or different sources are merged, i.e., the GP search is expanded only if required and the situational sources are included when the current data are insufficient.

In [258] is proposed a LSTM framework for VQA (Visual Question Answering) that combines an internal representation of an image's content with text-based information extracted from a general KB to answer a broad range of image-based questions. Given a image-question pair, a set of image attributes are predicted and used to extract relevant information from the external KB (DBpedia). Then, the paragraphs extracted from the KB are encoded and used to train an LSTM, by maximizing the probability of the correct answer. DBpedia is therefore used as situational source for providing targeted data which, integrated with current data, is able to answer questions referred to information not contained in the image.

The AQUA system [240] *partially* covers Situational Source Inclusion, i.e., it retrieves external information when is needed, but the retrieved data does not complement the existing Knowledge Base. AQUA's algorithm executes the query against the knowledge base, and if it does no succeed, the query is reformulated and used for launching a search engine that retrieve documents which satisfy the new query. So, these data are situational, since they are retrieved motivated by the insufficiency of the KB.

A.2.3 Unstructured Data Preprocessing & Situational Source Inclusion

A set of QA approaches fully cover Data Management by handling ***Unstructured Data Preprocessing and Situational Source Inclusion*** simultaneously. E.g., the work in [177] applies Local-As-View mappings to characterize elements of the sources schemata in an ontology. The ontology is semantically annotated, enabling to resolve problems of ambiguity. Also, the ontology accommodates situational data and is able to deal with their evolution, by exploiting the impact of feedback and monitored data for improving the user's quality of experience.

The study in [36] presents a framework to build domain specialized knowledge graphs, by fusing curated KBs with extracted knowledge, in a way that the resulting graph is used to answer a set of queries. In the use case, given the existing information about use of drones in the KB (YAGO2 ontology [110]), texts are retrieved from the Web and preprocessed for entity and relation extraction. The extracted triples are mapped to entities present in YAGO2, generating a drone graph that allows to continuously stay abreast of information around drones.

In [9], the knowledge sources are accessed by a Query Manager which is based on a shared ontology called CALO. The approach also has a Mediator reasoner, responsible for extracting information from the Web, so that extracted data are mapped into CALO ontology by means of translation axioms. In addition to this correlation, the Mediator provides suitable semantics to the data returned by Web extraction. The Situational Source Inclusion is motivated by insufficiency of current data: first, the answer is searched locally, and when the answering requires facts residing in other sources, the system breaks down for searching partial proofs in the the appropriate sources.

The authors in [107] present an approach to answering definition questions, in which the goal is to return as many relevant "nuggets" of information about a target concept as possible, by using a relational database and a Web dictionary as corpus. The retrieved documents are tokenized into individual sentences, discarding candidate sentences that do not contain the target term. If no answers are found by using database and dictionary lookup, the system employs traditional document retrieval to extract relevant nuggets, and then the results from all sources are integrated to produce a final answer.

In [230] is presented QuASE (QUestion Answering via Semantic Enrichment), a QA system that mines answers directly from the Web, and meanwhile employs Freebase as a significant auxiliary to further boost the QA performance. Answer candidates are detected from sentences extracted from the Web and linked to entities in Freebase, by means of entity linking tools. By linking answer candidates to the KB, similar answer candidates can be automatically merged, significantly reducing redundancy and noise. Freebase is used as situational source when integrated into a ML model, since it provides the information required for completely answering questions.

One problem faced in the QUARK system proposed by [245] is the lack of uniform conventions in notation by the knowledge resources. For example, for resources that deal with latitudes and longitudes, some may adopt a decimal notation, while others employ degrees, minutes, and seconds. Due to this, QUARK includes important agents that preprocess notations, converting one to another. Each knowledge resource acts as a situational source, invoked when is

needed, and the axiomatic theory is responsible for their correlation and integration. In True Knowledge [237], knowledge extraction includes a four-stage process of sentence extraction, simplification, translation, and bootstrapping that allows to extract high-quality facts for the knowledge base. The proposal has a general inference system that dynamically generates fact as needed. These facts can be seen as situational data, since they are generated based on a need, and complement the current knowledge for presenting the user with a concise explanation.

As previously mentioned, the QA system in [40] uses an early-answering strategy for answering a question from structured data and justifying the answer with a secondary corpus. However, if no acceptable justification can be found, or if the question cannot be answered with the structured collection, situational sources are invoked through a basic question strategy, and merged in order to produce a final result. Moreover, the approach includes an Entity Extractor component to eliminate unacceptable or unlike answer candidates (i.e., n-grams) from a retrieved passage, which includes, e.g., removing stopwords and n-grams that only appear in a single passage.

The study in [73] describes the implementation of *Watson*, the IBM's well-known QA system based on *DeepQA*, an architecture that performs at human expert levels of precision, confidence and speed. The sources for Watson include a wide range of encyclopedias, dictionaries, articles, literary works, including sources of specific information. Given a baseline corpus, DeepQA identifies related Web documents, extracts text nuggets from them, score the nuggets by relevance and merge the most informative ones into the expanded corpus. Preprocessing techniques are used to get more detailed analysis of the search results, such as named entity detection and soft filtering (to prune candidate answers).

The proposal in [70] integrates an internal DW with an external and unstructured data source. The external data are obtained through QA techniques, thus given a question, it is sent to a set of specialized nodes (DW and QA nodes, composed by ontologies) that process it. An Information Retrieval tool in the QA node retrieves the set of documents that is more likely to contain the answer and, once the running of each specialized node is finished, a semi-automatic mapping process is carried out for detecting connections between the QA and DW ontologies. Preprocessing is performed through a normalization process for obtaining the lemma of ontologies' classes and properties.

The following studies *fully* cover Preprocessing and *partially* cover Situational Source Inclusion. The medical QA system (MQAS) proposed in [66] establishes a mapping process between the question and answers on the basis of the datasets. Situational and specific-domain sources are employed for expanding the terms in the question, allowing to use these terms for discovering relationships in the knowledge sources. These sources, although situational, are not used for complementing the existing datasets. In a later preprocessing step, all textual descriptions of the data fields from the datasets are normalized and combined.

Similarly, the proposal in [264] investigates the integration of lexical and external knowledge (Wordnet and Web, respectively) to bridge the gap between query space and document space in QA. The Web can only provide words that occur frequently with the original query terms, but it lacks information on lexical relationships between these terms. To overcome this need, Wordnet is used as situational source to expand the query that are used for searching answer candidates. NL analysis is performed on the candidate answers to extract POS, base noun phrases, and named entities, thus minimizing the noise introduced by the external resources.

Discovering and including a situational source in an approach usually has a definite motivation: supporting human actions and decisions by offering a complete information. In this context, the next subsection presents the Timely Decision Support and discuss how QA systems handle this feature.

A.3 TIMELY DECISION SUPPORT IN QA

The last SDI feature (see Table 2.1) is **Timely Decision Support**. As stated by [29], if users were aware of events that impact their operations and relationships that are affected by such events, they would have the opportunity to take immediate action. Thus, a situational integration system must provide means to assist users in their decision processes. This involves three subfeatures, which we call *User Guidance, Decision-making Support*, and *Response Time Improvement*. Their requirements are detailed in Table A.3, and results are shown in Figure A.3.

Table A.3: SDI's Timely Decision Support Requirements

Timely Decision Support Features	**Requirement for "Complete Feature"**	**Requirement for "Partial Feature"**
Decision-making Support	Approaches present *full* coverage of this feature if (i) they provide valuable responses (such as alerts, recommendations, reports or predictions) that (ii) provide Situation Awareness to the user AND support his actions or decisions [19, 178, 56].	This feature is *partially* supported in approaches that do not provide explicit action support, but do provide opportune Situation Awareness to the user [178, 29, 244].
User Guidance	We have considered User Guidance in QA systems that presented user participation both in the system operation OR evaluation, i.e., by deciding which information are relevant or giving feedback on the responses returned [99, 3].	Not applicable.
Response Time Improvement	We consider this feature *fully* supported in studies that present any technique of time optimization for retrieving answers, e.g., search indexing or parallel processing.	Not applicable.

A.3.1 Decision-Making Support

Question answering and decision support systems have been independently developed for decades. However, the QA scenario is a friendly environment for decision processes to occur: with the development of high-performance QA systems, which combine natural language processing and information retrieval, users can directly interact with an information system to evaluate evidence gathered automatically [266]. With SDI, this evidence may contain situational data that are at the basis of decision-making, and most importantly, can be used to recommend an action to the user or trigger an alert about the situation identified.

Analyzing studies that only covered ***Decision-Making Support***, the proposal in [50] presents a query answering mechanism for answering ad-hoc queries, and by exploring heterogeneous definitions of indicators formulas, it supports evaluation of cross-organizations performances and produces meaningful comparisons. In [246], the topical interest and expertise from the cross CQA sources are integrated with the Bayesian model in an unified probabilistic, called MultiTEM. MultiTEM is applied to a specific task of expert user recommendation for a given a question, since an expert user tends to provide a good answer.

The algorithm of the medical QA system proposed in [66] creates answers that bridges the question and the corresponding datasets. Optimal answers for decision support are returned

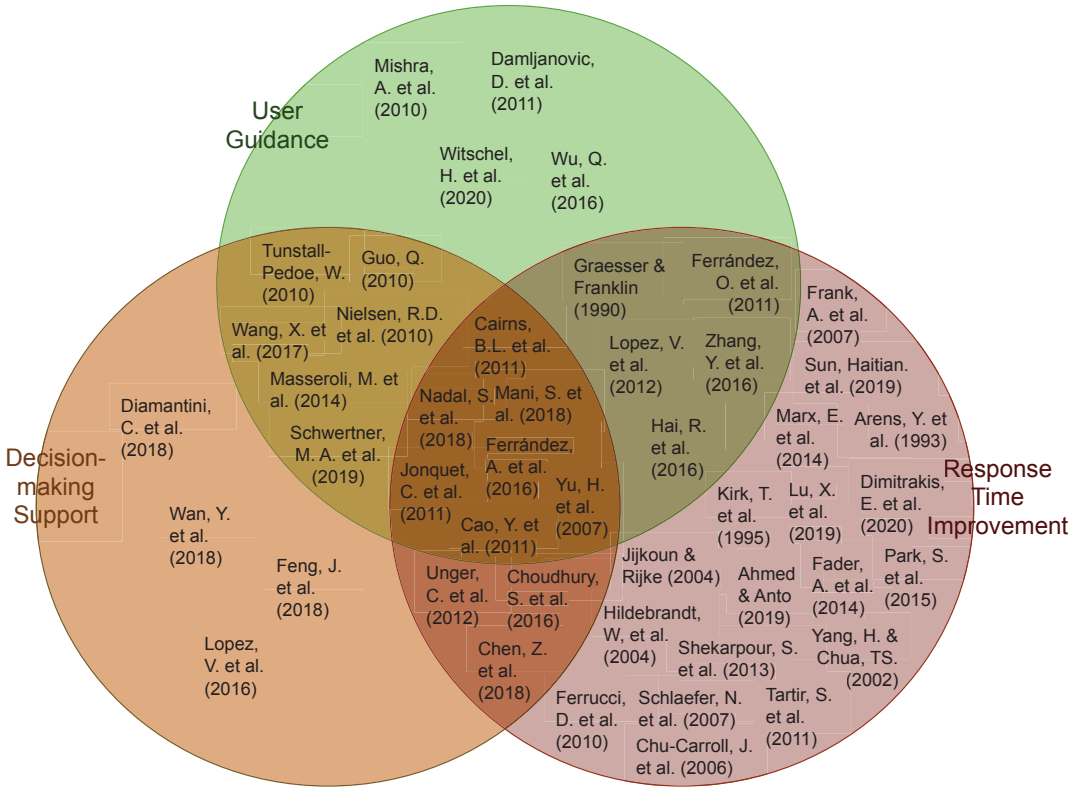


Figure A.3: Timely Decision Support in Question Answering systems.

to users, including the question, the topic searched, and the knowledge extracted from the datasets. For example, an output of the medical question *Why is my son having a continuous chest congestion?* generates a topic "symptom" and a set of the selected data fields from the datasets "procedure, diagnosis".

The QuerioDALI QA system in [153] was evaluated in two scenarios: an open-domain scenario (using DBpedia and Freebase datasets as QA KGs) and a Smarter Care scenario (containing enterprise data about patient conditions and biomedical ontologies). In the Smarter Care scenario, specifically, the situational integration satisfies the information need of healthcare professionals, by answering questions e.g. related to what are the side-effects for all the medications of a patient.

A.3.2 User Guidance

Once situational data are retrieved and returned, the user may decide that they are not suitable for the task at hand [3]. This feedback is useful, for example, for the QA system to redirect its search based on learning (i.e., ML techniques). Besides, user guidance can reduce system mistakes if the user informs it, before the Source Discovery process, which information he *is not* looking for.

A few approaches cover *User Guidance* individually. In the experiments of the study described in [256], each participant receives a first predefined query, and all subsequent ones require participants to write a question or select a subset of the KG nodes that were displayed. These experiments were based on a specific scenario (domain of books) and had a group of 5 participants that performed exploratory searches on the KG.

In [258], the model was evaluated on two publicly available VQA datasets[6] and involved ground truth answers generated by 10 human subjects. Similarly, in [171], human participants

[6]Toronto COCO-QA [198] and VQA [10].

have elaborated a database of questions containing 152 questions related to city domain, in order to evaluate the geographical system proposed.

In the FREyA system [42], the user has an important role to improve the performance of NLIs to ontologies. When the system fails to automatically generate the answer, it will prompt the user with a dialog comprising two tasks: *disambiguation*, in which the user resolves identified ambiguities; and *mapping*, in which the user maps query terms to ontology concepts suggested. Thus, the system learns from the user's selections, and improves its performance over time.

A.3.3 User Guidance & Decision-Making Support

Some studies have covered both ***Decision Support and User Guidance***: [212] presents an approach for natural language QA over a knowledge base containing medical texts information. For conducting the work, an EHR (Electronic Health Record) system dedicated to Oncology was used, which provided EHR records containing medical appointment data. From the involvement of professionals, it was possible to define sentences of interest, e.g. "what symptoms does the patient report?" and "is the symptom associated with which diagnoses?". The QA process returned a set of answers from the KB, with indication of the correct tuples that support the professional's needs.

In [252], Decision-Making Support is covered since relevant news' documents and representative images are retrieved and presented to the user, who can be aware of the current news. The experiments involved fifteen participants; each one of them selected three query questions from a list of questions provided, and gave evaluation scores to the results returned from the system. The Bio-SeCo application presented in [163] uses a highly efficient algorithm for rank aggregation, and consensus ranking methods to get a global ranking of results. The results are shown in the interface, so that users can find answers to their biomedical questions, and provide feedback about the relevance of the system and its ranking strategy.

In [93], historical data in medicine were used to structure an automatic diagnosis decision-making table, i.e., a kind of knowledge expression system where conditions are associated with decision rules. Potential diagnosis rules were extracted from the decision-making table, which can offer effective diagnosis service. The PAO ontology (see Subsection A.1.1) was provided to users as an access interface, where they entered the system and raised NL questions. Also, PAO was tested on the QA system in medicine and the generated answers were compared with manual answers of authoritative experts.

In [237], User Guidance is present by means of a process called User Assessment, in which users are able to contradict or endorse existing facts, optionally providing additional sources for the knowledge in platform. The system identifies whether there is a missing knowledge and provides the user with a link to add it, so the user may guide the process. Also, a query-processing engine is capable of tracing the path it followed to generate answers, in order to create a detailed explanation of how those answers were generated. The facts used as part of that proof can be extracted and presented to the user as concise explanation.

A.3.4 Response Time Improvement

Considering a QA system running on a server, the time required to compute answers and the overhead of the network requests must be taken into account, mainly because the user expects response times of around 1 second [52]. Since agility is crucial for SDI, it could favor the response delivery in QA. Moreover, the situation-aware aspect of SDI favors the acquisition of immediate information, and consequently, a holistic view of various activities, which will be an

important factor for making the "best" decision, especially when allied to the interaction feature of QA [84].

This subfeature is the most present one in the reviewed approaches. In the work proposed by [218] (an extension work of [219]), it was implemented parallelization over some components of the system, in order to speedup runtime. Parallelization is also present in [38] (multiple answering agents address the same question in parallel, with results combined) and in [119] (the system sends the question to six streams in parallel).

The QUEST system [155] computes answers in interactive time, with median run-time of 1.5 seconds. The PullNet framework [229] applies ADAM optimizer [128] in the learning process, as well as indexing for fast retrieval. In the Ephyra system [208, 209], duplicated text snippets retrieved from the Web are removed, in order to reduce the processing time. *Response Time Improvement* also appears in the OpenQA framework [160], where a cache service stores the results of processes, so that future requests to the same process can be executed faster.

In the IM system proposed by [129, 141], the query planning algorithms used provides a query interface to distributed structured information sources. The plan executor tries to access the sources in parallel, minimizing the time taken to provide answers. Also, a query processor allows to prune information sources in order to solve completeness and redundancy issues. Dealing with redundancy also improves time, since a minimal set of information sources is determined for answering the query.

Databases and ontologies are queried in the QA system proposed by [77]. Thus, answers can be returned by both MySQL and *Sesame* [26], which includes the SeRQL query language for handling RDF data. In the search process, path constraints are specified and added as further constraints to the `WHERE` clause of the SeRQL query, simplifying it and speeding up the query performance.

Given a query, the SIMS system [12] generates and executes a *query plan* for accessing the appropriate information sources. Before executing a query, the system performs a query reformulation to minimize the cost and the amount of data processed, so that generated subqueries are executed in parallel. Besides the reformulation and parallelism, the approach presents a cache mechanism for data that are required frequently or are very expensive to retrieve.

In IBM Watson [73] fast runtime indices are created using the Hadoop map-reduce framework[7], so that the system is able to answer more than 85% of the questions in 5 seconds or less. The systems in [53] and [189] include search indexes for offering faster question responses. Similarly, the OQA system in [63] uses a simple KB abstraction where ground facts are represented as string triples (argument1, relation, argument2). Triples are used from curated and extracted KBs, and they are stored in an inverted index that allows for efficient keyword search. Indexes are also used in the QA system proposed in [264], in the SemanticQA system [232], in [107], and in the hybrid QA system described in [8].

A.3.5 Decision-Making Support & Response Time Improvement

Some studies cover both features, although they do not cover User Guidance. The NOUS framework presented in [36] builds domain specialized KGs by fusing curated KBs with extracted knowledge. The user can visualize the resultant graph and a summarization of quality-related statistics, which provides an overview of the current situation (Situation Awareness). Also, the streaming graph mining algorithm developed in the proposal increases the execution speed of the proposal when discovering trends in streaming data.

[7]Available at: <`https://hadoop.apache.org/`>.

The HSIN framework developed in [33] for CQA platforms integrates and simultaneously learns the questions textual contents, their related categories information and user's social interaction. When a user poses a new question, HSIN uses the integrated information for ranking similar historical questions proposed by other users, along with corresponding answers. This recommendation provides Decision-Making Support through Situation Awareness. In addition, the authors used an optimization method to speed up for training time in their HSIN framework.

The proposal in [239] implements a prototype as a freely accessible web application, which allows users to enter their questions and receive answers. The answers are show in a tabular view if appropriate, and the view allows the user to enrich the generated answers by displaying further information for the returned resources. The web application thus contributes to Situation Awareness. With respect to Response Time Improvement, a BOA index [83] was created based on Lucene indexer[8], which allows for time-efficient search, e.g., it improves the mapping of properties in natural language queries compared to using a text index.

In all cases where *Decision-Making Support* was covered in addition to *Response Time Improvement*, the support was *partial* through Situation Awareness, which means that there was not an explicit assistance, but the approach was able to make the user aware of the current situation, thus contributing to *Decision-Making Support*.

A.3.6 User Guidance & Response Time Improvement

The study in [152] covers ***User Guidance and Response Time Improvement***, since the user participates in the evaluation of the system, and the Triple Mapping Component improves time by handling indexing and computational expensive queries. In [96], users can define quality metrics for data quality management in the Constance system. Also, the approach identifies potential foreign keys from the user query, and builds indexes on the corresponding attributes to improve the performance.

In the evaluation of QALL-ME framework [72], users received a brief description of the system (just to make them aware of its capabilities) and elaborated a set of questions containing 304 cinema questions referred to Italy region. The QALL-ME workflow is managed by the QA Planner, which orchestrates the web service components by receiving input parameters, including spatial-temporal context and pattern mappings. As the system response time is directly related to the size of the question pattern set, this set is reduced as much as possible to contain only the essential patterns, thus improving the response time.

In the model proposed by [88], the information sources were constructed with User Guidance: the authors simply approached 10 adults, asked them questions, and tape-recorded their answers. Also, as already mentioned, the model includes arc-search procedures and a constraint propagation component that provide a satisfactory solution to the convergence problem, since they reduce the node space to a minimal set of good answers. Pruning down non-informative answers enables Response Time Improvement.

The joint model proposed in [271] employs *User Guidance* as five questioners were asked to pose questions independently, in order to enrich the evaluation dataset. The model is based on Integer Linear Programming, which was implemented by using Gurobi[9], a fast and powerful mathematical optimization solver that has shown efficiency w.r.t. the average speed in processing questions.

[8]Available at: <http://lucene.apache.org/>.

[9]Available at: <https://www.gurobi.com/>.

A.3.7 User Guidance, Decision-Making Support & Response Time Improvement

Some approaches cover all features of SDI's Timely Decision Support together. One of them is [28], which presents the AskHERMES system for answering complex clinical questions. In the Summarization module of the system, the answers are grouped by content terms, thus helping physicians quickly and effectively browsing answer clusters (Response Time Improvement). In the evaluation phase, AskHERMES was compared with state-of-the-art systems, and manual evaluation of the output was performed by three physicians, aiming to examine how well each of the three systems answer the questions. The evaluation found that AskHERMES is competitive with the state-of-the-art systems and its answer presentation interface helps physicians easily obtain information from different points of view (Decision Support).

The work in [177] exploits end-user feedback and runtime data, with the overall goal of improving the quality of experience. Also, wrappers are modeled to accommodate different sources, which deal with query complexity, so that runtime is improved. The results of the study have shown that a great number of changes performed in real-world APIs could be semi-automatically handled by the proposed wrappers and the ontology, thus providing Decision Support when ingesting and analyzing the data.

The MiPACQ system proposed by [27, 183] was evaluated through a clinical dataset and human-annotated answers (User Guidance). The approach includes an Answer Summarization process that performs answer ranking and decides whether to present statistical information about subsets of the results or other potentially interesting aspects of the set of patient records returned (Decision Support). Lucene was used as indexing tool in the Information Retrieval module as time improvement method.

The QA system for IT support in [158] monitors the user's feedback after every dialog turn, and uses this feedback to improve its knowledge. W.r.t. Decision Support, the system provides IT support to users' questions by exploring multimedia data, and furthermore, it comprises a Resolution Automation process, i.e., the platform supports an automation to be attached to an answer. Response Time Improvement is achieved by means of Apache Lucene indexing that enables fast retrieval.

In [120], wrappers are developed for accessing the biomedical resources. This process is performed with a subject matter expert to determine which metadata fields must be processed (User Guidance). The approach provides Situation Awareness by means of the graphical interface that suggests relevant resources to be explored. Also, the Resource Index improves runtime information retrieval.

The MedQA system [267] presents a user with both Web definitions and MEDLINE sentences suitable for his question. These results are shown in an interface, which displays a summary of potential clusters of sentences, along with other relevant sentences that might be additionally important to the biologists or the physicians. The approach uses the indexing tool Lucene in the Document Retrieval step, a technique for Response Time Improvement. User Guidance was also covered in MedQA, since a group of physicians volunteered to participate in the study, evaluating the performance of the system and comparing it with other online information systems. Lastly, in the framework proposed by [70], the question is sent to specialized nodes and the fused output information is sent back to a GUI element, where a dashboard integrates external and internal data as result. The result provides Decision Support and allows the user to correct the information if needed. In the experiments, the QA system consisted of a indexation phase that aimed to prepare all the information required for the search phase, thus optimizing the time response.

APPENDIX B – PUBLICATIONS

This thesis is supported by the following productions:

- **Maria Helena Franciscatto**; Luis Carlos Erpen de Bona; Celio Trois; Marcos Didonet Del Fabro. *Built-up Integration: A New Terminology and Taxonomy for Managing Information On-the-fly.* In: Journal of Information and Data Management (JIDM), v. 15, p. 80, 2024.
- **Maria Helena Franciscatto**; Marcos Didonet Del Fabro, Luis Carlos Erpen de Bona; Celio Trois; Hegler Tissot. *Blending Topic-based Embeddings and Cosine Similarity for Open Data Discovery.* In: 24th International Conference on Enterprise Information Systems (ICEIS), p. 163, 2022.
- **Maria Helena Franciscatto**; Marcos Didonet Del Fabro, Celio Trois; Luis Carlos Erpen de Bona; Jordi Cabot; Leon Augusto Gonçalves. *Talk to Your Data: a Chatbot System for Multidimensional Datasets.* In: IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), p. 486, 2022.
- **Maria Helena Franciscatto**; Marcos Didonet Del Fabro, Celio Trois; Jordi Cabot; Leon Augusto Gonçalves. *Querying Multidimensional Big Data Through a Chatbot System.* In: The 37th ACM/SIGAPP Symposium On Applied Computing (SAC), Virtual Event, 2022.
- **Maria Helena Franciscatto**; Marcos Didonet Del Fabro, Celio Trois; Hegler Tissot. *Towards Open Data Discovery: A Comparative Study.* In: The 37th ACM/SIGAPP Symposium On Applied Computing (SAC), Virtual Event, 2022.
- Marco Antoni; Andrea Charão; **Maria Helena Franciscatto**. *Querying Brazilian Open Data Using a Hybrid NLP-based Approach.* In: 24th International Conference on Enterprise Information Systems (ICEIS), p. 120-130, 2021.