SpringerBriefs in Applied Sciences and Technology

PoliMI SpringerBriefs

More information about this subseries at http://www.springer.com/series/11159
http://www.polimi.it

Shima Zahmatkesh · Emanuele Della Valle

Relevant Query Answering over Streaming and Distributed Data

A Study for RDF Streams and Evolving Web Data

Springer

Shima Zahmatkesh
DEIB
Politecnico di Milano
Milano, Italy

Emanuele Della Valle
DEIB
Politecnico di Milano
Milano, Italy

ISSN 2191-530X ISSN 2191-5318 (electronic)
SpringerBriefs in Applied Sciences and Technology
ISSN 2282-2577 ISSN 2282-2585 (electronic)
PoliMI SpringerBriefs
ISBN 978-3-030-38338-1 ISBN 978-3-030-38339-8 (eBook)
https://doi.org/10.1007/978-3-030-38339-8

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Nowadays, Web applications have often to combine highly dynamic data streams with data distributed over the Web to provide relevant answers for their users. Social Media analysis and Web of Things are contexts that require this type of Web applications. Social Media analysis often needs to look up the profiles of influencers most mentioned in a stream of posts. Web of Thing applications often have to join streams of sensor observations with data about the platforms that host the sensors to extract samples that best represent a phenomenon. In those settings, responding in a timely fashion, i.e., being reactive, is one of the most important requirements. However, when trying to join data streams with distributed data on Web, the time to access and fetch the distributed data can be so high that applications are at risk of losing reactiveness.

In particular, this book focuses on RDF Stream Processing (RSP) engines because they offer a query language (namely, RSP-QL) that eases the development of this type of Web queries and caching features that keep RSP engines reactive if the distributed data is static. However, RSP engines are also at risk of losing reactiveness when the distributed data is evolving.

For this reason, this book introduces the ACQUA framework to address the problem of evaluating RSP-QL queries over streaming and evolving distributed data. ACQUA keeps a local replica of the distributed data and offers an expertly defined maintenance process to refresh the replica over time. The users of ACQUA can set a refresh budget to control the number of elements to refresh in the replica before each evaluation. If set correctly, the refresh budget guarantees by construction that the RSP engine is reactive. When the maintenance process has enough budget to refresh all the stale elements, the answers of the RSP engine are exact. Otherwise, the maintenance process tries to approximate the result. Notably, the maintenance process is designed to gracefully decrease the accuracy of the answer when the refresh budget diminishes.

The remainder of the book presents extensions of the ACQUA framework for relevant query answering. It first introduces ACQUA.F to reactively answer to queries that pose a filter condition on the distributed data. For instance, a social media analysis may ask for users with more than one million followers that are

mentioned in the last five minutes. Then, it brings in rank aggregation as a way to combine the maintenance processes proposed in ACQUA and those proposed in ACQUA.F.

Finally, the book focuses on continuous top-k queries and introduces AcquaTop. Consider, for instance, a mobile application for supporting people in parking in a crowded city. An RDF stream continuously reports the positions of the cars looking for a parking lot, while a Web service returns the number of free parking lots per city district. The continuous top-k query has to return the areas (around the car of the user's mobile App) where there is the largest number of free parking lots and the smallest number of cars looking for parking.

The authors of this book thank Dr. Daniele Dell'Aglio for his comments and support during the Ph.D. studies of Dr. Shima Zahmatkesh. They also thank Dr. Soheila Dehghanzadeh for reviewing Chap. 3. Last but not least, they acknowledge the contribution of Prof. Abraham Bernstein, Dr. Alessandra Mileo, and Dr. Shen Gao in shaping the ACQUA framework.

Milan, Italy
November 2019

Shima Zahmatkesh
Emanuele Della Valle

Contents

1 Introduction 1
1.1 Motivation 1
1.2 Problem Statement and Research Question 3
1.3 Approach and Contributions 4
1.4 Structure of the Book 5
References 6

2 Background 9
2.1 RDF Data Model and SPARQL Query Language 9
2.2 Federated SPARQL Engines 12
2.3 RSP-QL Semantic 14
2.4 Top-k Query Answering 17
2.5 Top-k Query Monitoring Over the Data Stream 19
2.6 Metrics 21
References 24

3 ACQUA: Approximate Continuous Query Answering in RSP 27
3.1 Introduction 27
3.2 Problem Statement 29
3.3 Proposed Solution 30
3.3.1 Window Service Join Method 31
3.3.2 Window Based Maintenance Policy 32
3.4 Experiments 34
3.4.1 Experimental Setting 34
3.4.2 Experiment 1—Comparison of Proposers 34
3.4.3 Experiment 2—Comparison of Maintenance Polices 36
3.5 Related Work 37
3.6 Conclusion 38
References 39

4 Handling Queries with a FILTER Clause 41
4.1 Introduction 41
4.2 Problem Statement 43
4.3 Proposed Solution 43
4.3.1 Filter Update Policy 44
4.3.2 ACQUA.F Policies 45
4.4 Experiments 46
4.4.1 Hypotheses 46
4.4.2 Experimental Setting 46
4.4.3 Experiment 1—Sensitivity for Filter Update Policy 47
4.4.4 Experiment 2—Sensitivity for ACQUA.F Policies 50
4.5 Outlook 52
References 53

5 Rank Aggregation in Queries with a FILTER Clause 55
5.1 Introduction 55
5.2 Rank Aggregation 57
5.3 Problem Statement 59
5.4 Rank Aggregation Solution 59
5.4.1 ACQUA.F^{+} Policy 60
5.4.2 WBM.F* Policy 60
5.5 Experiments 61
5.5.1 Research Hypotheses 62
5.5.2 Experimental Setting 63
5.5.3 Relaxing ACQUA.F Assumption 65
5.5.4 Experiment 1—Sensitivity to the Filter Selectivity 66
5.5.5 Experiment 2—Sensitivity to the Refresh Budget 70
5.6 Outlook 72
References 73

6 Handling Top-k Queries 75
6.1 Introduction 75
6.2 Problem Statement 79
6.3 Topk+N Solution 81
6.3.1 MinTopk+ 81
6.3.2 Updating Minimal Top-K+N Candidate List 82
6.3.3 Super-MTK+N List 85
6.3.4 Topk+N Algorithm 86
6.4 AcquaTop Solution 92
6.4.1 AcquaTop Framework 92
6.4.2 AcquaTop Algorithm 93
6.4.3 Cost Analysis 96

6.5 Evaluation 97
6.5.1 Experimental Setting 98
6.5.2 Preliminary Experiment 101
6.5.3 Research Hypotheses 103
6.5.4 Experiment 1—Sensitivity to the Refresh Budget 104
6.5.5 Experiment 2—Sensitivity to Change Frequency (CH) 106
6.5.6 Experiment 3—Sensitivity to K 106
6.5.7 Experiment 4—Sensitivity to N 108
6.5.8 Wrap up 109
6.6 Related Work 111
6.7 Outlook 112
References 113

7 Conclusion 115
7.1 Suggestions for Future Works 117
References 119

Index 121