Explaining Data Patterns using Knowledge from the Web of Data

Thesis

Explaining Data Patterns using Knowledge from the Web of Data

Ilaria Tiddi

Knowledge Media Institute
The Open University

This dissertation is submitted for the degree of
Doctor of Philosophy

November 2016

“*The moment you doubt whether you can fly, you cease for ever to be able to do it.*”
Peter Pan; or the Boy Who Wouldn’t Grow Up (1904)
J. M. Barrie

To my parents

(who understood that sometimes it is better not to understand)

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Ilaria Tiddi
November 2016

Acknowledgements

I spent months trying to think how I would write these pages – we all know this is the section that will be read the most. In the end, it probably took me the same amount of time it took to write the thesis itself. I promised a friend that the first sentence of the Acknowledgements would be "Graphs are everywhere", because graphs have been my Odi et amo *in these years. Well, I could not keep the promise. I had to reconsider this sentence.*

***

Patterns are everywhere. Thinking back right now, these 4 years have been all about patterns (for clarity: by "pattern", I mean "something recurrent"). Patterns have been part of my work, because I (almost desperately) tried to explain them automatically. But they have also been part of my days, without even me realising that. Funnily, when I thought of whom I should have thanked, I found a very clear, recurrent pattern. I hope the reader will agree.

First and foremost, I need to thank my two supervisors, **M**athieu d'Aquin and Enrico **M**otta, for their guidance, support and trust. I do not think that words can be sufficient to thank Mathieu for all the time he patiently dedicated to me. I am immensely grateful to him for all the times he let me in his office and listened to any kind of problem, complaint, fear, doubt I might have, and for all the supervisions we had in impossible places (even public transports!) around the world. And I am even more thankful to Enrico, who never mistrusted us in the work we were doing and, more especially, made a bet on me a long time ago, trusting that I could be able to arrive at the end of this journey. Legends say they fought to decide who would be my first supervisor: I am sure my experience would have been the same either ways.

Many thanks should also go to my exa**M**iners, Frank van Harmelen and Harith Alani, for their valuable comment on my thesis, for having made my Ph.D. dissertation an awesome and enjoyable experience that I will never forget. They might not remember this, but Harith was my mentor during my first conference and Frank was the inspiring opening talk to the unforgettable SSSW2013. How could I not close my Ph.D. with them? Also, they deserve a big "thank you" for having bravely coped with the crazy burocracy of the OU.

I am as much thankful to the wonderful **M**eD-team, just for being what we are – a messy and noisy group without which I could have never started my Mondays. A special thank is deserved by the best *compagno di banco* I ever had, Enrico D., and by Alessandro for having been my coding-teachers in many occasions, for having dealt with a constantly complaining "I-am-not-a-Computer-Scientist" Ph.D. student, and for having spent hours with me every time I was messing up with our servers.

I need to thank **KM**i for being such a perfect place for studying, working, partying, playing music...basically, living (for the record, I definitely spent more time at my desk than in my room, and I do not regret it). I am proud to be part of KMi, and I will never stop saying that. My only regret is to having met so many cool people, who then left KMi. But I like to think KMi as a family: we grow up together but then we need to let our friends go, knowing that it does not really matter the distance, the good souvenirs that we shared will always keep us close (well, social networks do help, too).

KMiers are only a part of the family I had these years, the **MK**-crowd. As some people know, Milton Keynes is a peculiar place to live, but this is the price for having such special and unique friends. It is quite hard to name everybody, every single friend has contributed to make me the person that I am today, and for that I am extremely thankful (I feel a bit sorry, too – more than sometimes I can be a very difficult person). I thank all the people that lived or orbited Perivale 46, for being friends and siblings; the football group for having coped with my despotic "Sir Alex" manners, and all the other friends that were always there for helping, laughing, listening, drinking, partying, chilling, relaxing, travelling and much more.

I could not have survived my Ph.D. without coming back to my Ro**M**e every now and then (everybody knows how much I am proud of being Roman, so I think it is nice to thank my hometown as a whole). I must have said that a millionth times now, without my mother backing my decisions up (well, most of them) nothing of this would have been possible. For that, no thanks will ever be enough. I must also thank my father, for being my first and biggest fan, and my brothers, for the special relation that makes us able to cross in places such as a random airport without even organising it. And because "Rome, sweet home", a great thank also goes to all the friends that made my homecoming always warm and enjoyable.

Finally, and most importantly, a truly, sincere, special thanks goes my $E\upsilon\tau\varepsilon\rho\pi\eta$, my Music muse, my Mountain (!) of these years. **M**anu, you are the M that walked these years with me day-by-day, that was always there (physically and remotely!), that celebrated my winning moments, that alleviated my (code) sufferings and dried my tears, forcing me to stand up and believe in myself when I did not want to. For that, and for all that will come, I thank you and I just add

the land at the end of our toes goes on, and on, and on...

Abstract

Knowledge Discovery (KD) is a long-tradition field aiming at developing methodologies to detect hidden patterns and regularities in large datasets, using techniques from a wide range of domains, such as statistics, machine learning, pattern recognition or data visualisation. In most real world contexts, the interpretation and explanation of the discovered patterns is left to human experts, whose work is to use their background knowledge to analyse, refine and make the patterns understandable for the intended purpose. Explaining patterns is therefore an intensive and time-consuming process, where parts of the knowledge can remain unrevealed, especially when the experts lack some of the required background knowledge.

In this thesis, we investigate the hypothesis that such interpretation process can be facilitated by introducing background knowledge from the Web of (Linked) Data. In the last decade, many areas started publishing and sharing their domain-specific knowledge in the form of structured data, with the objective of encouraging information sharing, reuse and discovery. With a constantly increasing amount of shared and connected knowledge, we thus assume that the process of explaining patterns can become easier, faster, and more automated.

To demonstrate this, we developed Dedalo, a framework that automatically provides explanations to patterns of data using the background knowledge extracted from the Web of Data. We studied the elements required for a piece of information to be considered an explanation, identified the best strategies to automatically find the right piece of information in the Web of Data, and designed a process able to produce explanations to a given pattern using the background knowledge autonomously collected from the Web of Data.

The final evaluation of Dedalo involved users within an empirical study based on a real-world scenario. We demonstrated that the explanation process is complex when not being familiar with the domain of usage, but also that this can be considerably simplified when using the Web of Data as a source of background knowledge.

Keywords: Knowledge Discovery, Linked Data, Explanation, Background Knowledge, Pattern Interpretation

Table of Contents

List of Figures

List of Tables

Part I

Introduction and State of the Art

Chapter 1

Introduction

This is the introductory chapter where we give an extensive overview of our thesis. Section 1.1 presents the general problem that motivates our work, along with its key concepts; Section 1.2 explains the research hypothesis that we want to validate; Section 1.3 details the general and specific research questions that we will be addressing in this work; Section 1.4 presents our research methodology and Section 1.5 outlines the approach we propose, as well as the specific contributions that we are bringing. Finally, Section 1.6 presents the structure of the thesis, some relevant publications, and an overview of the use-cases we have been using during our research.

1.1 Problem Statement

Knowledge Discovery (KD) is a long-established and interdisciplinary field focusing on developing methodologies to detect hidden regularities in large amounts of data [Fayyad *et al.*, 1996]. Those regularities, commonly known as patterns, are statements describing interesting relationships among a subset of the analysed data. With the data flood phenomenon that we are experiencing nowadays, where digital data are being published without interruption, Knowledge Discovery becomes crucial to provide human analysts with the useful and explicit knowledge contained in the data.

To reveal those patterns, the typical process in KD (sometimes also called KDD, standing for Knowledge Discovery in Databases) is to perform a sequence of operations, which can be summarised as:

- *data pre-processing*, where raw data are cleaned and pruned;
- *data mining*, where machine learning techniques are applied to reveal the patterns of interest;

- *data post-processing*, consisting in the interpretation, explanation and understanding of those patterns, so that they can be further exploited.

While much effort has been put in studying ways to automatically assist the experts in pre-processing and mining data, the post-processing step still mostly relies on manual analysis. Generally, domain experts use their own background knowledge to interpret the patterns discovered through data mining in order to explain (and possibly refine) them. Firstly, this means that the interpretation process is intensive and time-consuming. Second, some patterns are likely to remain unexplained, especially when the interpretation requires expertise from different domains, and/or the experts lack some of the background knowledge that is necessary to the explanation. What we aim to investigate in this thesis is how to improve this data post-processing step, and especially how to automatically assess an explanation to data mining patterns.

The last decade has seen a huge amount of information being openly published in the form of a Web of Data. Domain-specific knowledge has been made available in machine-readable formats but, most importantly, has been connected (under the name of Linked Data[1]) with the objective of encouraging information sharing, reuse and interoperability. If the information contained in the Web of documents used to be hidden and hard to automatically detect, the rise of the Web of Data now makes it possible to move one step closer to the automatic discovery of knowledge.

Given such a constantly increasing amount of published and connected knowledge, our assumption is that explaining patterns can be easier, faster and more automated. Thanks to the practices relying on semantic technologies available nowadays, the Web of Data can be automatically accessed in order to find the knowledge that is needed for the pattern interpretation process, so that the effort that the experts have to put into it can be reduced. The hypothesis that we aim to demonstrate with our research is that the data post-processing step of Knowledge Discovery can be improved with the support from the Web of Data, because its connected knowledge can facilitate the automatic access to the background knowledge required to explain data patterns.

Below, we define the basic terminology and core concepts of our work.

Data. Raw signals either automatically or manually produced. They are factual (i.e. they rely on something specific) and unorganised.

[1] http://linkeddata.org/

Patterns. Expressions providing a high-level description of a subset of data. According to [Fayyad *et al.*, 1996], patterns are meant to be:

- *non-trivial*: they should not be mere observations, but should include some inference;
- *understandable* by humans (assuming they have the right background knowledge);
- *novel*: they should bring some sort of new information to the humans;
- (potentially) *useful*: they should be beneficial for users or for a predefined task.

Background Knowledge. The external knowledge, usually found in external sources (such as manuals or the experts' mind) that gives a meaning to patterns. Background knowledge consists of statements that are not contextual to the patterns, but that make them novel and useful.

Knowledge. Patterns transformed into evidence by the addition of background knowledge. Knowledge is used to achieve tasks and make decisions. It is inferential and abstract.

1.2 Research Hypothesis

Before introducing our research hypothesis, we introduce a few scenarios, which will help to clarify our assumptions and objectives. These examples are part of the use-cases presented in Section 1.6.2. Let us imagine the following type of data patterns:

(A) Search rates of a specific term entered over a web search engine across time, such as, for instance, how many people have searched for the term "A Song of Ice and Fire"[2] in the last 10 years. The trend shows that the term was searched with a very regular frequency, especially in the past 5 years.

(B) Groups of geographical points, representing the UK districts from which some Open University students come, formed according to the Faculty the students belong to. Clusters show that in certain regions some faculties attract more students than others, e.g. the Business&Law Faculty attracts more students than average from the districts around London.

(C) Communities of research papers (the Open University publications), grouped according to the semantic similarity of their words, where papers about the same topic have been clustered together.

[2] *A Song of Ice and Fire* is an American series of fantasy novels written by George R.R. Martin firstly published in 1996. Its popularity considerably increased after its TV adaptation, *The Game of Thrones*, broadcasted by HBO since 2011. At the time of writing, the sixth novel is being written, and the sixth season of the TV series has not yet been released.

Based on the above, we can make the following observations.

[Obs1] *Data patterns obtained after a data mining process require an* ***explanation****.*
An explanation is in fact required to motivate in (A) why the popularity of "A Song of Ice and Fire" regularly increases or decreases, to explain in (B) why certain faculties attract more students in specific regions, or in (C) to find out which is the topic related to some specific documents.

[Obs2] *To find out the explanation, one needs to make some* ***hypotheses*** *aiming at interpreting the patterns.*
One could imagine that in (A) users become more interested in the term (associated to the novel) following some specific events, such as the release of the TV series "Game of Thrones". Similarly, one can also hypothesise for (B) that some faculties are more attractive due to the facilities offered by the location, e.g. London is a financial hub, or for (C) that all the keywords of documents in a given group are related to Mathematics.

[Obs3] *Those hypotheses can only be generated and validated with some* ***background knowledge****, which is generally part of the remit of an expert in the given domain.*
If one has never heard of the Game of Thrones TV series, or does not know the economical situation of London, or has never seen the keywords contained in the documents, none of the hypotheses for (A), (B) or (C) can be validated.

[Obs4] *The* ***Web of Data****, promoting shared and automatically accessible knowledge, is also a source of background knowledge.*
The Web of Data contains information about events, that can be related to a searched term (A); it also contains geographical information, which can reveal facilities about cities or districts (B); and it contains linguistic information, that can be used to derive that words contained in similar documents are part of the same topic (C).

For this reason, it appears possible that parts of the background knowledge required to explain a pattern are available through the Web of Data, and this can facilitate the interpretation process. Therefore, the main research hypothesis that we aim to demonstrate with this thesis is:

The process of pattern explanation can be improved by using background knowledge from the Web of (Linked) Data

We assume that the knowledge embedded in the Web of Data, which has been collected and published under the name of the Linked Data Cloud, can be harvested automatically, and that the extracted information can be used either to assist the experts with the knowledge they are missing to give a complete explanation, or to provide new knowledge to the non-experts. More generally, we aim to show whether and how the Web of Data can benefit the Knowledge Discovery process.

It is important to remark that our hypothesis is not to prove that the Web of Data can be a replacement for Machine Learning, but rather that we can combine its high-performance algorithms with semantic technologies to benefit the Knowledge Discovery process. A common misunderstanding is to think that patterns are produced purely according to the way the problem has been modelled within the data mining process (during a step called *feature selection*). On the contrary, while patterns are indeed the output of data mining, we state that data points happen to be in the same pattern because they have common characteristics, which can often be explained by some external knowledge, outside of the problem definition. In our illustrating examples, data behave differently according to some criteria (respectively: the popularity score, the faculty choice and the text similarity) but the explanation for each of these behaviours comes from external factors, such as the ones that we have presented. Assuming the required knowledge is represented within the Web of Data, it is then possible to exploit it to produce explanations.

Finally, our hypothesis is not that the Web of Data can replace the experts, but rather that the cost of the interpretation process can be reduced. We intend to show that Linked Data are a support for humans, either because they provide additional knowledge whenever the domain expert is not sufficient, or when the user is not familiar with the domain.

1.3 Research Questions

Once defined our research hypothesis, we define the main research question that we will investigate in this thesis.

How can we automatically explain patterns of data using background knowledge from the Web of Data?

To answer this, we articulated our research in a specific set of sub-questions, which are summarised as:

RQ1: What is an explanation? (Section 1.3.1)

RQ2: How to automatically find the required background knowledge in the Web of Data? (Section 1.3.2)

RQ3: Which process can be used to automatically generate explanations? (Section 1.3.3)

RQ4: How to assess the validity of such a process? (Section 1.3.4)

These sub-questions are presented more in details below.

1.3.1 RQ1: Definition of an Explanation

If we wish to produce explanations for patterns, the first thing we need to do is to formally define (the concept of) an explanation. Is it a correlation between events? Or is it a cause-effect relationship? If not, which are the differences between them, and how do we recognise them?

The idea here is that an explanation is more than a causal relationship between events, and that other factors might take part in the process. More specifically, explaining might not only mean finding the event that is correlated to another event (X caused Y), but also finding out why they have occurred together (X caused Y because of Z).

The question that needs to be investigated is therefore what else might be needed for an explanation to be complete. We want to identify which are the important actors that take part in an explanation, which are their roles and their interactions, so to be able to declare that, given a phenomenon encoded into a pattern, we will be giving a complete explanation to it. Once identified those components, we can then further reuse them as the building blocks upon which we can set up our automatic process.

This problem has been addressed in Section 2.1 and an answer to these questions is to be found in Section 2.1.2.

1.3.2 RQ2: Detection of the Background Knowledge

If we aim at producing explanations for a specific pattern automatically, the next question is where to find information about it in the Web of Data, and how to find it in an automatic and generic way. If we consider exploiting the Linked Data Cloud, we find ourselves in front of a vast ocean of structured and cross-domain information. The challenge is then how to strategically find the right piece of background knowledge in there, without incurring expensive computational costs. The question can also be sub-articulated in two parts.

(a) *Inter-dataset search problem.* One problem is detecting the background knowledge at dataset-level: automatically finding the right datasets in the large choice offered by

Linked Data might not be easy. Also, assuming Linked Data effectively contain the right datasets, where and how to automatically find them?

(b) *Intra-dataset search problem.* The second question is how to find the useful part of the knowledge for explaining a given pattern within a specific dataset. Not all the information therein represented might be useful to produce explanations. Which predicates do we need, and about which data? What are the criteria to use to decide that some predicates are the ones we are really interested in? Are the data in hand sufficient in their Linked Data form?

Solving these issues is challenging from a double perspective: (i) if there is a lack of datasets, the data in the pattern might not be described enough in Linked Data and the search for an explanation could be unsuccessful; (ii) because domains are dynamic, it might not be possible to find general interestingness criteria, since what is important for interpreting the data might be highly dependent on the context. We investigate these aspects in Chapter 4 and Chapter 6.

Finally, this question can be integrated within the more general discussions on Linked Data accessibility and trust: How much knowledge can be accessed using the available technologies? How reliable is the information provided? Is there enough coverage of the domains of interest? A preliminary study of this issue can be found in Section 8.3.4.

1.3.3 RQ3: Generation of the Explanations

Once the background knowledge about the data has been found, the next question to answer is: how do we use it to explain the patterns? What kind of mechanisms can automatically generate explanations? And how do we produce them so that they are presentable to the users? Can we expect that explanations will effectively bring new knowledge?

While the human process of pattern interpretation might trivially look like

observe data
↓
make hypotheses
↓
validate them with your background knowledge
↓
generate natural language explanations

automatically realising the same process is challenging. Automatic frameworks for logical inference and reasoning, which have been extensively studied in Artificial Intelligence, can be applied to generate explanations, but the biggest challenge is the data deluge, i.e. those systems might not be designed to deal with the amount of information available in the Web of Data, and we might therefore run into scalability problems. This question is mostly addressed in Chapter 3 and Chapter 4.

1.3.4 RQ4: Evaluation of the Explanations

The explanations having been generated, they are still considered hypotheses until they are verified. The last question we need to answer is about their quality and significance assessment. Are explanations good? How do we know? Which ones are more interesting, and why? Are interestingness measures enough, or do we need to take into account other aspects? And if so, how to define them? This evaluation step can also be articulated in two sub-questions.

(a) *Explanation evaluation*. This consists in finding the criteria to assess the interestingness and the general quality of an explanation, so that we can decide that a hypothesis is a sufficient explanation. Answers to this question can be found in Chapter 4, 5 and 6.

(b) *Method evaluation*. How do we evaluate that our interpretation process based on the Web of Data is efficient, when compared to domain experts? This question is answered in Chapter 7.

The challenge for this part is that we might encounter a lack of background knowledge, if a clear evidence for some of the generated hypotheses is missing, or a recursion issue, if a hypothesis requires a new piece of knowledge to be explained itself. These problems are also discussed in Chapter 6.

1.4 Research Methodology

To address the research questions presented above, our work focuses on designing and developing an automatic process that, given a pattern of data to be explained on one side, and the Web of Data on the other, is able to generate satisfactory explanations to the grouping of the items based on the knowledge automatically extracted from the Web of Data. To this end, our research is built upon the experimental methodology, commonly used in Computer Science to evaluate new solutions for a problem [Elio *et al.*, 2011].

Our research starts with an extensive literature review, presented in Chapter 2. This covers approaches from the different fields that contribute to our research. Focusing first on the explanations, we investigate work in Cognitive Science to understand which components we need for a process producing explanations, and work in Knowledge Discovery to analyse how to generate them automatically. We then focus on the way to exploit the available knowledge in the Web of Data, by trying to understand which Semantic Web technologies and which Graph Theory algorithms and models can be integrated in our process.

The design of the prototype aiming at validating our research hypothesis is then realised in an iterative way, following the idea of [Denning, 1981] that experimentation can be used in a feedback loop to improve the system when results do not match expectations. In the following chapters (Chapter 3, 4, 5 and 6) we approach each of our sub-questions, propose a solution and use the experimental evaluation to assess whether the system is producing the expected results, which in our case consist of meaningful explanations for a pattern generated from the Web of Data.

The final part of our methodology is then to assess the validity of our system in an empirical user study designed around a large real-world scenario. This is presented in Chapter 7. We evaluate the system by comparing the explanations generated automatically from the Web of Data with the ones provided by the set of users who took part in the study.

1.5 Approach and Contributions

The resulting system, that we called Dedalo, is an automatic framework that uses background knowledge automatically extracted from the Web of Data to derive explanations for data, which are grouped into some patterns according to some criteria.

Below we present the process in details, namely by showing some real-world applications of our approach, an overall picture of the process, and the contributions that our approach is bringing.

1.5.1 Applicability

The approach that we propose could benefit many real-world domains, namely those where background knowledge plays a central role for the analysis of trends or common behaviours. For instance, Dedalo could be exploited for business purposes such as decision making or predictive analytics, by providing the experts with the Linked Data information that they might miss to explain the regularities emerging from the raw data collected using data analytics methods. A practical application is the Google Trends scenario used throughout this thesis: namely, Linked Data could help in explaining the increased interest of the users towards some topics, which could be used to improve the user experience and profiling.

A second application is in educational fields such as Learning Analytics, where Dedalo could be helpful to accelerate the analysis of the learners' behaviours. This would allow universities to improve the way they assist people's learning, teachers to better support their students or improve their courses, as well as the staff to plan and take their decisions.

Finally, Dedalo could be applied in the medical contexts, by helping the experts in explaining patterns and anomalies requiring some external knowledge, e.g. the environmental changes affecting the spread of diseases.

Or course, these are only a few examples of the way the explanation of patterns through background knowledge from Linked Data can be useful.

1.5.2 Dedalo at a Glance

Figure 1.1 presents an overview of Dedalo, with indications about the chapters of the thesis describing each part. As one can see, every step requires the integration of knowledge from the Web of Data, which is the core aspect within our process.

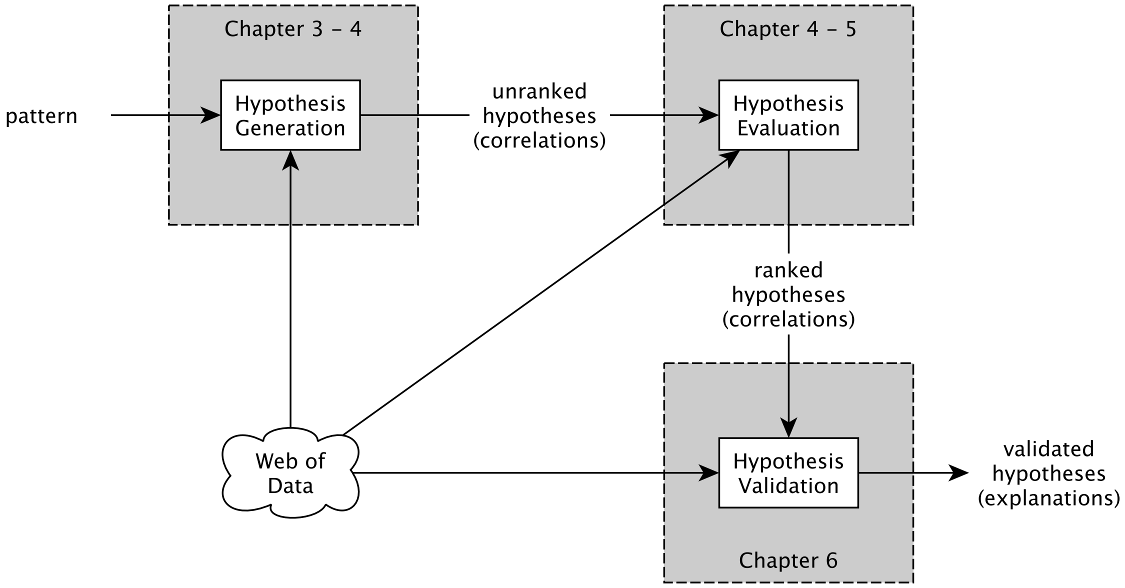


Figure 1.1 Overview of Dedalo according to the thesis narrative.

Hypothesis Generation. Assuming a pattern (any data grouped according to some criteria: Clusters, association rules, sequence patterns and so on), this first step is to search the Linked Data space for information about the data contained in the pattern, and then to generate some correlated facts (alternatively called *anterior events*, *hypotheses* or *candidate explanations* throughout this work), which might be plausible explanations for it.

We combine here several techniques from Machine Learning, Graph Theory, Linked Data and Information Theory to iteratively explore portions of Linked Data on-the-fly, so that only the part of information needed for the explanation is collected. By doing so, we avoid inconveniences such as dataset indexing or crawling, while comfortably keeping the resolution of not introducing any a priori knowledge within the process.

Hypothesis Evaluation. In this step, we evaluate the unranked facts so that we can assess which ones are valid and may represent the pattern.

We use Linked Data combined with techniques from Information Retrieval, Rule Mining and Cluster Analysis to define the interestingness criteria, giving the generated hypotheses a priority order. This step also includes the study of a Machine Learning model that predicts the likelihood of improving the quality of the hypotheses by combining several of them.

Hypothesis Validation. The final step consists in validating the ranked hypotheses so that they are turned into explanations that can be considered valuable knowledge.

The validation process exploits Linked Data and applies techniques from Graph Theory and Machine Learning to identify the relationship between a pattern and a hypothesis generated from Linked Data that is correlated to it.

1.5.3 Contributions of the Thesis

This thesis aims at being a contribution to the process of automatically discovering knowledge and at reducing the gap between Knowledge Discovery and the Semantic Web communities. From the Semantic Web perspective, our main contribution is that we provide several solutions to efficiently manage the vastness of the Web of Data, and to easily detect the correct portion of information according to the needs of a situation. From a Knowledge Discovery perspective, we show how pattern interpretation in the KD process can be qualitatively (in time) and quantitatively (in completeness) improved thanks to the use of semantic technologies.

Specific contributions, further detailed in the corresponding chapters, are as follows:

- we present a survey on the definition of explanation from a Cognitive Science perspective, and we formalise it as a small ontology (Chapter 2);
- we show how the interconnected knowledge encoded within Linked Data can be used in an inductive process to generate meaningful explanations (Chapter 3);

- we reveal how URI dereferencing can be combined with graph search strategies that access Linked Data on-the-fly and remove the need for wide data crawling (Chapter 4 and Chapter 6);
- we show that the Entropy measure [Shannon, 2001] is a promising function to drive a heuristic search in Linked Data (Chapter 4);
- we present some metrics and methodologies to improve and predict the accuracy of the explanations (Chapter 4 and Chapter 5);
- we detect which factors in the Linked Data structure reveal the strongest relationships between entities, and enclose them in a cost-function to drive a blind search to find entity relationships (Chapter 6);
- we present a methodology to evaluate the process of automatically generating explanations with respect to human experts (Chapter 7);
- we show how to identify the bias that is introduced in the results when dealing with incomplete Linked Data (Chapter 8).

1.6 Structure of the Thesis

The following section gives some additional information about this work: the structure, the publications on which the chapters have been based, and an overview of the specific use-cases that we will be mentioning throughout the book.

1.6.1 Structure

The material of this thesis is distributed in individual parts and chapters as follows.

Part I : Introduction and State of the Art

Besides the extensive introduction of Chapter 1, Chapter 2 provides the background for our thesis. We start by introducing a methodology to answer our first research question (how to formally define an explanation), and then we review the fields of Knowledge Discovery, Graph Theory and Semantic Web to identify which techniques can be chosen to answer our research questions. The literature review aims at revealing strengths and weaknesses of the current works, but also helps us to identify our research space and to highlight the novel aspects of our work.

Part II : Looking for Pattern Explanations in the Web of Data
This part provides the details of our framework. In Chapter 3, we show how we can use the background knowledge extracted from Linked Data in a process that produces hypotheses (i.e. candidate explanations) for patterns, and present the preliminary studies focused on manually building such background knowledge. In Chapter 4, we describe how we can automatically obtain such background knowledge and create meaningful candidate explanations. In Chapter 5, we present a Neural-Network approach that improves the generated hypotheses by combining them. In Chapter 6, we finally show how to automatically validate the hypotheses and assess them as explanations, by identifying the Linked Data relation that correlates them with the pattern.

Part III : Evaluation and Discussion
The final section presents the evaluation study that we have conducted with the users and the final discussions. In Chapter 7, we show the methodology that we chose to evaluate Dedalo and the details of the user study. Chapter 8 closes this thesis with some final remarks, including discussion of the results achieved by our approach, some limitations and future directions.

1.6.2 Publications

The chapters mentioned above are based on the following publications.

Chapter 2

▷ Tiddi, I., d'Aquin, M. and Motta, E. (2015) *An Ontology Design Pattern to Define Explanations*, The 8th International Conference on Knowledge Capture (K-CAP 2015), Palisades, New York, USA

Chapter 3

▷ Tiddi, I. (2013) *Explaining data patterns using background knowledge from Linked Data*, ISWC 2013 Doctoral Consortium, Sydney, Australia, Lara Aroyo and Natasha Noy

▷ Tiddi, I., d'Aquin, M. and Motta, E. (2013) *Explaining Clusters with Inductive Logic Programming and Linked Data*, Poster at ISWC 2013, Sydney, Australia, Eva Blomqvist and Tudor Groza

Chapter 4

- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2014) *Dedalo: looking for Clusters' Explanations in a Labyrinth of Linked Data*, 11th Extended Semantic Web Conference, ESWC 2014, Crete (*Best Student Paper Nominee*)
- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2014) *Walking Linked Data: A graph traversal approach to explain clusters*, Workshop: Fifth International Workshop on Consuming Linked Data (COLD2014) at International Semantic Web Conference 2014 (ISWC 2014), Riva del Garda, Italy (*Best Paper Award*)
- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2015) *Using Linked Data Traversal to Label Academic Communities*, Workshop: SAVE-SD 2015 at 24th International World Wide Web Conference (WWW2015), Florence
- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2015) *Data Patterns explained with Linked Data*, Demo at European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2015), Porto, Portugal

Chapter 5

- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2014) *Using Neural Networks to aggregate Linked Data rules*, 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW2014), Linköping, Sweden (*Best Paper Nominee*)

Chapter 6

- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2016) *Learning to Assess Linked Data Relationships Using Genetic Programming*, The 15th International Semantic Web Conference (ISWC2016), Kobe, Japan

Chapter 7

- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2015) *Interpreting knowledge discovery patterns using Linked Data*, Journal of Web Semantics (*under review*)

Chapter 8

- ▷ Tiddi, I., d'Aquin, M. and Motta, E. (2014) *Quantifying the bias in data links*, 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW2014), Linköping, Sweden

1.6.3 Datasets and Use-cases

For the purpose of clarity, we briefly introduce the datasets and use-cases that we will exploit in the next chapters. With this, it is our intention to help the reader in following the forthcoming narration, where several datasets will be used alternatively. Moreover, with a full picture of the data in hand, we believe that it will be easier to show how our approach is indeed applicable to contexts and domains of various natures.

We gave the use-cases an identifier and below provide some minimal information about them, especially which kind of explanation we have been looking for. More details such as how the datasets were formed are to be found in Appendix A.1.

#Red – *The Reading Experiences*. The use-case is built on some data from the *Reading Experience Database*[3] ontology, a record of people's reading experiences, including metadata regarding the reader, author, and book involved in a reader's experience as well as its date and location. The dataset consists in almost 400 people clustered in 10 groups according to the topics of the books they have read. In this scenario, we are interested in explaining why people read similar things, e.g. they might have some common features such as gender, religion or nationality which make them being interested into the same topics. The dataset has been used for the preliminary approach of Chapter 3.

#KMi.A – *The Knowledge Media Institute co-authorships*. This is a dataset of ca. 100 researchers from the Knowledge Media Institute (KMi) that have been clustered in 6 groups according to the papers they have co-authored. In this very restricted and well-understood scenario, we are interested in knowing what makes researchers being grouped together, e.g. it is likely that people who work on the same research topics are in the same cluster. The dataset has been used for the experiments of Chapter 4 and 5.

#KMi.P – *The Knowledge Media Institute publications*. Similarly, this dataset presents research papers from the KMi department, clustered according to the words in their abstracts. In this scenario, we are imagining that the reason for papers being together is because they are about the same topic. As above, the dataset is used in Chapter 4 and 5.

#KMi.H – *The Book Borrowing observations*. In this dataset, books borrowed by university students have been clustered according to the Faculty the students belong to. A likely explanation for the grouping of the books might be that they are related to the same topic(s). This is a relatively realistic and large use-case (close to 7,000 items) used as well in Chapter 3, 4 and 5, which we have used to explore Linked Data knowledge from different datasets.

[3] http://www.open.ac.uk/Arts/RED/index.html

#Lit – *World Literacy Rate observations*. The dataset consists in the set of world countries partitioned in 3 groups according to the gender literacy rate (i.e. where men are more educated than women, where women are more educated than men, and where the education average is equal). In this scenario, we look into explaining what countries of a same group do have in common. For instance, we can hypothesise that the countries with a low female education rate are the ones with the lowest human development. Experiments over this dataset are mostly in Chapter 4.

#OU.P – *The Open University publications*. Similarly to #KMi.P but on a larger scale, it consists in a dataset of 1,200 English words extracted from around 17,000 Open University paper abstracts. Words have been clustered into 30 communities according to their semantic similarity. In this scenario, we are interested in explaining why words happen to be together, and this reason, which is likely to be "they consists in words under the same topic", can also be used to automatically label research communities. Results for #OU.P are presented in Chapter 4.

#Tre – *The Google Trends dataset*. This dataset consists in 13 cases of Google Trends data[4], graphically showing how often a particular term (a word, or a group of words) has been searched on the Web in the last 10 years. In this scenario we have only two groups, the moments in which the term was popular (high points on the timeline) and the moments in which it was not (low points). What we are looking into discovering is what makes a term increasing its popularity with a specific regularity. Results for this dataset are partly shown Chapter 4 and 6, but more extensively in Chapter 7.

#OU.S – *The Student Enrolment dataset*. This dataset consists in the Open University students clustered according to the UK district they come from. Here, it turns out that in certain regions, some faculties attract more students than others, and we are looking into identifying the reason for that, possibly an eco-demographic explanation. The dataset is used in Chapter 8.

[4] http://www.google.com/trends/

Chapter 2

State of the Art

This State of the Art chapter, presented in a slightly unconventional way, is articulated around two main axes. Since it is our aim to develop a system that generates explanations, we first study on a conceptual level what is an explanation and which are its main components, as an answer to our first resection question (Section 1.3.1). Once we understand that, we can define the explanation model upon which we can build our system. Therefore, Section 2.1 is dedicated to surveying how works in Cognitive Science have defined an explanation, and successively at presenting the derived model. In the second part, we focus more practically on understanding which practices and techniques we need to include in our system, and therefore review the fields that we consider relevant to our research. We first introduce our research context by giving the fundamental notions of Knowledge Discovery, Graph Theory and Web of Data (Section 2.2); we then explore the resources of the Web of Data and the methods to access them (Section 2.3); and we give an overview of how knowledge from structured data can be discovered (Section 2.4). Finally, we discuss the limits of existing approaches and identify which techniques we can exploit in our work (Section 2.5).

2.1 A Cognitive Science Perspective on Explanations

Let us begin with a practical example. What we want to understand is why people search for the term “A Song of Ice and Fire” more at specific times, as in Figure 2.1.

If we are familiar with the context, one of the plausible explanations can be that people search more for “A Song of Ice and Fire” when a new season of the TV series Game of Thrones is released. Nevertheless, without the appropriate background knowledge (for instance, knowing what “A Song of Ice and Fire” or “Game of Thrones” are), one has no way to declare that the stated explanation is correct. This example makes us understand that

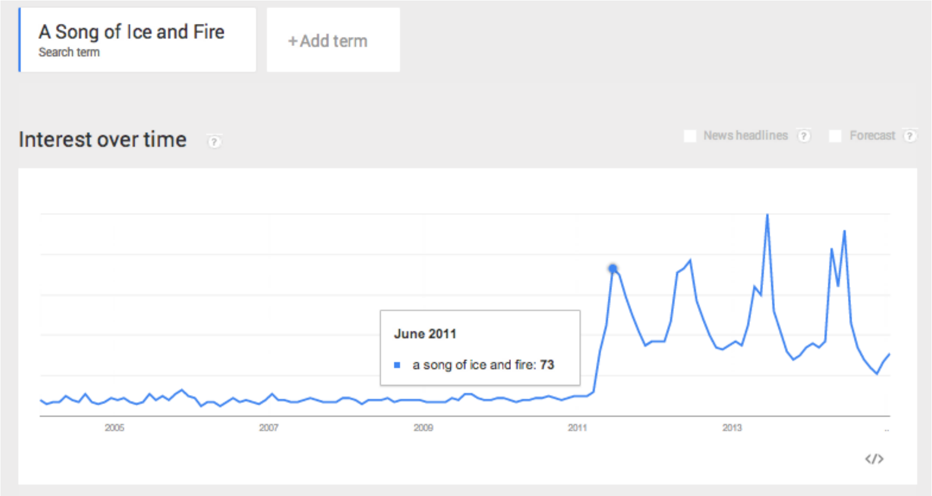


Figure 2.1 Google Trends result for the term “A Song of Ice and Fire”.

“explaining” does not only mean putting two events in a correlation (X co-occurred with Y), but also revealing their causal dependency based on the context in which they occur (X happened because of Y in some context C).

The approach we aim to create has therefore to make sure that the automatically generated explanations involve the right components, but identifying them can be challenging. In order to find them, we studied the areas that most deal with the organisation and understanding of knowledge, commonly grouped under the term of Cognitive Science.

2.1.1 Characterisations of Explanations

Cognitive Science is defined by [Gardner, 2008] as the science of understanding the nature of the human mind. In this work, dated 1985, Cognitive Science is presented as a field involving a series of disciplines interacting in such a way that they can be represented as a “cognitive hexagon” (Figure 2.2).

Interestingly, besides the pragmatic differences that those disciplines present, the way their experts see explanations is structurally the same. More precisely, an explanation is composed by the same four components (whose names change according to the discipline), which can be linguistically represented[1] as:

“When an event △ happens, then, due to a given set of circumstances ★, the event ▲ will also occur because of a given law ■”

[1] The original definition appears in [Maaløe, 2007]

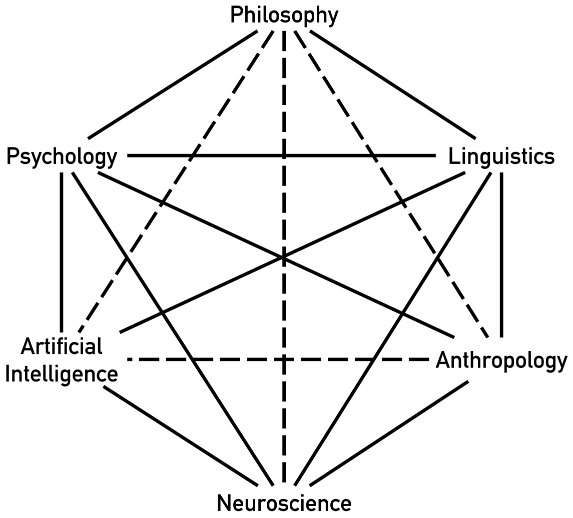


Figure 2.2 The cognitive hexagon, as in [Gardner, 2008]. Dotted lines are weaker connections between disciplines.

The rest of the section provides an overview of how explanations have been defined in these disciplines. It is worth mentioning that the structure of the survey presented hereafter is an organisational choice of our methodology, but many of the cited works, especially the contemporary ones, span over multiple areas. Additionally, we took into consideration Computer Science and Sociology rather than Artificial Intelligence and Anthropology, since they present a more direct connexion with the notion of explanation. Finally, we adopt the common nomenclature, i.e. the event ▲ to be explained is referred to as *explanandum/-nda* ("that which is explained"), and the premises to the explanandum, i.e. the prior events △ that make it happen, are defined as *explanans/-tia* ("that which does the explaining").

Explanations in Philosophy

Planets are near △, what is near does not twinkle ■; therefore, planets do not twinkle ▲ (context: ★: planets) – Aristotle

In Philosophy, defined as the "discipline of asking questions and checking answers" in [Gardner, 2008], an explanation is considered as the process where a set of initial elements (an event and some initial conditions) are described, deduced or put into a relation with an output phenomenon according to a set of (empirical or metaphysical) laws.

The works of [Ruben, 1990; Strevens, 2006; Psillos, 2007] provide interesting surveys on how explanations have been defined by intellectuals from the ancient times to contemporary days. Authors report that the first discussions on what an explanation is already appear among the Greek intellectuals. Thucydides[2] defines explanations as a process where facts (or "indisputable data", [ibid.]), are observed, evaluated based on our knowledge of the human nature and then compared in order to reach generalised principles of why some events occur. For Plato[3], an explanation is an expression of knowledge using the *logos*, and is composed by the Forms, consisting in the abstractions of the entities we know ("the unchanging and unseen world", [ibid.]) and the Facts, i.e. occurrences or states of affairs, which are the changing world we are used to see. Aristotle[4] sees explanations as a deductive process aiming at finding out the cause of why something happened, and this cause can be either a thing's matter, its form, its end, or its change-initiator (Aristotle's *4-cases doctrine*, as called later).

In the modern age of determinism, causality becomes prominent in characterising explanations. To know what caused an event means to know why it happened, which in turn means understanding the universe determined by natural laws. Descartes and Leibniz describe explanations as the process of demonstrating the mechanical interactions between God ("the primary efficient cause of all things"), the world things (the secondary causes), the laws of nature, and some initial conditions. Newton rejects the God component and puts the laws of nature as playing the central role. For him, explanations are the deductive process of finding the most general principles (the fundamental laws of nature) that account for the natural phenomena. The Empiricists then reject explanations that cannot be traced directly to experience. According to Hume, all causal knowledge stems from experience, and causation is just a regular succession of facts to be discovered by experience. Kant, reintroducing the metaphysical component into explanations, believes that the knowledge starts with experience, but does not arise from it: it is shaped by the categories of the understanding and the forms of pure intuition (space and time). John Stuart Mill rejects this view and defines explanations as only based on the laws of nature and on deduction. A fact is explained by deducing its cause – that is, by stating the laws of which it is an instance.

In closer times it is Carl Hempel, who revisits and extends Mill's work, which starts the contemporary discussions around explanations. Explaining an event is to show how this event would have been expected to happen, taking into account the laws that govern its occurrence, as well as certain initial conditions. Formally speaking, a singular explanandum E is explained if and only if a description of E is the conclusion of a valid deductive argument, whose explanantia involve essentially a set C of initial or antecedent conditions and a law-like statement L [Hempel and Oppenheim, 1948]. L can be either statistical (in inductive-statistical explanations) or laws of nature (in deductive-nomological explanations).

[2] cf. *History of the Peloponnesian War*, 4th cent. BC.

[3] cf. *Phedrus* and *Theaetetus*, 370 BC ca.

[4] cf. *Posterior Analytics*, 350 BC ca.

Salmon [Salmon, 1990] further extends the inductive-statistical explanation model proposing the statistical-relevance model, where explanations have to capture the dependencies between the explanandum and the explanans statistically, and the causal model, which sees explanations as mechanical processes of interacting components. An interactive view is also adopted by the Interventionists [Woodward, 2003], for which explanations are the process of manipulating the explanandum within a space of alternative possibilities to see if a relation holds when various other conditions change. The Unificationism approach of Friedman and Kitcher [Friedman, 1974; Kitcher, 1981] affirms that explaining a phenomenon is to see it as an instance of a broad pattern of similar phenomena. According to Friedman, explanations are based on both the scientific understanding of the world, but also on some assumptions (i.e. regularities in nature), which are the basic "unifiers". Similarly, Kitcher believes that a number of explanatory patterns (or schemata) need to be unified in order to explain a fact.

Explanations in Psychology

According to psychological theories ■, *Borderline Personality disorder* △ *can be explained by a childhood trauma* ▲ (context ★: a person's behaviour)

Psychology is the discipline that tries to understand the human (and animal) cognitive processes, i.e. the way they represent and process information [Wilson and Keil, 2001]. Explanations in this area are "an attempt to understand phenomena related to intelligent behaviours" [Bechtel and Wright, 2009] or, in other words, they consist in identifying which entities and/or activities produce regular changes from some start conditions to a termination condition.

Many works of psychology criticise the traditional philosophical-scientific approaches to explanation since they are too much focused on causality and subsumption under laws (deduction) [Hutten, 1956; Cummins, 2000; Bechtel and Wright, 2009]. Instead, psychological explanations accept over-determinism and dynamism, i.e. the antecedent conditions that explain a phenomenon can be more than one and do change over time. The explanantia are not laws of nature but human capacities that psychological methodologies have to discover and confirm.

In the literature, psychological explanations have been defined as "models" providing a theory and a law, by means of which it is possible to describe a phenomenon and relate it to other similar phenomena, and possibly to allow a prediction, intended as a future occurrence of the phenomenon [Hutten, 1956; Cummins, 2000]. The authors of [Marraffa and Paternoster, 2012] have highlighted how the mechanical aspect within the explanation process plays a central role into recent psychology work. Phenomena are often explained by decomposing them into operations localised in parts (components) of the mechanism. Mechanisms have spatial and temporal organisations that explain how entities are organised into levels and carry out their activities. In other words, explaining a phenomenon means revealing its internal structure by defining the organisational levels that are responsible of producing it, then identifying how those levels relate to other levels, and finally building a model to explain similar phenomena. This idea of interactive levels can be found in the major contemporary psychology trends, i.e. the connectionist (bottom-up) approach [Darden and Maull, 1977; Marr and Vision, 1982; Bechtel and Wright, 2009], that first specifies the system's components and then produces the explanation's model, and in the symbolic (top-down) one [Smolensky, 1988], that first identifies the phenomenon to be explained and then describes it according to the external symbols (syntactico-semantic rules or mathematical laws) manipulated by the humans.

Explanations in Neuroscience

Neuroimagining ■ *has proven that humans can do math calculations* ▲ *because some neurons actively respond to quantities* △ (context ★: human capacities)

Neuroscience is the general study of the brain and endocrine system [Friedenberg and Silverman, 2011]. Neuroscience explanations reject the concept of levels, as well as the synthetic a priori frameworks. Mentality is the central pivot between interpretation and rationality. Neuroscientists aim at understanding brains, i.e. providing a description of mental events at the implementational level [ibid.]. This requires assuming that beliefs are true and rational, which is in contrast with many philosophical views.

Explaining for neuroscience means fitting a mental phenomenon into a broadly rational pattern. Explanations are answers to the *what-if-things-had-been-different* questions: they describe what will happen to some variables (effects) when one manipulates or intervenes on some others (causes) [Davidson, 1990; Campbell, 2008; Woodward, 2008]. They can be "upper level explanations", when causes are macroscopic variables or environmental factors, and "lower level explanations", if causes are more fine-grained (neural, genetic, or biochemical) mechanisms. Also, explanations do not require any law or sufficient condition, and are simply generalisations that describe relationships stable under some appropriate range of interventions. Therefore, a typical approach to explanations consists in detecting how the variables respond to the hypothetical experiments in which some interventions occur, then establishing if other variables under the same intervention respond in the same way. The relevance of explanations is assessed by their stability, i.e. some relationships are more stable than others under some interventions or under some background conditions.

Explanations in Computer Science

Three authors submit three different papers to the conference C; however, only the first one is accepted ■*. The first author attends the conference* △*. Therefore, we induce that to be able to attend the conference C, one has to have both submitted a paper and have it accepted* ▲. (context ★: conference submission)

Computer Science is the science dealing with computer programs as a medium to perform human operations. Artificial Intelligence (AI) considered explanations in terms of inference and reasoning, and programs were the means to decipher connections between events, to make predictions and to represent some order in the universe. The process of generating explanations fits the two main reasoning patterns, abduction and induction, that Peirce describes in its *Lectures on Pragmatism* of 1903. According to Peirce, induction is the inference of a rule (major premise) starting from a case (minor premise) and its result (conclusion), while abduction is the inference of the case from a rule and a result [Fann, 1970]. Those models have been often put under the same umbrella-term of non-deductive, a posteriori inference, and Peirce himself in its later period saw abduction as only one phase of the process, in which hypotheses and ideas are generated, but then need to be evaluated by induction or deduction [Fann, 1970; Russell *et al.*, 1995; Schurz, 2008]. Moving on from such discussion, AI approaches to generate explanations (see [Russell *et al.*, 1995; Rothleder, 1996; Páez, 2009] for extensive surveys) show that the two inferences can be similarly modelled, so that one body of initial knowledge (divided into observations and prior knowledge) is mapped to another (the new knowledge), under constraints of certain criteria, which reveal their connection and confirm the hypothesis. The explanation process consists of first analysing the observations, then defining tentative hypotheses and finally proving them empirically.

Similarly to explanations in psychology, Artificial Intelligence explanations do not assume that a truth has been previously accepted in the prior knowledge. Nevertheless, in order to produce new knowledge, explanations need to be scientifically and empirically evaluated. Plausibility, usefulness and desirability are considered too subjective and are replaced by some formal constraints that the process has to respect – namely, efficiency (how easily it solves the problem), non-deducibility (it should introduce new knowledge), consistency (it must be consistent with what it is already known) and non-emptiness (it must introduce new knowledge). Some works attempting to merge the views from philosophy and Artificial Intelligence are surveyed in [Páez, 2009].

Explanations also play a central role in other areas of Computer Science such as Knowledge Management. Knowledge Management systems are used for intelligent tutoring or decision-support, and explanations provide insight into how their reasoning is accomplished. Users are presented solutions and explanations of a problem to be solved relying on some domain knowledge (see referenced works in [Southwick, 1991; Wooley, 1998; Alavi and Leidner, 1999]). This domain knowledge, related to facts, procedures or events, is defined as "tacit" because it is already in the individuals' mind. An inflow of new stimuli then triggers a cognitive process that produces explanations, which are the explicit knowledge communicated in symbolic forms. Explanations can be intended as "giving something a meaning or an interpretation to, or to make it understandable" [Wooley, 1998].

Explanations in Sociology

In Italy, it is hard for young people to live on their own ■. *Since the job opportunities are low* △, *the young people tend to stay at their parents' until later ages* ▲ (context ★: Italy's social world)

Sociology is the discipline that focuses on the links between the human's cognitive processes and his sociocultural world [Calhoun, 2002]. Weber and Durkheim, founders of the major two trends in sociology, consider the act of explaining as giving a meaning and justifying social facts, intended as observable regularities in the behaviour of members of a society [Durkheim, 2014; Weber and Heydebrand, 1994]. In their perspective, explanation is a rational (empirical) observation of social behaviours. The author of [Carter, 1998] rather prefers the idea of "comparing events", and defines an explanation as a comparison stating a similarity, a relationship, a connection between social phenomena. For [Brent *et al.*, 2000], explanation is also a form of reasoning for prediction, and it has to take into account that the structure of the social world and the social behaviours constantly change in time and space.

The general agreement of works in sociology is that explanations, based on social facts, have more pragmatic requirements with respect to scientific explanations. Formal constraints, such as mathematical formalisation or empirical falsifiability, leave space to descriptions and approximations of complex structures of the social world [Clayton, 1989; Gorski, 2004]. Explanations might be weaker or stronger depending on the set of social impacts that the rules express. For example, they can be proverbs (everyday rules to express what one knows), social theorems (rules summing up the human experience) or paradoxes [Brent *et al.*, 2000; Maaløe, 2007]. Some have also distinguished between social-anthropological and religious explanations, where "religious" means giving sense to phenomena using symbols of a tradition. Explanations are declared to be valid if a phenomenon can also appear due to other circumstances, and logically exhaustive if they are generalisable to all logically possible combinations of conditions encountered [Brent *et al.*, 2000; Maaløe, 2007].

Explanations in Linguistics

The expression *THE MY BOOK *is not allowed in English* ▲ *because the English grammar* ■ *does not allow double determiners, and both* THE *and* MY *are determiners* △ (context ★: English as a language)

Linguistics is the discipline that provides theories dealing with the nature of the human language. Linguistic explanations are descriptions focused on finding what rules of a language explain a linguistic fact [Hawkins, 1988]. Linguists do not aim at describing the processes but rather at explaining how language acquisition is possible, by identifying the grammatical patterns that occur in the human languages. Useful surveys on explanations in linguistics are to be found in [Hawkins, 1988; Culicover and Jackendoff, 2005; Haspelmath *et al.*, 2005; Itkonen, 2013].

A central assumption in linguistic theories is that a language is a set of elements (words, morphemes, phonemes) that meet a set of wellformedness conditions, called language universals. Language universals are (language-specific) grammatical characteristics that constitute the grammar of the language. The major approaches – the generative linguistics, which follows Chomsky's theories; and the functional linguistics, which is based on Greenberg's school – have different views on the roles and functions of the components of linguistic explanations. In the former case, only the generalisation of an explanation is regarded as interesting, since it shows that linguistics constructions are derivable from the general regularities of a language (*descriptive adequacy*), but some of those regularities are innate and have to be accepted a priori (*explanatory adequacy*). Searching for an explanation takes the form of new constraints on the descriptive frameworks, which are able to describe the languages. This claim to innateness and the view of explanations as constrained descriptions cannot be accepted by functional linguistics, for which external elements and realities are essential to understand linguistic facts. Functionalism is more interested in language regularities than in descriptive frameworks. The rejection of a priori language universals does not mean that functionalists think that nothing in language structure is arbitrary, but rather than it is worth attempting to explain, wherever possible, language facts using external non-linguistic factors [Haspelmath *et al.*, 2005].

2.1.2 The Explanation Ontology

The survey presented above shows how explanations are structured in a similar way in Cognitive Science. Explanations include two events, an anterior event △ and a posterior event ▲, a context ★ relating them and a law ■ that governs their co-occurrence. Such a structure can be formally represented as a small ontology, shown in Figure 2.3, which we call the

Explanation Ontology. This ontology, representing the main components of an explanation as they should be produced by our automatic process, is described below.

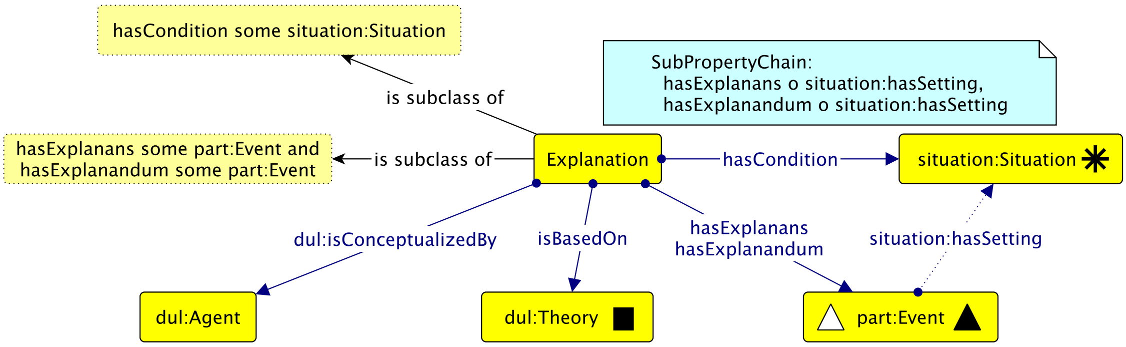


Figure 2.3 The Explanation Ontology that we propose to model explanations.

(a) *Explanation* is the explanation to be produced. The class has a few constraints represented as OWL class restrictions. In order to be complete, an explanation: (i) needs at least one antecedent event △, expressed as *hasExplanans* `some` *part:Event*; (ii) requires a posterior event ▲, expressed as *hasExplanandum* `some` *part:Event* and (iii) has to happen in a context that relates the two events, expressed as *hasCondition* `some` *situation:Situation*.

(b) the class *part:Event* is used to represent the antecedent event (the explanans △) and the posterior event (the explanandum ▲) involved in the explanation. To promote reusability, we reuse the *part:Event* class from the Participation[5] ontology design pattern [Gangemi and Presutti, 2009], which can be used to represent any binary relation between objects and events.

(c) The *situation:Situation* class from the Situation[6] pattern, designed to "represent contexts or situations, and the things that have something in common, or are associated" [Gangemi and Mika, 2003], is used to represent the context ★ under which the two events occur. The OWL axiom in the blue box above the object property *hasCondition* shows what we can infer about the situation in which the explanation is happening. If there exists a situation in which an explanation is contextualised (through *hasCondition*), then both the explanans and the explanandum do share this same situation (through *situation:hasSetting*), therefore they are in the same context.

(d) The *dul:Theory* class from the DOLCE+DnS Ultralite ontology[7] is used to represent the governing law ■. We use the term theory with the idea that it best embraces those

[5] http://ontologydesignpatterns.org/cp/owl/participation.owl
[6] http://ontologydesignpatterns.org/cp/owl/situation.owl
[7] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

concepts defined in Cognitive Science (laws, human capacities, human experiences, universals, and so on), that consist of something having a binding force on some events under analysis.

(e) The *dul:Agent* represents the entity producing the explanations. Since there is no agreement on how to "call" the act of explaining (e.g. deducing, inducing, inferring, describing have all been used in the literature), we have kept the generic object property label as *dul:isConceptualizedBy*.

2.2 Research Context

Once understood what an explanation is and designed a model for it, we can start reviewing works in the disciplines that most contribute to our research, in order to identify which techniques we can use, and to which extent.

Our research starts at the crossing between Knowledge Discovery and the Web of Data, but also spans to a certain extent the Graph Theory field and, as already pointed out, to Cognitive Science. Since it might not be straightforward to see how all these disciplines contribute to our work, Figure 2.4 gives a graphical representation of our research context.

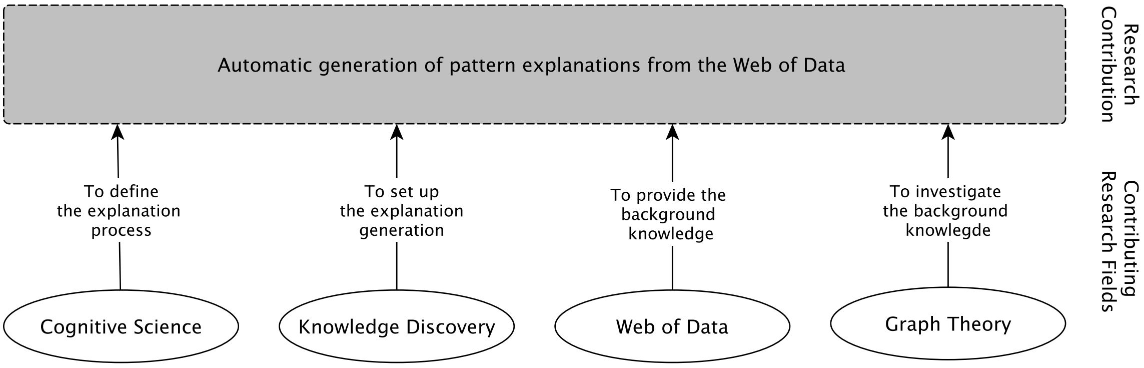

Automatic generation of pattern explanations from the Web of Data
Research Contribution
To define the explanation process
To set up the explanation generation
To provide the background knowledge
To investigate the background knowlegde
Contributing Research Fields
Cognitive Science
Knowledge Discovery
Web of Data
Graph Theory

Figure 2.4 Contributions that the analysed disciplines can bring to our research.

It is important to emphasise that these areas are not the ones we are contributing to, but rather the research fields whose techniques and overviews have to be investigated to develop our work. While we already presented how Cognitive Science helps us in clarifying what to we need to know when talking about explanations, we believe that:

(a) **Knowledge Discovery** has to be explored to find which process or technique is the most suitable to generate explanations;

(b) data and standards for structured data management (storing, querying, retrieving, traversing and revealing) from the **Web of Data** have to be explored so that we know how to access the background knowledge to produce explanations;

(c) **Graph Theory**, which provides the fundamental notions and advanced algorithms for graph management, can be beneficial to an efficient exploration of the Web of Data.

In the rest of this section, we provide the fundamental notions in those three areas.

2.2.1 The Knowledge Discovery Process

The Knowledge Discovery process is a framework aiming at developing methods and techniques to detect hidden patterns and regularities in large amounts of data [Fayyad *et al.*, 1996]. The process, shown in Figure 2.5, includes a set of sequential activities with their specific inputs and outputs, and which constitute themselves different research areas with specific problems and challenges. These activities, iteratively performed according to the domain problem, are articulated in (i) data pre-processing, (ii) data mining and (iii) data post-processing. Multiple iterations presuppose that the discovered knowledge and data selection improves at each iteration.

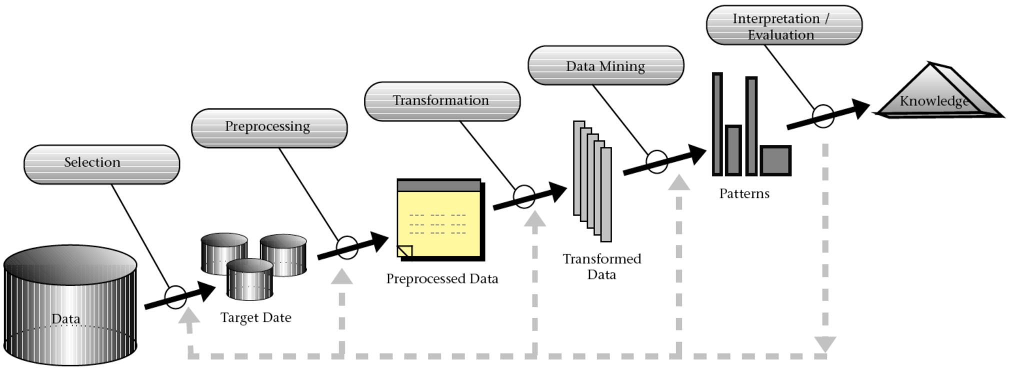


Figure 2.5 Chained activities in the Knowledge Discovery process as first appeared in [Fayyad *et al.*, 1996].

Data pre-processing. Data pre-processing is a set of steps in which data are selected, cleaned, transformed, organised, and structured so that they can be processed. These operations include removing noise and outliers, detecting missing or unknown information and deciding how to deal with it. The idea behind pre-processing is that results produced by a data mining algorithm can only be as good as the given input data. The background knowledge is assumed to be part of the input dataset.

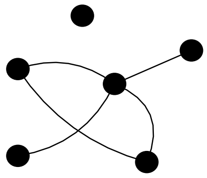
Data mining. The data mining process takes as input the pre-processed data and produce as output some information, called nuggets or patterns, defined as "statements describing an interesting relationship among a subset of the data" [Fayyad *et al.*, 1996]. At this stage, one has to choose which tasks to apply to the data (classification, summarisation, regression, clustering, and so on), has to decide which Machine Learning algorithm and parameters are appropriate, and finally has to learn the model that will extract interesting patterns, which may be represented as clusters, association rules, sequences or dependencies. Besides the technical choices, problems related to both the quantity (the number of patterns can be extremely large) and to the quality (not all of them are interesting) of the patterns have to be tackled.

Data post-processing. The last step, also called pattern interpretation, consists in understanding, validating and refining the discovered patterns so that a new iteration of the KD process can start (or the results can be presented to the user). As the interestingness of the patterns is not guaranteed, steps to be included in this process are pruning, summarising, grouping, and visualising. Also, the interpretation can be difficult since it might require knowledge from different domains. It would therefore require the human analyst to be sufficiently familiar with all of these domains.

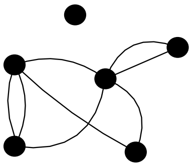
2.2.2 Graph Terminology and Fundamentals

Graphs are among the most studied mathematical data structures, and have found application in a wide range of disciplines – to name a few, Topological Mathematics, Computer Science, Electrical Engineering, Biology, Linguistics and Sociology [West *et al.*, 2001; Washio and Motoda, 2003; Giblin, 2013].

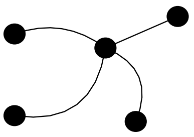
The basic notions on graphs, which are relevant to our research, are reviewed below.



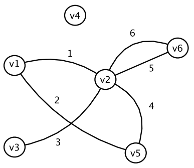
(a) The example shows the most generic type of **graph** $\mathcal{G}$, i.e. an ordered triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Psi)$ consisting of a non-empty set $\mathcal{V}$ of *vertices* (or *nodes*), a set of *arcs* (or *edges*) $\mathcal{E}$, and the incidence function Ψ which associates with an edge e an unordered pair of vertices such that $\Psi_e = (v, w)$, which are said to be joined by e [Bondy and Murty, 1976].



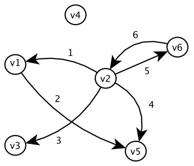
(b) A graph allowing loops, i.e. there exists an edge $e = (v, w)$ so that $v = w$, is called **multigraph**. The graph in the example has 2 loops.



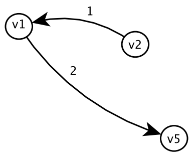
(c) A **subgraph** $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \Psi')$ is a portion of a graph, where $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{E}' \subseteq \mathcal{E}$ and $\Psi' \subseteq \Psi$. The graph in the example is a subgraph of both (a) and (b).



(d) A **labeled graph** is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Psi, \mathcal{A})$ where $\mathcal{A}$ is a set of labels that are assigned to vertices and edges through the labelling functions α and β. When these labels are numbers, as in the example, they are called weights, and the graph is called a weighted graph.



(e) A **directed graph** (or digraph) is an ordered triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Psi)$ where the incidence function Ψ associates with each arc an ordered pair of $v, w \in V$. In that case, we call v the *source* and w the *end* (or tail).



(f) A **path** (or walk) is a finite sequence $p = \{v_0, e_1, v_1 \ldots e_l, v_l\}$ of alternate vertices and edges, such that, for $1 \leq i \leq l$, the end of e_i are v_{i-1} and v_i. l defines the length of p. Two vertices $v, w \in V$ are said to be connected if there exists at least one path from v to w. The example shows a path of length 2 from v_2 to v_5 in (e).

Isomorphism

Two graphs $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic if there exists a bijective function $f \colon \mathcal{V} \to \mathcal{V}'$ between the vertex sets $\mathcal{V}$ and $\mathcal{V}'$ such that any two vertices v and w of $\mathcal{G}$ are adjacent in $\mathcal{G}$ if and only if $f(v)$ and $f(w)$ are adjacent in $\mathcal{G}'$ (also called "edge-preserving bijection") [Chartrand, 2006]. In the case of labeled graphs, isomorphism is both edge-preserving and label-preserving.

Topology The **degree** $d(v)$ of a vertex $v \in \mathcal{V}$ is the number of edges incident to v, and is referred to as the vertex neighbourhood. If the graph is directed, the $d(v)$ is the sum of the number of incoming edges $i(v)$ and the number of outgoing edges $o(v)$. The **density** of a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ is the ratio of edges in $\mathcal{G}'$ to the number of all possible edges among the vertices in $\mathcal{G}$.

Graph Search Algorithms

An important class of graph algorithms is the one aiming at finding the shortest path between two vertices, where shortest is intended as the "cheapest path", i.e. the one with the lowest cost in term of cumulative edge weights. Strategies differ in informativeness (if they use preference strategies, called *heuristics*, or not), completeness (if they are guaranteed to find a solution, if any), time complexity (how long do they take), space complexity (how much memory they require) and optimality (if there are more solutions, how accurate is the one with the lowest cost found first).

Uninformed search. The uninformed (also called blind or brute-force) search explores a graph without using any domain specific knowledge.

The simplest and most common strategy is the Breadth-First Search (BFS), which iteratively expands the shallowest unexpanded node, so that all the nodes at depth d are expanded before the nodes at depth $d+1$. The new nodes are put in a queue after all the previously found nodes. BFS is complete if the set of vertices is finite, and it is guaranteed to be optimal assuming the cost of the edges are all the same (because it will always find the shallowest one). However, time and space complexity are exponential with respect to the size of the graph.

Uniform Cost Search (UCS) modifies BFS by keeping nodes in a priority queue according to their cost $g(n)$ so that the one with lower cost is the first to be expanded. UCS is complete, optimal, but inefficient. UCS is logically equivalent to the very well-known Dijkstra algorithm (proposed independently by both [Dijkstra, 1959; Whiting and Hillier, 1960]). They expand exactly the same nodes in exactly the same order. However, Dijkstra's solution inserts all the nodes in the queue, which makes it computationally more expensive, while UCS lazily inserts the nodes during the search [Felner, 2011].

Depth-First Search (DFS) aims at expanding the deepest unexpanded node. The new nodes are put always at the front of the queue and, if the search hits a dead end, i.e. the node is not expandable, the search goes back and expands a node at a shallower level. It is generally considered as a better solution for more dense spaces, but not for graphs with infinite depths. For this reason, DSF is not complete, and is not optimal, since the path that is found is not guaranteed to be the shortest. Space complexity is linear but time complexity is exponential.

The Bi-directional Search (BiS) simultaneously starts from the initial and goal node until the two searches meet in the middle. Bi-directional search has been proven useful in computational savings for those problems where the goal node is known, since it can sensibly reduce the time and space complexity [Pohl, 1970; Dantzig, 1998]. However, it is not guaranteed to be optimal.

Informed Search. The idea behind the informed (or heuristic) search is to use some additional knowledge about the problem as "hints" to help the search method to find the most promising paths.

One of the most popular heuristic search method is the A* algorithm [Hart *et al.*, 1968], as demonstrated by the vast literature related to it [Newell *et al.*, 1959; Doran and Michie, 1966; Quinlan and Hunt, 1968; Kilssen, 1971]. In A*, nodes are queued according to the score assessed by $f(n)$, defined as an estimate of the cost of the path from the start node to the goal node, passing through the node n. $f(n)$ is the sum of the two functions $g(n)$, representing the actual cost from the start to n, and $h(n)$, being the estimate of the cost from n to the goal node. A* is optimal only if $h(n)$ is admissible, i.e. the estimate cost of n is never overestimated (more than its real cost; for instance, when reaching the goal, $h(n)$ must be 0). In the worst case, time complexity is exponential. Also, since A* maintains in a queue all the unexpanded states, the space complexity is the same. Generally, identifying a good heuristic helps in finding optimal solutions in a reasonable time. A second possible alternative is the beam search that trims the best j options at each point.

Table 2.2 Search strategies characteristics. d is the depth of the shallowest goal state. m is the maximum depth of the graph.

Strategy	Informed	Complete	Optimal	Time complexity	Space complexity
BFS	✗	✓	✓**	$O(b^d)$	$O(b^d)$
UCS	✗	✓	✓	up to $O(b^d)$	up to $O(b^d)$
DFS	✗	✗	✗	$O(b^m)$	$O(bm)$
BiS	✗	✓	✗	$O(b^{d/2})$	$O(b^{d/2})$
A*	✓	✓	✓	up to $O(b^m)$	up to $O(b^m)$
GS	✓	✗	✗	up to $O(b^m)$	up to $O(b^m)$

**(if costs are equals)

Using the heuristic function alone results in a greedy best-first search (GS) [Miller and Myers, 1985; Ukkonen, 1985; Myers, 1986; Wu *et al.*, 1990]. While it can sometimes vastly reduce computation times compared to other searches, the path optimality is not guaranteed, and it can get stuck in loops. Time and space complexity are exponential.

The presented algorithms and their main characteristics are summarised in Table 2.2.

Spreading Activation

Originated from psychological studies such as [Rumelhart and Norman, 1983], Spreading Activation (SA) is a technique to explore graphs currently adopted in different areas of Computer Science and particularly successful in Information Retrieval (IR) [Crestani, 1997].

The idea behind SA is that a graph (generally called *network* in IR) can be explored using a sequence of iterations until halted by the user or a termination condition. Each iteration consists of one or more*pulses* and a *termination check*. The sequence of actions which composes the pulse is what distinguish SA from other complex models to explore graphs [ibid.]. A pulse is composed of the following three phases:

- pre-adjustment (optional): a control phase to avoid the activation from previous pulses;
- spreading: a phase in which activation passages weave from a node to the ones connected to it;
- post-adjustment phase (optional) : a second control phase as the pre-adjustment;

To avoid the activation spreading all over the network, the processing technique can be constrained using information such as the diverse significance of the edges (e.g. labels or weights) using some form of heuristics or inference rules. This technique is called *Constrained Spreading Activation.*

2.2.3 Historical Overview of the Web of Data

As shown in Figure 2.4, the Web of Data contributes to our research by providing the background knowledge required to produce explanations. We give here a short overview of the basic notions.

Semi-structured Data. When the World Wide Web emerged and was promoted as the prime vehicle for disseminating information [Florescu *et al.*, 1998], the database community started addressing a new challenge: the one of managing data at a large (Web) scale [Gutierrez, 2011]. In its original view, the Web was a collection of documents connected using hyperlinks, represented using the HyperText Markup Language (HTML) [Berners-Lee and Connolly, 1995] and accessed using the Hypertext Transfer Protocol (HTTP) [Fielding *et al.*, 1999], that computers could access and exchange in order to promote interoperability [Berners-Lee, 1996].

Structured Data. The traditional hypertext Web, meant to be mostly human-accessible, revealed to be insufficiently expressive, since it was leaving implicit the semantic relationship between data linked across documents [Bizer *et al.*, 2009b; Rettinger *et al.*, 2012]. The manifesto for a structured and semantically meaningful Web (the "Semantic Web") proposed that machines should have a deeper understanding of the document contents, so that data access, organisation and management could be easier. Specifying the semantics of information on the Web was seen as a significant step towards automated data processing [Berners-Lee *et al.*, 2001].

Linked Data. A set of best practices (so-called *Linked Data principles*) for publishing and connecting structured data on the Web were presented by Tim Berners-Lee in 2006 [Berners-Lee, 2006]:

(1) *Use URIs as names for things*

(2) *Use HTTP URIs so that people can look up those names*

(3) *Provide useful information, using the standards (RDF, SPARQL)*

(4) *Include links to other URIs, so that they can discover more things*

In short, data providers are recommended to identify an entity through HTTP based on its unique identifier (*Uniform Resource Identifier* or URI) [Berners-Lee *et al.*, 1998], which provides an access to a structured representation of the entity. This data should be represented using the *Resource Description Framework* (RDF) model [Klyne and Carroll, 2004], which represents information in the form of triples ⟨subject, predicate, object⟩, and should be accessible using the SPARQL language and protocols [Correndo *et al.*, 2010; Feigenbaum *et al.*, 2013]. Furthermore, data should also be linked to other sources, to facilitate automatic data discovery and reuse.

Ever since these principles were presented, there has been much effort in both academia and industry to publish and connect datasets of multiple formats, sources and domains, so that fragmentary information could be aggregated into a multi-domain, shared knowledge, defined today as the Web of Data.

Open Data. While those principles were design guidelines for data publishers, Tim Berners-Lee introduced in 2010 a 5-star dataset rating model to encourage data owners to adopt Linked Data[8]. The model included an open-licence principle, i.e. the first star is assigned if data are published under open licences such as PDDL[9], ODC-by[10] or CC0[11], while the other four are assigned based on the conformance w.r.t. the Linked Data principles.

Nowadays many initiatives promote Open Data. To name a few, Open Knowledge[12] (formerly the Open Knowledge Foundation) campaigns for creating collaborative networks of Open Data from different domains at local level; the World Bank Open Data Initiative[13] provides free and open access to more than 1,168 datasets describing countries' development according to the wide range of World Bank socio-economical indicators; the US government constantly publishes new datasets within the Data.gov portal[14]; the Datahub[15] contains 13,940 descriptions of data documents, 2,369 of which can be identified as containing Linked Data; and finally, the Linking Open Data (LOD) community, started in 2007 with a dozen of datasets published under Linked Data principles, became in a few years a large space containing hundreds of datasets[16].

2.3 Consuming Knowledge from the Web of Data

The section presents an overview of the most common resources in the Web of Data, and the methods to access them.

2.3.1 Resources

A number of available resources, standards and best practices are nowadays established in the Web of Data community. We grouped them in (i) vocabularies, (ii) cross-domain Linked Data collections and (iii) indexes.

Dataset Vocabularies. Vocabularies help data consumers to have a meaningful view on the datasets they are using. The VoID schema[17] is the standard vocabulary adopted to

[8] http://5stardata.info/en/

[9] http://opendatacommons.org/licenses/pddl/

[10] http://opendatacommons.org/licenses/by/

[11] http://creativecommons.org/publicdomain/zero/1.0/

[12] https://okfn.org/

[13] http://datacatalog.worldbank.org/

[14] https://www.data.gov/

[15] http://datahub.io/

[16] Approximately 1,100 datasets as of October 2014 [Schmachtenberg *et al.*, 2014].

[17] http://www.w3.org/TR/void/

formally describe datasets in the Web of Data. It supports dataset metadata such as homepage, SPARQL endpoint, available protocols, and statistics about the resources (classes and properties, numbers of triples and so on). The VoID description has also been extended in the science domain within the Bio2RDF schema [Callahan *et al.*, 2013]. However, it has been demonstrated that the VoID vocabulary still struggles in being extensively adopted in the Linked Data community [Hogan *et al.*, 2012]. Also, a current open issue is related to the discoverability of datasets adopting such a standard. Other established vocabularies for dataset descriptions include DCAT [Maali *et al.*, 2014], the Prov-O [Lebo *et al.*, 2013] ontology, VOAF [Vandenbussche and Vatant, 2011] and Datanode [Daga *et al.*, 2014]. To overcome the issue of using too many vocabularies, the Linked Open Vocabularies (LOV) initiative[18] was promoted to be an observatory of the Web of Data vocabularies, gathering the freshly proposed ones, showing interconnections, versioning history and maintenance policy.

Cross-domain Linked Data Collections. As previously said, the LOD cloud is a large collection of datasets interlinked across domains. The central hub, as shown in the very well-known LOD diagram[19], is DBpedia [Auer *et al.*, 2007; Bizer *et al.*, 2009a], a RDF dataset of general knowledge extracted from the semi-structured Wikipedia articles. Freebase [Bollacker *et al.*, 2008] was also presented as a dataset of general knowledge, created from user inputs and existing RDF and Microformat datasets. Since DBpedia and Freebase are both automatically extracted, they contain a significant amount of inconsistencies, non-conformant data and syntactically incorrect triples. While attempts to improve DBpedia are constantly proposed (see [Paulheim and Gangemi, 2015] for an updated survey on systematic error identification), Freebase was discontinued by Google in March 2015. To date, the plan is to transfer Freebase to Wikidata [Vrandečić and Krötzsch, 2014] and release a new API to access the Google Knowledge Graph. Europeana [Haslhofer and Isaac, 2011] also provides aggregate data from specific content domains, and allows queries through customised APIs. The LOD cloud was also made available in 2012 in the context of the Billion Triple Challenge (BTC): the dataset consists of 1.4 billion triples, including large RDF datasets, RDFa and Microformats data from various domains.

Linked Data Indexes. Frameworks to explore Linked Data include Sindice [Oren *et al.*, 2008] and Sig.ma [Tummarello *et al.*, 2010], which crawl Linked Data resources (RDF, RDFa and Microformats) and allow access to data with custom user interfaces (note that

[18] http://lov.okfn.org/dataset/lov/

[19] http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

Sindice was discontinued in 2014); the LODCache[20], a collection of more than 50 billion Linked Data triples, which provides a SPARQL endpoint and dereferenceable URIs, but does not make data dumps available; DyLDO [Käfer *et al.*, 2012], a platform to monitor the dynamics of 80,000 Linked Data documents retrieved through URI dereferencing (data dumps are published weekly as N-Quads file); the DataHub portal[21], which gathers and archives data sources and their meta-data schemes to allow data description and discovery of data; and the LOD Laundromat [Beek *et al.*, 2014], which crawls the Linked Data Cloud, and re-publishes any (cleaned) Linked Dataset in a N-Triples or N-Quads format.

2.3.2 Methods

Figure 2.6 presents the different methods to access Web of Data resources.

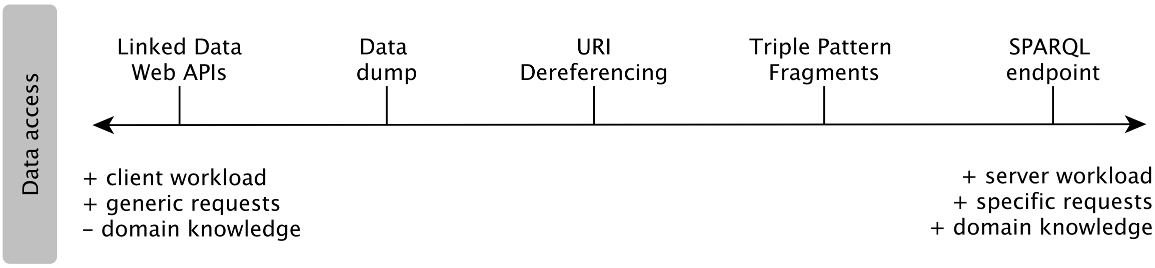


Figure 2.6 Types of data access in the Web of Data.

Typically, client-side applications ask for data that they analyse, visualise or mine to discover new information. Data are provided by remote servers, that clients query (they ask for accessing data stored in there), and that in turn respond with the results once the query is executed. As shown, each type of data access has different specificities: for instance, the workload for the server and the client varies, requests are more specific or generic, which also implies that different amounts of domain knowledge are required in the process.

Web APIs. Web APIs are the most popular approach for publishing and consuming open data on the Web. Based on the REST protocol [Fielding, 2000], they facilitate the interoperability and decoupling between services and applications, and generally offer a more granular access to data.

The combination of Linked Data and Web APIs has recently seen a lot of growing interest in the research community (see the survey of [Pedrinaci and Domingue, 2010], as

[20] http://lod.openlinksw.com/
[21] http://datahub.io

well as the various editions of the SALAD workshop[22]). Approaches allowing to perform CRUD operations on Linked Data resources include the Linked Data API[23] (implemented by ELDA[24] and Open PHACTS [Groth *et al.*, 2014]), The Data Tank[25], Restpark[26], the SPARQL Graph Store Protocol [Ogbuji, 2011] and the Linked Data Platform [Speicher *et al.*, 2014]. However, since custom specifications and formalisms are required to use those APIs, there is no established approach in the area. This problem has been highlighted by [Daga *et al.*, 2015], that introduce the BASIL[27] system as a mediator between SPARQL endpoints and applications: SPARQL queries are stored on the BASIL middleware, which then generates the corresponding API to be consumed through the canonical HTTP GET requests.

Data Dumps. Data dumps consist of a single file representations of a dataset (or part of it) fetched from the Web, which have to be locally deployed in order to speed up the query and search processes. Dumps are generally stored in triple stores, which also offer query interfaces to access data through the SPARQL language.

Virtuoso [Erling and Mikhailov, 2010] and Jena [Grobe, 2009] are currently the most used triple stores. However, studies on data dynamics have shown that Linked Data sources are not appropriate for long-term caching or indexing [Hausenblas *et al.*, 2009; Umbrich *et al.*, 2010; Auer *et al.*, 2012; Dividino *et al.*, 2013, 2014] since they are subject to frequent changes. [Käfer *et al.*, 2013] showed that almost 49.1% of the Linked Data cloud changes over a period of 29 weeks, and [Gottron and Gottron, 2014] also showed that the accuracy of indices built over the Linked Data sources drops by 50% after only 10 weeks. To address this problem, [Dividino *et al.*, 2015] recently proposes and evaluates different strategies for updating Linked Data datasets, by measuring the quality of the local data cache when considering iterative updates over a longer period of time, while [Nishioka and Scherp, 2015] uses K-means to classify static and dynamic entities over Linked Data, by identifying some temporal patterns with substantial changes. Data dumps also require time-consuming maintenance, involving manual support and expensive infrastructures. Datasets have to be fully loaded into a triple store, even if the amount of data required by the query is actually a much smaller part. Moreover, SPARQL endpoints are hosted at different physical locations, violating the data discovery principle for Linked Data.

[22] Services and Applications over Linked APIs and data – http://salad2015.linked.services/
[23] https://code.google.com/p/linked-data-api/
[24] http://www.epimorphics.com/web/tools/elda.html
[25] http://docs.thedatatank.com/4.3/spectql and http://docs.thedatatank.com/4.3/sparql
[26] http://lmatteis.github.io/restpark/
[27] http://basil.kmi.open.ac.uk/

URI Dereferencing. Dereferencing URIs is a practice in which a client application can access ("*look up*") information about a specific entity using its URI and obtain a copy of the corresponding published document. Those documents are called *subject pages*, because triples are generally organised by subject (i.e. triples where the entity is the subject), and contain references to other entities, which can be in turn dereferenced. This process is generally called *Link(ed Data) Traversal*, and has the main advantage that it allows the serendipitous discovery of data in a *follow-your-nose* fashion, i.e. without or with very little domain knowledge. Few approaches make effective use of this technique; for instance, the author of [Hartig, 2011, 2013] has focused on integrating the traversal of data links into query execution (LTBQE – Linked Data Based query extension). When executing a SPARQL query, only links that are relevant (identified via heuristic strategies) and available (some servers might stop responding) are traversed. The work of [Umbrich *et al.*, 2014] presents a Linked Data query engine which implements and extends the LTBQE framework with features for optimisation and reasoning. The traversal approach is also used to build Linked Data knowledge bases for learning systems, as in [Tiddi *et al.*, 2014b]. Pubby[28] is a Linked Data interface that transforms non-dereferenceable URIs of a SPARQL endpoint into resources supporting REST and content negotiation. Among its limitations, Pubby only supports SPARQL endpoints, therefore ontologies which are not uploaded on a server with a triple store cannot be accessed, and it is limited to those endpoints that support the DESCRIBE operator. Virtuoso also provides optional URI dereferencing facilities. Publishing subject pages does not require much memory and processing efforts: servers do not have to process complicated queries, while the applications do not have to access data that is not needed. While data access is much faster, however, executing more non-unidirectional queries (e.g. predicate- or object-based queries) is practically infeasible.

Triple Pattern Fragments. Triple Pattern Fragments (TPFs) are a novel, low-cost interfaces for RDF data providers that allow to move to the Web-client side the costs of executing SPARQL queries [Verborgh *et al.*, 2014a,b]. The general idea behind TPFs is that some complex queries can significantly reduce the server's availability and, in order to rebalance scalability, the important "fragments" of the query can be identified and executed by the clients (for instance the FILTER operator is executed on client-side because TPFs only support triple patterns). Moreover, to further improve adaptability, TPFs implement a non-blocking iterator strategy based on the triples' cardinality distribution (as in [Hartig, 2011]), so that the most complex parts of the query are executed first. Within less than two years, more than

[28] http://wifo5-03.informatik.uni-mannheim.de/pubby/

600,000 datasets have already been published as TPFs [Beek *et al.*, 2014]. Although TPFs provide a significant improvement regarding server availability, the query optimisation algorithm is still at an early stage, and the performance is affected by a delayed data transfer (i.e. queries are slower than on common SPARQL endpoints). To reduce the request execution time, [Acosta and Vidal, 2015] study how TPFs can be integrated with routing techniques, while [Van Herwegen *et al.*, 2015] use a predictor to identify the least expensive query path.

Public endpoints. Public SPARQL endpoints are the Web interfaces to access published datasets. While data are guaranteed to be up-to-date, the execution of the queries is performed entirely by the server, which therefore can suffer from overload with a too high demand or with too complex requests. The limitation on server availability has been extensively documented in [Buil-Aranda *et al.*, 2013], which showed that public endpoints were down on average more than 1.5 days for each month. As of end of 2015, relying purely on public datasets still remains impossible [Verborgh *et al.*, 2014a], but it is possible to monitor their availability through the SPARQLES portal[29] [Vandenbussche *et al.*, 2015]. Originally, public endpoints were operating independently, acting as centralised data warehouses. This quickly became an issue in the view of a Web of Data, and SPARQL query federation was promoted with two main ideas: (i) knowledge from distributed and heterogeneous datasets can be easily accessed and (ii) data fragmentation improves availability and query performance [Özsu and Valduriez, 2011]. A federated query is a query split into sub-queries, which are in turn sent to multiple endpoints, whose results are retrieved and then merged [Umbrich *et al.*, 2014]. To decide how to assign a sub-query to the relevant endpoint, the SERVICE operator is used [Buil-Aranda *et al.*, 2011]. Anapsid [Acosta *et al.*, 2011], SPLENDID [Görlitz and Staab, 2011], DARQ [Quilitz and Leser, 2008] and FedX [Schwarte *et al.*, 2011] are among the existing federated SPARQL engines guaranteeing efficient query execution and complete results. The performance of query federation is affected by the ineffective and time-consuming query processing, as well as endpoint result limitations (public endpoint generally return up to 10,000 results per query): some optimisation strategies have been presented by [Buil-Aranda *et al.*, 2014; Ibragimov *et al.*, 2015; Montoya *et al.*, 2015].

2.4 Towards Knowledge Discovery from the Web of Data

Graphs have been attracting attention in Knowledge Discovery for a long time. Their expressive power, as well as the complexity of data representation, access, and processing are

[29] http://sparqles.ai.wu.ac.at/

only some of the challenges Knowledge Discovery approaches have to face when working on graph data [Levinson, 1985; Thompson and Langley, 1991; Conklin and Glasgow, 2014; Segen, 2014]. In this section, we are interested in studying if graph techniques can be applied to discover knowledge from the Web of Data. We will first analyse general techniques for managing and mining graphs, and then we present specific techniques that have been applied in context of the Web of Data – namely, techniques for mining the graph of Linked Data.

2.4.1 Managing Graphs

For those domains where datasets are naturally modelled as networks of relationships (large-scale social networks, bioinformatics, Semantic Web and so on), the need for efficiently storing and executing queries is fundamental. Graph Management is a cross-disciplinary field that deals with providing (or improving) efficient graph databases.

An extensive survey of the field is presented by [Angles and Gutierrez, 2008]: the authors first compare graph database models against other standards (relational, semi-structured, semantic, object-oriented models); then define a set of important characteristics (data structures, query languages and integrity constraints), finally using them to evaluate the existing models. The work of [Dominguez-Sal *et al.*, 2010] compares instead some of the most popular Graph Database implementations: (i) Neo4j[30], a graph database that does not only rely on a relational layout of the data, but also on a network model storage (to store nodes, relationships and attributes); (ii) HypergraphDB [Iordanov, 2010], which stores graphs, and hypergraph structures (abstract graphs, in which the edges are substituted by hyperedges connecting an arbitrary set of nodes); (iii) the already mentioned Jena [Grobe, 2009], which handles RDF graphs (note that while other implementations exist, such as Sesame[31] or AllegroGraph[32], Jena currently remains the most used), and (iv) DEX [Martínez-Bazan *et al.*, 2007], an efficient graph database implementation based on bitmap representations of the entities.

Other works in Graph Management have focused on graph reachability queries based on: (i) executing queries to discover paths between nodes in large directed graphs [Agrawal *et al.*, 1989; Jagadish, 1990; Cohen *et al.*, 2003; Wang *et al.*, 2006; Trißl and Leser, 2007; Chen *et al.*, 2009]; (ii) graph matching [Ullmann, 1976; Larrosa and Valiente, 2002; Conte *et al.*, 2004; Cordella *et al.*, 2004]; and (iii) keyword search [Bhalotia *et al.*, 2002; Kacholia *et al.*, 2005; He *et al.*, 2007].

[30] http://dist.neo4j.org/neo-technology-introduction.pdf

[31] http://www.openrdf.org/

[32] http://franz.com/agraph/

From a Web of Data perspective, graph databases are powerful tools to store data and query them in a fast way. They provide a graph representation of the data (while RDF engines focus more on lists of triples), and are also optimised for graph traversal algorithms. However, graph databases lack an established standardised language, which make the development of domain-agnostic applications more expensive; they do not provide any inferential support on data and they do not exploit the properties of RDF, which might result in inefficient pattern matching queries on RDF-based graphs.

2.4.2 Mining Graphs

Efficient algorithms have been proposed not only to store and query graph structures, but also to mine them to extract useful and hidden information, such as patterns, trends, classes and clusters. Graph mining focuses on searching for frequent subgraphs in either a single graph [Kuramochi and Karypis, 2001, 2005; Fiedler and Borgelt, 2007; Bringmann and Nijssen, 2008], or a set of graphs [Inokuchi *et al.*, 2000; Kuramochi and Karypis, 2001; Vanetik *et al.*, 2002]. Approaches are divided into 5 main categories, described below.

- Heuristic searches [Holder *et al.*, 1994; Yoshida *et al.*, 1994; Jonyer *et al.*, 2002; Ketkar *et al.*, 2005; Basse *et al.*, 2010]. These approaches use canonical BFS, DFS or best-first strategies for direct matching (i.e. they do not simply compute the similarity of two graphs), and tend therefore to incur complexity issues.

- Inductive Logic Programming approaches [Dehaspe and Toivonen, 1999; Nijssen and Kok, 2001], based on Inductive Logic Programming, a field at the intersection between Machine Learning and Logic Programming [Muggleton *et al.*, 1992; Lavrac and Dzeroski, 1994]. These approaches use a set of positive and negative subgraph examples to generate hypotheses justified upon some background knowledge extracted from the graph. Due to the large search space, they tend to have computational issues.

- Inductive database approaches [Imielinski and Mannila, 1996; De Raedt and Kramer, 2001]. These approaches first generate the possible subgraphs and relations and then store them in an inductive database that can be efficiently and interactively queried a posteriori. While detecting basic patterns is a fast process, these approaches fail on more complex operations because they require a large memory usage. Moreover, in many cases the use of custom languages is necessary.

- Mathematics-based approaches [Nijssen and Kok, 2004; Maduko *et al.*, 2008], mostly divided in A-priori based methods [Inokuchi *et al.*, 2000; Kuramochi and Karypis, 2001; Inokuchi *et al.*, 2003] and pattern growth methods [Yan and Han, 2002; Huan *et al.*, 2003;

Yan *et al.*, 2004]. These approaches exploit the mathematical structure of the graph to execute a complete search on it.

- Kernel function-based approaches [Kashima and Inokuchi, 2002; Kondor and Lafferty, 2002; Gärtner, 2003]. These approaches focus on using kernel functions to identify a similarity between two graphs.

Other more general approaches have been focused on developing algorithms for clustering and classification of nodes, labels or subgraphs (see surveys of [Flake *et al.*, 2004; Schaeffer, 2007; Aggarwal and Wang, 2010]), solving the graph isomorphism problems (the NAUTY algorithm [McKay, 2007] is known to be the fastest algorithm for it), or defining mining measures to apply on graphs (e.g. support, information entropy, information gain, gini-index or minimum description length) [Agrawal *et al.*, 1994; Yoshida *et al.*, 1994; Yan and Han, 2002].

These works rely on graph-specific characteristics, such as pattern matching, traversal and reachability, and can be useful to efficiently reveal information in a Web of Data context. However, they are known for being computationally expensive, and the risk we can expect is that they might be even less efficient when dealing with the large amount of information of the Web of Data.

2.4.3 Mining the Web of Data

The Web of Data gives the possibility for Knowledge Discovery to improve its activities using the semantics embedded into the ontologies and datasets. Moreover, to the eyes of Knowledge Discovery experts, the Web of Data provides an opportunity to exploit an accessible, reliable and long-term data storage (and management) that can offer more flexibility with respect to their traditional, generally application-specific stores (e.g. MATLAB-files or relational databases).

Knowledge Discovery frameworks. An important area of research comprises Knowledge Discovery frameworks that integrate semantic knowledge. We divide them according to the three steps of the KD process.

In data pre-processing, semantic structures are exploited to improve the quality of the generated patterns. Approaches for data cleaning have relied on taxonomies [Srikant and Agrawal, 1995; Pohle, 2003; Tseng *et al.*, 2005], full ontologies [Liu *et al.*, 2004; Brisson *et al.*, 2005; Escovar *et al.*, 2006; Brisson and Collard, 2008; Józefowska *et al.*, 2010; Nebot and Berlanga, 2012] and more recently Linked Data [Garriga *et al.*, 2008; Zheng *et al.*, 2009; Khan *et al.*, 2010; Wang *et al.*, 2011; Abedjan and Naumann, 2013]. These works use external semantic knowledge to improve their results, but have a different objective (processing data to be mined during the data mining step) and motivation, namely that better results in a KD process can be obtained if the semantics is used to refine the input data.

Data mining applications based on Linked Data cross-domain knowledge and heterogeneous sources have been largely studied and recently surveyed in [Ristoski and Paulheim, 2014]: most of the approaches consist of extracting RDF data in a supervised (using SPARQL and federated queries) [Fanizzi *et al.*, 2012; d'Amato *et al.*, 2012; d'Aquin and Jay, 2013; Mencıa *et al.*, 2013; Ristoski, 2015]) or unsupervised way (with a search limited to a fixed depth or neighbourhood) [Grimnes *et al.*, 2008; Cheng *et al.*, 2011; Paulheim and Fümkranz, 2012; Zhang *et al.*, 2012; Jones *et al.*, 2014], and then transforming such data into feature vectors on which the state-of-the-art machine learning algorithms can be directly applied. As in the previous case, our intention is to avoid supervision and to reduce the constraints to exploit the Web of Data, therefore these approaches cannot be considered.

Semantics can also help the refinement of the generated patterns, as a support for the analysis of results. This is the idea brought forward by works in the post-processing step, as well as our work. We can distinguish here works of different types:

(1) Works that have been used as a support for interpretation, enrichment and exploration of patterns. While these share the same goal of our research, they fail in automatically selecting the correct external knowledge, e.g. they either use hand-crafted domain ontologies [Stankovic *et al.*, 2010; Russ *et al.*, 2011; d'Aquin *et al.*, 2012] or manually selected Linked Data [Mulwad *et al.*, 2010; Zapilko *et al.*, 2011; Paulheim, 2012; d'Aquin and Jay, 2013].

(2) Works to prune association rules or frequent itemsets [Marinica and Guillet, 2010; Anusha and Reddy, 2012; Nandhini *et al.*, 2012; Ramesh *et al.*, 2013]. In this case, the external knowledge is also manually selected. Also, these works are targeted for a specific audience, namely experts in biology or medical computer science. This simplifies the task of selecting the external knowledge, and also reduces the risks of dealing with inconsistent information in the large knowledge bases.

(3) Works to support the experts' for data analytics. While [Brisson *et al.*, 2005; Trajkovski *et al.*, 2008] use ad-hoc domain knowledge, a small portion of pre-defined Linked Data sources are used by [Novak *et al.*, 2009; Parent *et al.*, 2013]. While our idea is to target domain-agnostic audience, these work target a more domain-aware audience.

Machine Learning for Web of Data. A second area has focused on integrating Machine Learning techniques with the Web of Data. For a long time, the Semantic Web was looking at Machine Learning only as a tool to enrich or extend ontologies on the schema level (ontology construction, learning, matching and evolution [Maedche and Staab, 2004; Bloehdorn *et al.*, 2006; Grobelnik and Mladenić, 2006; Euzenat *et al.*, 2007]).

For Machine Learning experts, however, the Semantic Web is a constant new challenge due to its size, the open-world assumption, and the noisy/incomplete data herein represented. One of the most active areas of research is Statistical Relational Learning (SRL), which uses inductive reasoning and logical inference applied to Semantic Web data to assign individuals to classes, predict features of instances, predict links, and clustering instances. Similarity-based methods [d'Amato *et al.*, 2006; Janowicz, 2006; Janowicz *et al.*, 2007; d'Amato *et al.*, 2008], multi-layered networks, kernel machines [Fanizzi and d'Amato, 2006; Bloehdorn and Sure, 2007; Fanizzi *et al.*, 2008; Bicer *et al.*, 2011; Lösch *et al.*, 2012], multivariate prediction models [Thor *et al.*, 2011; Krompaß *et al.*, 2015] and graphical models [Getoor, 2007] have been extensively studied in this context (see the survey of [Rettinger *et al.*, 2009]).

Another promising area is the recently-emerged Link Mining, which puts a strong emphasis on links as a medium to build predictive or descriptive models of Linked Data [Getoor and Diehl, 2005] for object ranking, group detection, classification, link prediction and sub-graph discovery.

Graph analysis techniques have been exploited in the Semantic Web to study how the contents embedded in RDF graphs are interlinked. This is the focus of Link Structure Analysis, aiming at analysing the topology of RDF graphs, i.e. degree distribution, connectivity, link types or graph diameter. Approaches have carried out analysis at the ontology [Gil *et al.*, 2004; Hoser *et al.*, 2006; Theoharis *et al.*, 2008; Zhang, 2008] or instance level [Hausenblas *et al.*, 2008; Ge *et al.*, 2010; Guéret *et al.*, 2010; Joslyn *et al.*, 2010].

These works agree with our vision that the Web of Data can be exploited because of its powerful characteristics, e.g. the graph structure openly accessible, the links between cross-domain knowledge, the support for inference or the semantic relationships between data. Although they present interesting solutions for the study and exploration of the graph of the Web of Data, they fail in two main aspects: first, the expert is still required to select the required information from the Web of Data, as well as to validate the obtained results; second these are techniques meant to improve the Web of Data, rather than to improve the discovery of knowledge.

Navigation of the Web of Data. A third area of investigation is more graph-oriented, and sees in the Web of Data an opportunity to develop strategies that can deal with distributed, semantically meaningful and cross-domain graphs.

Studies in the area of Query Languages for Graph Databases [Wood, 2012] have focused on developing or improving languages to navigate graphs by taking into account the semantic relations expressed by the paths [Abiteboul *et al.*, 1997; Buneman *et al.*, 2000; Kasneci *et al.*, 2008; Harris *et al.*, 2013]. The SPARQL language and queries are also investigated in the Semantic Web community, e.g. SPARQL queries have been extended to the Web with basic graph patterns and operators [Bouquet *et al.*, 2009; Harth and Speiser, 2012; Hartig, 2012; Umbrich *et al.*, 2014] and languages for navigation and query across distributed datasets have been proposed by [Fionda *et al.*, 2012; Hartig and Pirrò, 2015]. Although distributed navigation is one the common features these works have with ours, their main focus is purely the improvement of the efficiency of the SPARQL language.

The idea of abstracting users from the representation of data and to allow them to agnostically discover knowledge in a large number of Linked Data source was also carried forward by approaches using Spreading Activation, to improve user queries [Freitas *et al.*, 2011a,b], exploratory searches [Marie *et al.*, 2013a,b] or Web-based reasoning [Akim *et al.*, 2011]. The same work also presents the drawbacks of using Spreading Activation at a large scale such as the Web of Data. Of particular interest, in this case, is the idea of using on-the-fly graph exploration, e.g. through dereferencing, without introducing any a priori knowledge.

A body of work has also focused on applying pathfinding algorithms on distributed semantic graphs using the information about their topology, following the approach pioneered by [Eliassi-Rad and Chow, 2005] on domain ontologies. [De Vocht *et al.*, 2013] use A* on Linked Data, a limited DFS is used by [Schuhmacher and Ponzetto, 2014] and random walks are used by [Moore *et al.*, 2012]. We consider the usage of graph algorithms as a valid opportunity; however, these approaches introduce some a priori knowledge about the sources to be used, namely these are a small portion of Linked Data that is pre-selected.

2.5 Summary and Discussion

Although there is no common agreement on the definition of an explanation, when reviewing the disciplines in Cognitive Science we showed that those have a similar way of structuring explanations. We gave an overview of such structures, including the elements, the interactions and the characteristics that explanations have according to each of the disciplines, and presented a general model that could unify those views.

Motivated by the idea of using the Web of Data as a source of knowledge, we have reviewed the most recent methods to store and access data from it. We have seen how most of the approaches rely on data indexing and introduce a priori knowledge to select the desired data sources, while very few use link traversal, a less computationally expensive technique which allows the serendipitous discovery of knowledge.

In order to explore the Web of Data automatically, we reviewed Graph Theory techniques that offer scalable ways to manage and access graph-structured data. However, we have seen how most of these approaches are highly sensitive to scalability issues, which can be relieved by using the right heuristics accessing only relevant portions of the data.

Finally, we presented approaches where the knowledge from the Web of Data was used as a support for Knowledge Discovery. We demonstrated how the knowledge from the Web of Data has been successfully used as a support for improving results in both data pre-processing and data-mining, while in data post-processing the expert's knowledge is still required to interpret results. On the other hand, ontological knowledge has been used to induce hypotheses in Inductive Logic Programming-based frameworks, which show potential for adapting such technique for our scenario.

Based on this, we can conclude that a successful approach to automatically generate explanations from the Web of Data should possess the following properties:

① it has to generate meaningful explanations, where meaningful means that they have to include the same components as the Cognitive Science model presented in Section 2.1;

② it has to explore the Web of Data on-the-fly, so that the computational costs are limited and only the necessary background knowledge is introduced;

③ it has to induce explanations automatically from the Web of Data, without any assistance from a domain expert.

In our thesis, we will approach these constraints in the following way:

❶ complete explanations can be obtained by designing a process in which its sub-processes aim at finding each of the components of the Explanation Ontology;

❷ we can combine the link traversal and graph searches to avoid the retrieval of unnecessary information from the Web of Data;

❸ we can apply the idea of Inductive Logic Programming, which induce candidate hypotheses to justify observations based on some background knowledge, to automatically generate candidate hypotheses and explain a specific pattern using background knowledge that is automatically extracted from the Web of Data;

Let us step back to the scenario of Section 2.1: we aim to automatically explain why "A Song of Ice and Fire" becomes popular on the web with a very specific regularity. In terms of the Explanation Ontology, such an observation consists in the explanandum ▲. To obtain a complete explanation, our process needs to automatically identify what △, ★ and ■ are.

The first step that has to be achieved is the generation of candidate explanantia for the observation, i.e. some plausible events that have caused the fact that some dates belong to the pattern of "popular dates". The Web of Data, which provides the background knowledge about the dates, can be accessed using link traversal combined with a heuristic search, with the aim of collecting information about events that have happened during those dates. As in Inductive Logic Programming, we can use induction to generate candidate explanantia △ learnt from the set of positive and negative examples (popular and non-popular dates) and the background knowledge built from the Web of Data. For example, both "people search for A Song of Ice And Fire when a new Game of Thrones season is released" and "people search for A Song of Ice And Fire when an episode of the TV comedy series *Veep* is aired" are plausible since they are both correlated to "A Song of Ice and Fire" according to Linked Data information. Challenges to be faced are the identification of a heuristic function for the traversal, as well as the assessment of the validity of the generated candidates. We address them in Chapter 4 and Chapter 5.

If no context ★ is specified, there is then no way to distinguish whether the induced fact △ and ▲ are co-occurring by coincidence (as in the case of "A Song of Ice and Fire" and the "Veep TV series"). The second step of the process is therefore to identify the context ★ that relates the anterior and posterior events, to remove implausible explanations. This can be achieved by using an uninformed search across Linked Data that identifies the contextual relationship between △ and ▲, but requires us to study what is the best way to assess such relationships in the graph of Linked Data. This is achieved in Chapter 6.

Finally, for an explanation to be complete, the process has to identify the theory ■, i.e. the assumption behind which the pattern has been created. Ontology-level searches can be used to partly identify the theory (e.g. "A Song of Ice and Fire" is a novel, "Game of Thrones" is a TV series, and TV series can based on novels); however, building a full theory requires knowledge from different domains, that might not be existing as representations in the Web of Data (for instance, "people search over the Web for what they are interested in"). We study this aspect in Chapter 8.

In the second part of this thesis we will present the challenges and the solutions we proposed to the first two processes, that we have implemented in Dedalo, a framework to automatically generate pattern explanations from the Web of Data. As for the last process, we will discuss on its feasibility in the third part of the work.

Part II

Looking for Pattern Explanations in the Web of Data

Chapter 3

Generating Explanations through Manually Extracted Background Knowledge

This chapter presents our preliminary work, in which we focused on answering our third research question (Section 1.3.3), i.e. how to automatically generate explanations with background knowledge from the Web of Data. Section 3.1 shows how we identified Inductive Logic Programming as a valid framework to generate pattern explanations automatically; Section 3.2 presents the foundations of Inductive Logic Programming; Section 3.3 describes how we shaped our problem as an Inductive Logic Programming one, and Section 3.4 presents some preliminary results. Section 3.5 discusses the limitations of the approach and the next steps to undertake.

3.1 Introduction

The goal of our research is the creation of a system that automatically generates explanations for a given pattern using the background knowledge from the Web of Data. With such information we could, for instance, automatically assess that the popularity of the term "A Song of Ice and Fire" increases when a new "Game of Thrones" season is released. In this scenario, one of the first required steps is to find a process that automatically generates explanations.

In the previous chapter, we have seen how frameworks based on inductive reasoning have been successfully applied to generate hypotheses based on some ontological knowledge. Our idea is therefore to study their feasibility in the context of the Web of Data and, more specifically, we explore the possibility of using the Inductive Logic Programming framework (ILP) [Muggleton *et al.*, 1992; Muggleton and De Raedt, 1994] to automatically generate explanations from Linked Data.

In Machine Learning, Inductive Learning systems start from some positive and negative evidence to learn general rules, also called "induced theories" [Lavrac and Dzeroski, 1994]. Emerging from both Machine Learning and Logic Programming [Lloyd, 2012], the field of Inductive Logic Programming introduced the idea that the learning process could be improved by adding some prior knowledge about the body of evidence. This additional feature has been widely recognised as one of the strongest points of ILP, when compared to other forms of Inductive Learning [Lisi and Esposito, 2009].

Nevertheless, it has been highlighted that the ILP frameworks fail in organising the background knowledge in a well-formed conceptual model. For this reason, fields such as Onto-Relational Learning and Statistical Relational Learning proposed to introduce ontologies in the background knowledge because, on the contrary, those were a more natural mean to convey conceptual knowledge [Chandrasekaran *et al.*, 1999]. Based on this idea, we study the integration of cross-disciplinary knowledge from the Web of Data in the background knowledge of an ILP framework in order to generate explanations for patterns.

The aim of this chapter is to demonstrate that Inductive Logic Programming is a good candidate for automatically generating pattern explanations, and that Linked Data provide valid background knowledge for this purpose. To this end, we start from some data (the evidence) organised into patterns, and then use ILP to induce theories that explain why certain data points belong to a specific pattern. As already said, the background knowledge upon which the theories are built consists of information found in Linked Data. Through this process, we can show that meaningful explanations can be automatically obtained based on the knowledge provided from the Web of Data.

3.2 The Inductive Logic Programming Framework

Inductive Logic Programming is placed at the intersection between Machine Learning and Logic Programming. From Logic Programming, ILP inherited the knowledge representation framework - namely, the use of first-order logic clauses to represent data. From Machine Learning, it inherited the learning mechanism, i.e. the derivation of some rules based on some positive and negative examples.

3.2.1 General Setting

The objective of ILP is to construct first-order logic clausal theories, called hypotheses, which are derived by reasoning upon a set of negative and positive examples, plus some additional background knowledge about them. The process is typically carried out using a search in a space of possible hypotheses. More precisely, the task of ILP is defined as:

- Given $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$, a set $\mathcal{E}$ of training examples represented as ground facts, and divided into positive examples $\mathcal{E}^+$ and negative examples $\mathcal{E}^-$;
- Given $\mathcal{B}$, some background knowledge about the examples $e \in \mathcal{E}$;
- Find a hypothesis $\mathcal{H}$, so that $\mathcal{H}$ is *complete* and *consistent* with respect to the background knowledge $\mathcal{B}$ and the examples in $\mathcal{E}$.

The sets $\mathcal{E}$, $\mathcal{B}$ and $\mathcal{H}$ are logic programs, i.e. they are sets of clauses with an atom as head and a set of atoms as body, in the form $h \leftarrow b_1, b_2 \ldots b_i$. Also, the two sets of positive and negative examples usually contain only ground clauses (clauses with empty bodies).

To check the completeness and consistency requirements, ILP uses a coverage function, which returns *true* if the example e is satisfied by $\mathcal{H}$ with respect to $\mathcal{B}$. We note:

$$covers(\mathcal{H}, \mathcal{B}, e) = \textit{true iff } \mathcal{H} \cup \mathcal{B} \models e$$

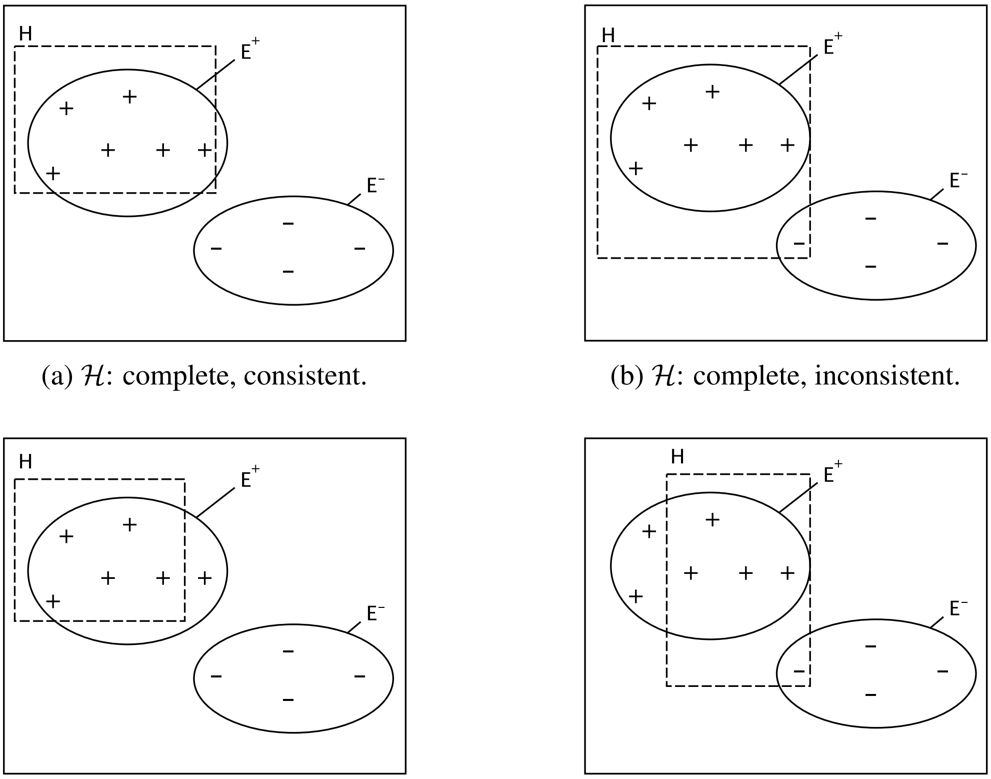
meaning that e is a logical consequence of $\mathcal{H} \cup \mathcal{B}$.

Consequently, we say that:

- *Completeness* with respect to $\mathcal{B}$ is guaranteed when a hypothesis $\mathcal{H}$ covers all the positive examples $\mathcal{E}^+ \subseteq \mathcal{E}$, so that $covers(\mathcal{B}, \mathcal{H}, e) = \forall e \in \mathcal{E}^+$;
- *Consistency* with respect to $\mathcal{B}$ is guaranteed when a hypothesis $\mathcal{H}$ covers none of the negative examples, so that $\neg covers(\mathcal{B}, \mathcal{H}, e) = \forall e \in \mathcal{E}^-$.

These criteria require then that $\mathcal{H}$ and $\mathcal{E}^+$ agree on the examples that $\mathcal{H}$ covers: in other words, a hypothesis acts as a classifier of examples that are tested against the oracle $\mathcal{E}^+$ (see Figure 3.1). Criteria to check the validity of a classifier $\mathcal{H}$ are generally classification accuracy, transparency, statistical significance and information content [Lavrac and Dzeroski, 1994].

Of course, not all the learning tasks can produce complete or consistent hypotheses: this means that ILP systems need to include a noise-handling mechanism that prevents overfitting by dealing with imperfect data such as noisy training examples or missing, sparse or inexact values in the background knowledge.



(a) $\mathcal{H}$: complete, consistent.

(b) $\mathcal{H}$: complete, inconsistent.

(c) $\mathcal{H}$: incomplete, consistent.

(d) $\mathcal{H}$: incomplete, inconsistent.

Figure 3.1 Accuracy of a hypothesis based on the completeness and consistency criteria.

3.2.2 Generic Technique

In Inductive Logic Programming, as well as in Inductive Learning, induction (or generalisation) is seen as a search problem, where a hypothesis has to be found in a partially ordered space of hypotheses [Mitchell, 1982]. This process requires three steps:

(i) *Structuring*. In a first instance, an ILP algorithm constructs an ordered lattice of all the possible hypotheses, ordered from the most general (the ones that cover more training examples) to the most specific (the ones that cover one training example only).

(ii) *Searching*. The ordered space is then searched using some refinement operators, which are functions computing the generalisation or specification of a clause, according to whether the search is performed in a bottom-up or top-down way, respectively.

(iii) *Bounding*. Finally, in order to reduce the computational complexity, some bias (e.g. in heuristically directing the search or in the language expressing the hypotheses) is defined to constrain and reduce the search in the space.

The generic ILP algorithm works as follows: candidate hypotheses are kept in a queue; hypotheses are then repeatedly deleted from the queue and expanded using the refinement operators; finally, if they are valid according to the declared bias, the new expanded hypotheses are added to the queue. This process continues until a stop-criterion is satisfied.

3.2.3 A Practical Example

Reusing the example introduced earlier in this thesis, let us imagine that we want to automatically learn why during some dates (considered as the positive examples) people look for “A Song of Ice and Fire”, while on some other dates (the negative examples) people do not. We note the concept to be learnt as `isPopular(X)`, with `X` being a date. Suppose we have some background knowledge about this problem, e.g. which TV series episodes have been aired on those dates, as of Table 3.1.

Table 3.1 Background knowledge $\mathcal{B}$. `GoT` and `HIMYM` are respectively the “Game of Thrones” and “How I met Your Mother” TV series.

$\mathcal{B}$
happenedOn(‘2013-06-09’,‘GoT-s03e10’)
happenedOn(‘2014-06-15’,‘GoT-s04e10’)
happenedOn(‘2014-06-15’,‘HIMYM-s08e23’)
happenedOn(‘2014-03-31’,‘HIMYM-s09e24’)
TVseries(‘GoT-s03e10’,‘GoT’)
TVseries(‘GoT-s04e10’,‘GoT’)
TVseries(‘HIMYM-s08e23’,‘HIMYM’)
TVseries(‘HIMYM-s09e24’,‘HIMYM’)
seasonFinale(‘GoT-s03e10’)
seasonFinale(‘GoT-s04e10’)
seasonFinale(‘HIMYM-s08e23’)
seasonFinale(‘HIMYM-s09e24’)

Suppose also that we are given some positive and negative examples, i.e. dates in which “A Song of Ice and Fire” was popular or not, as in Table 3.2.

Table 3.2 Examples $\mathcal{E}$ for `isPopular(X)`.

$\mathcal{E}^+$	$\mathcal{E}^-$
isPopular(‘2013-06-09’)	isPopular(‘2013-05-06’)
isPopular(‘2014-06-15’)	isPopular(‘2014-03-31’)

Believing $\mathcal{B}$, we can induce that “A Song of Ice and Fire” is popular for those dates in which a season finale of the TV series “Game of Thrones" has been aired. Therefore:

`isPopular(X)` $\leftarrow$ `happenedOn(X,Y)` $\wedge$ `TVseries(Y,‘GoT’)` $\wedge$ `seasonFinale(Y)`

Note that $\mathcal{H}$ is complete with respect to $\mathcal{B}$, because all the examples in $\mathcal{E}^+$ are season finales *and* are part of the TV series “Game of Thrones”; and it is also consistent, because none of the negative examples is satisfied by $\mathcal{H}$.

3.3 The ILP Approach to Generate Explanations

Once presented the ILP framework, we formulate our problem as an ILP program in the following way.

Given:

- a dataset $R = \{r_0, ..., r_m\}$, corresponding to the totality of training examples $\mathcal{E}$;
- a set of mutually disjoint clusters (patterns) $C = \{C_0, C_1, \ldots, C_n\}$, so that $C_i \cap C_j = \emptyset$, which are extracted from R (i.e. $C_i \subseteq R$);
- a background knowledge $\mathcal{B}$ of ground clauses, extracted from Linked Data, that give information about each item $r_i \in R$;

Find:

- for a chosen cluster C_i, some hypotheses $\mathcal{H}$ that explain, according to $\mathcal{B}$, why items r_j belong to C_i and not to the rest of R. Note that $C_i = \mathcal{E}^+$ and $(R \setminus C_i) = \mathcal{E}^-$. In this manner, $\mathcal{H}$ is in the form of

$$\texttt{in\_cluster(}r_j\texttt{)} \leftarrow b_1 \wedge b_2 \wedge \ldots \wedge b_m$$

where r_j is an item of C_i. This, of course, assumes that the background knowledge $\mathcal{B}$ contains enough information about the items in R, so to construct $\mathcal{H}$. To make sure of that, in a first stage the Linked Data knowledge that populates $\mathcal{B}$ is manually extracted.

Our process is then articulated as follows.

Initial Dataset. A set of items partitioned into clusters is provided (or created). Following the example of explaining a trend popularity, a dataset might be a group of dates such as $R = \{$`‘2013-05-06’, ‘2013-06-09’,‘2014-03-31’,‘2014-06-15’`$\}$, partitioned into dates in which the web search rate for “A Song of Ice and Fire” was high, i.e. $\mathcal{E}^+ = \{$`‘2013-06-09’,‘2014-06-15’`$\}$ and dates in which the rate was low, i.e. $\mathcal{E}^- = \{$`‘2013-05-06’, ‘2014-03-31’`$\}$.

Background Knowledge Population. For each of the items in R, some triples involving them are extracted from some datasets in Linked Data and then encoded as first-order clauses, as in Table 3.3. RDF is-a relationships are encoded as clauses of arity 1, while instance relations are encoded as clauses of arity 2.

Table 3.3 $\mathcal{B}$ built based on Linked Data.

clusters	`in_cluster('2013-06-09')`
RDF predicates	`happenedOn('2013-06-09','GoT-s03e10')`
RDF is-a relations	`seasonFinale('GoT-s04e10')`

Hypothesis Generation. The ILP framework can then reason upon the examples and the background knowledge, so that to generate hypotheses in the form of

[+*cov*, -*cov*] `in_cluster(X)` ← `happenedOn(X,Y)` ∧ `TVseries(Y,'GoT')` ∧ `seasonFinale(Y)`

which is interpreted as: "items `X` are part of the cluster because they are dates in which an episode `Y` happened, and this episode is part of the TV series Game of Thrones, but is also a season finale. +*cov* is the number of positive examples covered by the generated hypothesis, while -*cov* is the number of negative examples covered by $\mathcal{H}$.

Hypothesis Evaluation. The hypotheses evaluation is performed in this preliminary study using two common rule interestingness measures, the F-Measure and the Weighted Relative Accuracy (see definition below). While the F-Measure is the typical Information Retrieval measure for accuracy, WR_{acc} puts more emphasis on unusualness, i.e. the relative accuracy of $\mathcal{H}$, which can be beneficial to obtain valid hypotheses for patterns of small sizes.

First, *Precision* and *Recall* are defined as:

$$P = \frac{|\mathcal{H} \cap \mathcal{E}^+|}{|\mathcal{H}|} \quad (3.1) \qquad\qquad R = \frac{|\mathcal{H} \cap \mathcal{E}^+|}{|\mathcal{E}^+|} \quad (3.2)$$

where $|\mathcal{H}|$ is total coverage of $\mathcal{H}$ and $|\mathcal{E}^+|$ is the total number of positive examples (the size of the cluster to be explained). The F-Measure is very well-known to be the harmonic mean of P and R as:

$$F = 2 \times \frac{P \times R}{P + R} \tag{3.3}$$

where F is in the range $[0,1]$.

WR_{acc} is a rule evaluation measure proposed by [Lavrač *et al.*, 1999] as a trade-off between the relative accuracy of a rule, i.e. how many examples are well predicted compared to the ones that should be predicted, and its generality, i.e. how frequent the rule is, which is used as a "weight". WR_{acc} is in the range $[-1,1]$.

In our scenario, WR_{acc} is defined as:

$$WR_{acc} = \frac{|\mathcal{H}|}{|\mathcal{E}|}\left(\frac{|\mathcal{H} \cap \mathcal{E}^+|}{|\mathcal{H}|} - \frac{|\mathcal{E}^+|}{|\mathcal{E}|}\right) \tag{3.4}$$

where $|\mathcal{E}|$ is the number of training examples (both positive and negative), and $|\mathcal{H}|$ and $|\mathcal{E}^+|$ are as defined above.

The generated hypotheses are then ranked according to the chosen score, so that the ones with the highest scores can be considered as the most representative for the cluster.

3.4 Experiments

To test the approach described above, we considered two different case studies based on the #Red and the #KMi.H datasets. We remind that in the first case we have people grouped according to what they have read, and in the second case we have books grouped according to the course attended by the students that have borrowed the book. An overview of the clusters that we are trying to explain is in Table 3.4. For other information about the datasets, e.g. how they were obtained, we invite the reader to refer to Appendix A.1.

Table 3.4 Cluster details for #Red and #KMi.H.

#Red clusters (number of people and what they read)

26	Romanticism ∨ Walter Scott	15	18th century women writers
110	Jane Austen	68	Victorian novelists
4	Tasso ∨ epic poetry	16	Marx ∨ philosophy
38	Samuel Johnson	35	Shakespeare
17	Milton ∨ Christian writers	39	Scottish existentialism

#KMi.H clusters (number of borrowed books, and what they are about)

171	Information Systems	374	Computing
319	Mechanics	405	Mechanic Engineering
101	Computer games programming	99	Music Technology
282	Control systems	404	Electrical Engineering

As said in the introduction, the objective of this evaluation is to show that we can obtain meaningful pattern explanations with Inductive Logic Programming based on background knowledge from Linked Data. To do that, we study if explanations improve in F-Measure or WR_{acc} score when adding more Linked Data information in $\mathcal{B}$. The process consisted in building the training examples, building the background knowledge using Linked Data and finally generating the hypotheses.

3.4.1 Building the Training Examples

Given a dataset of items partitioned into mutually disjoint clusters, in this first step we built Prolog-like clauses such as `in_cluster(X)` for each item `X` in the dataset.

```
(a) case #Red:                    (b) case #KMi.H:
in_cluster(lord_byron).           in_cluster(bookXXX).
in_cluster(lady_byron).           in_cluster(bookYYY).
in_cluster(james_clunie).         in_cluster(bookZZZ).
```

As our final objective was to generate hypotheses $\mathcal{H}$ for each cluster (where each $\mathcal{H}_i$ was a plausible reason of the elements belonging to that cluster), the training examples were divided into $\mathcal{E}^+$ and $\mathcal{E}^-$ in a one-vs-all manner. For instance, we proceeded so that (i) $\mathcal{E}^+$ =(cluster-Austen) vs. $\mathcal{E}^-$ =(cluster-Milton $\vee$ cluster-Marx $\vee \ldots$), then (ii) $\mathcal{E}^+$ =(cluster-Marx) vs. $\mathcal{E}^-$ =(cluster-Milton $\cup$ cluster-Austen $\vee \ldots$), and so on.

3.4.2 Building the Background Knowledge

In the second step, we manually explored one or more Linked Data datasets to extract additional information about the training examples, with the objective of building the background knowledge.

The background knowledge $\mathcal{B}$ for #Red was retrieved using the RED ontology predicates concerning each reader. Predicates and type relationships were then transformed as:

⟨red:Lord_Byron *red:originCountry* red:England⟩ ⇒ `originCountry(lord_byron,england).`
⟨red:Lord_Byron *rdf:type* red:Reader⟩ ⇒ `reader(lord_byron).`
⟨red:England *rdf:type* red:Country⟩ ⇒ `country(england).`

In the same way, we manually retrieved additional information from Linked Data. We used the *owl:sameAs* property to navigate through other datasets, such as DBpedia, and to get new pieces of information using its SPARQL endpoint. Assuming the triple ⟨red:Lord_Byron *owl:sameAs* db:GeorgeByron⟩, we used the object value db:GeorgeByron to retrieve new statements, such as:

⟨db:GeorgeByron *dbo:birthPlace* db:England⟩
⟨db:GeorgeByron *dbo:influenced* db:EmilyBronte⟩

consequently adding them to $\mathcal{B}$ as new Prolog clauses.

The external knowledge about the books in #KMi.H required us more navigation. We used the *bibo:isbn10*[1] property as equivalence statement to navigate to the British National Bibliography dataset[2] (BNB). From there, we retrieved the topics that were associated to each book. Since topics in the BNB dataset are expressed with the Library of Congress Subject Headings[3] (LCSH) vocabulary, we then used the LCSH endpoint, locally loaded, to retrieve the broader concepts of each topic, using the SKOS[4] *skos:broader* property. The insight behind extending $\mathcal{B}$ with the LCSH taxonomy is that subjects provided by the BNB dataset might in fact be too narrow (since specific to just few books), while the broader terms could be representative of more books, and they could therefore facilitate the induction. An example of the navigation across datasets is shown in Figure 3.2.

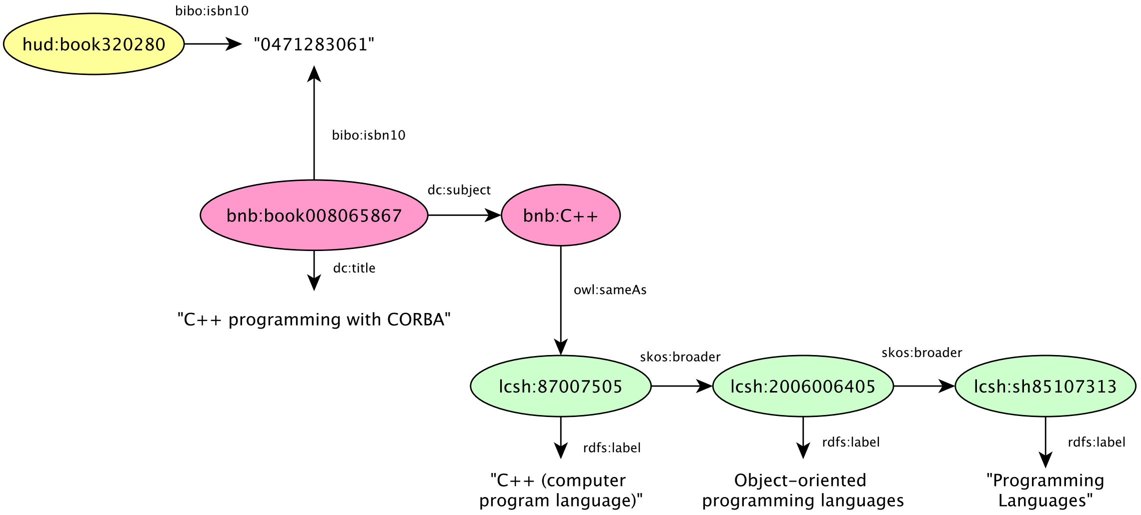


Figure 3.2 Linked Data navigation example. The different colours correspond to the #KMi.H (yellow), the BNB (pink) and the LCSH (green) datasets.

[1] The BIBliographic Ontology (http://bibliontology.com/specification) is a vocabulary providing many concepts and properties for describing citations and bibliographic references

[2] http://bnb.data.bl.uk/sparql

[3] http://id.loc.gov/authorities/subjects.html

[4] http://www.w3.org/TR/swbp-skos-core-spec/

3.4.3 Inducing Hypotheses

In this step, we use the ILP Aleph system[5] to generate hypotheses for each of the clusters.

We ran several experiments, each of which with a different background knowledge $\mathcal{B}$. Our assumption is that the more external information is added to $\mathcal{B}$, the more interesting hypotheses are likely to be found. We expect, for instance, that the background knowledge built with DBpedia and RED predicates will induce hypotheses that are more accurate than the ones induced with a background knowledge built on RED only. Similarly, we expect more interesting hypotheses from a $\mathcal{B}$ built with additional BNB and LCSH information. Table 3.5 shows an example of different background knowledges that we build from #Red.

Table 3.5 Different $\mathcal{B}$s built with Linked Data statements in the #Red experiment.

Background knowledge		Clause example
$\mathcal{B}_1$	RED only	`originCountry(lord_byron,england)`
$\mathcal{B}_2$	$\mathcal{B}_1 \cup$ *dbo properties*	`influenced(lord_byron,emily_bronte)`
$\mathcal{B}_3$	$\mathcal{B}_2 \cup$ *dc:subject*	`subject(lord_byron,bisexual_writers)`
$\mathcal{B}_4$	$\mathcal{B}_1 \cup$ *rdf:type*	`politician(lord_byron)`
$\mathcal{B}_5$	$\mathcal{B}_4 \cup$ *rdf:subclassOf*	`person(lord_byron)`
$\mathcal{B}_6$	$\mathcal{B}_1 \cup$ *yago:class*	`lgbtWritersFromUK(lord_byron)`

Once having generated all the input data, we ran Aleph to obtain hypotheses for each of the clusters. To facilitate readability, we use below the label of the cluster instead of the `in_cluster(X)` notation. The best hypotheses for #Red, that we scored with WR_{acc}, are shown in Table 3.6.

Table 3.6 Explanations induced in the #Red experiments.

hypothesis	WR_{acc}
`austen(X):- religion(X,anglican)`	0.025
`austen(X):- female(X)`	0.020
`marx(X):- religion(X,quaker)`	0.007
`milton(X):- religion(X,anglican) ∧ male(X) ∧ country(X,england)`	0.025

Table 3.7 presents some of the hypotheses we obtained with #KMi.H. Here, we compared the accuracy of explanations generated with background knowledge from the BNB dataset only (the ones with the clause `topic(X)` in the body), with the ones generated when enriching it with the LCSH broader concepts (clause `broader(X)`). As shown, in the latter case the hypotheses have a larger coverage.

[5] http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html

Table 3.7 Best generated hypotheses for #Kmi.H, including their WR_{acc} and F scores.

hypothesis	WR_{acc}	F
`digital_media(X):- topic(X,computer_games_design)`	0.001	0.139
`digital_media(X):- topic(X,computer_graphics)`	0.002	0.13
`digital_media(X):- broader(X,computers)`	0.007	0.143
`electronic_eng(X):- topic(X,digital_control_systems)`	0.0005	0.02
`electronic_eng(X):- topic(X,systems_analysis)`	0.0005	0.03
`electronic_eng(X):- topic(X,automatic_control)`	0.006	0.19
`electronic_eng(X):- broader(X,engineering_instruments)`	0.012	0.34
`electronic_eng(X):- broader(X,algorithms)`	0.006	0.34
`electronic_eng(X):- broader(X,dynamics)∧broader(X,machine_theory)`	0.011	0.33
`mechanics(X):- topic(X,project_management)`	0.002	0.074
`mechanics(X):- topic(X,production_management)`	0.005	0.169
`mechanics(X):- broader(X,technology)`	0.003	0.121
`mechanics(X):- broader(X,industrial_management)`	0.007	0.208
`mechanics(X):- broader(X,management)`	0.010	0.242

3.4.4 Discussion

The #Red experiments seem to present weak results in terms of evaluation score, i.e. a low WR_{acc} for the generated hypotheses. It is important to keep in mind that this score introduces the weight of the true negative examples, which are obviously a huge amount when compared to the small size of the analysed cluster. With that said, the results for the Austen cluster are fairly strong when compared to the sample set ($|R|$ is actually 1,230), and show how ILP can explain a pattern, e.g. "people read Jane Austen because they are Anglican women".

The #KMi.H experiment is certainly a simpler use-case in terms of possible explanations to be found. Although the domain might seem trivial, it was our intention to prove that when enough background knowledge about the examples is presented, it is possible to obtain improved results. The difference between the two datasets consists in the amount of positive and negative examples homogeneously described in the background knowledge (that is, a particular property concerns a good percentage of the example set). Such a larger coverage demonstrates that both the WR_{acc} and F-Measure are indeed significantly improved.

Both experiments confirmed our intuition that the proposed approach could naturally combine different sources of background knowledge (i.e. different datasets) to produce explanations of patterns found in the data. For example in #Red, the information about the gender of readers comes from the RED ontology, while the information about their religion is coming from DBpedia. In #KMi.H, adding more background knowledge demonstrates that we can give more accurate explanations about the patterns ("*1 book out of 3 in the Electronic Engineering cluster is about Machine Theory*" rather than "*1 book out of 5 is* *about Automatic Control*”). The sensibility of the relative accuracy is much more visible with the #KMi.H results, whose WR_{acc} scores significantly increase when the background knowledge is extended.

While this suggests that, by adding even more information, better explanations can be obtained, it also reveals that hypotheses for explanation have to be made more understandable. In #Red, we require some information to reveal the relations between the readers and their topic (what connects Jane Austen with Anglican women? Is Jane Austen Anglican too?). In the case of #KMi.H, this relation is already more visible to our eyes (the faculty of Engineering and the Engineering books are in fact about the same topic), but an automatic approach should be able to express it anyway, no matter how trivial such relation is.

Finally, while WR_{acc} has proven to be a good starting point, the F-Measure should be preferred because it can be more effective in “unclear situations”, i.e. with smaller clusters or when no obvious explanation is visible because of a lack of information.

3.5 Conclusions and Limitations

In this chapter, we have presented a preliminary idea about building a process that automatically generates explanations using background knowledge found in the Web of Data. We have shown in Section 3.1 how we identified induction as the reasoning process to obtain explanations for patterns, and how the framework of Inductive Logic Programming was a good candidate for the scope. We continued with Section 3.2 where we introduced the general framework of ILP, and with Section 3.3 in which we designed our problem as an ILP one. Finally, we tested the approach in Section 3.4 on two use-cases, #Red and #KMi.H, presenting and discussing the obtained results.

The results and final discussion have revealed the limitations of the current approach and consequently the next steps that we need to undertake. We present them below.

➊ **Data is manually selected.** Focusing on proving the validity of the Inductive Logic Programming approach, the background knowledge in this chapter has been semi-manually selected. More precisely, when expanding the background knowledge with information from other Linked Data datasets, we intentionally chose the properties that we believed would be forming interesting hypotheses. This mostly means that we drove the ILP process to induce the explanations that we wanted (for instance, we believed that the religion could be influencing a reader’s book choice). With that said, our objective is to make this selection of background knowledge an automatic process, where knowledge in Linked Data is agnostically accessed. In order to detect what strongly connects the data in a pattern, the next step is to find a good way to automatically detect the useful background knowledge in Linked Data.

❷ **Aleph is not scalable.** Inductive Logic Programming has shown to be poorly scalable when the background knowledge is large. This means that designing a process in which we first find the valid Linked Data information and then build the background knowledge to use Aleph is risky, since the computational complexity might be too high and the hypotheses might never be obtained. However, we have shown that by adding information in the background knowledge *little by little*, we were able to sensibly improve the hypothesis accuracy. The next step to take in this direction is therefore to design an inductive process in which information from Linked Data is iteratively added until a satisfying explanation is found.

❸ **Explanations are not complete.** Finally, we have seen that hypotheses are still unclear to the reader, e.g. we do not really know what is the connection between Jane Austen and Anglican women. Rolling back to the Explanation Ontology of Section 2.1.2, we can now say that the ILP process allows us to find some plausible anterior events for a pattern. However, we are missing the context that relates an anterior and a posterior event, as well as the theory behind them. This means that our system has to include some Linked Data-based processes to automatically detect the contextual relation of two events, as well as the theory governing their occurrence.

Through the next chapters, we will see how we propose to solve the issues of manual selection, of scalability and of context definition. Regarding the automatic definition of a theory behind the explanations, we will comment on its feasibility in the third part of this work.

Chapter 4

Generating Explanations through Automatically Extracted Background Knowledge

In this chapter we show how we extended the Inductive Logic Programming idea into the framework we propose, that we called Dedalo. We designed it as a process to automatically find pattern explanations by iteratively building the required background knowledge from Linked Data. Dedalo is inspired by Inductive Logic Programming and integrates new features such as a heuristic greedy search and Linked Data Traversal. These allow us to answer our second research question (Section 1.3.2), i.e. how to find in the Web of Data the background knowledge that we need to generate explanations. After the introduction in Section 4.1 of the problems tackled in this chapter, we present Dedalo's foundations in Section 4.2 and the designed approach in Section 4.3. In Section 4.4, we evaluate the performance of the process according to different criteria while, in Section 4.5, we conclude by discussing some of the limitations of the approach.

4.1 Introduction

In the previous chapter we have seen how Inductive Logic Programming is a promising solution if we aim to automatically explain patterns. With that said, ILP has shown two main drawbacks: first, it requires the background knowledge about the evidence to be manually selected, which means introducing a priori knowledge of the problem; second, it incurs considerable computational issues when the background knowledge is too large and rich. This means that adding the entire knowledge from Linked Data in the process is not feasible;

in addition, most of this knowledge would certainly be irrelevant. Thus, it is necessary that we detect and select only the salient information that we need to build the background knowledge for the induction process.

The solution we present here is to redesign the inductive process to make it not only more suitable for the Web of Data, but also able to automatically detect the relevant background knowledge. Our idea is based on two key aspects. First, we avoid scalability issues by increasing the background knowledge from Linked Data iteratively. This is achieved using a Link Traversal strategy, which uses the links between entities to blindly explore the graph of Linked Data. By "blindly", we mean that we use URI dereferencing to discover new resources on-the-fly (which possibly belong to unknown data sources) that can serendipitously reveal new knowledge. Second, in order to know which is the right piece of information that has to be extracted, such Link Traversal is driven by a greedy search whose ultimate scope is to find relevant information about the items of a pattern that we want to explain. These two aspects are finally combined with an inductive reasoning process to find out hypotheses that potentially explain a pattern.

The resulting process, Dedalo, is able not only to automatically navigate throughout Linked Data, without knowing them in advance or in their entirety, but also to cleverly use this information to explain patterns of data. In terms of the Explanation Ontology that we previously presented, achieving this task allows Dedalo to find candidate anterior events for a specific observation (the pattern).

4.2 Problem Formalisation

In short, the process derives candidate explanations about a pattern based on information heuristically found in Linked Data. In an ILP fashion, Dedalo uses the items of the pattern that needs to be explained as the positive examples to learn from, while the items in the rest of the dataset are used as negative examples. As in ILP, Dedalo aims at finding the candidate explanation that covers a maximum number of positive examples and a minimum number of negative examples. Candidate explanations are derived by reasoning upon the background knowledge built using statements automatically extracted from Linked Data.

4.2.1 Assumptions

We based Dedalo on a few assumptions, which gave the bases to develop the process.

Linked Data are a graph of knowledge. The Linked Data hub is a directed and labelled (multi)graph of RDF entities and properties, that can be navigated on-the-go using URI dereferencing. This means that one can start from one entity and “walk” through a complete, possibly cross-datasets graph simply by following the RDF properties that link entities to each other.

Entities share Linked Data paths. In Linked Data, as in any other graph, entities can have common paths, which connect them to another entity. A path, in this case, is intended as a chain of contiguous RDF properties from one entity to another: the assumption here is that some of those “walks to a specific entity” will be more shared than others.

Paths are Linked Data explanations. The final insight is that if a path to a Linked Data entity is more commonly shared among the original entities belonging to the pattern we are trying to explain than to its negative examples, then both the path and the entity can be used as an explanation to the grouping of the positive examples.

4.2.2 Formal Definitions

We present here the basic terminology that we will use for the rest of our work. We invite the reader to refer to Figure 4.1 and Section 4.2.3 for an illustrative example of the problem.

ILP terminology

- $\mathcal{E}$ is a set of initial items, divided in a disjoint set of positive and negative examples, such that $\mathcal{E} = \{\mathcal{E}^+ \cup \mathcal{E}^-\}$.
- $\mathcal{E}^+ = \{e_0, \ldots, e_k\} | \mathcal{E}^+ \subseteq \mathcal{E}$ is the pattern that we want to explain, used as positive evidence;
- $\mathcal{E}^- = \{e_0, \ldots, e_n\}$ is the set of remaining items in $\mathcal{E}$, so that $\mathcal{E}^- = \mathcal{E} \setminus \mathcal{E}^+$. $\mathcal{E}^-$ can be composed of items from $|K|$ different patterns, i.e. $\mathcal{E}^- = \bigcup\limits_{i \in K} \mathcal{E}_i^-$
- $\mathcal{B}$ is the background knowledge from which we learn, composed of knowledge from Linked Data.

Graph terminology

- $G = \{V, E, L\}$ is any subgraph of the Linked Data Cloud, where V is the set of RDF entities, E the set of RDF properties and L the set of labels that are known in G at a given moment.

- $\overrightarrow{p_i} = \langle p_0 \cdot p_1 \ldots p_l \rangle$ is a path, i.e. a chain of sequential RDF properties, whose length is defined as *length*$(\overrightarrow{p_i}) = l$;
- $R_i \subseteq V$ is the set of entities that share $\overrightarrow{p_i}$, also called roots of $\overrightarrow{p_i}$;
- $V_i \subseteq V$ is the set of entities where $\overrightarrow{p_i}$ can terminate. For each $v_j \in V_i$, there exists a set $R_{i,j} \subseteq R_i$ of those roots that share $\overrightarrow{p_i}$ to v_j;
- $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$ is a candidate explanation (alternatively called hypothesis) shared by the items of $R_{i,j}$.

Problem Definition

Inspired by the ILP approach, we define our problem as follows. Given:

- the initial observations divided into one positive pattern and its counterexamples, as $\mathcal{E} = \{\mathcal{E}^+ \cup \mathcal{E}^-\}$;
- the background knowledge $\mathcal{B} = \{\varepsilon_{1,1}, \ldots, \varepsilon_{i,j}\}$, consisting in the space of all possible candidate explanations $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$ known in G at a specific moment;

find the explanation $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$ that at the same time is:

- the most *complete*, i.e. it maximises the number of roots in $R_{i,j}$ that are also positive examples, i.e. $|R_{i,j} \cap \mathcal{E}^+| \approx |\mathcal{E}^+|$;
- the most *consistent*, i.e. it minimises the number of roots that are negative examples, i.e. $|R_{i,j} \cap \mathcal{E}^-| \approx \emptyset$;

across all the possible explanations in $\mathcal{B}$. We call this explanation *top*($\mathcal{B}$).

4.2.3 An Example

Let us take again the example of why a term such as "A Song of Ice and Fire" is popular at specific times.

In Figure 4.1, G is a RDF graph that includes a set $\mathcal{E}$ of 5 entities, each of which corresponds to a week time frame where some TV series episodes have been broadcasted. Links from a week to a broadcasted event are labelled *ddl:linkedTo*. From the evidence, we know that the first three weeks are the ones where more people were searching for "A Song of Ice and Fire", and therefore they constitute $\mathcal{E}^+$, while in the last two the term did not have such a popular search rate, so they constitute the counterexamples $\mathcal{E}^-$. From the figure, one can see that some of the weeks share paths, whose characteristics are:

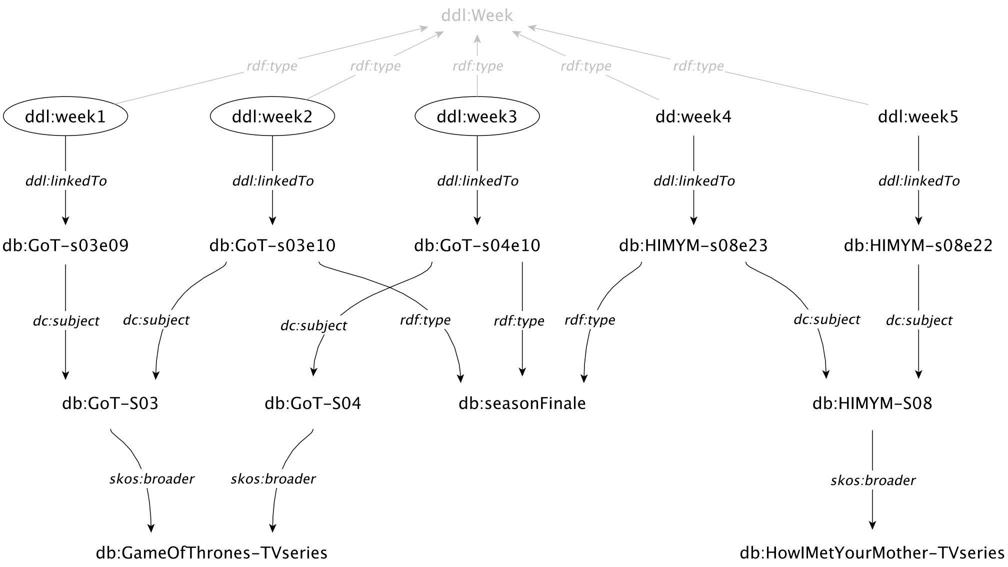


Figure 4.1 Graph example. Entities with a border belong to $\mathcal{E}^+$.

(1) $\overrightarrow{p_1} = \langle$ddl:linkedTo.rdf:type$\rangle$
R_1={ddl:week2, ddl:week3, ddl:week4}
V_1={db:seasonFinale}

(2) $\overrightarrow{p_2} = \langle$ddl:linkedTo.dc:subject.skos:broader$\rangle$
R_2={ddl:week1, ddl:week2, ddl:week3, ddl:week4, ddl:week5}
V_2={db:GameOfThrones-TVseries, db:HowIMetYourMother-TVseries}.

From those, the following candidate explanations $\varepsilon_{i,j}$ can be derived:

- From $\overrightarrow{p_1}$, we derive that there is a set of weeks linked to a season final episode of a TV series, i.e. $\varepsilon_{1,1} = \langle$ddl:linkedTo.rdf:type $\cdot$ db:seasonFinale$\rangle$; this is the only one we can derive, because $\overrightarrow{p_1}$ only has one possible value for V_1;
- For $\overrightarrow{p_2}$ and v_1=db:GameOfThrones-TVseries, $R_{2,1}$ ={ddl:week1, ddl:week2, ddl:week3}. We derive $\varepsilon_{2,1} = \langle$ddl:linkedTo.dc:subject.skos:broader $\cdot$ db:GameOfThrones-TVseries$\rangle$;
- Similarly, for $\overrightarrow{p_2}$ and v_2 =db:HowIMetYourMother-TVseries, $R_{2,2}$ ={ddl:week4, ddl: week5}. We derive $\varepsilon_{2,2} = \langle$ddl:linkedTo.dc:subject.skos:broader $\cdot$ db:HowIMetYourMother-TVseries$\rangle$

According to the completeness and consistency constraints defined above, the best explanation is the one where the number of items in $\mathcal{E}^+$ is maximised and the number of items outside it is minimised. In this case, the one that best fits is of course $\overrightarrow{p_2}$, since it is the one shared only by elements in $\mathcal{E}^+$.

4.3 Automatic Discovery of Explanations

Once the process is formally defined, the second step is to focus on how to build it. The section first introduces the specific challenges we address, with their proposed solutions; then, it presents the approach step by step; finally, it shows the detailed algorithm we designed.

4.3.1 Challenges and Proposed Solutions

In automatically inducing explanations from Linked Data, we face two specific challenges, which are addressed as follows.

Challenge 1. *If we index Linked Data in their entirety to create the background knowledge to reason upon, we will have to face scalability issues. For this reason, we propose to build the background knowledge iteratively, using Linked Data Traversal.*

As highlighted several times, one of the main advantages of Linked Data is the possibility of automatically exploring things by looking them up through dereferencing. Once dereferenced, entities reveal their connections to other entities that can be, in turn, further dereferenced. This means that the bigger is the number of dereferenced entities, the more RDF statements can be collected, and the bigger will be the size of the background knowledge that Dedalo uses to induce hypotheses, and of course the higher are the chances to find a good one. The search for candidate explanations needs therefore to be iteratively repeated over a constantly increasing background knowledge $\mathcal{B}$, with the hope of finding new (possibly) better candidate explanations.

Building $\mathcal{B}$ iteratively means exploring a graph G starting from the initial set of items $\mathcal{E}$ that we have in hand, and then gradually traversing entities following their links to other entities. In this way, no a priori knowledge is introduced in the process: there is no need of knowing which data sources exist in Linked Data; which ones should be used; which properties should be selected. The graph exploration is simply carried out by following existing links between Linked Data entities. Such traversal relies on the assumption that if data are connected (e.g. through *owl:sameAs*, *skos:exactMatch*, *rdfs:seeAlso* or vocabulary reuse), then data sources can be easily and naturally spanned to gather new, unknown knowledge, that we encode in $\mathcal{B}$ as new candidate explanations.

Challenge 2. *Even if we explore Linked Data iteratively, we risk spending too much time (and consequently, getting into scalability problems again) if we do not find the right direction to good explanations. For that, we can use a heuristic strategy to "greedily" navigate Linked Data searching for the best explanation.*

Even within an iterative process, one of the problems we face is that the size of the graph rapidly and significantly increases because of the number of relations existing between Linked Data entities. This means that the graph exploration has to be held heuristically so that only the portion of Linked Data that is expected to be useful to explain a particular pattern is explored. A second problem is that even once we have chosen a direction to follow in the graph, we do not know if the values that will be found will lead to produce good explanations. In other words, while we will not be able to determine which is the best explanation for a pattern (*global optimum*), we should be able to easily identify the very good ones (*local optima*).

Similar problems have been effectively solved with greedy algorithms: even without finding an optimal solution, a greedy search can yield very good solutions in a short time. A greedy search is a best-first search strategy that tries to minimise the cost to reach a goal state, by expanding first the node that is believed to be closest to it. With that in mind, what we propose is to explore Linked Data by relying on a greedy search, whose aim is to look in the graph for the node(s) v_j, which is likely to reveal the most appropriate explanations for the pattern. "Appropriate", in our case, means that the completeness of the explanation is maximised and its consistency is minimised. This estimation is stated based on the set of paths $\overrightarrow{p}$ known in G at a specific iteration: at each step, the greedy algorithm will choose to expand the values V_i of the path $\overrightarrow{p_i}$ that is judged to be the most likely to reveal new, and better, explanations.

As already indicated in Section 2.2.2, greedy algorithms are up to $O(b^m)$-complex in time and space; however, with a good heuristic function this complexity can be significantly reduced. In this scenario, another challenge for our process is to identify the strategy that most reduces these costs.

4.3.2 Description of the Process

The process that we propose is to repeat the steps shown in Figure 4.2 (we call the whole loop an *iteration*) until a termination condition is met. This is expressed by the user in terms of time spent, number of iterations achieved or a specific score to be reached by best explanation. An iteration then includes:

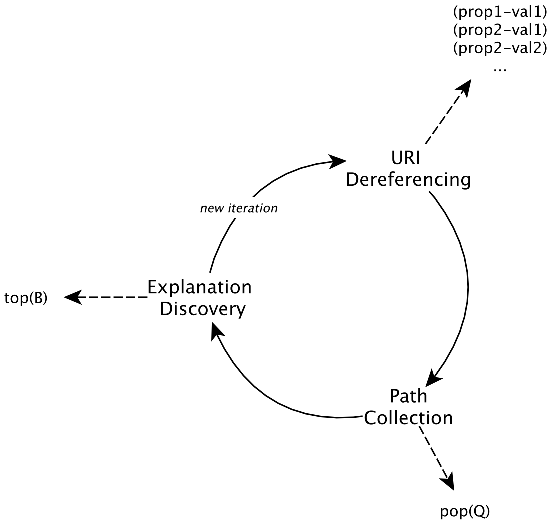


Figure 4.2 Schema of an iteration.

(1) *Graph Expansion*, where we dereference new entities to reveal new property-value couples;

(2) *Path Collection*, where we build new paths that can be followed in the following iterations;

(3) *Explanation Discovery*, where we create and evaluate new explanations.

At each iteration, the process collects new explanations and outputs the one with the highest score. All the explanations collected by the process up to that point are kept in a queue in a descending order based on their score. This means that, within a new iteration, two scenarios are possible: either we have found a better explanation, that covers more positive examples than the previous one and has therefore a better score, or no better hypotheses are found, and we keep the last iteration's hypothesis as the best one. In other words, with more iterations, results can only increase in quality. Therefore, Dedalo can be considered an *anytime process*.

Graph Expansion. An iteration starts by dereferencing the set V_i of ending values of a path $\overrightarrow{p_i}$ that has been chosen from a queue Q of possible paths to be followed. We call this operation *pop*(Q). Initially, the graph G is only composed of elements from $\mathcal{E}$, so $V = \mathcal{E}$ and $E = \emptyset$ (see Figure 4.3). Since Q is empty, we dereference elements of $\mathcal{E}$. When dereferencing the set of entities corresponding to the values in V_i, we obtain each of the pairs (property-value) linked to the entity. For instance, after the first iteration (Figure 4.4), we

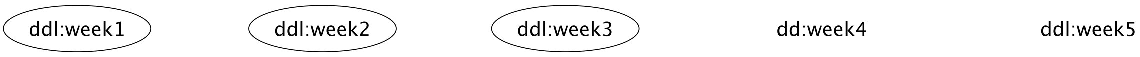


Figure 4.3 G at iteration (1).

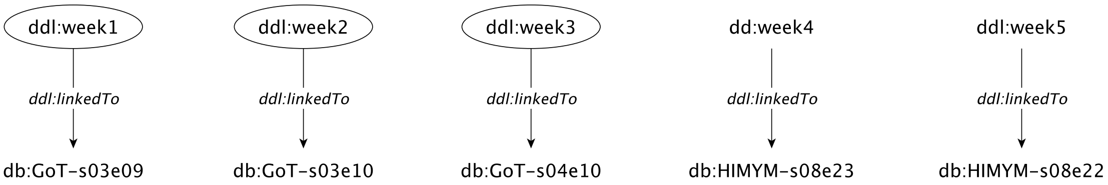


Figure 4.4 G at iteration (2).

obtain pairs as (ddl:linkedTo, db:GoT-s03e09) for ddl:week1, (ddl:linkedTo, db:GoT-s03e10) for ddl:week2, and so on. The value of each pair is added to V, while the property is added to E. Figure 4.5 and Figure 4.6 show an example of how G can be further increased throughout iterations.

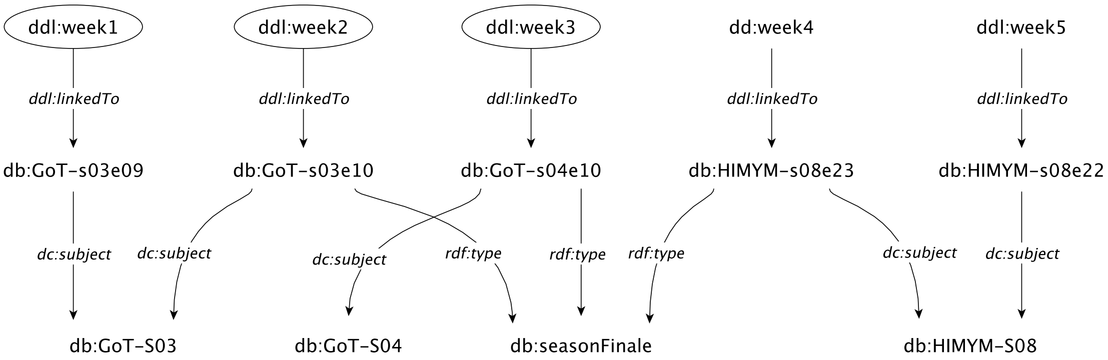


Figure 4.5 G at iteration (3).

As hinted by the namespaces, values might or might not be part of the same dataset, and this only depends on the representation of the entity that has been dereferenced. In this example, we have shown how from a dataset of weeks the search spread to DBpedia simply by following entity links (the property *ddl:linkedTo*).

Path Collection. Given a node $v_j \in V_i$, there exist a path $\overrightarrow{p_i}$ of length l that has led to it (unless v_j belongs to $\mathcal{E}$, then $l = 0$). For each of the (property-value) pairs revealed by the look up, the former are concatenated to the $\overrightarrow{p_i}$ that led there, to generate new paths of length $l + 1$. Those are added to the queue Q of paths, and the best one will be returned by $pop(Q)$ at the next iteration for further extending the graph. For instance, consider being in iteration (3) of Figure 4.5 and having reached the entity db:GoT-S04 from the path $\overrightarrow{p_1} =$

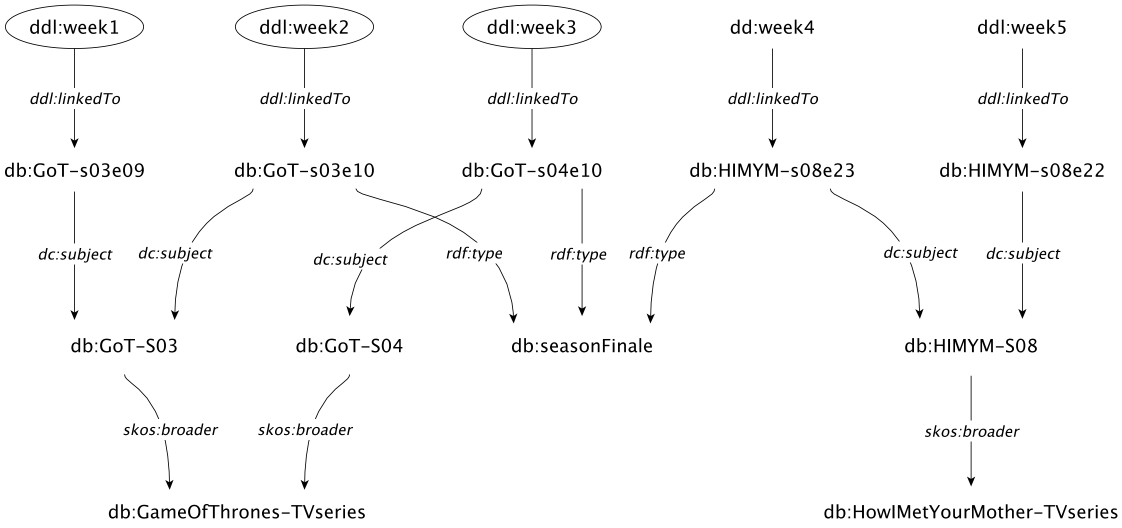


Figure 4.6 G at iteration (4).

$\langle$ddl:linkedTo.dc:subject$\rangle$. By dereferencing db:GoT-S04, the new property *skos:broader* is extracted from the discovered pair (skos:broader, db:GameOfThrones-TVseries); thus, a new path $\overrightarrow{p_2} = \langle$ddl:linkedTo.dc:subject.skos:broader$\rangle$ is built and added to Q. When dereferencing the other values of V_1, as for examples db:HIMYM-S08, the path $\overrightarrow{p_2}$ exists already in Q, then the value of the pair (skos:broader, db:HowIMetYourMother) is added to the set V_2 of ending values for $\overrightarrow{p_2}$.

Explanation Discovery. Before starting a new iteration, we build from each of the new paths the set of candidate explanations and then evaluate them[1]. Explanations are added to $\mathcal{B}$ (the list of possible explanations known at that given iteration), and the one(s) with the highest score is saved as *top*($\mathcal{B}$) for that current iteration.

To generate explanations, the paths $\overrightarrow{p_i}$ are chained to each of their ending values $v_j \in V_i$. We can obtain two types of explanations:

(1) $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$, if the last property p_l of $\overrightarrow{p_i}$ is an object property, as in

$$\varepsilon_{1,1} = \langle \text{ddl:linkedTo.rdf:type} \cdot \text{db:seasonFinale} \rangle$$

indicating that the popularity increases for those episodes that are at the end of a season;

(2) $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot \leq \cdot v_j \rangle$ or $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot \geq \cdot v_j \rangle$, if p_l is a numbered datatype property, e.g.

$$\varepsilon_{3,1} = \langle \text{ddl:linkedTo.airedIn} \cdot \leq \cdot 2014 \rangle$$

meaning that the popularity increases for the episodes aired before the year 2014.

[1] cf. the next subsection for the evaluation measures we used.

Note that the length of the path $\overrightarrow{p_i}$ in the explanations gives an insight of how much the graph has been traversed, i.e. how far has the search gone, and possibly how many iterations have been achieved[2]. In our examples, the best explanation for iteration (2) of Figure 4.4 is $\varepsilon_{1,1} = \langle$ddl:linkedTo.rdf:type · db:seasonFinale$\rangle$ of length $l = 2$. At the following iteration, in Figure 4.5, when new explanations will be built and evaluated, the best explanation detected is in fact $\varepsilon_{2,1} = \langle$ddl:linkedTo.dc:subject.skos:broader · db:GameOfThrones-TVseries$\rangle$ (length $l = 3$), since it represents more items in the pattern and less outside of it.

The iterative process here described is executed until a termination condition (time, number of iterations or a desired explanation score) is met. Next, we proceed by presenting the evaluation measures that we use to assess the validity of the hypotheses.

4.3.3 Evaluation Measures

To evaluate the generated explanations, we consider one of the three following possibilities.

Already presented in Section 3.3, the **F-Measure** (F) is the harmonic mean of Precision (P) and Recall (R), now redefined as:

$$P(\varepsilon_{i,j}) = \frac{|R_{i,j} \cap \mathcal{E}^+|}{|R_{i,j}|} \qquad (4.1) \qquad\qquad R(\varepsilon_{i,j}) = \frac{|R_{i,j} \cap \mathcal{E}^+|}{|\mathcal{E}^+|} \qquad (4.2)$$

The F-Measure evaluates a candidate explanation $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$ as $0 < F(\varepsilon_{i,j}) < 1$ based on the roots $R_{i,j}$ of a path $\overrightarrow{p_i}$, i.e. which are the entities that are connected to v_j through $\overrightarrow{p_i}$ and how many of those are in $\mathcal{E}^+$.

As from Section 3.3, we also consider the **Weighted Relative Accuracy** (*WR_{acc}*), that we redefine as:

$$WR_{acc}(\varepsilon_{i,j}) = \frac{|R_{i,j}|}{|\mathcal{E}|} \times \left(\frac{|R_{i,j} \cap \mathcal{E}^+|}{|R_{i,j}|} - \frac{|\mathcal{E}^+|}{|\mathcal{E}|} \right) \qquad (4.3)$$

This means that WR_{acc} also takes into consideration how big is the pattern $\mathcal{E}^+$ compared to the full dataset $\mathcal{E}$. As said, $-1 < WR_{acc}(\varepsilon_{i,j}) < 1$.

Fuzzy F-Measure (*FFM*) is a weighted variant of the F-Measure, which directly exploits the importance of items within the pattern they belong to.

[2] This is however highly dependent on the dataset.

As mentioned already in Chapter 3, both the WR_{acc} and the F evaluations are designed for classification tasks entailing a binary relation between an item and the pattern the item does (or does not) belong to. We introduce the *FFM* measure with the idea that items count differently within patterns (e.g. the concepts of cohesion, separation and silhouette in Cluster Analysis [Rousseeuw, 1987]), and that WR_{acc} and F clearly ignore such a "weight". For example, if ddl:week1 has a very high search rate for "A Song of Ice and Fire", then the evaluation measure needs to favour the events (and connected information, e.g. db:GameOfThrones-TVseries) that happened that week, giving those a higher priority than events happened on other weeks, in which the search rate is much lower. Such a priority is formalised by defining a weight for each of the items $e \in \mathcal{E}$ based on their distance d_e with the centre of the pattern they belong to. This means that our process assumes that the items given as input come with a distance metric that is used to compute their distance d_e to the center of the cluster (e.g. Euclidean distance to the centroid or absolute difference from the points' average in the case of one-dimensional data). This weight is then shaped depending on whether $\mathcal{E}$ belongs to the pattern to be explained:

$$w_e^+ = \frac{d_e}{max\{d_i \mid i \in \mathcal{E}^+\}} \iff e \in \mathcal{E}^+ \tag{4.4}$$

or to the rest of the dataset $\mathcal{E}$:

$$w_e^- = \frac{d_e}{min\{d_i \mid i \notin \mathcal{E}^+\}} \iff e \notin \mathcal{E}^+ \tag{4.5}$$

In this way, we make sure to favour both items that are in $R_{i,j} \cap \mathcal{E}^+$ (d_e in this case is positive) but also the ones that erroneously appear in $\mathcal{E} \setminus \mathcal{E}^-$ (d_e is negative).
Then, for an explanation $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$ shared by $|R_{i,j}|$ roots, we define:

(1) *Fuzzy True Positives* (*FTP*), the sum of weights w_e^+ for all the root items of $R_{i,j}$ which appear in $\mathcal{E}^+$:

$$FTP = \sum_{e \in R_{i,j} \cap \mathcal{E}^+} w_e^+ \tag{4.6}$$

(2) *Fuzzy False Positives* (*FFP*), the sum of weights w_e^- for all the root items of $R_{i,j}$ that are not in $\mathcal{E}^+$:

$$FFP = \sum_{e \in R_{i,j} \setminus \mathcal{E}^+} w_e^- \tag{4.7}$$

(3) *Fuzzy False Negatives* (*FFN*), the sum of weights w_e^+ for all the items of $\mathcal{E}^+$ not in $R_{i,j}$:

$$\mathit{FFN} = \sum_{e \,\in\, \mathcal{E}^+ \setminus R_{i,j}} w_e^+ \tag{4.8}$$

Finally, the Fuzzy Precision (FP) and Fuzzy Recall (FR) are defined similar to their usual definitions, such as:

$$\mathit{FP} = \frac{\mathit{FTP}}{\mathit{FTP} + \mathit{FFP}} \tag{4.9}$$

$$\mathit{FR} = \frac{\mathit{FTP}}{\mathit{FTP} + \mathit{FFN}} \tag{4.10}$$

and the Fuzzy F-Measure is defined as:

$$\mathit{FFM} = 2 * \frac{\mathit{FP} * \mathit{FR}}{\mathit{FP} + \mathit{FR}} \tag{4.11}$$

4.3.4 Final Algorithm

The final algorithm is shown in Algorithm 1 below.

Functions

- *add*(e_i, *list*) adds an element to a list;
- *pop*(*list*) gets and removes the best element from the list;
- *top*(*list*) gets the best element from the list;
- *expand*(v_i) dereferences the entity that labels a node v_i and returns a list of pairs (property-value);
- *concat*($item_i, item_j$) concatenates two unspecified items;
- *values*(p_i) returns V_i for $\overrightarrow{p_i}$;
- *roots*(p_i) returns R_i for $\overrightarrow{p_i}$;
- *roots*(p_i, v_j) returns $R_{i,j}$ for $\overrightarrow{p_i}$ and v_j;
- *evaluate*(e_i) evaluates a new explanation;
- *log*(e_i) outputs the best explanation *top*($\mathcal{B}$).

Algorithm 1 Dedalo's complete algorithm

```
stop = false                                          ▷ E.g. iterations or time limit
$V$, $Q$, $\mathcal{B}$ = list()                      ▷ Empty lists for collections
while (not stop) do
    top_p = pop($Q$)                                  ▷ The best path $\vec{p}$
    if top_p is null then                             ▷ Initially, expand $\mathcal{E}$
        V = $\mathcal{E}$
    else                                              ▷ Then, get the best path's ending values
        V = values(top_p)
    end if
    for value in V  do                                ▷ Graph Expansion
        predicates=expand(value)
        for (prop,item) in predicates do
            new_path = concat(top_p, prop)            ▷ Path Collection
            add(newPath, $Q$)
            explanation = concat(new_path, item)      ▷ Add new explanations
            evaluate(explanation)                     ▷ Evaluation
            add(explanation, $\mathcal{B}$)
        end for
    end for
    top_explanation=top($\mathcal{B}$)
    log(top_explanation)                              ▷ Best explanation for that iteration
end while                                             ▷ Start new iteration
```

4.4 Experiments

We divided this section into several evaluations. First, we aim at evaluating the best heuristic function for the greedy search: we evaluate different heuristic functions on the same datasets, to see which ones reach the highest score for the best candidate explanations in the shortest time. Second, we look at the type of explanations that can be reached throughout iterations. This helps us in assessing if the iterative process is justified, how many meaningful explanations are found when compared with an expert's knowledge, and if we indeed use knowledge across datasets as expected. We are also interested in knowing about the performance of the Fuzzy F-Measure, when compared to the other evaluation measures. Finally, to assess the validity of the process in terms of computational costs, we conduct a quantitative evaluation based on time spent, paths and graph size.

4.4.1 Use-cases

Before showing the experiments in details, we briefly describe the datasets that we have used. While the details of how those have been obtained are described in Appendix A.1, we are particularly interested in showing the reader the background knowledge $\mathcal{B}$ that is built after a fixed number of iterations in each use-case, so that the candidate explanations that will be shown in the following sections will be clearer. Note that, for clarity, $\mathcal{B}$ is shown as a tree where the elements of $\mathcal{E}$ are at the top (the positive examples are marked with a circle); however, this is a simplified version, and the graph that was explored had of course a much more complicated structure.

The use-cases we consider here are as follows.

#KMi.A – Already presented in Chapter 3, the dataset consists of 6 clusters of researchers grouped according to the publications that they have in common. An example of background knowledge is shown in Figure 4.7. Two clusters are used from #KMi.A: the "Semantic Web people" and the "Learning Analytics people".

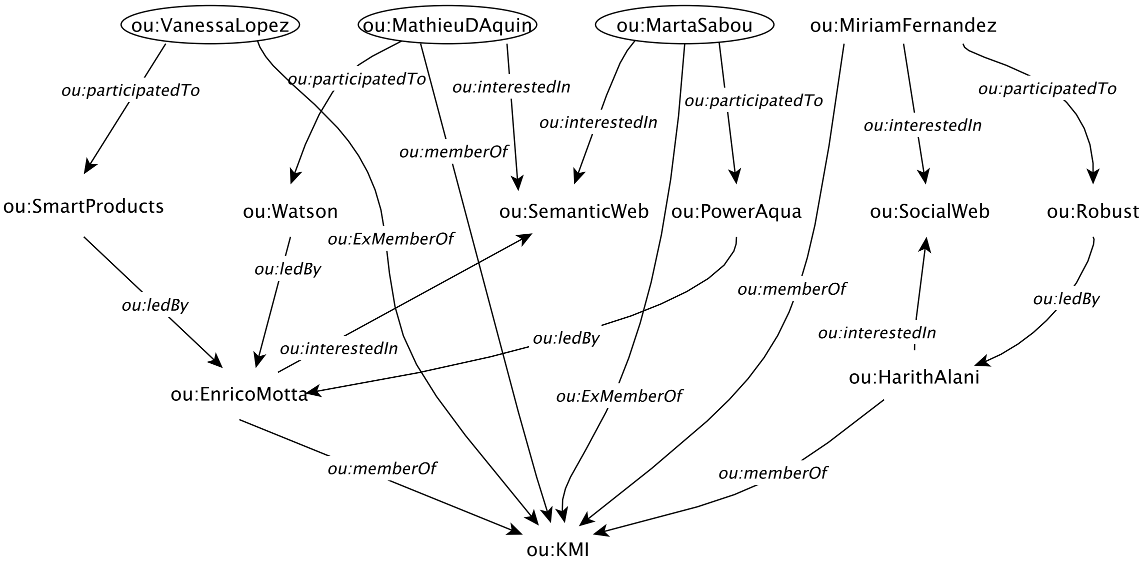


Figure 4.7 Extract from the background Knowledge for #KMi.A.

#Lit – The dataset consists of three clusters of countries grouped according to their male and female literacy rate: countries where women are more educated than men, countries where men are more educated than women, and countries where the education rate is approximately equal. In Figure 4.8, we show how we are trying to explain the cluster of countries where women are less educated.

#OU.P – It consists of 30 clusters of keywords extracted from research papers from the Open University, that have been clustered according to their co-occurrences. An example of the

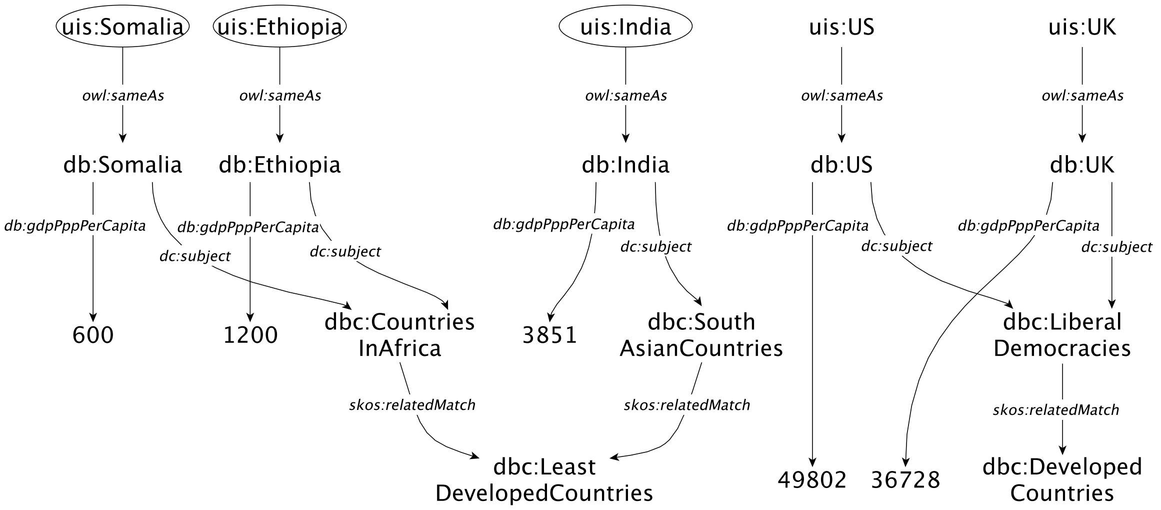


Figure 4.8 Extract from the background Knowledge for #Lit.

background knowledge to explain a cluster about Philosophy is shown in in Figure 4.9.

We also used #Tre, that we have used as running example throughout the chapter, and some clusters from #KMi.P and #KMi.H, that we have already seen in the experiments on ILP of Section 3.4.

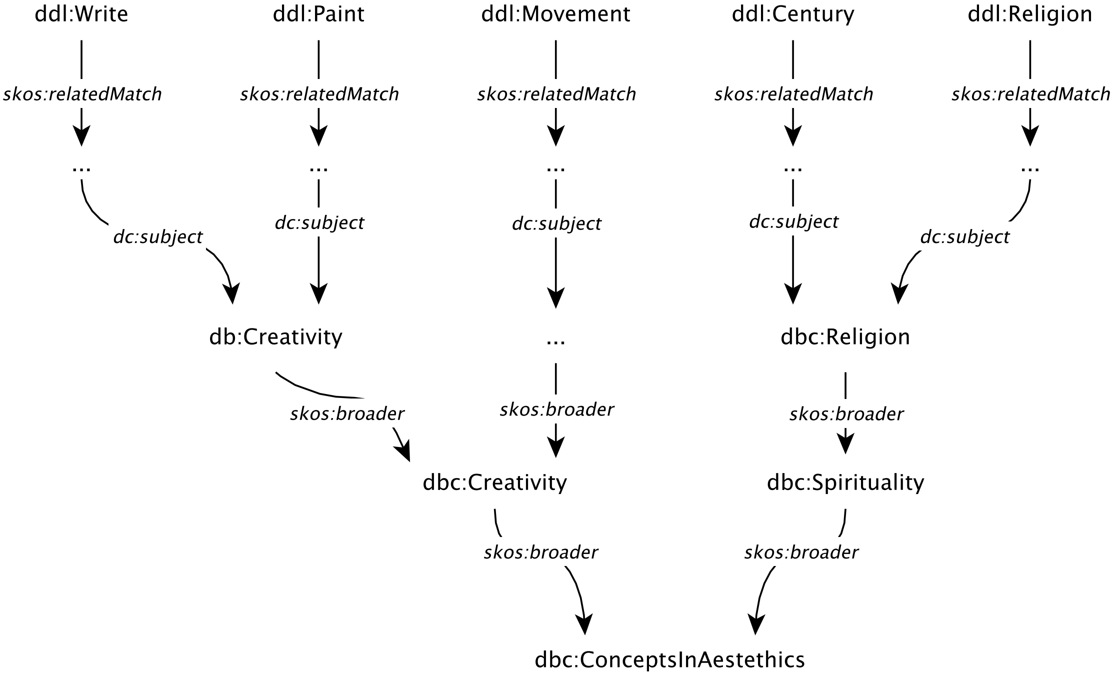


Figure 4.9 Extract from the background Knowledge for #OU.P.

4.4.2 Heuristics Comparison

It is well known that the greedy search is neither optimal nor complete, and can fall into dead-ends from which it is hard to roll back. Also, because we retain the graph nodes in memory (as the greedy search), the process can be computationally expensive due to the time and space complexity. However, with a good heuristic function, those issues can be sensibly reduced. Our objective is then to minimise this complexity: to do so, we adapt some existing measures to identify the most effective one, where effective means a measure giving the best candidate explanation score in the shortest time. We invite the reader to use Figure 4.6 as a reference for the examples.

1. *Shortest Path.* The baseline to which we compare the other measures is the length of $\overrightarrow{p_i}$. Shortest Path *SP*($\overrightarrow{p_i}$) assumes that the best paths are the shortest: it counts the number l of properties composing $\overrightarrow{p_i}$, favouring the shortest ones.

$$SP(\overrightarrow{p_i}) = \frac{1}{l} \tag{4.12}$$

   Ex. If $\overrightarrow{p_1} = \langle$ddl:linkedTo$\rangle$ and $\overrightarrow{p_2} = \langle$ddl:linkedTo.dc:subject$\rangle$, then *SP*($\overrightarrow{p_1}$) $>$ *SP*($\overrightarrow{p_2}$).

2. *Path Frequency. Freq* estimates the frequency of a path $\overrightarrow{p_i}$ in G by considering the size of its root set R_i. It assumes that the most important paths are the most frequent.

$$Freq(\overrightarrow{p_i}) = \frac{|R_i|}{|\mathcal{E}|} \tag{4.13}$$

   Ex. With $\overrightarrow{p_2}$ as above, and $\overrightarrow{p_3} = \langle$ddl:linkedTo.rdf:type$\rangle$, then *Freq*($\overrightarrow{p_2}$) $>$ *Freq*($\overrightarrow{p_3}$) because the number of roots for the latter is lower.

3. *Pointwise Mutual Information. PMI* is used in Information Theory and Statistics to measure the discrepancy of a pair of random variables X and Y given their joint distribution $P(X|Y)$ and individual distributions $P(X)$ and $P(Y)$. In our scenario, we measure the probability that $\overrightarrow{p_i}$ is shared by its roots compared to the positive examples $\mathcal{E}^+$.

$$PMI(\overrightarrow{p_i}) = log\left(\frac{|R_i \cap \mathcal{E}^+|}{|\mathcal{E}| \times |R_i|}\right) \tag{4.14}$$

   Ex. With $\overrightarrow{p_2}$ and $\overrightarrow{p_3}$ as above, then *PMI*($\overrightarrow{p_2}$) $>$ *PMI*($\overrightarrow{p_3}$) because $\overrightarrow{p_2}$ has more roots belonging to $\mathcal{E}^+$ than $\overrightarrow{p_3}$.

4. *Adapted TF-IDF.* We adapted the very well-known TF-IDF measure to evaluate the relevance of a path $\overrightarrow{p_i}$ (the *term*) in a given pattern $\mathcal{E}^+$, compared to the frequency of the

path across the rest of the $|K|$ patterns (the *documents*) into which $\mathcal{E}$ is divided.

$$TF\text{-}IDF(\overrightarrow{p_i}) = \frac{|R_i \cap \mathcal{E}^+|}{|\mathcal{E}^+|} \times log \frac{|K|}{|\{\mathcal{E}_j \in K | R_i \cap \mathcal{E}_j \neq \emptyset\}|} \tag{4.15}$$

Ex. *TF-IDF*($\overrightarrow{p_3}$) $>$ *TF-IDF*($\overrightarrow{p_2}$) since $\overrightarrow{p_3}$ is less frequent and more specific to elements in $\mathcal{E}^+$ than $\overrightarrow{p_2}$.

5. *Delta function.* We developed a function comparing the number of values V_i for a $\overrightarrow{p_i}$ and the number of patterns in the dataset. Delta (Δ) assumes that the best $\overrightarrow{p_i}$ is the one having one different end value $v_j \in V_i$ for each pattern in $\mathcal{E}$. A path's score is better the closer the cardinality of $|V_i|$ is to $|K|$, i.e. the number of patterns into which the dataset is split.

$$\Delta(\overrightarrow{p_i}) = \frac{1}{1 + |V_i| - |K|} \tag{4.16}$$

Ex. Given the two patterns of popular and not popular weeks (so $K = 2$), $\overrightarrow{p_4} =$ ⟨ddl:linkedTo.dc:subject. skos:broader⟩ is the one getting a better score, because $|V_4| = 2$, so $\Delta(\overrightarrow{p_4}) = 1$. On the other hand, Δ disfavours $\overrightarrow{p_1}$ attributing it a lower score since its values are too sparse (i.e. $|V_1| = 5$).

6. *Entropy.* Entropy [Shannon, 2001] (defined by the Greek letter *eta*, H) is an Information Theory measure analysing the performance of communication channels, and has been applied to network graphs in a broad range of disciplines [Dehmer and Mowshowitz, 2011; Mowshowitz and Dehmer, 2012]. Given a random process $X = \{x_0, x_1, ..., x_n\}$ with n possible outcomes, the amount of uncertainty removed by equiprobable messages increases monotonically with the number of existing messages, meaning that the bigger is n, the less information is gained (and the more X is uncertain). Considering this, we used a naïve adaptation of Shannon's Entropy, in which the random process X corresponds to $\overrightarrow{p_i}$, while its n possible outcomes are the values $v_j \in V_i$.

$$H(\overrightarrow{p_i}) = -\sum_{i=1}^{j=|V|} \frac{|R_{i,j}|}{|E|} log \frac{|R_{i,j}|}{|E|} \tag{4.17}$$

Ex. There is less information gain with $\overrightarrow{p_3}$ because it only has one possible outcome. The gain of information is much higher with $\overrightarrow{p_2}$, whose number of values increases the path uncertainty. Therefore, $H(\overrightarrow{p_2}) > H(\overrightarrow{p_3})$.

7. *Conditional Entropy.* Conditional Entropy measures the information gain of a random variable X given the knowledge of a variable Y. In this scenario, $H(\overrightarrow{p_i}|\mathcal{E}^+)$ measures

how much information gain $\overrightarrow{p_i}$ brings, given the items known in $\mathcal{E}^+$ (i.e. how specific $\overrightarrow{p_i}$ and its values are in $\mathcal{E}^+$).

$$H(\overrightarrow{p_i}|\mathcal{E}^+) = -\sum_{i=1}^{j=|V|} \frac{|R_{i,j} \cap \mathcal{E}^+|}{|E|} log \frac{|R_{i,j} \cap \mathcal{E}^+|)}{|R_{i,j}|} \tag{4.18}$$

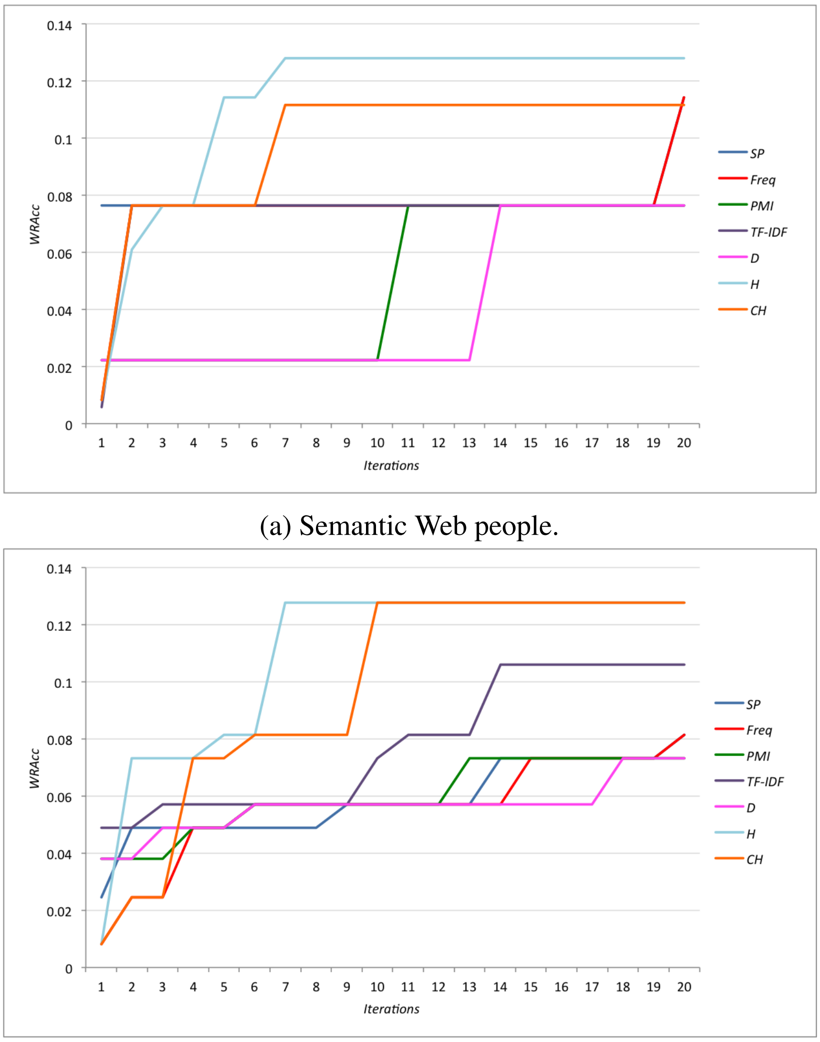
With $\overrightarrow{p_2}$ and $\overrightarrow{p_3}$, then $H(\overrightarrow{p_2}) > H(\overrightarrow{p_3})$ because $\overrightarrow{p_2}$ has more roots in $\mathcal{E}^+$ and more uncertainty than $\overrightarrow{p_3}$.

Results. We compared the measures above presented when applying Dedalo's implemented Algorithm 1 on the #KMi.A, #KMi.P and #KMi.H datasets. Our aim is to see which is the fastest at reaching the best explanation given a fixed number of iterations. For this evaluation, we use the *WR*$_{acc}$ score.

In Figures 4.10, 4.11, 4.12, the X axis represents the number of iterations that the process was run for, and the Y axis represents the score of the best explanation *top*($\mathcal{B}$) found at that given iteration. As we explained, each improvement of the *WR*$_{acc}$ score means that a new explanation has been found.

The experiments clearly show that Entropy outperforms the other measures. This means that we can use Entropy to reduce redundancy (i.e. avoiding to follow wrong paths) and to detect the best explanations in a short time. Conditional Entropy, showing a very good performance as well, is the second best performing in 5 of out 6 experiments. In Figure 4.12b, Conditional Entropy even finds better explanations for the pattern. Probably, the reason can be attributed to the fact that items of that pattern had explanations that were more specific when compared to the rest of the dataset.

As for the other measures, *PMI*, *TF-IDF* and Δ have the worst performances, possibly because the use-cases are too homogeneously composed and each entity, regardless which pattern it belonged to, had approximately the same properties. For instance, *TF-IDF* works relatively well in the case illustrated in Figure 4.10b. In that case, we were dealing with a more heterogeneous pattern of data. *SP* and *Freq* are good in finding candidate explanations in the first iterations, but then they plateau and take time before getting any improvement. In other words, they are able to find the correct path only if this appears first in the queue by chance. Shortest Path also seems to have a better performance on big patterns with items having smaller numbers of properties, as shown in Figures 4.12a and 4.12b. Given such a visible difference of performance between Entropy and the other measures, we can consider it as the right candidate to use for our process. With that said, the possibility of combining it with other measures can be foreseen as a plausible future work.



(a) Semantic Web people.

(b) Learning Analytics people.

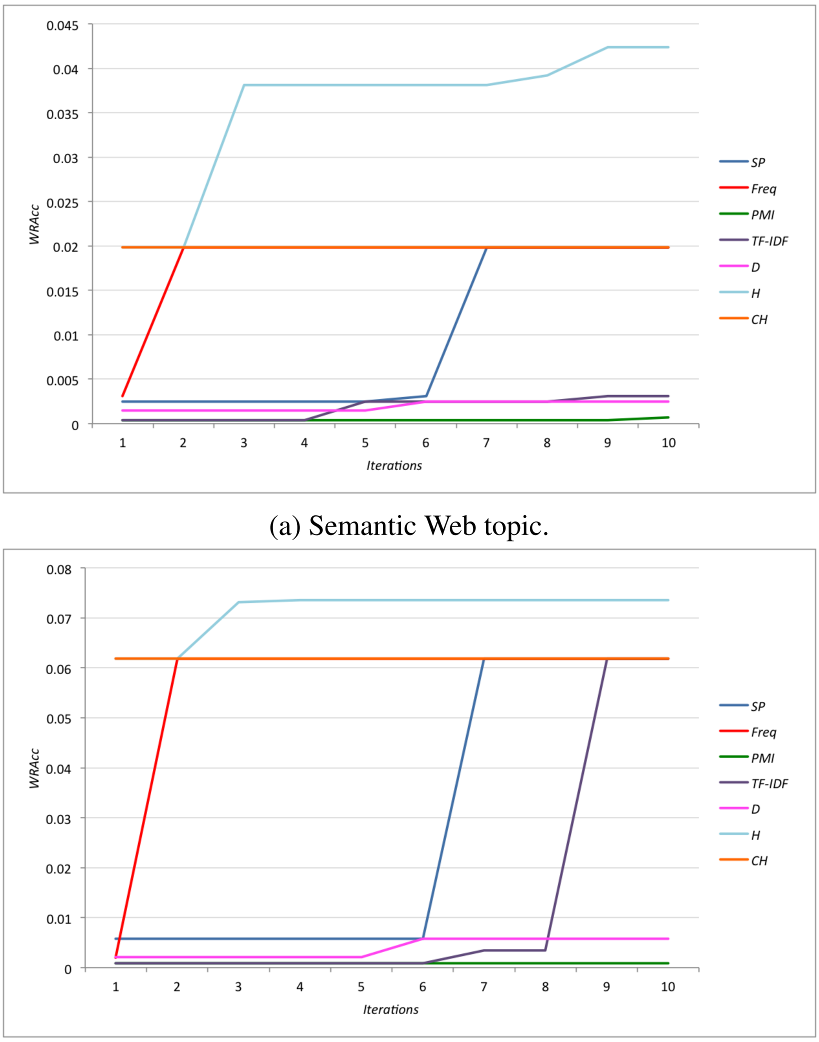
Figure 4.10 #KMi.A results. The process was run for 20 iterations.

The experiments also showed an apparent phenomenon that the bigger the dataset, the lower is WR_{acc}. This can probably be explained by the fact that it is harder to find strong candidate explanations in a larger population.

4.4.3 Best Explanations

Here, we present some analysis on the type of best explanations that we have found in different use-cases.

Explanation understandability. Examples of the best explanations $top(\mathcal{B})$ for the datasets #KMi.A, #KMi.P and #KMi.H are presented in Table 4.1. The left column shows the size of



(a) Semantic Web topic.

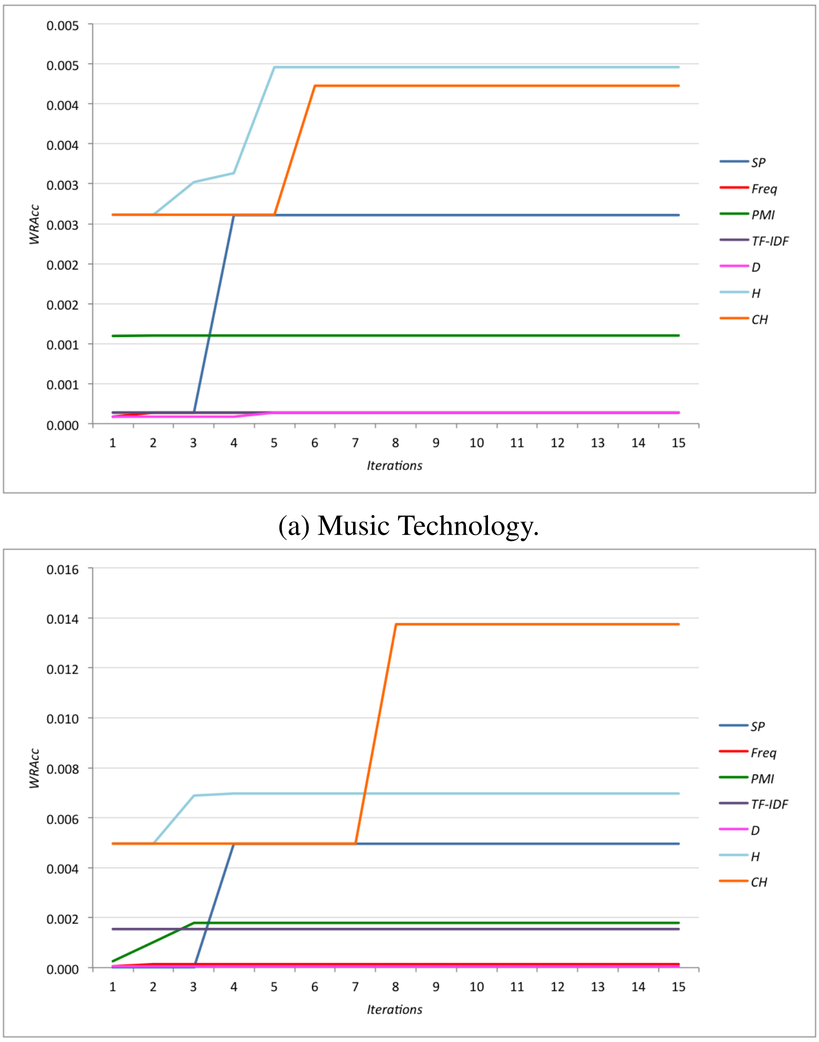
(b) Learning analytics topic.

Figure 4.11 #KMi.P results after 10 iterations.

the pattern and the label of the cluster, that we manually defined based on our knowledge of the KMi department. Interestingly, the results demonstrate that our process is agnostic to the type of pattern (type of clustering process, size or domain), since candidate explanations comparable to the human-provided ones are obtained in each of the use-cases.

In the first case of #KMi.A, the second explanation for the items' grouping is that people who worked on Semantic Web-related themes are together because they have all been part of a project whose director was someone working himself on the SmartProducts project[3] (with a WR_{acc} score of 0.128), which is more distant in G than the first explanation (those people are associated to the Semantic Web topic, $WR_{acc} = 0.076$). We remind here that WR_{acc} can go

[3] http://www.smartproducts-project.eu/



(a) Music Technology.

(b) Theatre.

Figure 4.12 #Kmi.H results for 15 iterations.

into negative values, therefore our results can be considered acceptable. Also, lower WR_{acc} scores are likely to happen when the cluster is very small when compared to the rest of the dataset. What it is interesting to notice is that such explanations could only be given by someone knowing the KMi department well enough to affirm that those people worked in projects under the same director. Typically, this is an example in which explanations are hidden and only an expert with the right background knowledge can provide it.

We also remark how, by using the connections between datasets in the Linked Data cloud, we also get better explanations. Referring to the first example for #KMi.H: first, we get explanations from the British Library dataset ("books borrowed by students of the Music Technology faculty are about sound recording", $WR_{acc} = 0.002$); then, we extend the graph,

Table 4.1 Examples of explanations for #KMi.A, #KMi.P and #KMi.H.

#KMi.A – People working on		*WR*$_{acc}$
Semantic Web (22)	$\varepsilon_{1,1} =$ ⟨tag:taggedWithTag · ou:SemanticWeb⟩	0.076
	$\varepsilon_{2,1} =$ ⟨org:hasMembership.ox:hasPrincipalInvestigator.org:hasMembership · ou:SmartProducts⟩	0.128
Learning Technology (23)	$\varepsilon_{1,1} =$ ⟨org:hasMembership · ou:open-sensemaking-communities⟩	0.073
	$\varepsilon_{2,1} =$ ⟨org:hasMembership.ox:hasPrincipalInvestigator.org:hasMembership · ou:SocialLearn⟩	0.127

#KMi.P – Papers on		*WR*$_{acc}$
Learning Analytics (601)	$\varepsilon_{1,1} =$ ⟨dc:creator.org:hasMembership · ou:StoryMakingProject⟩	0.038
	$\varepsilon_{2,1} =$ ⟨dc:creator.org:hasMembership.ox:hasPrincipal-Investigator.tag:isRelatedTo · ou:LearningAnalytics⟩	0.042
Semantic Web (220)	$\varepsilon_{1,1} =$ ⟨dc:creator · ou:EnricoMotta⟩	0.061
	$\varepsilon_{2,1} =$ ⟨dc:creator.ntag:isRelatedTo · ou:SemanticWeb⟩	0.073

#KMi.H – Books borrowed by students from		*WR*$_{acc}$
Music Technology (335)	$\varepsilon_{1,1} =$ ⟨dc:subject · bnb:SoundsRecording⟩	0.002
	$\varepsilon_{2,1} =$ ⟨dc:creator.bnb:hasCreated.dc:subject · bnb:SoundsRecording⟩	0.005
	$\varepsilon_{3,1} =$ ⟨dc:creator.owl:sameAs.skos:broader.skos:broader.skos:broader · lcsh:PhysicalScience⟩	0.005
Theatre (919)	$\varepsilon_{1,1} =$ ⟨dc:subject · bn:EnglishDrama⟩	0.004
	$\varepsilon_{2,1} =$ ⟨dc:creator.owl:sameAs.skos:narrower · lcsh:EnsembleTheatre⟩	0.007
	$\varepsilon_{3,1} =$ ⟨dc:creator.bnb:hasCreated.dc:subject · bnb:EnglishDrama⟩	0.013

reach the Library Of Congress dataset and find a better explanation, such as "books borrowed by students of the Music Technology faculty are about a narrower topic of Physical Science" (*WR*$_{acc} = 0.005$). This shows that more accurate explanations can be found using Linked Data connections among datasets and domains, consistently with the findings in Chapter 3.

Explanation improvement. To have an idea of how explanations are improving with iterations and the growth of the background knowledge, we use the data from the #OU.P use-case. Table 4.2 gives an overview of how explanations improve for three chosen groups of keywords (what the papers are about), by showing the best explanation at each iteration i, as well as its F-Measure, and the size of $R_{i,j}$.

Again, within few iterations we automatically span from our dataset to DBpedia, and manage to build explanations that generalise the groups and explain why words appear together. Thanks to Entropy, we detect that paths based on the properties *dc:subject* and *skos:broader* are the most promising for climbing up the taxonomy of DBpedia concepts

Table 4.2 Examples of explanations for three clusters in #OU.P.

i	*top*($\mathcal{B}$)	F
1	⟨skos:relatedMatch · db:Century⟩	0.08
2	⟨skos:relatedMatch.dc:subject · dbc:Writing⟩	0.11
3	⟨skos:relatedMatch.dc:subject.skos:broader · dbc:Problem_solving⟩	0.13
4	⟨skos:relatedMatch.dc:subject.skos:broader.skos:broader · dbc:Creativity⟩	0.18

i	*top*($\mathcal{B}$)	F
1	⟨skos:relatedMatch · db:Scale⟩	0.03
2	⟨skos:relatedMatch.dc:subject · dbc:ConceptsInPhysics⟩	0.10
3	⟨skos:relatedMatch.dc:subject.skos:broader · dbc:Physics⟩	0.14
4	⟨skos:relatedMatch.dc:subject.skos:broader.skos:broader · dbc:FieldsOfMathematics⟩	0.18
5	⟨skos:relatedMatch.dc:subject.skos:broader.skos:broader.skos:broader · dbc:Mathematics⟩	0.21

i	*top*($\mathcal{B}$)	F
1	⟨skos:relatedMatch · db:Social⟩	0.06
2	⟨skos:relatedMatch.dc:subject · dbc:Concepts_in_Logics⟩	0.11
3	⟨skos:relatedMatch.dc:subject.skos:broader · dbc:Logic⟩	0.15
4	⟨skos:relatedMatch.dc:subject.skos:broader.skos:broader · dbc:Abstraction⟩	0.18

and induce strong candidate explanations. With this strategy, we can see how *top*($\mathcal{B}$) significantly improves in a short time (in the second example, we get from "3% of the words match the DBpedia concept db:Scale" to "more than 20% are words about subcategories of Mathematics"). Other examples of best explanations for #OU.P are given in Table 4.3.

Table 4.3 Explanations found by Dedalo after 5 iterations on #OU.P, their Precision (P), Recall (R) and F-Measure (F). For readability, the apex on skos:broader indicates a chain of the property.

top($\mathcal{B}$)	P	R	F
⟨skos:relatedMatch.dc:subject.skos:broader^3· dbc:Geology⟩	0.94	0.20	0.33
⟨skos:relatedMatch.dc:subject.skos:broader^3· dbc:Chemistry⟩	0.70	0.28	0.23
⟨skos:relatedMatch.dc:subject.skos:broader^3· dbc:Astronomy⟩	0.43	0.15	0.22
⟨skos:relatedMatch.dc:subject.skos:broader^3· dbc:Mathematics⟩	0.37	0.15	0.21
⟨skos:relatedMatch.dc:subject.skos:broader^2· dbc:Creativity⟩	0.24	0.14	0.18
⟨skos:relatedMatch.dc:subject.skos:broader^2· dbc:Abstraction⟩	0.17	0.19	0.18
⟨skos:relatedMatch.dc:subject.skos:broader^2· dbc:Management⟩	0.29	0.12	0.17
⟨skos:relatedMatch.dc:subject.skos:broader^3· dbc:Leadership⟩	0.24	0.13	0.17

Explanations with datatypes. Examples of candidate explanations with datatype properties are obtained from the #Lit dataset and shown in Table 4.4. In those cases, given a numerical value $v_j \in V_i$, we create two alternative explanations: (1) $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot \geq \cdot v_j \rangle$ and (2) $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot \leq \cdot v_j \rangle$. Given the roots in R_i, we check if the value each of them is walking to through $\overrightarrow{p_i}$ is greater or smaller than the value v_j, and subsequently evaluate either (1) or (2) accordingly.

Let us consider the graph G in Figure 4.8 as an example. For $\overrightarrow{p_1} = \langle$owl:sameAs.dbp:gdpPppPerCapita$\rangle$, the entity uis:Ethiopia ends to the value $v_1 = 1,200$, uis:Somalia ends to the value $v_2 = 600$ and uis:India to $v_3 = 3,851$. Since creating two alternate explanations for each value can make the candidate explanation space larger, we eventually only keep the one with the best score with respect to the pattern $\mathcal{E}^+$, as Table 4.4 shows. Other explanations with datatype properties are shown in Table 4.6.

Table 4.4 Example of the production of explanations for numeric values on the #Lit use-case.

$\varepsilon_{i,j}$	F
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \geq \cdot$ 600$\rangle$	0.75
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \leq \cdot$ 600$\rangle$	0.50
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \geq \cdot$ 1200$\rangle$	0.57
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \leq \cdot$ 1,200$\rangle$	0.80
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \geq \cdot$ 3,851$\rangle$	0.33
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \leq \cdot$ 3,851$\rangle$	**1.00**
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \geq \cdot$ 49,802$\rangle$	0.00
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \leq \cdot$ 49,802$\rangle$	0.75
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \geq \cdot$ 36,728$\rangle$	0.00
$\langle$owl:sameAs.dbp:gdpPppPerCapita$\cdot \leq \cdot$ 36,728$\rangle$	0.85

Fuzzy F-Measure Performance. We compare here explanations evaluated by the F-Measure with the ones evaluated by the Fuzzy F-Measure. As said, our intention is to show that items in a pattern have different importance, and when not taking this into consideration, inaccurate explanations might be wrongly validated as the best ones. In Table 4.5, we show how the evaluations performed with the F-Measure and with the Fuzzy F-Measure differ. We manually chose some of the best explanations (at iteration 20) for four search terms of #Tre, and then compared the score each of the measures have attributed to them, as well as their ranking in $\mathcal{B}$.

As one can see, the Fuzzy F-Measure is more precise in ranking the correct candidate explanations, while the normal F-Measure, which tends to give the same importance to each item within the cluster, misevaluates the correct explanations putting them much lower in the rank.

4.4.4 Time Evaluation

Finally, we are interested evaluating Dedalo's performance on a time perspective. For that, we use the three KMi datasets, #Lit and #Tre. Unless specified, all the experiments have

Table 4.5 Chosen candidate explanations at iteration 20 in #Tre. The Fuzzy F-Measure score *S(ff)* and ranking *R(ff)* significantly improve over the ones of the normal F-Measure (*S(fm)* and *R(fm)*).

#Tre search term: Brazil	*S(ff)*	*S(fm)*	*R(ff)*	*R(fm)*
$\varepsilon_{1,1}$ = ⟨ddl:linkedTo · dbp:2014FIFAWorldCup⟩	0.88	0.52	1	4
$\varepsilon_{2,1}$ = ⟨ddl:linkedTo.dc:subject · dbc:FIFAWorldCupTournaments⟩	0.87	0.64	2	1
$\varepsilon_{2,2}$ = ⟨ddl:linkedTo.dc:subject · dbc:2014FIFAWorldCup⟩	0.82	0.50	4	6

#Tre search term: Obama	*S(ff)*	*S(fm)*	*R(ff)*	*R(fm)*
$\varepsilon_{1,1}$ = ⟨ddl:linkedTo.dc:subject · dbc:UnitedStatesSenateElectionsInWestVirginia⟩	0.72	0.12	1	100+
$\varepsilon_{1,2}$ = ⟨ddl:linkedTo.dc:subject · dbc:HistoryOfTheUS(1991-present)⟩	0.71	0.16	5	100+
$\varepsilon_{1,3}$ = ⟨ddl:linkedTo.dc:subject · dbc:USpresidentialElectionInWashington(state)⟩	0.66	0.20	6	100+

#Tre search term: A Song of Ice and Fire	*S(ff)*	*S(fm)*	*R(ff)*	*R(fm)*
$\varepsilon_{1,1}$ = ⟨ddl:linkedTo.dc:subject.skos:broader.skos:broader · dbc:ASongOfIceAndFire⟩	0.65	0.39	7	100+
$\varepsilon_{2,1}$ = ⟨ddl:linkedTo.dc:subject.skos:broader · dbc:GameOfThrones(TVseries)⟩	0.65	0.39	7	100+
$\varepsilon_{3,1}$ = ⟨ddl:linkedTo.dc:subject · dbc:GameOfThrones(TVseries)⟩	0.65	0.38	7	100+

#Tre search term: Daniel Radcliffe	*S(ff)*	*S(fm)*	*R(ff)*	*R(fm)*
$\varepsilon_{1,1}$ = ⟨ddl:linkedTo.dc:subject.skos:broader · dbc:HarryPotter⟩	0.35	0.20	18	100+
$\varepsilon_{1,2}$ = ⟨ddl:linkedTo.dc:subject.skos:broader · dbc:21centuryBritishChildrenLiterature⟩	0.35	0.20	18	100+

been run on a 2.9 GHz Intel Core i7 machine with 8GB of RAM.

In a first instance (see Figure 4.13), we compared the time each of the heuristic measures needed to reach the same explanation, i.e. the best one after a fixed number of iterations (20, 10 and 15 respectively for #KMi.A, #KMi.P and #KMi.H). In most of the examples, relatively to the scale of the dataset, Entropy is among the fastest measures also in time, while Conditional Entropy appears slightly slower.

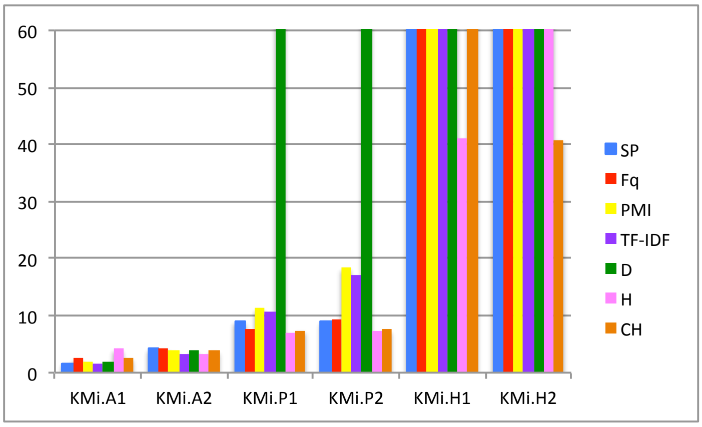


Figure 4.13 Time (in seconds) the heuristics need to reach the chosen $\varepsilon_{i,j}$.

Secondly, we are interested in knowing how much time it takes to reach the same explanation as a human would naturally give, how much it fits the pattern $\mathcal{E}^+$ and how far it is from the sources, as well as how big is the graph at the moment of the discovery. For that, we use the #Lit dataset since it is a more of a real-world scenario. This is a preliminary step for the broader evaluation of Chapter 7, in which we evaluate explanations obtained automatically with the ones given by the users. Table 4.6 shows the results we had after 10 iterations for the main #Lit groups. Time is evaluated in seconds taken to reach the explanation *top*($\mathcal{B}$). In 10 iterations, the graph G has 3,742,344 triples and the size of Q is 671 paths.

Table 4.6 Best explanations found for #Lit, the time it has taken to find them, and their Precision, Recall and F-Measure.

top($\mathcal{B}$) for countries where men are more educated (♂)	P	R	F	Time
⟨skos:exactMatch.dbp:hdiRank · ≥ · 126⟩	0.85	0.85	0.85	197”
⟨skos:exactMatch.dc:subject · dbc:LeastDevelopedCountries⟩	0.92	0.65	0.76	524”
⟨skos:exactMatch.dbp:gdpPppPerCapitaRank · ≥ · 89⟩	0.55	0.92	0.68	269”
⟨skos:exactMatch.dc:subject skos:broader · dbc:CountriesInAfrica⟩	0.79	0.61	0.69	40”
⟨skos:exactMatch.dbp:populationEstimateRank · 76⟩	0.49	0.96	0.62	201”
⟨skos:exactMatch.dbp:gdpPppRank · ≥ · 10⟩	0.44	0.91	0.59	235”

top($\mathcal{B}$) for countries where women are more educated (♀)	P	R	F	Time
⟨skos:exactMatch.dbpedia:hdiRank · ≤ · 119⟩	0.64	0.64	0.64	198”
⟨skos:exactMatch.dbp:gdpPppRank · ≤ · 56⟩	0.48	0.88	0.62	236”
⟨skos:exactMatch.dbp:populationEstimateRank· ≥ · 128⟩	0.67	0.50	0.57	203”
⟨skos:exactMatch.dbp:gdpPppPerCapitaRank · ≤ · 107⟩	0.48	0.76	0.56	267”
⟨skos:exactMatch.dbp:gdpPppPerCapitaRank · ≥ · 100⟩	0.47	0.64	0.54	267”
⟨skos:exactMatch.dc:subject.skos:broader · dbc:LatinAmericanCountries⟩	0.50	0.49	0.49	542”

To get the best explanation for the group ♂, the process requires less than 200”. The explanation shows that the 88% of the countries in ♂ are ranked more than 126 in the Human Development Index[4] (HDI). Based on statistics on life expectancy, education and income, the HDI ranks countries from the most to the least developed one. The lower the country is in the rank, the less developed it is. Similarly, the best explanation for ♀ is that the 63.4% of its countries are among the 119 most developed countries. It is important to recall that such an explanation would have not been found without any reasoning upon numerical values.

On the other hand, we remark how the process found object property-based explanations as good as the data property ones: for instance, the second best explanation for ♂ is that the 75% of the group is labelled in DBpedia as least developed countries (i.e. those countries share the path $\overrightarrow{p_i} = \langle$skos:exactMatch.dc:subject$\rangle$ to the common node dbc:LeastDevelopedCountries), which can be considered as an alternative formulation of the HDI explanation. While we showed that good datatype explanations could be obtained in our process, in the rest of the work we will focus on object property explanations because those

[4] http://en.wikipedia.org/wiki/Human_Development_Index

can be exploited by the graph traversal.

Finally, we use the #Tre use-case to evaluate the time Dedalo needs when dealing with much larger datasets. Those experiments have been run on a 16-cores Intel Xeon E5640 2.66GHz machine with 28G of RAM operating under Linux Red Hat 6. Table 4.7 shows the time taken by each of the tests (labelled with the searched term), as well as the number of explanations that have eventually been found.

Table 4.7 #Tre experiments details: trends are ordered by the time of the process, in the second column. The third column shows the number of explanations found after 20 iterations.

Searched term	Time	Num. $\varepsilon_{i,j}$
Daniel Radcliffe	3h30m04s	233,700
Brazil	3h38m31s	104,092
A Song of Ice and Fire	5h48m10s	78,407
Obama	6h44m42s	150,194
Dakar	6h45m12s	32,784
Turkey	6h47m43s	156,996
Hurricane	6h57m48s	116,558
Rugby	7h07m43s	184,870
Wembley	7h15m57s	146,732
Italy	7h20m13s	55,210
Germany	7h28m35s	399,072
Taylor	9h50m07s	114,364
How I met your Mother	12h10m22s	17,131

While the time does not seem to be affected by the number of explanations evaluated, it remains of course influenced by the Link Traversal, which relies on the availability of external servers, as well as the amount of data that are retrieved. In order to avoid the same information being fetched more than once, we therefore included in Dedalo a caching system to access some of the data locally. We are aware that more effort could be put in reducing the time of the process; nevertheless, we consider this manageable for the current work, and leave the optimisation of this process to future work.

4.5 Conclusions and Limitations

In this chapter we have presented a process to automatically induce candidate explanations for a given pattern using Linked Data.

We have shown how we based the process on the Inductive Logic Programming framework, which presents interesting features such as automatically inducing facts based on some background knowledge, but how we had to adapt it to make it able to deal with a more complex problem space as the one of the Web of Data. To face the scalability issues, we proposed to create a process that iteratively revises and increases the background knowledge for induction using new Linked Data statements extracted on-the-fly through Link Traversal. Moreover, to avoid wasting time in exploring parts of the graph that are not useful to explain a pattern, we proposed to use a greedy search based on entropy, that the experiments have revealed being the right heuristic to drive this search, as it is able to find accurate explanations in a short number of iterations. Finally, we showed how the accuracy of the candidate explanations could be improved by taking into account the importance of the items within the pattern that we are aiming at explaining during the evaluation of the explanations.

The presented approach has limitations that open new issues. We present them below, followed by a recall of the issues that we have left open so far.

❶ **Explanations are atomic.** In the ILP framework, $\mathcal{B}$ is used to build a lattice, in which statements are combined with each other and organised from the most general to the most specific ones (in terms of accuracy). In such a manner, the ILP approaches make sure that the explanation that is finally derived is the most accurate, i.e. it is the one that covers the maximum number of positive examples and the least of negative examples. In the process presented, however, we have shown atomic explanations that were composed by only one path and one ending value. While combining explanations might sensibly increase the explanation accuracy, combining each one of them is not possible without falling into scalability issues. A step to take in this direction is to find a strategy that is able to aggregate explanations only if their combination is likely improve on the accuracy of the atomic candidate explanations. This challenge is the focus of the next chapter.

❷ **Statistical accuracy might not be enough.** So far, we have only used predefined measures, as WR_{acc}, F-Measure or the Fuzzy F-Measure. While they have proven to be effective in reaching good explanations, we do not know how much the results that we obtain are dependent on the information that is declared in Linked Data. What if we missed explanations, because they were not declared in the dataset? Can we assume the information is homogeneously distributed across the data we look at? If there is a bias that might be introduced when adding more Linked Data knowledge, this has to be taken into account to properly evaluate our explanations. The challenge, discussed more in Chapter 8, consists in assessing this bias and possibly finding an evaluation measure able to take this aspect into consideration so to improve the accuracy of candidate explanations.

❸ **Explanations are (still) not complete.** Since the use-cases in this chapter were more real-world scenarios, the candidate explanations produced were with no doubt more human-understandable. With that said, if one was not familiar with some of the domains we discussed, it would not be possible to understand the explanations that were found, which then would remain unclear. In other words, we still cannot consider that we produce complete explanations with respect to the Explanation Ontology, since we are missing the context that connects the pattern with the explanation that Dedalo has found, and the theory governing such explanation. As already mentioned, the former challenge is discussed in Chapter 6 and the latter in Chapter 8.

Chapter 5

Aggregating Explanations using Neural Networks

In this chapter we show how we improve Dedalo by making it able to generate combined candidate explanations. Because combining a large set of explanations is too expensive, we present how we introduced in the process a trained Neural Network that can determine the likelihood that two explanations, if aggregated, improve their accuracy with respect to a pattern. After an overview of the chapter in Section 5.1, we discuss the problem and challenges in Section 5.2, and present the Neural Network approach details in Section 5.3. Section 5.4 and Section 5.5 present respectively the experiments and the limitations of the proposed approach.

5.1 Introduction

In order to be fully comparable to the Inductive Logic Programming frameworks, the process as described in Chapter 4 is missing a major feature, that is, the faculty of producing explanations composed by aggregated statements. When looking at the results of Section 4.4, it is clear that we are dealing with large amounts of hypotheses that, although being interesting and representative for each pattern, are "atomic" – composed in fact of a single path $\overrightarrow{p_i}$ and its ending value v_j. From the ILP frameworks, we know that better hypotheses can be obtained with aggregated facts, which in our case means that we should automatically combine Linked Data explanations[1].

With that said, the complexity and scale of trying each possible combination in $\mathcal{B}$ is much too high, especially when the explanation set is very large. Time and computational

[1] Note that in this chapter we will use *combination* and *aggregation* indiscriminately.

costs increase exponentially the more hypotheses we have in hand; besides, most of the combinations might simply not be interesting, i.e. they would not produce explanations that better represent the pattern. These quantitative and qualitative problems are well known in data mining, especially for those tasks dealing with association rules and frequent itemsets: to cope with them, results are generally processed a posteriori by human experts, whose role is to keep the interesting rules and filter out the unclear ones. To automatically assist the experts, many strategies have been proposed, ranging from using interestingness measures for rules [Geng and Hamilton, 2006] to including ontological knowledge [Marinica and Guillet, 2010; Anusha and Reddy, 2012; Nandhini *et al.*, 2012; Ramesh *et al.*, 2013].

With the scope of creating a process which does not rely on the experts, our challenge is to find a way to predict which explanations are worth aggregating, so that to save time and computations. In order to do so, it is first necessary to detect what knowledge about the atomic explanations is important for making predictions. This question is addressed in this chapter, where we present a Neural Network-based approach to predict whether two atomic hypotheses, if combined, will lead to a better pattern explanation. By "better", we mean that the accuracy of the combination is higher than the atomic explanations on their own. We use statistical information about the candidate explanations (Precision, Recall and F-Measure), as indicators for a prediction. The trained Neural Network is then integrated in Dedalo as an automatic post-processing step.

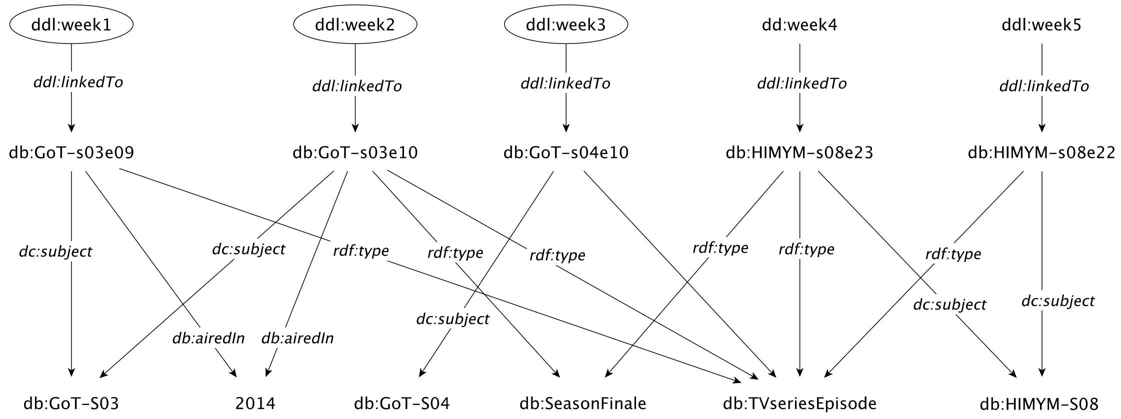


Figure 5.1 Example of $\mathcal{B}$ for the "A Song of Ice and Fire" web popularity.

5.2 Motivation and Challenges

The section presents our motivation and challenges based on the toy-example presented in Figure 5.1.

5.2.1 Improving Atomic Rules

Let us take again the "A Song of Ice and Fire" trend to illustrate the problem. Given the set $\mathcal{E}$ of 5 weeks (where the first three correspond to $\mathcal{E}^+$) and the background knowledge $\mathcal{B}$ of Figure 5.1, the explanations that we can obtain with Dedalo at the current moment are scored and ranked according to their F-Measure, as in Table 5.1.

Table 5.1 Atomic candidate explanations induced from Figure 5.1.

Explanation	F
$\varepsilon_{1,1}$=⟨ddl:linkedTo.dc:subject · db:GoT-S03⟩	0.8
$\varepsilon_{2,1}$=⟨ddl:linkedTo.db:airedIn · 2014⟩	0.8
$\varepsilon_{3,1}$=⟨ddl:linkedTo.rdf:type · db:TVseriesEpisode⟩	0.75
$\varepsilon_{1,2}$=⟨ddl:linkedTo.dc:subject · db:GoT-S04⟩	0.5
$\varepsilon_{3,2}$=⟨ddl:linkedTo.rdf:type · db:SeasonFinale⟩	0.4
$\varepsilon_{3,3}$=⟨ddl:linkedTo.rdf:type · db:HIMYM-S08⟩	0

Based on that, we can derive the following:

- some explanations, such as $\varepsilon_{3,2}$, are frequent but not really meaningful with respect to the pattern $\mathcal{E}^+$, as items outside it share it too;
- some explanations are redundant, such as $\varepsilon_{1,1}$ and $\varepsilon_{2,1}$ that share the same roots;
- some explanations, such as $\varepsilon_{1,1}$ and $\varepsilon_{1,2}$, if in a *disjunction*, would give a better explanation, since the union of their roots is more representative of $\mathcal{E}^+$ (the term is popular when either the third or fourth season of the Game of Thrones TV series is on, i.e. $\varepsilon_{1,1} \vee \varepsilon_{1,2} = 1$);
- some, such as $\varepsilon_{1,1}$ and $\varepsilon_{3,1}$, if in a *conjunction*, would give a better explanation, since the intersection of their roots is more representative of $\mathcal{E}^+$ (the term is popular during TV series episodes that are from the third season of the Game of Thrones TV series, i.e. $\varepsilon_{1,1} \wedge \varepsilon_{3,1} = 1$).

As said, our objective is to obtain the most representative explanation for a pattern, as Inductive Logic Programming frameworks would do. While in a simplified example such as the one above, trying each possible combination is not an issue, when dealing with thousands of explanations extracted from the Linked Data graph, the time and scale of the process increase exponentially. Trying each conjunction or disjunction becomes a strenuous process. Therefore, we require a process that is able to detect "promising combinations" or, in other words, a process that predicts which pairs of explanations is worth aggregating and which ones can be avoided.

In this scenario, the second issue is what makes a good prediction. Which are the indicators for it? Is there any information about the atomic explanations that we induced that we can use to predict an aggregation?

5.2.2 Rule Interestingness Measures

In typical KDD subfields such as Association Rule Mining, Frequent Itemset Mining or Sequential Pattern Mining, researchers constantly try to produce new, scalable approaches to post-mine large sets of rules (or patterns). As said, the atomic explanations obtained by Dedalo can be compared to such patterns.

Although numerous measures have been proposed to evaluate rule interestingness, there is no agreement on a formal definition of it. Standard measures include Accuracy in classification-oriented Predictive Induction, Precision and Recall in Information Retrieval, Sensitivity and Specificity in Medical Data Analysis, Support and Confidence in Association Rule Learning. The very first part of our work consists in verifying whether the quality of the induced explanations is being affected if we use different evaluation measures.

For this purpose, we tested some probability-based objective interestingness measures presented in the literature. As expected, the results showed that the probability-based measures have common behaviours. Some measures are precision-like and favour explanations that apply in a high percentage of cases within the pattern (they are generally called *rule reliability* measures); some others are recall-like, and prefer comprehensive explanations that cover a large subset of the items in the dataset (*rule generality* measures); finally, some of those are F-Measure-like and give a good trade-off between generality and reliability. Below we show the summary of the measures surveyed in [Geng and Hamilton, 2006] that we tested.

1) **Reliability.**
   $top(\mathcal{B})$=⟨ddl:linkedTo.dc:subject · db:GoT-S03⟩
   Measures: Added Value, Confidence, Conviction, Information Gain, Leverage, Lift, Odd's Ratio, Precision, Prevalence, Yule's Q, Yule's Y, Two Way Support.

2) **Generality.**
   $top(\mathcal{B})$=⟨ddl:linkedTo.rdf:type · db:TVseriesEpisode⟩
   Measures: Coverage, Recall, Relative Risk, Specificity, Support.

3) **Trade-off.**
   $top(\mathcal{B})$=⟨ddl:linkedTo.rdf:type · db:TVseriesEpisode⟩

Measures: Accuracy, Correction, Cosine, F-Measure, Gini Index, Jaccard, Klosgen, Laplace, Linear Correlation Coefficient, Novelty, Weighted Relative Accuracy.

5.2.3 Neural Networks to Predict Combinations

The assumption we make is that the measures described above can be used to train a Neural Network model, whose aim is to predict the likelihood of a combination score improvement. Given a pair of two explanations, statistically described by the aforementioned measures, the Neural Network can learn how to predict if their disjunction or conjunction will bring an improvement (i.e. a combined explanation whose accuracy score is higher than the ones of the single atomic explanations) or not.

Artificial Neural Networks (ANNs) are a well-established Machine Learning approach, imitating the learning mechanisms of brain neurons, to approximate real-valued, discrete-valued and vector-valued target functions [Michalski *et al.*, 2013]. We believe that ANNs fit our problem for the following reasons:

- ANNs are connections of units producing one single output from a set of inputs. If we consider each measure as the input unit (the neuron) and a pair of explanations as input data, the desired output that we can obtain from a Neural Network is a boolean answer of whether the combination (conjunction or disjunction) of two given explanations is worth it (1) or not (0).
- ANNs can induce the dependencies of features from some training data to learn models that predict outputs for future values. Therefore, we can easily detect which measures are important for a combination of two explanations, and which ones are not.
- ANNs are data-driven and self-adaptive methods, which adjust their model when learning from the input data. This means, ANN will iteratively use each pair description to gradually improve the prediction of a combination.

5.3 Proposed Approach

In this section, we first show how we train the Neural Network and then present how we integrate it in Dedalo as a post-processing step.

5.3.1 A Neural Network Model to Predict Aggregations

We built the Neural Network model with the following steps: (i) we chose the measures to use as features for learning; (ii) we created a dataset of training and test examples; (iii) we trained and tested the model on them; (iv) we included the trained model in a process to combine explanations.

Feature Selection. The general idea we bring forward is that there are some unrevealed relationships between the features of a pair of explanations, and these relationships can be the key to decide whether the combination is promising (what we define as a "good prediction") or not. To prove that, we experimented different sets of measures as features, with the aim of finding out the most efficient setting for the Neural Network training.

Coherently with the analysis provided in Section 5.2.2 (as well as in [Geng and Hamilton, 2006]), we reduced the set of features to the three measures of Precision, Recall and F-Measure. Besides those "structural" features, the final set of features upon which we built the model also included some "informative" features, namely the absolute difference between them. Those are presented in Table 5.2.

Table 5.2 Neural Network features. The input is a pair of explanations $(\varepsilon_{i,j}, \varepsilon_{n,m})$.

Type	Feature	Description
Structural features	P1	Precision of $\varepsilon_{i,j}$
	R1	Recall $\varepsilon_{i,j}$
	F1	F-Measure $\varepsilon_{i,j}$
	P2	Precision $\varepsilon_{n,m}$
	R2	Recall $\varepsilon_{n,m}$
	F2	F-Measure $\varepsilon_{n,m}$
Informative features	$\lvert$P1 – P2$\rvert$	Difference between P1 and P2
	$\lvert$R1 – R2$\rvert$	Difference between R1 and R2
	$\lvert$F1 – F2$\rvert$	Difference between F1 and F2

Training and Test Sets. In order to train the Neural Network model, we built the dataset fusing the explanations induced by Dedalo in Chapter 4. A dataset of 60,000 combinations was created automatically as follows:

- we randomly selected pairs of atomic explanations induced from 6 different sets of explanations, namely #KMi.A$_1$, #KMi.A$_2$, #KMi.P$_1$, #KMi.P$_2$, #KMi.H$_1$ and #KMi.H$_2$ that were already presented in the experiments of Chapter 3;
- we calculated the F-Measure of their disjunction and conjunction (for a total of 2 nominations per pair);
- we reported as a bi-dimensional vector of boolean answers whether the F-Measure of the new combined pair improved (1) or not (0) with respect to the F-Measure of the single explanations.

The resulting dataset was evenly partitioned, e.g. we obtained 15,000 combinations for each of the four possible outputs $[1,1]$, $[1,0]$, $[0,1]$, $[0,0]$ (where $[1,1]$ means that both the disjunction and the conjunction are worth doing, while $[0,0]$ means that none of them is worth). This was finally divided into a training and a test sets of 30,000 combinations each.

Learning. The Neural Network model was generated using the Neuroph[2] Java library. Table 5.3 reports the technical characteristics of the model resulted after the tuning that we performed with the objective of minimising the Mean Squared Error (MSE).

Table 5.3 Technical details about the trained Neural Network.

Neural Network	
ANN model	Feedforward Multilayer Perceptron
Learning Rule	Resilient Backpropagation
Activation function	Sigmoid function
Input neurons	9
Hidden Neurons	12
Output Neurons	2
Learning	
MaxIterations	3,000
Learning Rate	0.2
Tuning set split	70%
Validation set split	30%
Max Err.	0.1

In the testing phase, the model reported a MSE rate of 0.24 and a Root-Mean-Squared Deviation[3] of 0.49. With such a level of accuracy we confirmed our insight that using the measures as indicators for predicting the result of a combination is, to a certain extent, feasible and therefore the Neural Network model can be applicable to our purposes. The next section describes the Neural Network-based process to aggregate candidate explanations.

5.3.2 Integrating the Model in Dedalo

Once the Neural Network model (NNet) is trained, our goal is to use it in a process for detecting, in a large set of candidate explanations, which ones are worth combining. We define a prediction value indicator p for each explanation pair ($\varepsilon_{i,j}$, $\varepsilon_{n,m}$) as:

$$p = nnet(\varepsilon_{i,j}, \varepsilon_{n,m}) * max(F(\varepsilon_{i,j}), F(\varepsilon_{n,m})) \tag{5.1}$$

[2] http://neuroph.sourceforge.net

[3] https://en.wikipedia.org/wiki/Root-mean-square_deviation

The indicator takes into account the prediction returned by the Neural Network and combines it with the highest F-Measure in the pair under analysis, i.e. $max(F(\varepsilon_{i,j}), F(\varepsilon_{n,m}))$. In this manner, we favour pairs with a high probability of obtaining an improvement over a good F-Measure rather than the ones with a very low F-Measure.

As our objective is to obtain the best explanation of a pattern within the shortest time, the aggregation process also uses the explanation *top*($\mathcal{B}$) with the best F-Measure to start predicting the combinations. Given the set $\mathcal{B}$ of candidate explanations and *top*($\mathcal{B}$), each explanation is paired with *top*($\mathcal{B}$) to obtain its prediction score. If this is positive, we produce the actual conjunction or disjunction of the pair (*combine*(*top*($\mathcal{B}$), $\varepsilon_{i,j}$) returns either (*top*($\mathcal{B}$) $\vee$ $\varepsilon_{i,j}$) or (*top*($\mathcal{B}$) $\wedge$ $\varepsilon_{i,j}$), or both) and add them to $\mathcal{B}$.

Algorithm 2 Candidate Explanation Aggregation

```
stop = false
$\mathcal{B} \leftarrow$ list()                                          ▷ List of atomic rules
while (not stop) do
    $P \leftarrow$ list()                                        ▷ List of predictions
    top_explanation ← top($\mathcal{B}$)
    for explanation in $\mathcal{B}$ do                                  ▷ NNet prediction
        p ← nnet(top_explanation, explanation)
        if ($p > 0$) then
            add((top_explanation, explanation), $P$)     ▷ Add the pair if it is worth it
        end if
    end for
    for (exp1, exp2) in $P$ do                         ▷ Combine promising pairs
        new_combination ← combine(exp1, exp2)         ▷ Evaluate combination
        add(new_combination, $\mathcal{B}$)                               ▷ Add to $\mathcal{B}$
    end for
end while
```

The process, detailed in Algorithm 2, is iteratively reproduced in order to improve the *top*($\mathcal{B}$) score, until a predefined condition is met.

5.4 Experiments

To test our approach, we compared our Neural Network-based process to some strategies that might be adopted in common rule aggregation processes. We aim at analysing two characteristics in each of them: (i) the speed in reaching a new (better) F-Measure and (ii) the accuracy of it, intended as the level of improvement obtained compared to the previous best F-Measure.

5.4.1 Comparing Strategies for Rule Aggregation

Those strategies are presented below.

Random. Our baseline consists of the selection of a random explanation to be combined with *top*($\mathcal{B}$). While this is certainly a fast process, our assumption is that the new combination will not be very accurate.

AllComb. The most naïve approach to explanation aggregation is to try each and every possible combination in $\mathcal{B}$, and then to detect the one(s) with the highest score. This strategy should favour accuracy, but is likely to have a drawback on speed.

Top100. A naïve approach would also consider that the best score improvement is to be found in the top 100 rules of the list, therefore making every possible combination among them. While this strategy can be a good in avoiding redundant or useless combinations, the accuracy might not be guaranteed.

First. This strategy naturally assumes that the only first rule will provide the best improvements, when combined to one of the rules in $\mathcal{B}$. Therefore, at each iteration, *top*($\mathcal{B}$) is combined to all the rules in $\mathcal{B}$. Speed might not be guaranteed.

δ. Expecting computational and time problems, this strategy is refined upon the previous one. Meaningless aggregations are here filtered out, by putting a threshold δ to the F-Measure of the aggregation to put in $\mathcal{B}$. We set the threshold to the highest score, i.e. the F-Measure of *top*($\mathcal{B}$) at every iteration. In other words, we keep only combinations that improve over the current best score.

Besides those strategies, we chose two different settings for the Neural Network-integrated approach.

NNet-0. Given the positive prediction of a pair, the pair is combined a priori. While this strategy's accuracy is high, it might not be fast.

NNet-50. Given the positive prediction of a pair, the pair is combined only if the prediction is higher than 50% of the highest score at the current iteration. This avoids creating meaningless combinations, favouring speed, but might result in a reduced accuracy.

5.4.2 Results and Discussion

The strategies were tried on the datasets of #KMi.A, #KMi.P and #KMi.H. We selected 2 sets of combinations for each of them, for a total of 6 experiments. Table 5.4 presents information about each of them: the size of the initial set $|\mathcal{E}^+|$ of items grouped and the size of the full dataset $|\mathcal{E}|$, the number of atomic explanations from which we started ($|\mathcal{B}|$), the best explanation (its Precision, Recall and F-Measure scores) before the aggregation process, the allocated RAM and seconds that we have used to run the experiments. Because our goal is to improve the explanation accuracy, the following results do not report the explanations, but just their scores. Results in their entirety are publicly available online[4], while some examples of combinations are given at the end of the section.

Table 5.4 Experiments set-up information.

Test	$\|\mathcal{E}^+\|$	$\|\mathcal{E}\|$	$\|\mathcal{B}\|$	P	R	F	RAM	time (sec.)
#KMi.A_1	22	92	369	0.69	0.72	0.71	4G	60”
#KMi.A_2	23	92	511	1.0	0.43	0.61	4G	60”
#KMi.P_1	220	865	747	0.55	0.54	0.55	4G	75”
#KMi.P_2	601	865	1,796	0.76	0.19	0.31	4G	160”
#KMi.H_1	335	6,969	11,937	0.69	0.12	0.20	10G	2,500”
#KMi.H_2	919	6,969	11,151	0.93	0.07	0.13	10G	3,000”

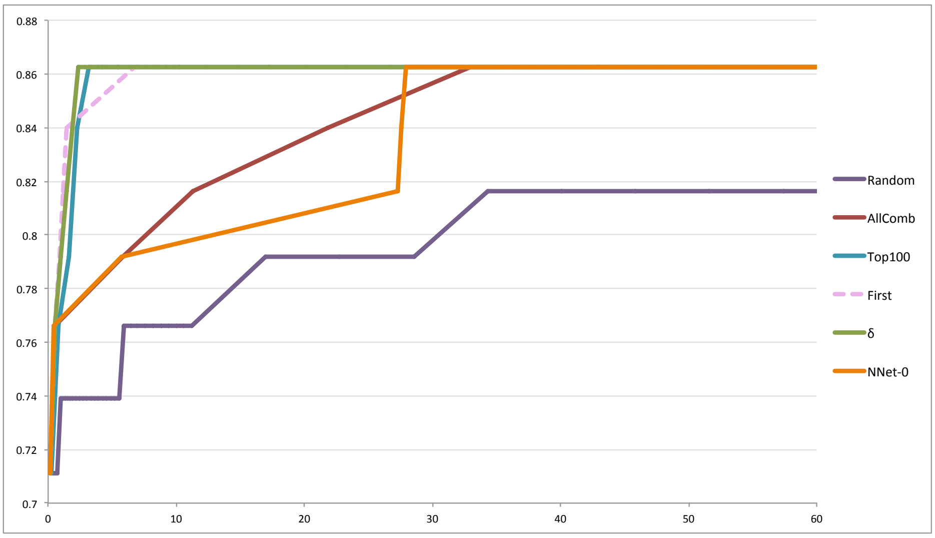
At each iteration, the best score among the set of combinations and the time spent is logged. The results are summarised below, in Figures 5.2, 5.3 and 5.4. The X axis is the time past since the beginning of the process (the speed criterion), while the Y axis is the score of the aggregated explanations (the accuracy criterion).

In Figure 5.2a, First, δ and Top100 find the best score before the NNet-0 and the AllComb approach, which can be explained by the small number of explanations that have to be combined. The test-case size also explains why the random strategy does fairly well in Figure 5.2a. In fact, as soon as the set of candidate explanations is larger, as in Figure 5.2b, the strategies First, δ Top100 and Random take much more time to find the best score, because they have to deal with a lot of redundant combinations that could be avoided. Given the small size of the explanation sets in the #KMi.A test, there is no need to make a distinction between NNet-0 and NNet-50.

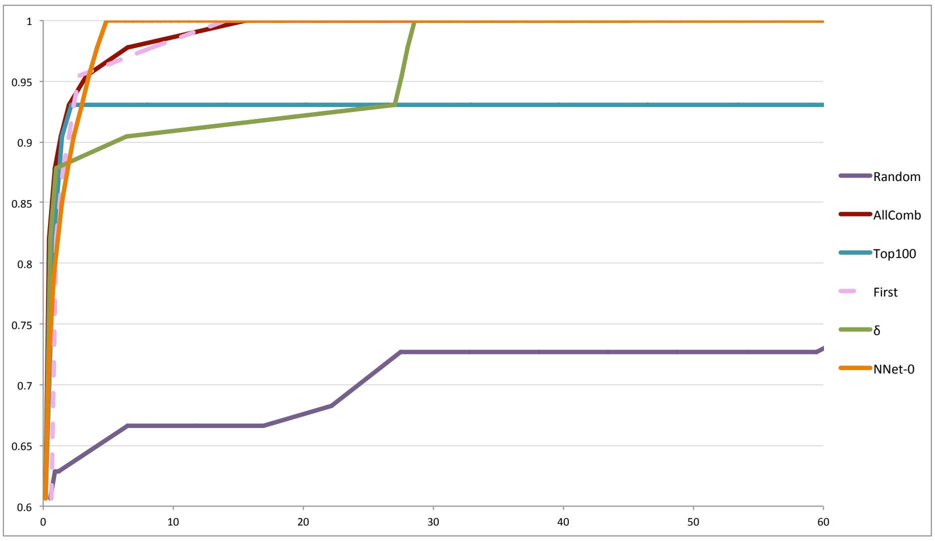
As the #KMi.P sets of explanations are medium-sized, Figure 5.3a still presents a good performance in terms of time/accuracy of AllComb, First, δ and Top100. From this example, however, it emerges that Top100 is limited as, after a certain number of combinations among the best explanations, it is not possible to obtain any further score improvement. This demonstrates that the best score increases are not to be found by combining only the explanations with the best scores.

As soon as the number of combinations becomes larger (see Figure 5.3b), despite being

[4] http://people.kmi.open.ac.uk/ilaria/experiments/nnets/



(a) #KMi.A_1, Semantic Web researchers.

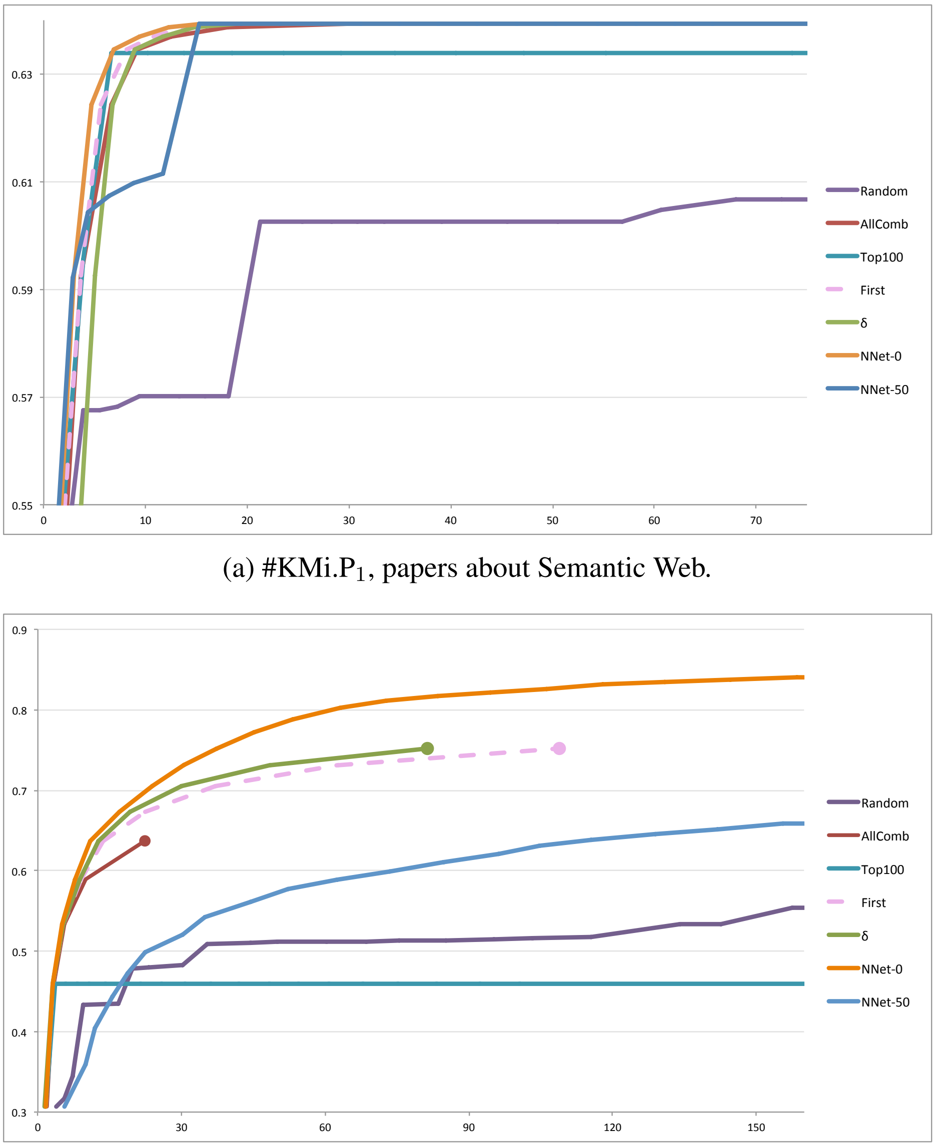


(b) #KMi.A_2, Learning Analytics researchers.

Figure 5.2 #KMi.A results.

fast and accurate in a first instance, none of AllComb, First or δ is able to get to the end of the process, as they encounter memory issues (highlighted with a dot at the end of the lines). Meanwhile, the two Neural Network approaches do not show any issue and both reach the end of the process.

The benefits of the NNet-based approaches in terms of the time required and explanation accuracy are much more visible in Figure 5.4, where the number of explanations to be combined is much higher. NNet-0 and NNet-50 are the strategies reaching the best scores in the shortest time, without falling into memory issues. As in the previous examples, AllComb,



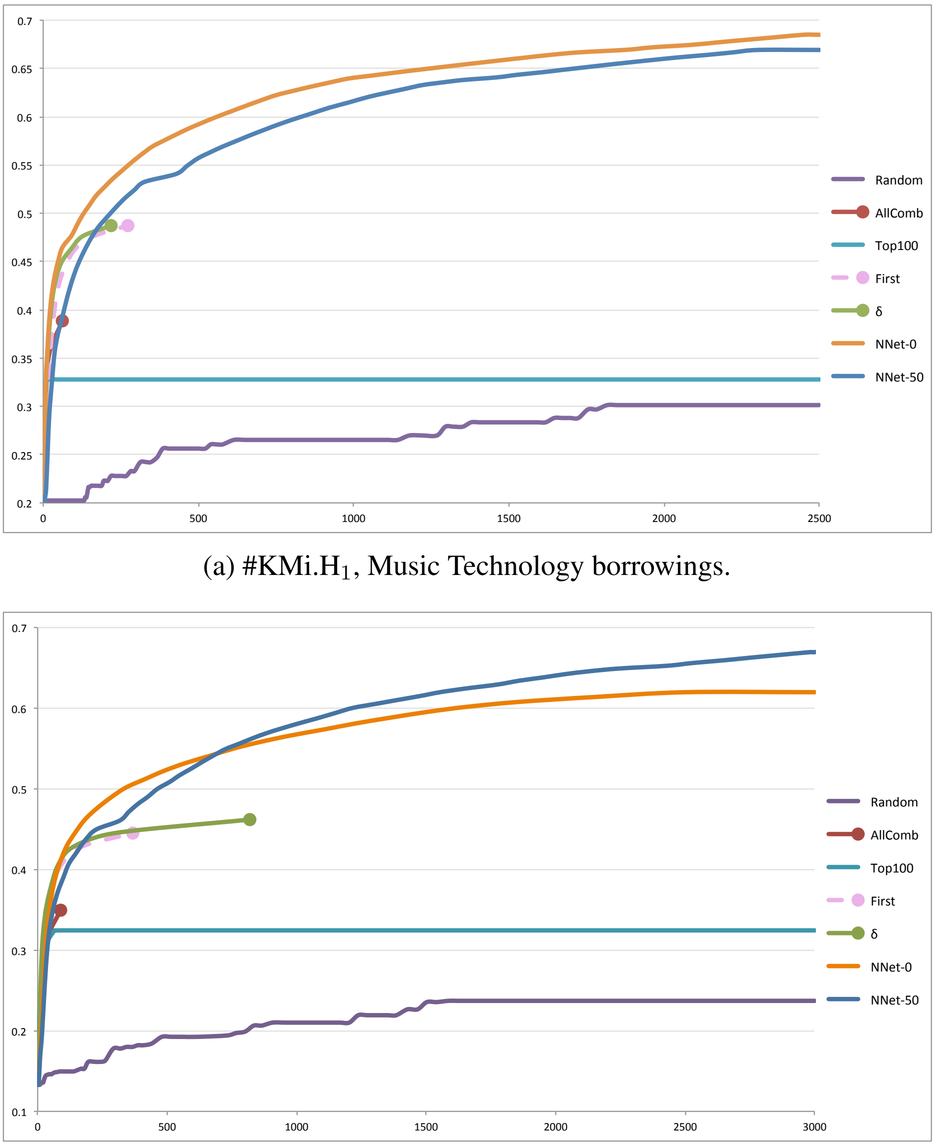
(a) #KMi.P_1, papers about Semantic Web.

(b) #KMi.P_2, papers about Learning Analytics.

Figure 5.3 #KMi.P results. In (a), AllComb, First and δ are almost as fast as NNet-0 because the explanation set size is still small enough. In (b), they cannot reach the end.

First and δ present an initial fast improvement but cannot run until the end. This means that those strategies are indeed accurate, but their realisation is not conceivable without high computational efforts.

As for the comparison between NNet-0 and NNet-50, in our settings, we did not experienced any computational issues. NNet-0 is in general faster, probably because the filtering used in NNet-50 introduces an overhead at this scale. In the case of Figure 5.4b, the higher accuracy reached eventually by NNet-50 is probably to be attributed to the increasing amount



(a) #KMi.H$_1$, Music Technology borrowings.

(b) #KMi.H$_2$, Theatre borrowings.

Figure 5.4 #KMi.H results. The memory issues of AllComb, First and δ are much more visible here, as well as the better performance of the two Neural Network-based approaches.

of candidate explanations to be aggregated, making the filtering more useful.

In Table 5.5 we give a summary of the characteristics of each strategy: speed, explanation accuracy and the computational efforts required, where ✗✗ corresponds to “not performing at all”, while ✓✓ means “fully performing”.

The results presented above show that learning how to predict an aggregation likelihood is not only feasible, but also that significant results (in terms of accuracy of a rule) can be achieved using very simple informative and structural information about the rules. While the

Table 5.5 Summary of speed, accuracy and computational efforts for the analysed strategies.

	Speed	Accuracy	Scalability
Random	✓✓	✗	✓✓
AllComb	✓	✓✓	✗✗
Top100	✓	✗✗	✓
First	✓	✗✗	✗✗
δ	✓	✗✗	✗✗
NNet-0	✓✓	✓✓	✓✓
NNet-50	✓	✓	✓✓

method proposed is indeed valuable, we remarked that one of its limitations is the complexity added to the readability of the aggregated rules. For instance, an aggregated explanation obtained by the NNet-0 model for #KMiA$_1$ at the end of the process (60”) is as follows,

$\varepsilon_{1,1}$ = ⟨ou:hasMembership · ou:KMi⟩ ∨ ⟨ou:interestedIn · outag:language-and-speech-technology⟩ ∨ ⟨ou:interestedIn · outag:social-web⟩ ∨ ⟨ou:interestedIn · outag:uncertain-reasoning⟩ ∨ ⟨ou:hasMembership · ou:LuceroProject⟩ ∨ ⟨ou:hasMembership.ox:hasPrincipalInvestigator · ou:EnricoMotta ⟩

which can be interpreted as “those researchers are clustered together because they either belong to the Knowledge Media Institute, or they are interested in Language and Speech technologies, or in Social-Web, or in Uncertain Reasoning, or they are part of the Lucero Project or another project that was led by Enrico Motta”. Given this example, it is easy to foresee that explanations also become more complex with more rules to aggregate. For example, in the case of #KMi.H$_1$ the explanation

$\varepsilon_{2,1}$ = ⟨dc:subject · bnb:ElectronicMusic⟩ ∨ ⟨dc:subject · bnb:MotionPictureMusic⟩ ∨ ⟨dc:subject · bnb:VocalMusic⟩ ∨ ⟨dc:subject · bnb:PopularMusic⟩ ∨ ⟨dc:subject.skos:broader · lcsh:AudioSequencers⟩ ∨ ⟨dc:subject.skos:broader · lcsh:CCTV⟩

composed of 5 aggregated rules was obtained after 27”; however, the best explanation after 2,500” was composed of 46 atomic rules. This means that the aggregated explanations we are dealing with at the end of the Neural Network process are exceedingly complex, especially when compared to an explanation that an expert would give (see, for instance, the explanations given by the users during Dedalo’s evaluation, cf. Chapter 7).

Considering that our goal is to obtain explanations that can be comparable to humans, we intentionally decided that the challenges we are going to tackle next will be in a different direction, more specifically we will focus more on identifying more complete (but atomic) explanations with respect to our Explanation Ontology. With that said, an interesting path to follow in the future would be to study ways to reduce the complexity added by the aggregation process by using, for instance, external knowledge sources. This is left for future work.

5.5 Conclusions and Limitations

This chapter presented how we improved Dedalo's process by making it able to produce explanations from aggregated Linked Data statements. We proposed and showed an approach using an Artificial Neural Network model to predict whether two candidate explanations, if aggregated, can lead to the creation of a new explanation, which better represents the pattern that we are observing. To evaluate our approach, we compared it with some other rule combination strategies, which usually tend to fail in speed, scalability or accuracy. The results show that the NNet-based strategies significantly outperform the other strategies, because they reduce time and computational efforts.

While the work achieved so far is a good step in the direction of automatically explaining patterns with Linked Data, we have new open issues that we need to tackle, as below.

❶ **The Neural Network has to be integrated.** The process that we have presented is applied in a post-processing phase: a set of atomic Linked Data explanations is produced within a certain amount of iterations, and only after that the Neural Network process is applied. It would be interesting to integrate the model directly within the Linked Data Traversal process, so that given a set of new atomic explanations found in Linked Data, we can generate more complete (and aggregated) pattern explanations thanks to the Neural Network prediction. This would also be beneficial in pruning the set of Linked Data explanations during the process and in reducing their complexity.

❷ **The model feature set might be trivial.** While we were able to use statistical measures to establish good predictions, there are other characteristics that can be used as new features to improve the model's accuracy. More specifically, there are topological features that paths and values have in the graph of Linked Data, which could make a prediction more accurate. This aspect is related to the challenge that we discuss in Chapter 6, where we attempt at finding a contextual relationship between the pattern and a candidate explanation.

Besides those, we remind that some issues from the previous chapters are still open.

❸ The candidate explanations (atomic or aggregated) are still only **statistically assessed**, and they assume no information lacks in Linked Data.

❹ We are still unable to automatically assess what are the **context** and the **theory** completing a generate explanation.

Chapter 6

Contextualising Explanations with the Web of Data

This chapter contributes to answering our last research question (Section 1.3.4), i.e. how to assess the validity of the generated explanations. We focus on how to find the context that relates an induced explanation and a pattern in the Web of Data using a blind graph search. To do so, we need to learn a cost-function supporting the detection of the strongest relationships between Linked Data entities. The Chapter is articulated as follows: Section 6.1 gives an overview; Section 6.2 presents the problem of finding strong relationships in detail; Section 6.3, presents the evolutionary algorithm that we propose to learn the best cost-functions, which are then presented and compared in Section 6.4. Finally, in Section 6.5 we discuss the approach limitations before closing the second part of our work.

6.1 Introduction

The previous chapters focused on the induction of candidate explanations, by showing how to obtain them or how to improve them through aggregation. In terms of the Explanation Ontology (Section 2.1.2), we can say that, so far, we are able to automatically produce anterior events, but we still do not know which context relates them to the posterior event, i.e. the pattern. The goal of this chapter is to identify such context using the Web of Data. In other words, we have to identify what is the relationship between a pattern and a candidate explanation.

Identifying the relationship between entities, sometimes referred to as entity relatedness, is a well-known problem for a wide range of tasks, such as text-mining and named-entity disambiguation in Natural Language Processing or ontology population and query expansion in Semantic Web activities. One of the approaches to identify such a relation is to estimate it using metrics based on some background knowledge such as Wordnet[1], Wikipedia [Strube and Ponzetto, 2006; Gabrilovich and Markovitch, 2007; Witten and Milne, 2008; Yeh *et al.*, 2009] or the Web [Chen *et al.*, 2006; Sahami and Heilman, 2006; Bollegala *et al.*, 2007; Cilibrasi and Vitanyi, 2007]. With those, we could for instance identify that there is a very strong relationship between the novel "A Song of Ice and Fire" and its writer George R. R. Martin, while the relationship between the novel and Kit Harington, which portrays the fictional character of John Snow in the TV series, is much weaker. Although those measures succeed in quantitatively measuring the strength of a relationship, they fail in qualitatively expressing it: they can tell how much two entities are related, but not how.

From a Web of Data perspective, the entity relatedness can be identified in the graph of Linked Data as a path between two given entities, and graph search techniques for pathfinding can therefore be used to reveal which paths exist between them. In our specific case, we could take the value of a candidate explanation (e.g. the db:GameOfThrones-TVseries) and a second value, which characterises the pattern (e.g. we could choose to map our search term to the DBpedia resource db:ASongOfIceAndFire(novel)) and find their relation, e.g. the path of Figure 6.1, which reveals that the two entities are connected through their broader category dbc:ASongOfIceAndFire(topic). As shown in Chapter 2, most of the existing pathfinding techniques in Linked Data use informed search methods, i.e. they pre-compute a portion of the graph (a limited number of datasets), and therefore violate the principle of incremental and serendipitous exploration of Linked Data through link traversal. If we aim at performing an efficient uninformed graph search, we need to find a cost-function that identifies the correct paths, i.e. the relationships that best represent the context that relates two entities.

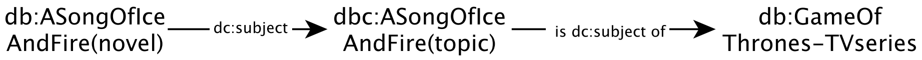

db:ASongOfIce AndFire(novel)
dc:subject
dbc:ASongOfIce AndFire(topic)
is dc:subject of
db:GameOf Thrones–TVseries

Figure 6.1 Example of Linked Data context between "A Song of Ice and Fire" and the TV series "Game of Thrones".

Our assumption is that the structure of the Linked Data graph can be used by a cost-function to successfully evaluate paths. While one could intuitively think that the shortest paths reveal the strongest connections, this assumption does not hold anymore within the Linked Data space, where entities of different datasets are connected by multiple paths of similar lengths. Our challenge, therefore, is to find which Linked Data structural information we need in order to design a cost-function that objectively assesses the value of a path. More

[1] https://wordnet.princeton.edu/

specifically, we aim at discovering which topological and semantic features of the traversed nodes and properties can be used to reveal the strongest relationships between entities.

The approach we propose in this chapter is to use a supervised method based on Genetic Programming whose scope is to learn the path evaluation function to integrate in a Linked Data pathfinding process to identify an explanation's context. Our idea is that, starting from a randomly generated population of cost-functions created from a set of topological and semantic characteristics of the Linked Data graph, the evolutionary algorithm will reveal which functions best compare with human evaluations, and will show us what is really important to assess strong relationships in the Web of Data context.

We compare and discuss the learnt cost-functions in our experiments, where we demonstrate not only that good results are achieved using basic topological features of the nodes of the paths as they are being traversed, but also how those results can be improved through introducing a very small amount of semantic knowledge, as the vocabularies that label the edges connecting the nodes.

6.2 Problem Statement

The context between a candidate explanation and a pattern can be identified using a uniform-cost search based on Link Traversal. We remind from Section 2.2.2 that the uniform-cost search (ucs) is a best-first strategy where the cost-function $g(n)$ chooses the next node to expand based on the cumulative cost of the edges from the start to that node. As said, the search does not require to hold the whole graph in memory, which makes it particularly suitable with large graphs, as in our case.

Based on that, we designed a Linked Data pathfinding process consisting in a bi-directional uniform-cost search whose aim is to find the path $\overline{p_i} = \langle n^l \dots n^r \rangle$ that best represents the relation between two entities n^l and n^r, where the former corresponds to a Linked Data entity matching the pattern under observation, and the latter to the end value of an explanation $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$. We use the $\overline{p_i}$ notation to distinguish the path expressing the context, which is a sequence of nodes and edges, from the path $\overrightarrow{p_i}$ expressing a candidate explanation, which is in fact only a chain of edges. Note that, in a space as the one of Linked Data, where the number of entities retrieved at each node expansion exponentially grows, bi-directionality is needed to significantly reduce the search space, as well as the time and computational efforts.

Let us assume that, in our running example about weeks of popularity for the term A Song of Ice and Fire, we have obtained an explanation such as "A Song of Ice and Fire is popular in those weeks when an episode of Game of Thrones is aired". At this stage, we want to identify which is the strongest relationship between n^l_1 = db:ASongOfIceAndFire(novel), that we chose to represent the term searched by the users, and n^r_2 = db:GOT-TV-series(episodes), which was obtained out of the induction process. The process takes as input n^l and n^r, and gives as output the path which best expresses their relationship, along with a score indicating the strength of it (see Figure 6.2).

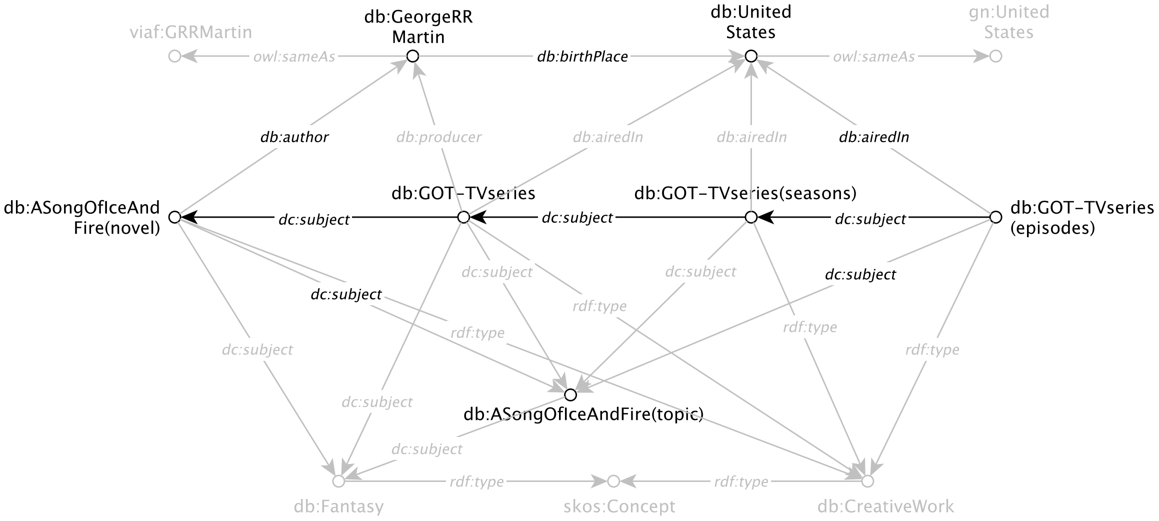


Figure 6.2 An example with multiple paths between db:ASongOfIceAndFire(novel) and db:GOT-TVseries(episodes).

The Linked Data pathfinding process then consists in:

(1) *Entity dereferencing*. Entities are dereferenced to find all the entities they are linked to, as during the induction process. Contrary to it, however, we do follow both incoming and outgoing RDF properties of an entity. In the example of Figure 6.2, both n_1 and n_2 are linked to 4 entities.

(2) *Bi-directional search*. Given the two nodes n^l and n^r, two uniform-cost searches ucs^l and ucs^r are performed simultaneously. Their objective is to iteratively build two search spaces, a right-directed one from n^l and a left-directed one from n^r, to find a common node n^c. Since the bi-directionality does not guarantee optimality, e.g. db:ASongOfIceAndFire(topic) might be found before db:GOT-TVseries(season), we collect all the common nodes found until a certain number of iterations is reached.

(3) *Path building*. Given a common node n^c, we build the two subpaths $p^j = \langle n^j \dots n^c \rangle$ with $j \in \{l, r\}$, and then merge them into the Linked Data path $p = \langle n^l \dots n^c \dots n^r \rangle$. Each

path identifies a relationship between the initial two entities. The graph of Figure 6.2 represents all the paths existing between n_1 and n_2 after a few iterations.

(4) *Path scoring*. The cost of each path is evaluated as an aggregation (most often a sum, see Section 6.3.2 for alternatives) of the costs of the paths from n^l to n^c and from n^r to n^c. The one with the lowest cost, highlighted in the figure, is chosen as the strongest relationship between the input entities.

Challenge. From the process described above, it becomes clear that a good path evaluation function is necessary to choose among a set of alternative paths between two entities, and to avoid computational efforts or inconclusive searches.

The question arising here is what is the best strategy to find the most representative relationships, and if we can exploit the information in the Web of Data to guide the two searches in Linked Data in the right direction, so that they can quickly get to convergence. In fact, when looking at the paths in Figure 6.2, an interesting observation can be made: the node corresponding to the entity db:GameOfThrones-TVseries is much less “popular” than other nodes, as the ones labelled as ASongOfIceAndFire(topic) or db:UnitedStates. This information could be used in the example above to automatically assess that the best relationship is the one we highlighted, which, although being longer, better specifies the relation between n_1 and n_2. In other words, the structural features of the graph could be a good insight to drive our blind search in Linked Data.

Given this, the challenge is: what makes a path important? Which are the topological or semantic features of a node or an edge, which we should care about when deciding if a path is better than another? To reformulate the problem: Can we use the structure of a graph to assess relationship strengths?

Proposed Approach. Our proposition is to use a supervised Genetic Programming (GP) approach to identify the cost-function that best performs in ranking sets of alternative relationship paths. Our idea is that, starting from a random population of cost-functions created on a set of features that are the structural characteristics of the graph, the evolutionary algorithm will learn the best cost-function based on a benchmark of human-evaluated relationship paths.

The next section presents our attempt to use Genetic Programming to discover what are the features of the Web of Data that matter in a graph search when assessing the strongest relationships between entities.

6.3 Learning Path Evaluation Functions through Genetic Programming

The section presents the Genetic Programming approach that we use to discover the cost-function upon which we can build the process to identify the explanation contexts.

6.3.1 Genetic Programming Foundations

Inspired by Darwin's theory of evolution, Genetic Programming (GP) is an Artificial Intelligence technique that aims at automatically solving problems in which the solution is not known in advance [Poli *et al.*, 2008]. The general idea is to create a population of computer programs, which are the candidate solutions for a problem, that the Genetic Programming algorithm stochastically transforms ("evolves") into new, possibly improved, programs in a random way. Such randomness guarantees the GP to be very successful at proposing novel and unexpected solutions to a given problem.

Algorithm 3 Generic GP Algorithm

```
Create a random population of programs based on the set of primitives
repeat
    Execute each program
    Assess with the fitness measure how good the program is
    Randomly select one or two programs
    Create a new individual from those using the genetic operations
    Add the new individual to the population
until An acceptable solution is found, or some stopping conditions are met
return The best program in the current population
```

Programs are generally represented as trees of primitive elements, where the internal nodes (that are generally operations) are called functions, while the leaf nodes (that can be constants or variables) are called terminals. The process is sketched in Algorithm 3. An initial population is randomly generated using the predefined primitives. For each program, a fitness function measures how good the program is with respect to the problem. After that, a new population is created by adding programs created with one of the three following operations:

i) reproduction, in which a new child program is generated by copying a randomly selected parent program;

ii) crossover, where a child program is generated by combining randomly chosen parts from two randomly selected parent programs;

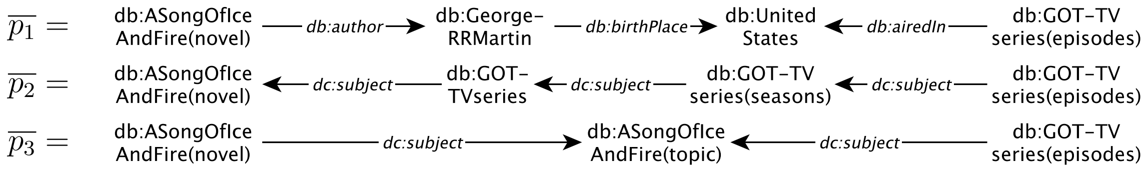
iii) mutation, where a new child program is generated by randomly altering a randomly chosen part of a selected parent.

This process is iterated until a termination condition is met, typically either a maximum number of generations is reached, or a satisfying, possibly optimal solution (e.g. a desired fitness) is found. Along with the primitive set, the fitness and the termination condition, a set of parameters such as the population size, the probabilities of performing the genetic operations, the selection methodology or the maximum size for programs need to be decided to control the GP process.

The GP framework can be then applied to find out which is the cost-function that has the best performance in evaluating (and ranking) alternative paths between two Linked Data entities. One of the main advantages of Genetic Programming is that it comfortably deals with wide search spaces: this means that we can create a large population of possible cost-functions whose primitives are Linked Data structural features without worrying about scalability issues, and obtaining a cost-function that is nearly optimal in ranking alternative Linked Data paths. Moreover, the execution times for a GP run are very short when compared to traditional classification methods, which makes the learning process more flexible, so that we can refine and improve parameters or primitives according to our needs. In the same way, this flexibility allows us to impose new constraints (or remove some) on the fitness function depending on the success or failure of the GP executions. Finally, and most importantly, the GP is not a black-box method but its results are human-understandable, which means that we can interpret the found solution and identify the structural features of a Linked Data path that matter for a successful search.

6.3.2 Preparatory Steps

The Genetic Programming process we show requires some preparatory steps before the actual run. For a better understanding, we invite the reader to use as a reference the graph of Figure 6.2 and the three following paths:


$\overline{p_1} =$
db:ASongOfIce AndFire(novel)
db:author
db:George-RRMartin
db:birthPlace
db:United States
db:airedIn
db:GOT-TV series(episodes)
$\overline{p_2} =$
db:ASongOfIce AndFire(novel)
dc:subject
db:GOT-TVseries
dc:subject
db:GOT-TV series(seasons)
dc:subject
db:GOT-TV series(episodes)
$\overline{p_3} =$
db:ASongOfIce AndFire(novel)
dc:subject
db:ASongOfIce AndFire(topic)
dc:subject
db:GOT-TV series(episodes)

Process. Let $P_i = \{\overline{p_1}, \ldots, \overline{p}_{|P_i|}\}$ be the set of $|P_i|$ alternative paths between two Linked Data entities, $D = \{P_1, \ldots, P_{|D|}\}$ the set of $|D|$ examples that have been ranked by humans, and $G = \{g_1, \ldots g_{|G|}\}$ a starting population of randomly generated cost-functions g_j. The Genetic Programming algorithm iteratively evolves the population into a new, possibly improved one, until a stopping condition is met. The evolution consists first in assigning the cost-functions a fitness score, which in our case reflects how "good" a cost-function is in ranking paths compared to the human evaluators. For instance, assuming 3 users have agreed on ranking $\overline{p_2}@1$, $\overline{p_3}@2$ and $\overline{p_1}@3$, those functions scoring the three paths in the same order will obtain the highest score. Then, reproduction, mutation and crossover are applied to some randomly chosen individuals, and the generated children are added to the new population. Once the new population reaches the same size as the current one, it is replaced by the evolved population, and a new generation starts.

Primitives. Terminals and functions are called the primitive set. A terminal can be:

1) a constant, i.e. a randomly chosen integer in the set $Z = \{0, \ldots, 1000\}$;
2) a combination of an edge-weighting function $w(e)$ (with e being the edge) and one aggregator a. We call this combination $a.w$ an *aggregated terminal.*

Edge weighting functions assign a weight to each edge of the path, based on the information of its starting node (the source). We define 10 edge-weighting functions, which we divide in topological and semantic terminals. Topological terminals focus more on the Linked Data graph plain structure, and are as follows.

- *Fixed Weight* (1): the edge is assigned a score of 1. This is equivalent to performing a breadth-first search, where nodes are queued and explored in the order they are found.
- *Indegree* (IN): the edge is weighted according to the number of incoming links of its source. For instance, the edge *db:birthPlace*(db:GeorgeRRMartin, db:UnitedStates) of Figure 6.2 has a weight of 2, since the source db:GeorgeRRMartin has 2 incoming links. This feature is chosen to understand the importance of "authority" nodes, i.e. the ones with many incoming links.
- *Outdegree* (OU): the edge is weighted according to the number of outgoing links of its source, e.g. the weight in the previous example is 2. OU helps us studying the importance of "hub" nodes that point to many other nodes.
- *Degree* (DG): an edge is weighted based on the degree of the source, i.e. the sum of IN and OU. To the previous example, DG would assign a score of 4.

- *Conditional Degree* (CD): the weight attributed to the edge depends on the RDF triple from which the edge has been generated. In fact, each edge $e(u, v)$ is generated from a dereferenced RDF triple, either ‹u, e, v›, as in the case of *db:birthPlace*(db:GeorgeRRMartin, db:UnitedStates), or ‹v, e, u›, as for *db:producer*(db:GeorgeRRMartin, db:GOT-TVseries). The CD terminal returns either the indegree or the outdegree of the source depending on whether the triple represents a back or a forward link. Therefore, CD would return 2 in the former case (the indegree of the node for db:GeorgeRRMartin) and 2 in the latter case (its outdegree). The conditional degree analyses the importance of paths going through large hubs, which are also common to many other paths.

We define semantic terminals those features that are more specific to Linked Data than to common graphs. For that, we first considered the vocabulary usage, then analysed the most frequent RDF properties, as provided by both Linked Open Vocabularies[2] and LODStats[3]. Note that, since we rely upon entity dereferencing to traverse Linked Data, we only considered the most frequent object properties.

- *Namespace Variety* (NS): an edge is weighted depending on the number of namespaces of its source node. For instance, the node db:GeorgeRRMartin has the two namespaces *owl:* and *db:* for its three links, while the node db:GOT-TVseries has the 3 namespaces *dc:*, *db:* and *skos:* for its 5 links. Namespaces variety is intended to analyse the use of vocabularies when semantically describing an entity. Note that, while initially we considered incoming and outgoing namespaces separately, we did not find any substantial difference in the process, and eventually reduced the two terminals to one.
- *Type Degree* (TD): the edge weight depends on the number of *rdf:type* declared for the source entity. For example, db:ASongOfIceAndFire(novel) has a type degree of 1 but, assuming this was declared as a skos:Concept too, its score would be 2. TD focuses on the taxonomical importance of an entity, with the idea that the more a node is generic (i.e. the entity belongs to many classes), the less informative the path might be. Since *rdf:type* is unidirectional, there is no need to distinguish between in- and outdegree.
- *Topic Outdegree* (SO): the edge weight is assigned by counting the number of outgoing edges labelled as *dc:subject* or *skos:broader* of the starting node. The edge *db:author*(db:ASongOfIceAnd- Fire(novel), db:GeorgeRRMartin) has a score of 2. The topic outdegree focuses on authority nodes in topic taxonomies (controlled vocabularies or classification codes).

[2] http://lov.okfn.org/dataset/lov/terms

[3] http://lodstats.aksw.org/

- *Topic Indegree* (SI): similarly, the edge weight is assigned by counting the number of incoming *dc:subject* edges. The same edge has a score of 1 in this case. SI considers hub nodes in controlled vocabularies.
- *Node Equality* (SA): the edge is weighted according to how much its source is connected to external datasets, based on the number of links labelled as *owl:sameAs*, *skos:exactMatch* or *rdf:seeAlso*. For instance, db:UnitedStates is connected to the corresponding entity gn:UnitedStated in Geonames[4] so, according to the SA weight, the edge *db:airedIn* (db:UnitedStates, db:GOT-TVseries(episodes)) is scored 1. SA considers the importance of inter-dataset connections. Since those properties are bi-directional, we do not distinguish between in- and outdegree.

Aggregators, shown below, are functions that decide how to merge the edge weights across a path. We chose four aggregators:

- SUM: given a path $\overline{p_i}$ of l length, it returns the sum of the $w(e)$ for each of the l edges of the path. For instance, with SUM.IN, the score for the path $\overline{p_1}$ is $1 + 2 + 4 = 7$ because db:ASongOfIceAndFire(novel) has 1 incoming link, db:GeorgeRRMartin has 2 incoming links and db:UnitedStates has 4.
- AVG: given a path $\overline{p_i}$ of l length, it returns the average weight across the path. In the same case as above, AVG.IN returns 2.33 for $\overline{p_1}$.
- MIN: it returns the lowest $w(e)$ across the edge weights of the paths. For $\overline{p_1}$, MIN.IN returns 1.
- MAX: it returns the highest $w(e)$ across the edge weights of the paths. For $\overline{p_1}$, MAX.IN returns 4.

To generate an individual, the aggregated terminals are randomly combined through the GP function set. Table 6.6 shows the set of arithmetic operators that we chose.

Table 6.1 Function set for the GP process.

Addition	$x + y$	(binary operator)
Multiplication	$x \times y$	(binary operator)
Division	$x \mathbin{/} y$	(binary operator)
Logarithm	$log(x)$	(unary operator)

These functions help in assessing how much each cost-function has to take into account the terminals in its body, e.g. g_4 = SUM.1 + (1/AVG.TD) is interpreted as a function acting almost as a breadth-first search, with a small added value from the average type degree.

[4] http://www.geonames.org/

6.3.3 Step-by-Step Run

Once defined the preliminaries, the GP process is executed as described below.

Initialisation. At first, the GP randomly creates a population of programs (the cost-functions). Figure 6.3 shows some random cost-functions that we can generate based on the primitives previously presented. The cost of a path will be assessed by traversing the cost-function body recursively starting from the root node, and evaluating each node only when the values of all its children are known.

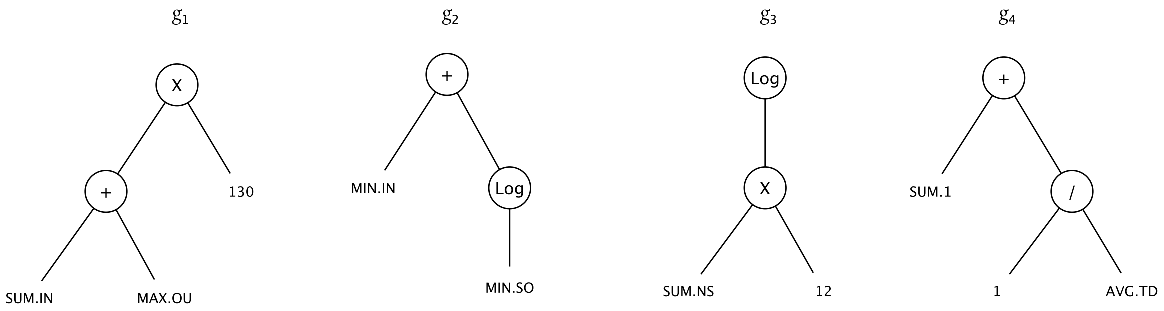


Figure 6.3 Cost-function examples at generation 0.

Fitness Evaluation. As said, the fitness measures how good a cost-function is in ranking the sets of alternative paths between entities based on a gold standard dataset. Such fitness is measured with the Normalised Discounted Cumulative Gain ($nDCG$), generally used in Information Retrieval to assess the quality of rankings provided by the web search engines based on the graded relevance of the returned documents[5]. The closer it gets to 1, the more the engine judges the documents as relevant as the human evaluators did. We apply the same idea by considering a path as a document, therefore evaluating first the DCG for a path p_k at rank k as:

$$DCG(p_k) = rel_1 + \sum_{m=2}^{k} \frac{rel_m}{log_2(m)} \tag{6.1}$$

where rel_m is the relevance score given to p_m by human evaluators. The normalised score $nDCG(p_k)$ for a path is then assessed by comparing $DCG(p_k)$ to its ideal score $iDCG(p_k)$, assessed by the gold standard.

[5] https://www.kaggle.com/wiki/NormalizedDiscountedCumulativeGain

The function $avg(P_i)$ then averages each $nDCG(p_k)$ in the set P_i, so that to obtain the performance of the function for the i-th pair, as follows:

$$avg(P_i) = \frac{\sum\limits_{p_k \in P_i} nDCG(p_k)}{|P_i|} \tag{6.2}$$

The fitness of a function is finally obtained by averaging each $avg(P_i)$ of all the $|D|$ pairs of the dataset:

$$f(g_j) = \frac{\sum\limits_{P_i \in D} avg(P_i)}{|D|} \tag{6.3}$$

We also add a penalty to avoid long and complex cost-functions, by comparing the length l of a function with its ideal length L. The fitness of a function is finally defined as

$$fw(g_j) = f(g_j) - (w \times (l - L)^2) \tag{6.4}$$

where w is the penalty weight.

Selection, Crossover and Mutation. After assessing the fitness of each cost-function, the GP process stochastically selects some programs to be the parents of the next generation. For the parent selection, we used a tournament selection, in which n random programs are selected from the current population, and the one with the best fitness is chosen as parent. Because we do not choose a greedy strategy for selection, programs with an inferior fitness still have a chance of being selected and bring their genetic material to the next generation. We then perform the three following operations.

1) *Reproduction.* Given a cost-function parent, a new individual is copied in the new generation without alterations. Consider the case in which g_1 has been selected in a 3-sized tournament with g_2 and g_3: g_1 is then copied into g_5 and passed to the next generation, as in Figure 6.4 (left).

2) *Crossover.* The crossover operation generates two children cost-functions by swapping two subtrees from the parents. In our example, let us assume that g_2 and g_3 are selected for crossover. Let us also assume that the *log* operator of g_2 and the *X* operator of g_4 are the nodes randomly picked for the crossover. The two subtrees are then inverted and the two children g_6 and g_7 are added to the new generation, as in Figure 6.4 (middle).

3) *Mutation.* Mutation modifies a node (the "mutation point") of the parent. Let assume that the cost-function g_4 is selected as parent. One of its nodes is then randomly picked and

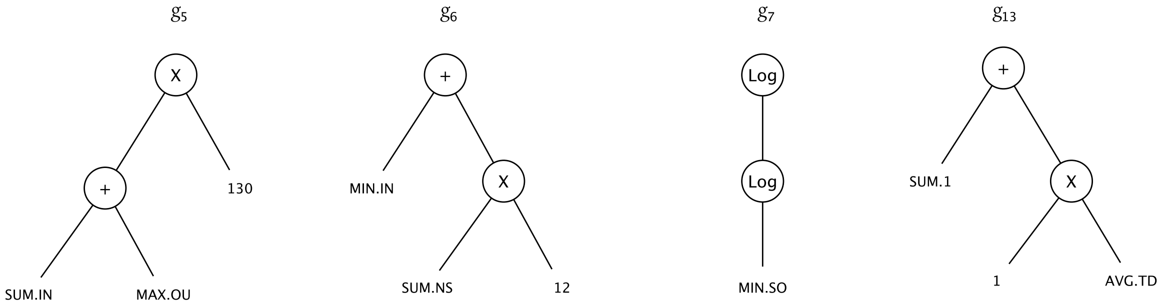


Figure 6.4 Cost-functions at generation 1, after one reproduction, two crossovers and one mutation.

mutated, and the new child is added to the next generation. We have designed different kinds of mutations, as follows:

3a) if the mutation point is a constant x, the node is mutated with a new constant y that is either higher or lower than x in the range of $y = \{x - 100,\ x + 100\}$. For instance:

$$g_8 = \text{SUM}.1 + (\mathbf{18}/\text{AVG.TD}) \quad \leftarrow 1 < (y = 18) < 101$$
$$g_9 = \text{SUM}.1 + (\mathbf{-18}/\text{AVG.TD}) \quad \leftarrow -99 < (y = -18) < 1$$

3b) if the mutation point is an aggregated terminal, the node is mutated by either modifying its aggregator, or by modifying its edge-weighting function, or by replacing the full aggregated terminal with a new constant. For instance:

$$g_{10} = \text{SUM}.1 + (1/\mathbf{MAX}.\text{TD}) \quad \leftarrow \textit{new edge-weighting } w$$
$$g_{11} = \text{SUM}.1 + (1/\text{AVG}.\mathbf{IN}) \quad \leftarrow \textit{new aggregator } a$$
$$g_{12} = \text{SUM}.1 + (1/\mathbf{20}) \quad \leftarrow \textit{new constant}$$

3c) if the mutation point is a function, we randomly choose to replace the node either with a new terminal (constant or an aggregated terminal) or with a new function, in which case we remove or add a child depending on whether the arity of the new function is different from the one of the mutation point. For example:

$$g_{13} = \text{SUM}.1 + (1 \times \text{AVG.TD}) \quad \leftarrow \textit{arity was the same}$$
$$g_{14} = \mathbf{MAX.OU +} (\text{SUM}.1 + (1/\text{AVG.TD})) \quad \leftarrow \textit{added child}$$
$$g_{15} = \boldsymbol{log}(1/\text{AVG.TD}) \quad \leftarrow \textit{deleted child}$$
$$g_{16} = \text{SUM}.1 + \mathbf{MAX.NS} \quad \leftarrow \textit{new aggregated terminal}$$
$$g_{16} = \text{SUM}.1 + \mathbf{20} \quad \leftarrow \textit{new constant}$$

The mutated child is then passed in the new generation, as g_{13} in Figure 6.4. Since the new generation has reached the same size of the previous one, the new individuals replace the old ones and a new generation can start. In a non-greedy strategy, a new generation might not necessarily mean that all the programs improve their fitness with respect to the parents; however, an overall fitness improvement will be obtained throughout generations, by recombining genetic pieces of fit parents into new, superior individuals.

Training and Testing. Finally, in order to identify the best cost-function, we articulate the learning process as described. First, we randomly split the dataset D into a training set and a test set. Then, we run the GP process on the training set and store a small set of fittest individuals, i.e. the cost-functions that performed better in ranking paths, while the rest are discarded. Third, the surviving individuals are tested on the test set, and if their fitness is not consistent with the one of the training set, we screen them out. This helps in avoiding overfitting and in obtaining more valid cost-functions. Finally, for each run, we keep the individual with the highest fitness on the overall dataset.

6.4 Experiments

The section introduces our experimental scenario, describing the dataset we built and the control parameters for the Genetic Programming learning process. Then, it presents the obtained results, including the discovered cost-function.

6.4.1 Experimental Setting

As previously mentioned, the fitness is assessed on a dataset composed by sets of alternative paths between random pairs of entities. In order to create more variety in the final dataset, so that the learnt functions would not be overfitted to a specific dataset, we used different types of entities, randomly extracted from different Linked Data sources, namely:

1) *Events*: 12,630 events (ranging from battles to sport to music events) from the YAGO dataset [Suchanek *et al.*, 2007];

2) *People*: 8,185 people from the Virtual International Authority File (VIAF) dataset[6];

3) *Movies*: 999 movies from the Linked Movie Database[7];

4) *Geodata*: 1,174 world countries and capitals from Geonames and the UNESCO datasets.

[6] http://viaf.org/

[7] http://www.linkedmdb.org/

To make sure those entities were connected to other datasets, we used the DBpedia SPARQL endpoint as a pivot, i.e. we chose a desired `?_class` (event, country, person etc...) and the `?_dataset` we wanted to retrieve it from, and then ran the simple query:

```
select distinct ?same where {
  #select the DB entities of the class we want
  ?entity a ?_class.

  #select each entity's sameAs
  ?entity <http://www.w3.org/2002/07/owl#sameAs> ?same.

  #keep only sameAs of the dataset we want
  FILTER(strStarts(str(?same), ?_dataset)).
}
ORDER BY RAND()  # make sure we get random entities
```

By doing so, we made sure to span at least to another dataset (DBpedia) when finding paths, therefore guaranteeing more variety.

Table 6.2 Control Parameters for the GP runs.

Population size	100 individuals
Max. generations	300
Termination condition	Until the max generation
Selection type	5-sized tournament
Crossover rate	0.65
Mutation rate	0.15
Reproduction	10% population size
Elitism	10% population size
Penalty weight w	0.001
Ideal length L	3
Validation Split	70%-30%
Best individuals for testing	5

Next, given a random pair, we ran a bi-directional breadth-first search limited to 30 cycles in order to find a set of paths between them. We discarded the pairs for which no path was found. 8 judges were asked to evaluate each set, assigning the paths rel scores between 2 (“highly informative”) and 0 (“not informative at all”), and discarded the pairs whose agreement was below 0.1 according to the Fleiss’k rating agreement[8]. An example of a path to be ranked, showing that the movie “The Skin Game” and the actress Dina Korzun are both based in Europe, is presented in Figure 6.5. The final dataset consisted of 100 pairs, whose paths were assigned a score corresponding to the average of the scores given by the users.

[8] https://en.wikipedia.org/wiki/Fleiss%27_kappa

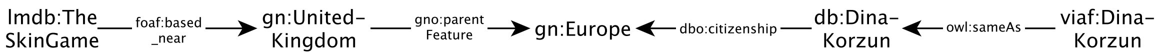


Figure 6.5 A path example.

Finally, Table 6.2 presents the control parameters we used during the GP process. In order to not bias the generation process, we also generated trees without limit in depth, and equally distributed the probability of functions, constants and aggregated terminals being chosen.

6.4.2 Results

First, we present the results of different runs of the Genetic Programming learning process presented in the previous Section. Table 6.3 shows the unweighted fitness on the training set (f_{tr}) and test set (f_{ts}) of the best cost-functions learnt during 3 different runs (and therefore on different dataset cuts). Experiments are divided into GP runs with topological terminals only (T), or with both topological and semantic terminals (S).

Table 6.3 Best cost-functions for different runs.

Run	Fittest g_j	f_{tr}	f_{ts}
T_1	$log(log(\textsc{min.cd} \times \textsc{min.cd}))/\textsc{max.cd}$	0.79	0.79
T_2	$log(\textsc{min.cd})/(\textsc{avg.cd} + 87)$	0.77	0.78
T_3	$\textsc{min.cd} \times (\textsc{min.cd} \ / \ \textsc{max.cd})$	0.78	0.72
S_1	$\textsc{min.ns} + (\textsc{sum.ns} \ / \ log(log(\textsc{sum.si})))$	0.88	0.83
S_2	$\textsc{min.ns} + (\textsc{min.cd} \ / \ log(log(\textsc{sum.si})))$	0.88	0.86
S_3	$\textsc{min.ns} + (log(\textsc{max.in}) \ / \ log(log(\textsc{sum.si})))$	0.87	0.86

Results first show that some terminals, i.e. the conditional degree CD, the namespace variety NS and the topic indegree SI, are recurrent across different runs, which shows the stability of our learning process. Given the regularity we noticed of MIN.NS, we ran a third block of experiments (N) in which we used only the topological terminals and MIN.NS. Results of three runs are reported in Table 6.4.

Table 6.4 Best cost-functions with topological features and MIN.NS.

Run	Fittest g_j	f_{tr}	f_{ts}
N_1	$(log(\textsc{max.ns}/\textsc{max.cd})/\textsc{avg.ns}) + \textsc{min.ns}$	0.82	0.81
N_2	$((\textsc{min.dg}/\textsc{sum.cd})/\textsc{sum.ou}) + \textsc{min.ns}$	0.79	0.77
N_3	$\textsc{min.ns}/(log(\textsc{max.cd})/\textsc{avg.ns})$	0.83	0.75

While results confirm again the importance of the MIN.NS aggregated terminal, we observe that both the T- and the N-functions, based mostly on topological features, have a lower performance when compared to the S-ones, that include semantic terminals too. Nevertheless, the S-functions confirm the importance of MIN.SC.

We then looked at the performance of the above functions on the full dataset. We compared them with measures that we have found in the literature, namely RelFinder (RF [Heim *et al.*, 2009]), RECAP [Pirrò, 2015] and the two measures presented by the Everything is connected Engine (EICE [De Vocht *et al.*, 2013]) and by Moore et al. (M&V [Moore *et al.*, 2011]). Figure 6.6 presents a discrete plot of the $avg(P_i)$ score (Y axis) that each of the functions obtained on the examples in D (X axis), in a descending order based on $avg(P_i)$ (from the examples were the function succeeded the most to the ones where they succeeded the least). Note that the continuous curve is only used to facilitate readability and the comparison of functions.

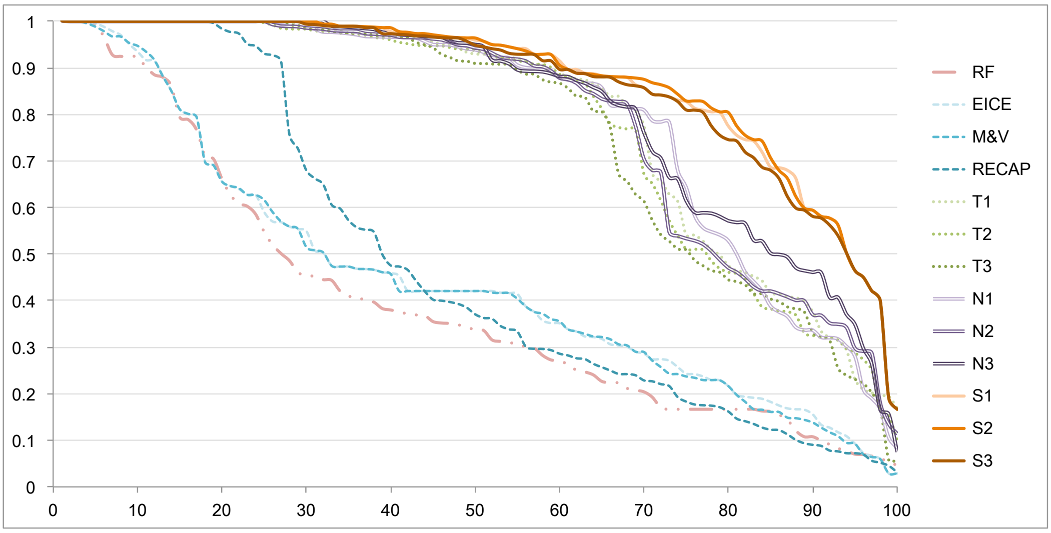


Figure 6.6 $avg(P_i)$ of the fittest functions on the full dataset, for each of the experiment blocks T, TN and S.

We finally verified if the cost-functions were affected by the type of entities in the dataset D. We removed, in turn, pairs whose entities belonged to one of the Linked Data sources presented in Section 6.4.1, and then calculated the functions' fitness $f(g_i)$ of Equation 6.3 on the filtered dataset. Results, presented in Table 6.5, confirm that the S-functions are consistent even with different datasets.

Discussion. What can be noticed from Figure 6.6 is a considerable difference between the existing approaches, which are based on ad-hoc information theoretical measures, and the

Table 6.5 Overall fitness $f(g_i)$ of the functions across datasets. $D\backslash$ indicates from which dataset the entities were removed; s indicates the size of the filtered dataset.

$D\backslash$	s	RF	RECAP	EICE	M&V	T_1	T_2	T_3	S_1	S_2	S_3
Geonames	23	0.455	0.457	0.584	0.605	0.756	0.710	0.641	0.909	**0.911**	0.887
Yago	77	0.371	0.513	0.432	0.424	0.819	0.820	0.819	**0.883**	0.881	0.880
VIAF	69	0.378	0.492	0.418	0.414	0.832	0.828	0.801	0.880	**0.888**	0.880
Unesco	79	0.390	0.457	0.442	0.437	0.784	0.764	0.750	0.858	**0.860**	0.849
LMDB	73	0.439	0.382	0.438	0.435	0.740	0.728	0.705	0.846	**0.847**	0.843
∅ (tot)	100	0.399	0.481	0.449	0.446	0.784	0.763	0.749	0.871	**0.873**	0.865

ones that were automatically learnt through Genetic Programming. Indeed, the combination of several topological characteristics sensibly improves a cost-function performance, as demonstrated by the overall fitness of $f(g_i)$ of T_1, T_2 and T_3 (see Table 6.5, last row). This means that the ranking the T-functions give for a set of path is much more similar to the ones of a human evaluator than the ones attributed by hand-crafted measures. The low performance of the existing measures suggests that they are not suitable to correctly evaluate paths that connect entities across several Linked Data datasets, as the ones we have collected in our experiments. A slight improvement can also be observed with the N-functions: the overall fitness $f(g_i)$ for them improves roughly by $0.02 - 0.04$ when compared to the T-functions. With that said, the Figure clearly shows that adding some semantic information is the key to obtain more precise results, as the S-function overall fitnesses $f(S_1)=0.86$, $f(S_2)=0.88$ and $f(S_3)=0.87$ demonstrate (i.e. fitness improvement is ca. $0.09 - 0.11$).

A function to assess Linked Data contexts. Finally, we analyse the S_2 cost-function, which reports the best performance:

$$S_2 = \text{MIN.NS} + \frac{\text{MIN.CD}}{log(log(\text{SUM.SI}))} \tag{6.5}$$

and observe that the terminals here included are the same that we had already noted as being the most recurrent among the different runs of Tables 6.3 and 6.4. As can be seen from its shape, this function prioritises paths that:

- pass through nodes with rich node descriptions (the higher MIN.NS is, the more relevant the path is considered);
- do not include high-level entities (that have many incoming *dc:subject*/*skos:broader* links, since many other entities are also of the same category), since the higher SUM.SI is, the lower the path score is;

- include only specific entities (not hubs) for paths with a small number of topic categories. Indeed, because of the use of the double log function, the ratio between MIN.CD and $log(log(\text{SUM.SI}))$ is negative if SUM.SI ≤ 10. However, the addend $\frac{\text{MIN.CD}}{log(log(\text{SUM.SI}))}$ becomes a positive factor when SUM.SI > 10.

In other words, the function prioritises specific paths (e.g. a movie and a person are based in the same region) to more general paths (e.g. a movie and a person are based in the same country).

Consistently with the discussion at the beginning of the chapter, such function is then integrated in the Linked Data pathfinding process, that we can use to assess the context between a pattern and its candidate explanations.

6.5 Conclusion and Limitations

In this chapter, we presented a supervised method based on Genetic Programming in which we learnt the cost-function to detect strong relationships between Linked Data entities, which in our case correspond to the contextual relationship between a pattern and a candidate explanation. With the assumption that the topological and semantic structure of Linked Data can be used in a blind search to identify the strongest connections, we used Genetic Programming to generate a population of cost-functions that was evolved iteratively, based on how well the individuals compared with a human evaluated training data. The results showed the validity of our idea that successful path evaluation functions are built using basic topological features of the nodes traversed by the paths, and that a little knowledge about the vocabularies of the properties connecting nodes in the explored graph is fundamental to obtain the best cost-functions. We analysed the obtained functions to detect which features are important in Linked Data to find the strongest entity relationships, and finally presented the cost-function that we learnt and we will integrate in Dedalo's Linked Data pathfinding process. Below, we present the last limitations and open issues.

❶ **The Link Traversal is forward-constrained.** Although we consider both backward and forward links for the uniform-cost search, dereferencing is clearly an obstacle to retrieve backlinks. Also, there are very few datasets that allow non-unidirectional queries with dereferencing. This means that improved cost-functions are likely to be found if using alternative discovery techniques for graphs (e.g. graph indexing), but this implies introducing a priori knowledge of the problem, which goes against the base principles set for this work.

❷ **Linked Data introduce bias.** As mentioned already in several parts of our work, this issue became much more visible with this piece of work. The blind search highly relies on the information that is declared in Linked Data. A sensible bias exists towards some information (RDF properties, vocabularies, namespaces...), and this can affect the quality of our results – namely the shape of the cost-functions. For example, a cost-function relying on types and topics, which are largely declared in the datasets that we have used, might not be as performant on Linked Data dataset of more specific domains such as bioinformatics, that might have a completely different taxonomic shape. While one possibility is to extend the GP process to new datasets, in Chapter 8 we also discuss how to measure the bias introduced by data links.

❸ **Explanations cannot be complete.** The previous point is also related to another important limitation of our approach: i.e. we cannot obtain the last component of an explanation (i.e. the theory) only by relying on the Web of Data. A deeper discussion about the problem is presented in Chapter 8.

❹ **Dedalo has to be evaluated.** Although explanations are not fully complete, we believe that the chapter we presented makes a considerable step towards automatically generating meaningful explanations. To prove that, the next chapter is focused on evaluating Dedalo on both its aspects, i.e. the induction of explanation and the definition of a context, against human judgment.

Part III

Evaluation and Conclusion

Chapter 7

Evaluating Dedalo with Google Trends

This chapter presents the user evaluation study that we designed to answer the second part of our last research question (Section 1.3.4), namely how to assess the validity of the process we designed. More specifically, we evaluated Dedalo in its two main processes, the induction of explanations and the assessment of their contextual relationship, which were presented throughout Part II. A brief summary of the whole system and an introduction on what we aim to achieve with the user evaluation is given in Section 7.1. Then, we present in detail the two evaluation studies that we performed, respectively in Section 7.2 and Section 7.3. Finally, we discuss the conclusions from the chapter and the second part of this thesis in Section 7.4.

7.1 Introduction

Starting from the hypothesis that knowledge from the Web of Data could be used to automatically explain patterns of data, the second part of this thesis focused on presenting the proposed approach, Dedalo.

We focused on showing how candidate explanations for a pattern could be generated by learning directly from the data in the pattern (or not in it) in a Machine Learning fashion, and by reasoning upon background knowledge manually extracted from Linked Data, within the Inductive Logic Programming framework (Chapter 3). Because in an ILP scenario manual effort was still required, we re-adapted some of the ILP features to design an automatic process which could exploit the Web of Data without incurring into scalability issues (Chapter 4). The result was an anytime process to induce Linked Data explanations, in which the required background knowledge was iteratively collected using a Link Traversal strategy, that we exploited to avoid the introduction of any a priori knowledge. We also demonstrated that a heuristic-based graph search could be beneficial to drive such Linked Data traversal to the best explanations in the shortest time. We then focused on how to improve the induced explanations. First, we showed how better explanations could be obtained using a Neural Network trained model that predicts the likelihood of good aggregated explanations (Chapter 5). We also showed how the complexity of their readability decreased with the number of aggregations performed, and therefore focused on how to automatically exploit the structural features of the Web of Data to provide us with contextual information, which enriches the generated explanations by revealing the connections between them and the pattern they are related to (Chapter 6).

Having designed Dedalo and demonstrated how it can perform in use-cases of different domains, our final objective is to assess the validity of what we proposed with respect to the users. In this chapter, we present the empirical evaluation of Dedalo, which we based on the most real-world scenario among our datasets – the Google Trends (#Tre). We remind the reader that the dataset consists of the web search rates of some terms that showed an interesting behaviour (e.g. the term popularity increases and decreases regularly), divided into groups of popular and not-popular dates (most of the details on how the data were created can be found in Appendix A.1). Two empirical studies, conducted in December 2014 and June 2015 and advertised in both internal (the Knowledge Media Institute, the Open University) and public fora (mailing lists, social networks), were designed with the idea of evaluating Dedalo's two main processes, i.e. the induction of the candidate explanations and the definition of the most relevant context for a candidate explanation. In both cases, the evaluation required users assessing how good were Dedalo's results.

The next two sections describe the two empirical studies in detail, including the experimental setting (i.e. data preparation, evaluation interface design, tasks to achieve, evaluation measurements), the analysis of the participation to the study (participants details and users' agreement, which can help us to interpret the results of the user study), and the final discussion about results, on both the achievements and the unsatisfactory performances.

By doing so, we will be able to assess the validity of the approach that we propose in this thesis and, more specifically to demonstrate that the Web of Data can be explored to identify explanations that reach a reasonable degree of agreement with the human evaluators. We conclude with some general considerations about this evaluation chapter, before moving to limitations, future work and conclusions in the next chapter.

7.2 First Empirical Study

As already mentioned, the first study was conducted to evaluate the explanations induced by Dedalo by comparing them with the ones provided by human experts.

7.2.1 Data Preparation

A very first step, whose results were already partly shown in Table 4.7 of Chapter 4, was to run Dedalo on each trend of #Tre and collect the set of induced Linked Data explanations.

Trends were manually chosen because they were showing an interesting behaviour (e.g. recurring at very specific moments of the year), and were selected on the basis of the existence of at least one explanation, as assessed by two human judges. This was done to avoid presenting the evaluators random or unclear trends, for which no valid explanation existed. Also, we intentionally diversified the trends' topic and popularity (e.g. popular movies/TV series, people, events) to see how Dedalo could cope with different examples[1]. We finally selected 30 trends, and created for each of them a dataset of 559 weeks (January 2004 to September 2014) and the corresponding rate of the search for that given trend during that week, as provided by the Google Trends API Endpoint[2].

The second step was to partition weeks in a group of peaks and another group of non-peaks. A week was considered as a "peak" if its search rate value was above a moving average (commonly used in time series), whose window size was fixed to 52 weeks. In other words, a given week was considered as a peak if the term, during that week, was searched significantly more with respect to its annual average. We cut the series into yearly subsets with the aim of detecting those peaks that would not be taken into account with a constant average: for instance, when the "Game of Thrones" TV series was not popular, the search rate for "A Song of Ice and Fire" increased only when a new book was released, but this rate is much smaller when compared to any search after the release of the TV series. Each week was then weighted, so that its weight w_i could be used when evaluating the explanations with the Fuzzy F-Measure. For each week, w_i was evaluated as the normalised distance between the week's search value and the moving average. To reduce noise in the cluster of peaks, we manually estimated a threshold δ related to the moving average, so that if the search value is higher than the moving average plus a certain proportion of it, then the week was considered as a peak. Figure 7.1 shows the peak group for "A Song of Ice and Fire", including the corresponding moving average.

[1] We invite the reader to refer to Annexe A.1 for more details about which trends were initially selected.

[2] https://www.google.com/trends/

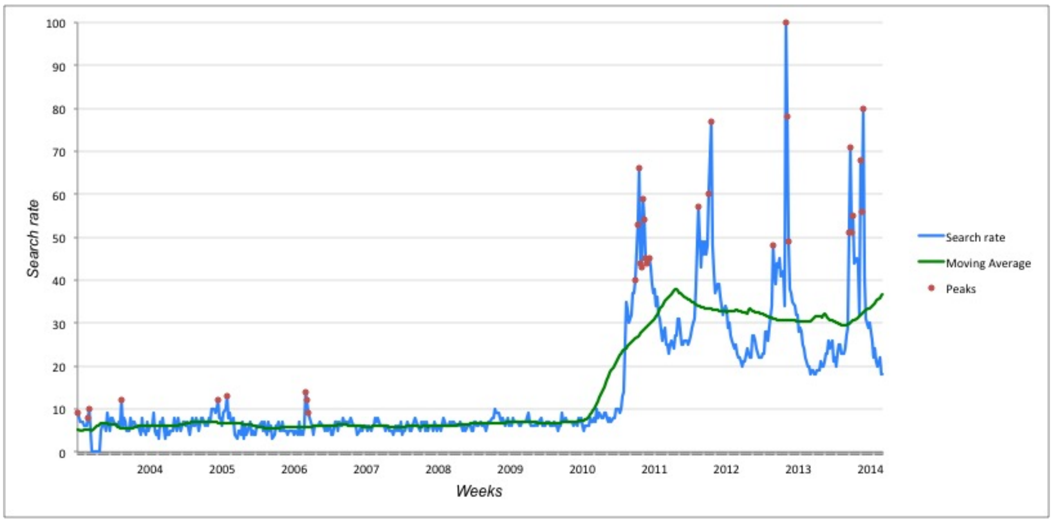


Figure 7.1 Peaks for “A Song of Ice and Fire”. The threshold δ that has been chosen is δ =1.5.

The set of weeks was then transformed in an RDF dataset. We create entities of type ddl:Week linked to events happened during that time. This was achieved by using a SPARQL query on DBpedia, where we selected any entity that had a declared date property between the beginning (`?_startDate`) and the end (`?_finalDate`) of the week:

```
select ?a where {?a ?p ?date.
filter(?date >= ?_startDate^^xsd:date
&& ?date <= ?_weekEndDate^^xsd:date)}
```

The initial dataset of URIs is necessary for Dedalo to be able to start the Linked Data traversal, and the creation of such a synthetic dataset of weeks and events is only motivated by the fact that currently there is no similar data available in Linked Data.

For each of the trends, Dedalo was given as input a file including several URIs, corresponding to dates (both from the “popular dates” and “non-popular dates” group), their weight w_i as defined by Equations 4.4 and Equation 4.5, and a boolean indicator of whether they belonged to the group of popular dates or not. Tests were run for 20 iterations, i.e. for 20 times Dedalo had selected the best path $\overrightarrow{p_i}$ from the queue Q, dereferenced its ending values V_i and evaluated new explanations with the Fuzzy F-Measure. This measure was chosen mostly because it could produce better results on the #Tre dataset, as reported in Section 4.4.

We then discarded 17 trends for which we could not find, in the resulting explanations provided by Dedalo, any plausible explanation according to our judgment. For instance, no possible explanation could be found for any of the trends related to tennis events, nor for Star

Wars. We leave for the following sections the discussion on the possible reasons for such a deficiency. The scenario finally presented to the users included 13 trends, as for Table 7.1.

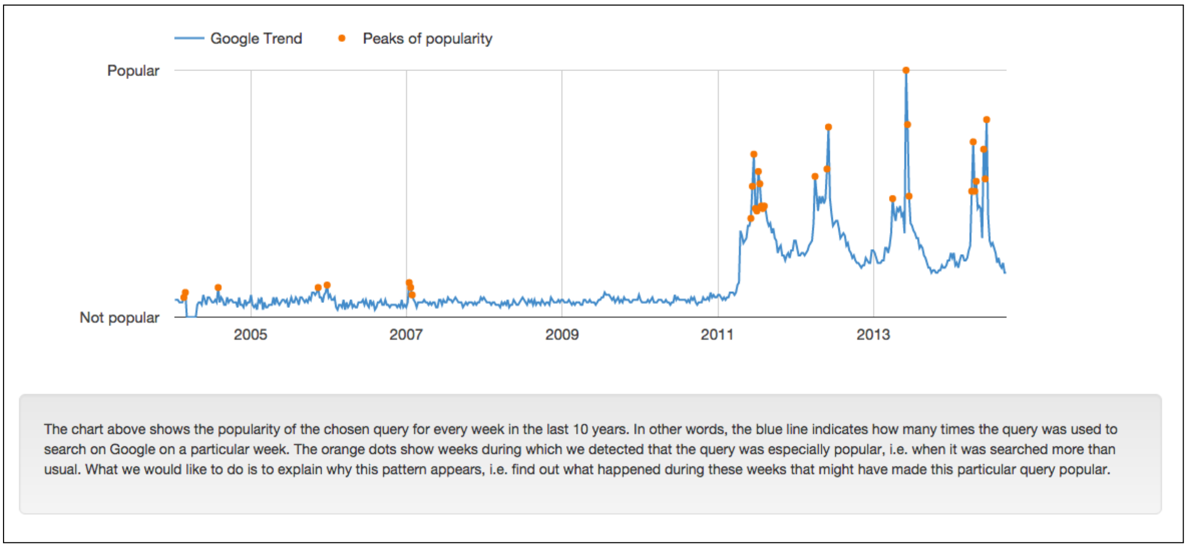
Table 7.1 Successful ✓ and unsuccessful ✗ trends chosen for the empirical study.

✓	A Song of Ice and Fire	✗	How I met your Mother	✗	Taylor
✓	Brazil	✓	Hurricane	✓	Turkey
✗	Dakar	✓	Italy	✓	Wembley
✗	Daniel Radcliffe	✓	Rugby		
✓	Germany	✓	Obama		

To avoid biasing our users, the trends selected for the empirical evaluation comprehended a variety of results including both cases where Dedalo's performance was really satisfactory (case ✓), i.e. the correct explanations were ranked among the best 10, and the ones that we considered as a failure (case ✗), since the explanations we were expecting were lower than the 10th position in the rank.

7.2.2 Evaluation Interface

The obtained data were presented through a javascript-based interface, still available online[3]. Users were required to accomplish three tasks, to be repeated on each trend, but they were left free to choose the desired number of trends to perform. Given a specific trend, the user was presented a double chart (as shown in Figure 7.2), including a blue line chart of the



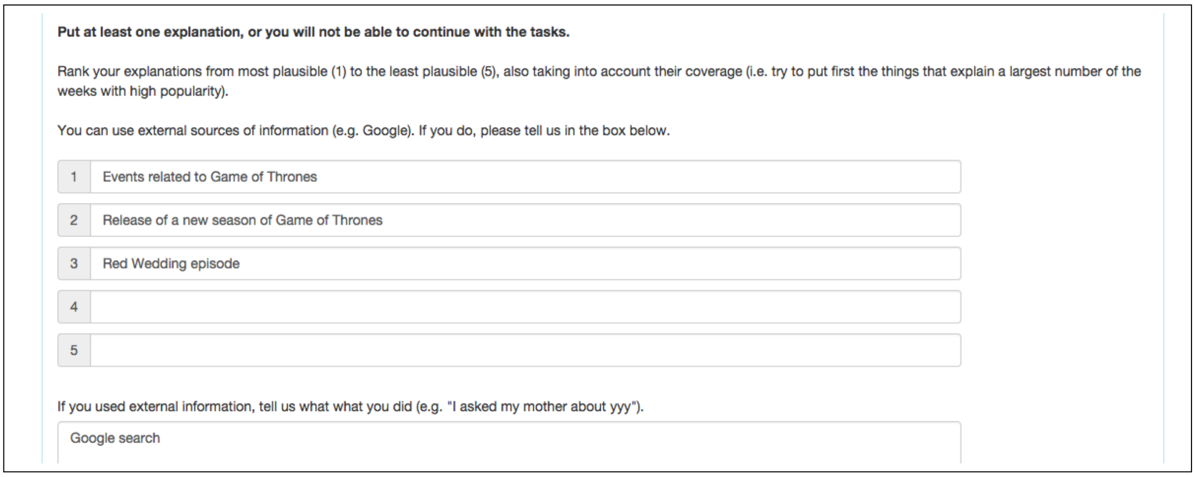
The chart above shows the popularity of the chosen query for every week in the last 10 years. In other words, the blue line indicates how many times the query was used to search on Google on a particular week. The orange dots show weeks during which we detected that the query was especially popular, i.e. when it was searched more than usual. What we would like to do is to explain why this pattern appears, i.e. find out what happened during these weeks that might have made this particular query popular.

Figure 7.2 First Empirical Study: trend plot.

[3] http://linkedu.eu/dedalo/eval/

time and popularity of the trend, corresponding to the normal Google Trends chart, and a scatterplot chart showing which dates we detected as peaks of popularity. The plot was obtained using the Google Chart API[4]. The user was then encouraged by the guidelines to think about some explanation(s) that might best represent the peaks, that he would further compare with the ones found by Dedalo.

Task 1. As indicated by the guideline "*give your own explanations for this pattern*", the task for the user was to find at least one and a maximum of 5 explanations for the peaks that he or she was shown. "Finding an explanation" was intended as trying to find a general reason for the trend's popularity or, in other words, which was the topic (or event) common to most of the peak dates.



Put at least one explanation, or you will not be able to continue with the tasks.

Rank your explanations from most plausible (1) to the least plausible (5), also taking into account their coverage (i.e. try to put first the things that explain a largest number of the weeks with high popularity).

You can use external sources of information (e.g. Google). If you do, please tell us in the box below.

1 Events related to Game of Thrones

2 Release of a new season of Game of Thrones

3 Red Wedding episode

4

5

If you used external information, tell us what what you did (e.g. "I asked my mother about yyy").

Google search

Figure 7.3 Task 1: providing explanations.

We now define this set of user-defined explanations $U = \{eu_1, \ldots, eu_l\}$, with $1 < l < 5$. In Figure 7.3, for instance, U is composed of three explanations:

eu_1: "Events related to Game of Thrones"

eu_2: "Release of a new season of Game of Thrones"

eu_3: "Red Wedding episode"

In order to emulate Dedalo's inductive process, the user was asked to insert explanations from the most generic to the most specific one. For example, the explanation eu_2, which is related to each time a new Game of Thrones season is released, explains more than one peak and comes in fact before eu_3, which is about the "Red Wedding" episode and is related only

[4] https://developers.google.com/chart/

to one peak. The same applies to eu_1, which covers a much larger number of peaks because it includes any event related to the TV series Game of Thrones (and not only the seasons).

In order to study how intensive the explanation process could be when not being a domain expert, the user had to provide in the final box information of whether he relied on external knowledge to find the explanations.

Task 2. The first 20 explanations found by Dedalo were then presented to the user, whose task was explained in the guidelines as "*map your explanations to the ones generated by Dedalo*". We call this set of Dedalo-defined explanations $D = \{ed_1, \ldots, ed_l\}$, where $l = 20$.

To avoid biasing the user, and to facilitate his comprehension, the elements in D appeared in a random order, and as natural language sentences. Explanations might or might not have been related to the given trend, and no evidence was given to the user about which ones could be correct or not. Some examples of explanations[5] in D for "A Song of Ice and Fire" were:

ed_1: Weeks including events whose topic is a subcategory of the category "Game of Thrones (TV series)"

ed_2: Weeks including events whose topic is a subcategory of the category "Rugby in Argentina"

ed_3: Weeks including events whose topic is the category "Game of Thrones (TV series) episodes"

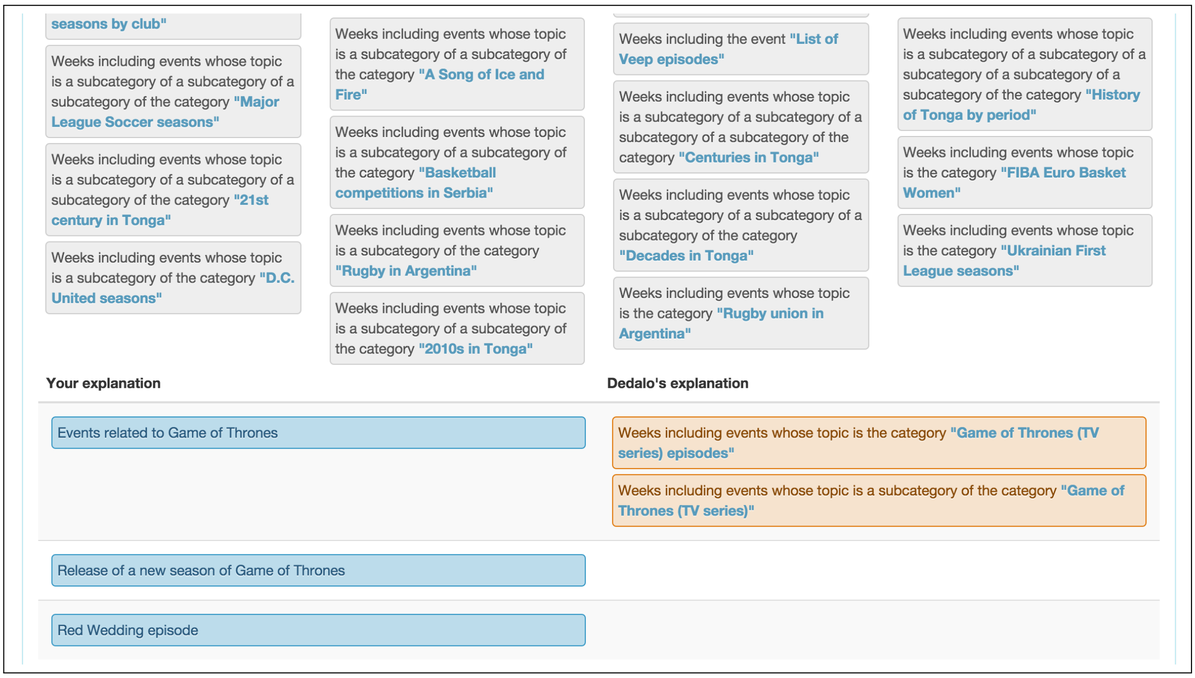
ed_4: Weeks including events whose topic is a subcategory of a subcategory of a subcategory of a subcategory of a subcategory of the category "History of Tonga by period"

ed_5: Weeks including events whose topic is a subcategory of a subcategory of the category "A Song of Ice and Fire"

The user's task was to find and map the explanation(s) in D that best fitted the ones he/she gave in U. Given one eu_i, the user was allowed to map more than one of Dedalo's explanations ed_i; however, mapping several eu_i to one ed_i was not allowed. We call $U' \subseteq U$ the set of user-given explanations with at least one mapping in D, and $D' \subseteq D$ the subset of Dedalo-defined explanations with at least one mapping in U.

In the example of Figure 7.4, we mapped events about Game of Thrones (eu_1) with both ed_1 and ed_3, that we considered being "topically" closer to the TV series than to the seasons (eu_2) or one specific episode (eu_3). Those last two remained unmapped, since we believed that no similar explanations appear in the list. Therefore, $U' = \{eu_1\}$ and $D' = \{ed_1, ed_3\}$.

[5] Note that the bizarre formulations for an explanation provided by Dedalo is due to the fact that we automatically translate the induced Linked Data paths into natural language expressions. We use the agreement between users (see Section 7.2.5) to make sure that this might not affect too much the users when performing their evaluation.



seasons by club"

Weeks including events whose topic is a subcategory of a subcategory of a subcategory of the category "Major League Soccer seasons"

Weeks including events whose topic is a subcategory of a subcategory of a subcategory of the category "21st century in Tonga"

Weeks including events whose topic is a subcategory of the category "D.C. United seasons"

Weeks including events whose topic is a subcategory of a subcategory of the category "A Song of Ice and Fire"

Weeks including events whose topic is a subcategory of a subcategory of the category "Basketball competitions in Serbia"

Weeks including events whose topic is a subcategory of the category "Rugby in Argentina"

Weeks including events whose topic is a subcategory of a subcategory of the category "2010s in Tonga"

Weeks including the event "List of Veep episodes"

Weeks including events whose topic is a subcategory of a subcategory of a subcategory of a subcategory of the category "Centuries in Tonga"

Weeks including events whose topic is a subcategory of a subcategory of a subcategory of the category "Decades in Tonga"

Weeks including events whose topic is the category "Rugby union in Argentina"

Weeks including events whose topic is a subcategory of a subcategory of a subcategory of a subcategory of the category "History of Tonga by period"

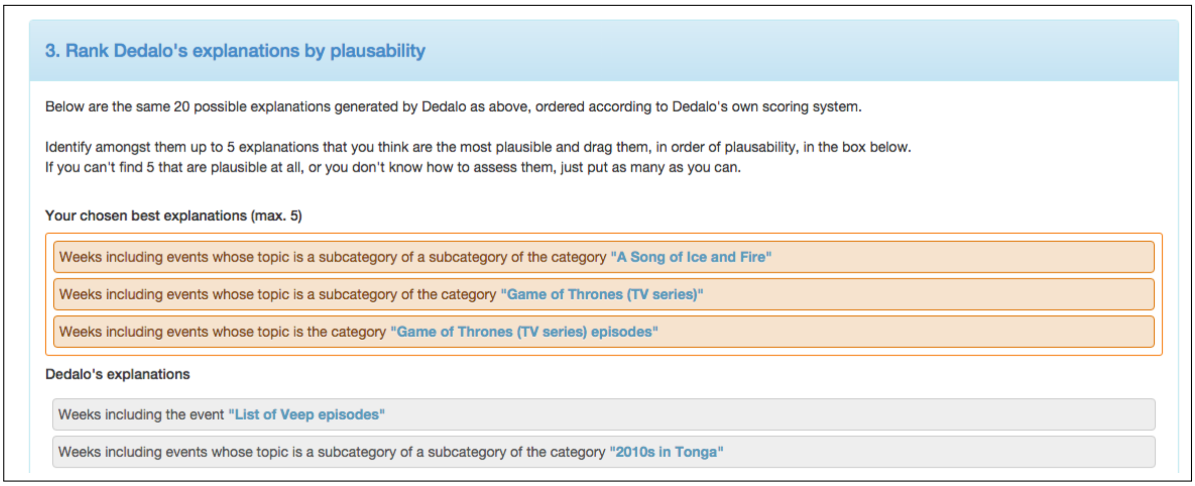
Weeks including events whose topic is the category "FIBA Euro Basket Women"

Weeks including events whose topic is the category "Ukrainian First League seasons"

Your explanation	Dedalo's explanation
Events related to Game of Thrones	Weeks including events whose topic is the category "Game of Thrones (TV series) episodes" Weeks including events whose topic is a subcategory of the category "Game of Thrones (TV series)"
Release of a new season of Game of Thrones	
Red Wedding episode	

Figure 7.4 Task 2: matching explanations.

Task 3. The user was shown again Dedalo's top 20 explanations, in the order they were ranked by the Fuzzy F-Measure, and was asked to "*rank Dedalo's explanations by plausibility*". The task consisted in choosing among the explanations in D the 5 that best explained the trend, and giving them a rank i from the most representative (1) to the least one (5). We call this set $R = \{ed_1, \ldots, ed_l\}$, where $1 < l < 5$ and $R \subset D$. For example, in Figure 7.5, R is composed of:



3. Rank Dedalo's explanations by plausability

Below are the same 20 possible explanations generated by Dedalo as above, ordered according to Dedalo's own scoring system.

Identify amongst them up to 5 explanations that you think are the most plausible and drag them, in order of plausability, in the box below.
If you can't find 5 that are plausible at all, or you don't know how to assess them, just put as many as you can.

Your chosen best explanations (max. 5)

Weeks including events whose topic is a subcategory of a subcategory of the category "A Song of Ice and Fire"

Weeks including events whose topic is a subcategory of the category "Game of Thrones (TV series)"

Weeks including events whose topic is the category "Game of Thrones (TV series) episodes"

Dedalo's explanations

Weeks including the event "List of Veep episodes"

Weeks including events whose topic is a subcategory of a subcategory of the category "2010s in Tonga"

Figure 7.5 Task 3: ranking explanations.

1 : (ed_5) Weeks including events whose topic is a subcategory of a subcategory of the category "A Song of Ice and Fire" (the novel)

2 : (ed_1) Weeks including events whose topic is a subcategory of the category "Game of Thrones (TV series)"

3 : (ed_3) Weeks including events whose topic is the category "Game of Thrones (TV series) episodes"

Here, ed_5 was prioritised because it is more general and better represents the group of peaks to explain: for instance, it comes before ed_1, which does not relate to events about the books.

In the case where no reasonable explanation was found, the user was allowed to leave the rank empty.

7.2.3 Evaluation Measurements

In order to measure Dedalo's performance compared to the users (our gold standard), we defined some "indicators", presented below.

Dedalo's Success. The first indicator, based on Task 1 and Task 2, estimates how good Dedalo was in inducing explanations according to the users. Dedalo's success $\overline{\text{DSX}}$ is defined based on the subset of explanations U' for which the user found at least one mapping in D:

$$\overline{\text{DSX}} = \frac{\sum_{i=1}^{j} |U'|}{j} \tag{7.1}$$

That is, the total number of users' explanations eu that have been mapped to at least one explanation given by Dedalo, normalised on the j number of users evaluating that specific trend. $\overline{\text{DSX}}$ ranges from 0 to 5, with 0 being the worst case (Dedalo did not find any good explanation) and 5 being the best one (all the users' explanations were found). Assuming the example of Figure 7.3 with $|U| = 3$ and $|U'| = 1$, and a second user with $|U'| = 2$, then $\overline{\text{DSX}} = 1.5$, i.e. Dedalo performed rather badly.

Dedalo's Surprise. Complementary, the results of Task 2 also tell us how wrong Dedalo was, i.e. how many of the users' explanations were not found. In this case, we count the set of explanations in U that were not mapped with at least one of D. Dedalo's Surprise $\overline{\text{DSP}}$ is as follows:

$$\overline{\text{DSP}} = \frac{\sum_{i=1}^{j} |U \backslash U'|}{j} \tag{7.2}$$

with j being again the number of users participating in the evaluation of a trend. $\overline{\text{DSP}}$ also ranges from 0 to 5, this time 0 being the best case (Dedalo missed no explanation) and

5 being the worst (all the explanations were missed). Given the user of Figure 7.4 with $|U \setminus U'| = 2$, and a second user with $|U \setminus U'| = 1$, then $\overline{\text{DSX}} = 1.5$ and therefore Dedalo performed moderately well.

User Surprise. With this indicator, we aim at quantifying how much the users did not know, by detecting the explanations that they could not think about (therefore they got surprised). The User Surprise $\overline{\text{USP}}$ looks at how many relevant explanations have been identified by Dedalo, but not by the user. To do so, $\overline{\text{USP}}$ compares D' (Dedalo's explanations that the user had found at least one mapping for) and R (the set of Dedalo's explanations ranked by the user in Task 3) as:

$$\overline{\text{USP}} = \frac{\sum_{i=1}^{j} |R \setminus D'|}{j} \tag{7.3}$$

with j being the number of users. $\overline{\text{USP}}$ ranges from 0 to 5, 5 being the best case (many explanations were found by Dedalo but not by the user, who was more "surprised") and 0 being the worst (none of Dedalo's explanations was new to the user, therefore no surprise at all). In the example of Figure 7.5, the explanation ed_5 is in R (explanations found by Dedalo) but not in D' (explanations found by the user). This means that we only thought about the TV series when looking for an explanation for the trend, but not about the book. While we did miss some knowledge, Dedalo did not, and that was indirectly admitted when the explanation ed_5 was chosen and put in the ranking during Task 3.

7.2.4 Participant Details

This first empirical study involved 83 users differing in age, education and background:

- *gender*: 25 females, 58 males;
- *age*: 10 users being less than 25 years old, 56 users between 25 and 34 years old, and 17 being 35 years old or more;
- *job*: 49 users from the academic domain and 34 from other sectors.

A specific overview of the participants per trend is given in Table 7.2.

Besides showing a huge involvement of academic staff in the study (which is certainly due to the diffusion channels used to call for participants), as well as a high difference between the participants' gender, we can observe some expected behaviours in the way trends were chosen: in general, TV series and movies were the most popular, followed by countries known for sport, politics or cultural events. As we expected, trends with the lowest popularity

Table 7.2 User preferences and some more details about age, gender and job.

Trend	Tot.	Age			Gender		Job	
		<25	25–34	>35	F	M	Academic	Other
A Song of Ice and Fire	37	5	28	4	9	28	23	14
How I met your Mother	33	2	28	3	6	27	25	8
Brazil	32	3	27	2	11	21	23	9
Daniel Radcliffe	27	1	21	5	8	19	21	6
Italy	25	0	20	5	6	19	18	7
Obama	24	0	19	5	7	17	18	6
Germany	23	2	19	2	6	17	17	6
Hurricane	21	0	16	5	4	17	16	5
Turkey	21	2	16	3	6	15	14	7
Wembley	21	0	18	3	5	16	16	5
Dakar	19	2	14	3	4	15	14	5
Rugby	17	1	14	2	2	15	12	5
Taylor	14	0	13	1	2	12	11	3

were the ones whose topic was either less popular or very specific. In the former case we have trends such as “Rugby” and “Dakar”, that are topics (rugby and motor rallies) generally not as popular as football. In the latter, we have “Wembley”, that refers to the Wembley stadium in London (so it is U.K. specific), and “Taylor”, that refers to the American singer Taylor Swift, whose popularity in Europe is not as high as in the U.S. (also, she might be more popular among people younger than the ones who were involved in the study). Alternatively, the scarce interest in this last trend could be attributed to its ambiguity, since “Taylor” can actually refer to either the singer Taylor Swift, or the actress Elizabeth Taylor, or any other person called Taylor.

Gender also affects the choice of the trends: namely, women seemed to be more interested in movies and TV series than in sports.

The age ranges do not bring any meaningful information about the preferences: very few people outside the range 25-34 were involved in the study, which could probably be strictly related to the number of participants working in the academic domain.

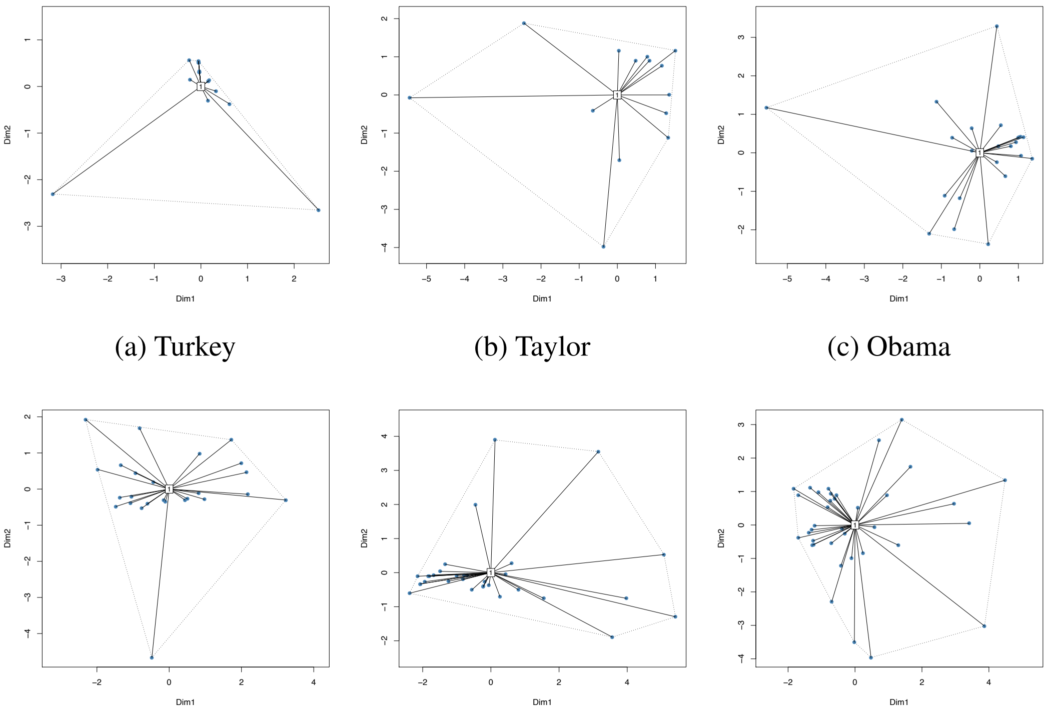
7.2.5 User Agreement

Before analysing the system’s results (Section 7.2.6), we first measure the degree of agreement among the participants, to ensure that there was a robust enough grounding for the evaluation. A homogeneous consensus means that the users know enough about a trend, and therefore we can rely on their judgement. However, if the user-provided explanations U vary too much, the users are not agreeing among each other, in which case we might also expect

Dedalo not to perform well either. Note that our evaluation assumes that the users were not affected by the readability of the explanations.

User Spread. Here, we evaluate the spread of the explanations given by the users. In order to assess it, we normalised the text of the explanations in U (using stopwords removal, stemming and term frequency weighting) and represented users as vectors of words. We then use k-means clustering (with $k = 1$) to obtain the trend mean vector (i.e. the centroid), from which we could estimate the spread. Finally, we plotted all the points on a 2D-space using multidimensional scaling and the Euclidean distance as distance function between each user and the centroid. The more the points are spread, the more heterogeneous are the explanations those users gave. Plots for some of the trends are shown in Figure 7.6.

What one can notice here is that there are cases where users agreed on the explanations they gave, resulting in a cluster of tightly grouped points close to the centroid (Figures 7.6a, 7.6b, 7.6c), and some others where the users' explanations were more varied, therefore points are more distant on the plot. In the case of Figure 7.6a, it is evident that the explanations found by the users for the trend "Turkey" are very homogeneous, since



(a) Turkey

(b) Taylor

(c) Obama

(d) Italy

(e) Brazil

(f) A Song of Ice and Fire

Figure 7.6 User spread according to the similarity of the explanations they provided, ordered by user's cohesion.

the majority of the points are very close to the centroid, and only a few are distant. In fact, a vast majority the users used the two words *thanksgiving* and *christmas*, while users who are far from the centroid happened to have given unique explanations such as "*Many new films were on the cinema*". Similarly, we observe a general agreement on the explanations for "Taylor" (Figure 7.6b), where the most popular words are *taylor* and *swift*. "Obama" (Figure 7.6c) is also an example of cohesive group: the most common words we identified are *campaign*, *presidential* and *election*, employed by the majority of the users, while further from the centroid we find users who gave unique explanations as, for instance, "*Nobel prize*".

In the second group, "Italy, "Brazil" and "A Song of Ice and Fire" (Figures 7.6d, 7.6e, 7.6f) are trends that show how users gave more miscellaneous explanations, resulting in clusters that are more relaxed (they also have a larger scale in the figures). The reason for that is likely to be the different nature of the peaks in the "popular dates" group: For instance, the peaks for "Italy" include different editions of the Football World Cup, but also L'Aquila earthquake and the Costa Concordia cruise disaster, and since they all belong to different topics, reaching a common agreement between users is harder (Figure 7.6d); similarly, peaks for "Brazil" include the Football World Cups and the 2013 riots and demonstrations against the local government, while for "A Song of Ice and Fire" (Figure 7.6f) we have both peaks related to the book releases and the TV series premieres.

Topic Variety. The variety of domains behind a set of explanations is another indicator of the user agreement, and can help in assessing whether Dedalo is more performant in cases of more homogeneous trends. For instance, if every user mentioned the Game of Thrones TV series and only one did mention the Red Wedding episode, then we cannot really consider the latter as relevant for "A Song of Ice and Fire". To identify this variety, we built an undirected graph for each trend, in which the nodes, consisting of each explanation, were connected by an edge if they had words in common. This gave us an initial idea of which words were used more or less often. We then used the Louvain Community Detection algorithm implemented within the Gephi tool[6] to create communities of explanations according to the topic they covered.

Figure 7.7 shows the communities of topics detected for "A Song of Ice and Fire". In this network, the node size increases according to how much they are connected to other explanations (i.e. their degree), while the intensity of the edges increases with the number of common words between the two explanations. The Network Partitioning algorithm is not only useful to detect the preferred words within the explanations, but also to identify

[6] https://gephi.github.io/

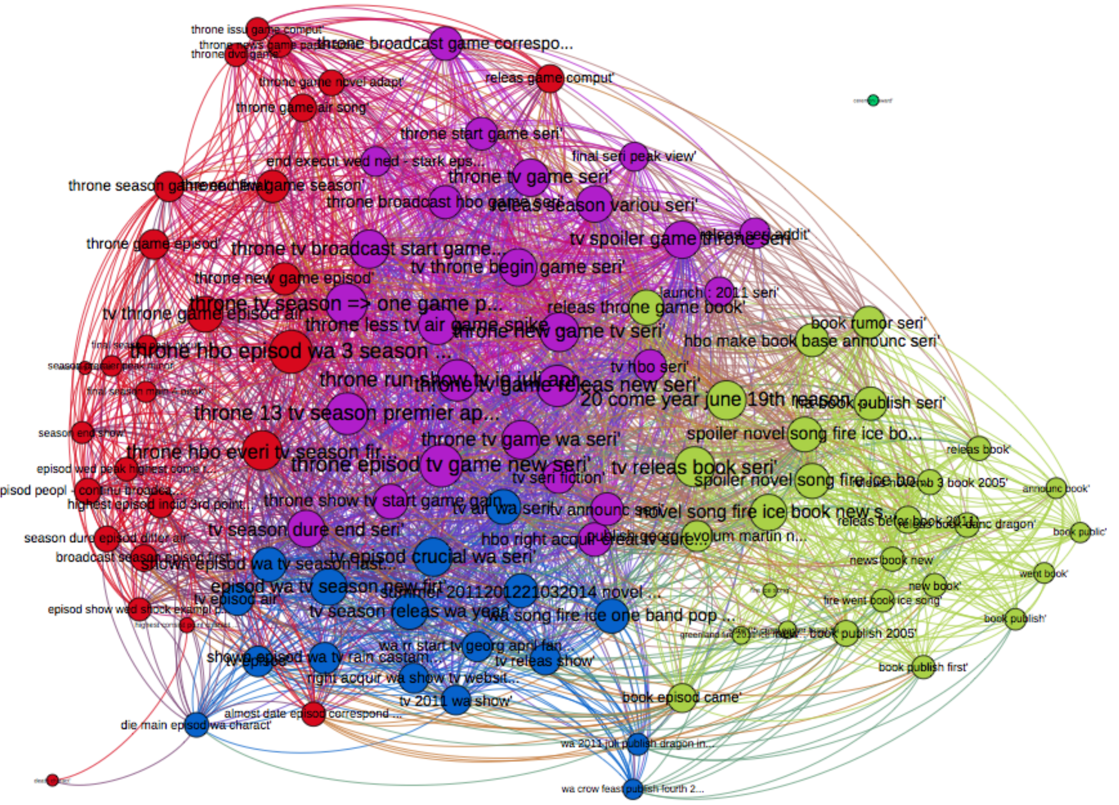
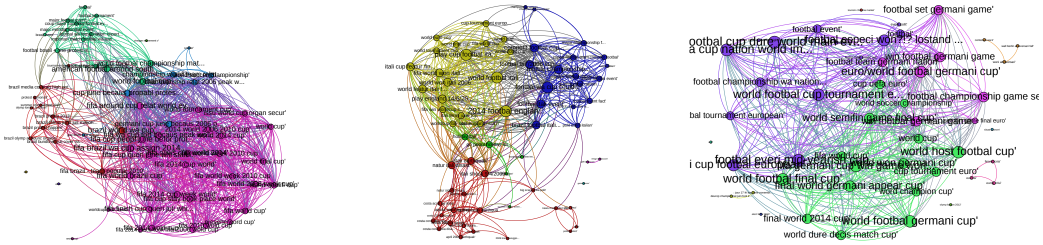

throne broadcast game correspo...
releas game comput'
throne game novel adapt'
throne game air song'
throne start game seri'
end execut wed ned - stark eps...
final seri peak view'
throne tv game seri'
throne broadcast hbo game seri'
releas season variou seri'
throne game episod'
throne tv broadcast start game...
tv spoiler game throne seri'
tv throne begin game seri'
throne new game episod'
launch 2011 seri'
releas throne game book'
book rumor seri'
throne tv season => one game p...
tv throne game episod air'
throne less tv air game spike'
throne new game tv seri'
hbo make book base announc seri'
throne hbo episod wa 3 season ...
tv hbo seri'
throne run show tv guid app...
throne tv game releas new seri'
20 come year june 19th reason'
book publish seri'
throne 13 tv season premier ap...
throne tv game wa seri'
spoiler novel song fire ice bo...
releas book'
throne hbo everi tv season fir...
throne episod tv game new seri'
tv releas book seri'
tv seri fiction'
spoiler novel song fire ice bo...
book 2005'
announc book'
episod peopl contin broadca...
highest episod incid 3rd point...
throne show tv start game gain...
tv air wa seri'
tv announc seri'
novel song fire ice book new s...
releas befor book 2012'
announc dragon'
book public'
season dure episod dure air'
tv season dure end seri'
hbo right acquir publish georg...
volum martin n...
news book new'
broadcast season episod first...
shown episod wa tv season last...
tv episod crucial wa seri'
new book'
went book'
episod wa tv season new first'
summer 20112012221032014 novel ...
fire went book ice song'
episod show wed stock exampl...
tv episod air'
tv season releas wa year'
wa song fire ice one band pop ...
book publish'
book publish 2005'
show episod wa tv rain castam...
wa n start tv georg april fan...
tv releas show'
book publish first'
right acquir wa show tv websit...
tv 2011 wa show'
book episod came'
almost date episod correspond ...
die main episod wa charact'
wa 2011 jul publish dragon in...
wa crow feast publish fourth 2...

Figure 7.7 Communities of explanations for “A Song of Ice and Fire”.

which are the topics that the user-given explanations covered. In fact, if only few nodes float apart in the graph, this means that there are explanations that only few users chose, and they might not be related to any other explanation. This is the case of the small green node at the top of Figure 7.7: it corresponds to the explanation “*ceremony award*” that only one user gave. The rest of the network is rather densely connected, meaning that there was, in general, a homogeneous agreement on the trend’s explanations. With that said, four communities (respectively in red, purple, blue and yellow) emerged. To know what they were about, we looked at the node labels and extracted a general topic: the yellow community was the only one related to the release of the books (*book* and *novel* are the most shared words), while the others, that are related to the TV series, represented different subsets of it: the blue community concerns single aired *shows*, the red one the *seasons* of the series, and the purple one the general TV *series*. It is worth mentioning here that those communities related to the same topic (with a different granularity) and matched some of the explanations found by Dedalo: For example, ed_5 relates to the category “A Song of Ice and Fire” and corresponds to the yellow group; ed_1 relates to the category “Game of Thrones (TV series)” and corresponds to the purple community; and ed_3 relates to the category “Game of Thrones (TV series) episodes” and corresponds to both the blue and the red groups.



(a) Brazil (b) Italy (c) Germany

Figure 7.8 Communities of explanations for “Brazil”, “Italy” and “Germany”.

Some other types of explanation communities are given in Figure 7.8. Figures 7.8a and 7.8b present the communities detected for “Italy” and “Brazil”, for which Dedalo had given less clear results, probably because the peak group included events of different natures. In both cases, users generally agreed on the explanations, resulting in some densely connected communities; nevertheless, we observed that the communities corresponding to unique events (as the cruise disaster or the riots) remain apart and less connected. As a different case, the trend “Germany” of Figure 7.8c was much more uniform in the explanations (mainly related to football events) given by the users, as shown by the three uniformly connected clusters.

Finally, for the case of “Taylor” in Figure 7.9, for which Dedalo was not able to find any

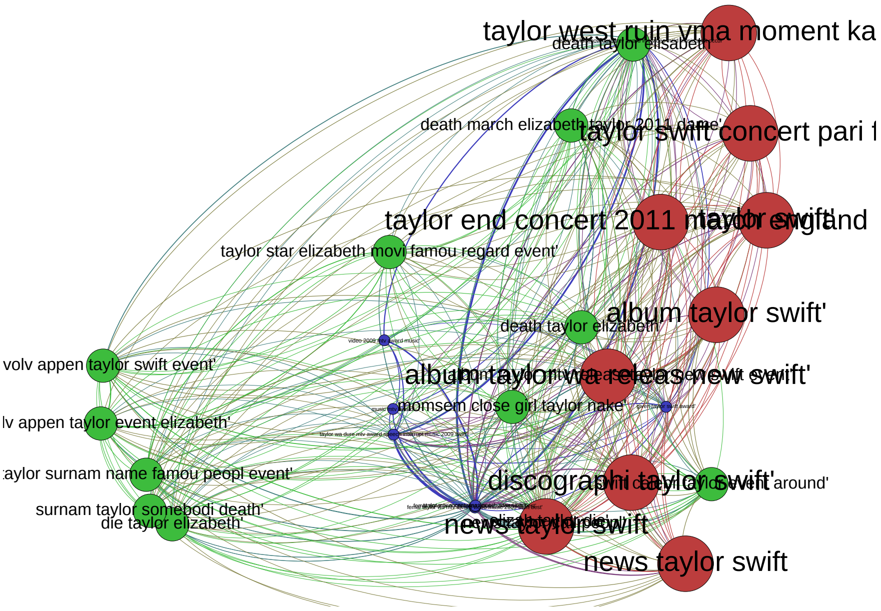

taylor west ruin vma moment ka
death taylor elisabeth'
death march elizabeth taylor 2011 dame'
taylor swift concert pari f
taylor end concert 2011 manor england
taylor swift'
taylor star elizabeth movi famou regard event'
album taylor swift'
death taylor elizabeth'
volv appen taylor swift event'
album taylor wa releas new swift'
momsem close girl taylor nake'
lv appen taylor event elizabeth'
aylor surnam name famou peopl event'
discographi taylor swift'
around'
surnam taylor somebodi death'
die taylor elizabeth'
news taylor swift
news taylor swift

Figure 7.9 Communities of explanations for an ambiguous trend, “Taylor".

proper explanation, the users' explanations have resulted in the following communities: (i) death of Elizabeth Taylor, (ii) Taylor Swift album release and (iii) awards given to the singer. Since (i) and (ii) are not related (they do not even correspond to the same person in fact), and since (i) and (iii) are specific to one single event, it is now more understandable that it was harder for Dedalo to find significant explanations.

7.2.6 Results, Discussion and Error Analysis

In this part, we used the success and surprise indicators defined in Section 7.2.3 to evaluate Dedalo's performance.

On Dedalo's knowledge. First, we analysed what and how much Dedalo "knew" (or did not). To do so, we compared $\overline{\text{DSX}}$ and $\overline{\text{DSP}}$ (Dedalo's success and surprise), as in Figure 7.10. We finally distinguished 3 different groups of performance.

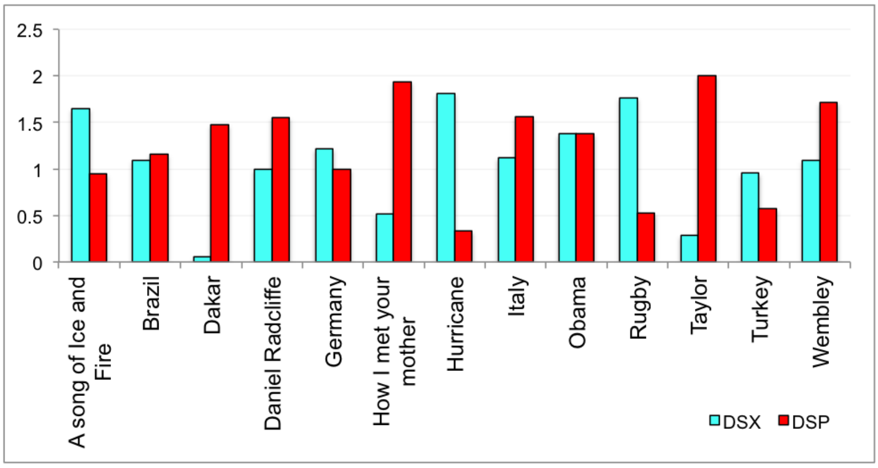


Figure 7.10 Dedalo's Success Rate ($\overline{\text{DSX}}$) and Surprise Rate ($\overline{\text{DSP}}$).

A first group includes trends where Dedalo's success (in blue) was higher than its surprise (in red); see the trends "A Song of Ice and Fire", "Germany", "Hurricane", "Rugby" and "Turkey". What we understand from this is that users declared that Dedalo was able to find a high number of explanations (high $\overline{\text{DSX}}$), while missing few (low $\overline{\text{DSP}}$). In those cases, we considered Dedalo's performance to be satisfactory.

A second group consists in trends with an acceptable performance, i.e. "Brazil", "Daniel Radcliffe", "Italy", "Obama" and "Wembley". Here, Dedalo's missed explanations ($\overline{\text{DSP}}$) were more than the ones successfully found (or just as much; see "Obama"); however, the $\overline{\text{DSX}}$ was still reasonable. This means that users declared at the same time that many explanations were missed by Dedalo (high $\overline{\text{DSP}}$), but many others were actually found (high $\overline{\text{DSX}}$). Such a performance is again motivated by the nature of the trend: some of the peaks relate to events that are too specific to be identified by induction, and this prevented Dedalo from finding explanations that the users, on their side, could find more easily. For example, Dedalo did find the generic explanations for both "Italy" and "Brazil", i.e. the ones related to the Football World Cup are indeed identified and ranked high; Nevertheless, the unique events such as Italy's disasters or Brazil's riots could not be found.

The last group, for which the $\overline{\text{DSX}}$ rate was really low, corresponds to the "negative examples" (cfr. the cases marked ✗ in Table 7.1), for which Dedalo was able to find the right explanations, but not to rank them among the best 10.

On the users' knowledge. In this case, we focused on what users knew and what they missed. Since we could not directly measure the surprise of the users, we compared the number of explanations that the users did not think about ($\overline{\text{USP}}$), with the ones Dedalo had found ($\overline{\text{DSX}}$), and estimated if and how much users did not know.

Again, the idea we brought here was that if an explanation ed_i did not appear in D' of Task 2, but did appear in the ranked ed_i in R of Task 3, then this explanation was one the user did not think about. We also put $\overline{\text{USP}}$ and $\overline{\text{DSX}}$ in relation to the usage of external background knowledge to see whether there was a correlation with the two indicators. Results are presented in Table 7.3.

In general, we observed that there were always 1 or 2 explanations per trend that the users missed. More specifically, we distinguished four behaviours (♣, ♦, ♥, ♠) among the users, presented below.

Case ♣. Trends where users knew the topic enough not to need too much external knowledge, but still missed some explanations. In those cases, $\overline{\text{USP}}$ was more than $\overline{\text{DSX}}$, but the percentage of external knowledge was low, which confirmed that few users needed it. See, for instance, "Obama" (it was easy to relate Obama to the U.S. presidential elections, but not to the federal elections), "Brazil" (assuming one was not expert enough to know the exact times of the World Cup), and "Turkey", for which users probably thought about the country while the trend clearly refers to the searches for turkey recipes during Thanksgiving and Christmas.

Case ♥. Popular trends the users knew about, for which external knowledge was only a support to confirm their explanations, and for which they missed very few explanations. This was the case of "A Song of Ice and Fire", "Italy", "Rugby" or "Daniel Radcliffe", whose $\overline{\text{USP}}$ was lower than $\overline{\text{DSX}}$, and the usage of external knowledge was high.

Table 7.3 Comparison between the user surprise ($\overline{\text{USP}}$) and Dedalo’s success ($\overline{\text{DSX}}$). We also include the use of external knowledge (EK), measured as the percentage of users that filled the final box in Task 1 with a non-negative answer (i.e. they relied on some external knowledge to give their explanations).

Case	Trend	% EK	$\overline{\text{USP}}$	$\overline{\text{DSX}}$
♦	Taylor	78.57	0.42	0.28
♥	A Song of Ice and Fire	62.16	0.70	1.65
♥	Rugby	58.82	1.35	1.76
♥	Italy	52.00	0.52	1.12
♥	Daniel Radcliffe	51.85	0.40	1.00
♠	Wembley	47.61	1.09	1.09
♦	Dakar	47.36	0.42	0.05
♣	Obama	45.83	1.79	1.37
♣	Brazil	43.75	1.34	1.09
♥	How I met your Mother	36.36	0.36	0.51
♠	Hurricane	33.33	1.57	1.81
♣	Turkey	23.80	1.00	0.95
♠	Germany	21.73	0.73	1.21

Case ♦. Trends, like “Taylor” or “Dakar”, that the users did not know enough, where their surprise was high ($\overline{\text{USP}}$ higher than $\overline{\text{DSX}}$). Considering that in Dedalo’s top 20 explanations neither Taylor Swift nor the Paris-Dakar rally were mentioned, we concluded that users might have misunderstood those trends and wrongly ranked some explanations in Task 3. The high percentage of external knowledge usage confirms this idea.

Case ♠. Miscellaneous trends of an average popularity, such as “Germany”, “Hurricane” or “Wembley”, for which $\overline{\text{USP}}$ is lower, but still balanced, when compared to $\overline{\text{DSX}}$. The usage of external knowledge is low in those cases.

Error Interpretation and Possible Solutions. We finally analysed those cases in which Dedalo did not perform well, and found three major reasons.

In the case of “How I met your Mother”, the pattern was big and noisy: the number of peaks actually not related to the trend was too elevated, and this prevented Dedalo from inducing the correct explanation(s). This was therefore a *pattern quality problem*. The same explains why some of the #Tre trends, related either to too specific entities (people such as “Carla” or “Beppe”, TV series such as “Sherlock”) or unique events (“Volcano”, “AT&T”, “Bitcoin”, “Star Wars”[7]) were excluded from the user evaluation. While noisy patterns may be refined to obtain good explanations, specific events cannot be explained using an inductive

[7] Only two out of the seven current Star Wars movies have been released after 2004, when Google Trends started the data collection.

process, and finding a solution for this goes beyond our research scope.

In the case of “Dakar”, Dedalo’s explanations mainly mentioned the KFC restaurant chain. It turned out that a recurring U.S. event sponsored by KFC happens at the same time as the Paris-Dakar rally. To confirm this, we compared the “Dakar” trend with the one corresponding to this event[8]. The error here is considered to be due to a *spurious correlation*, and can be improved when filtering out explanations that do not have any contextual relationship with the pattern using, for instance, the web-relatedness metrics (see Section 7.3.1 below).

In the case of “Taylor”, the lack of information within Linked Data about Taylor Swift, as well as the ambiguity of the word, made the derivation of a general explanation more difficult. Similarly, the tennis-related trends had to be discarded since the related events did not have a proper date declaration in DBpedia, and they could not be retrieved by the initial SPARQL query. Such a problem is considered a *Linked Data quality problem*, which is related to Linked Data bias as discussed in Chapter 8.

7.3 Second Empirical Study

To evaluate Dedalo’s performance in ranking contextual relationships between an explanation and a pattern using the cost-function of Equation 6.5, we conducted a second empirical study, in which we could compare Dedalo’s results with the users’ gold-standards rankings.

7.3.1 Data Preparation

To prepare the data for this second study, the first step we had to achieve was to remove the spurious correlations, i.e. those explanations which were not related to the pattern, but which were ranked at high positions anyway. For that, we used the Normalised Google Distance (NGD, or “Google similarity”), a measure of semantic distance between two terms x and y calculated based on $f(x)$ and $f(y)$, i.e. the number of web pages containing respectively x and y, and $f(x, y)$, the number of web pages containing both x and y as reported by the Google search engine [Cilibrasi and Vitanyi, 2007]. The idea is that terms with similar meanings (or from similar domains) are likely to co-occur more often in web pages, while terms with dissimilar meanings will co-occur less frequently. The lower the score, the more the terms are related. In our scenario, we used the NGD to measure how “distant” an explanation is from its pattern, and discard it above a given threshold. To calculate the

[8] Just compare the trends “Dakar” and “Big Bash” on http://www.google.com/trends/

NGD for an explanation $\varepsilon_{i,j} = \langle \overrightarrow{p_i} \cdot v_j \rangle$, we searched for how frequently its ending value v_j occurred with the trend on the Web. For example,

NGD_1(“A Song of Ice and Fire”, “Game of Thrones TV series”)$= 0.33$
NGD_2(“A Song of Ice and Fire”, “Game of Thrones TV series episodes”)$= 0.55$

means that the term “A Song of Ice and Fire” appears in more pages with the term “Game of Thrones TV series” than with the term “Game of Thrones TV series episodes”, and therefore the first two terms are more related. Finally, for each trend, we kept only the 50 most-related explanations according to NCG. This helped already in filtering out a considerable amount of unrelated explanations.

The second step was to find a set of alternative contexts between an explanation and its trend, so that users could rank them and we could evaluate the Linked Data pathfinding process. For that, we chose a DBpedia entity t_i corresponding to the trend in hand (in our case, $t_1 =$ db:ASongOfIceAndFire(novel)) and paired it with ending value v_j of each of the 50 explanations. We then ran a bi-directional breath-first search in order to find the set of alternative Linked Data paths $\overline{p_i}$ connecting t_i and v_j. This search was limited to 30 cycles and if no path was found the pair was discarded. We finanlly obtained 27 pairs with their alternative paths, distributed per trend as in Table 7.4.

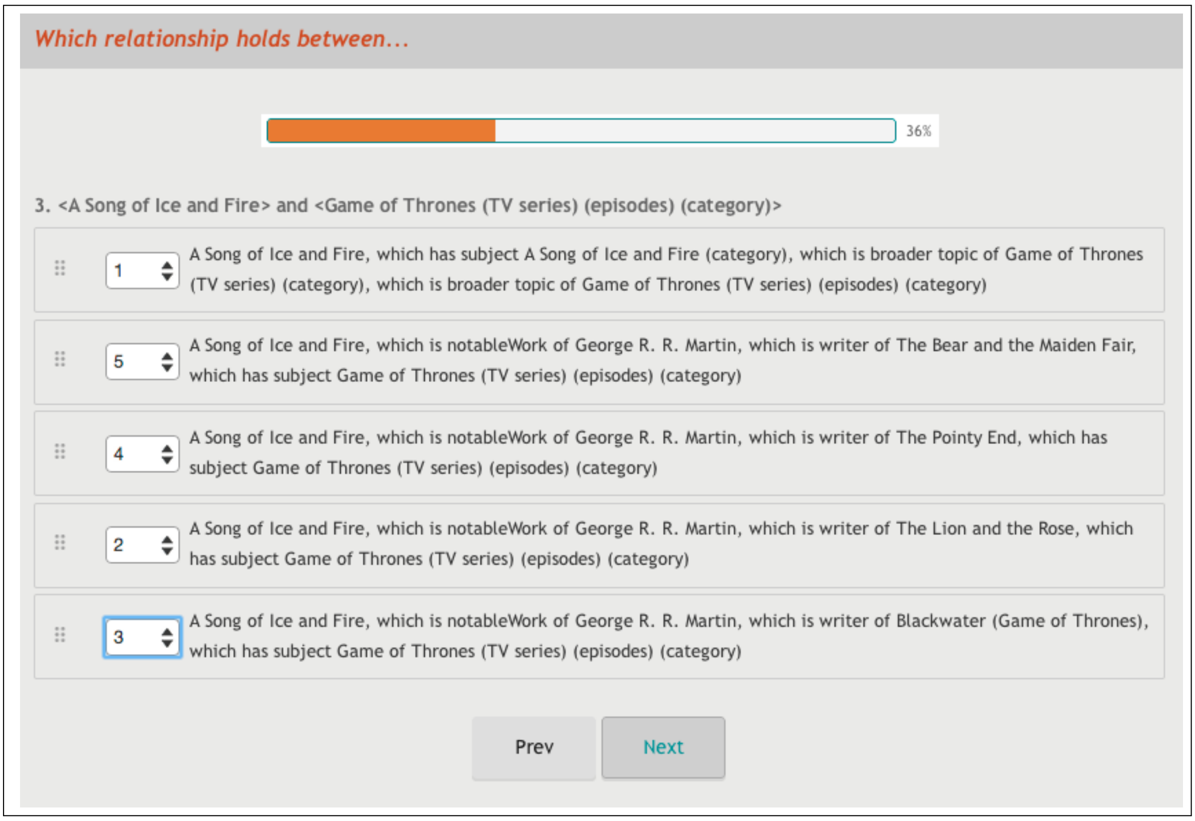
Table 7.4 Number of paired explanations per trend.

Trend	pairs	Trend	pairs
Hurricane	7	Wembley	2
Obama	6	Germany	1
A Song of Ice and Fire	3	How I met your Mother	1
Brazil	2	Italy	1
Daniel Radcliffe	2	Rugby	1

7.3.2 Evaluation Interface

The second evaluation was built based on a SurveyMonkey[9] interface, in which users were required to rank the 27 sets $P_i = \{\overline{p_1}, \dots, \overline{p_n}\}$ of alternative Linked Data paths between a pair trend-explanation from rank 1 (the most relevant connection) to rank n (the weakest relationship). Because we were mostly looking into assessing the strongest relationship, users were asked to focus on ranking properly the first three relationships only. Figure 7.11 shows how some paths were ranked for the pair db:ASongOfIceAndFire(novel)-dbc:GameOfThrones(TVseries). Again, Linked Data paths were automatically translated into natural language sentences to improve user readability.

[9] https://www.surveymonkey.com



Which relationship holds between...

3. <A Song of Ice and Fire> and <Game of Thrones (TV series) (episodes) (category)>

1 A Song of Ice and Fire, which has subject A Song of Ice and Fire (category), which is broader topic of Game of Thrones (TV series) (category), which is broader topic of Game of Thrones (TV series) (episodes) (category)

5 A Song of Ice and Fire, which is notableWork of George R. R. Martin, which is writer of The Bear and the Maiden Fair, which has subject Game of Thrones (TV series) (episodes) (category)

4 A Song of Ice and Fire, which is notableWork of George R. R. Martin, which is writer of The Pointy End, which has subject Game of Thrones (TV series) (episodes) (category)

2 A Song of Ice and Fire, which is notableWork of George R. R. Martin, which is writer of The Lion and the Rose, which has subject Game of Thrones (TV series) (episodes) (category)

3 A Song of Ice and Fire, which is notableWork of George R. R. Martin, which is writer of Blackwater (Game of Thrones), which has subject Game of Thrones (TV series) (episodes) (category)

Figure 7.11 Second Empirical Study: ranking paths.

To obtain more homogeneity in the judgments and avoid the typical long-tail problem (i.e. few pairs judged by many, and many pairs ranked by few), we created 3 different surveys with 10 pairs each, so that once we had reached 10 participants for one survey (meaning that 10 judges had ranked the 10 pairs in it), we could close the survey and change the redirection link[10] to a new survey.

7.3.3 Evaluation Measurements

During the second evaluation, we looked into assessing how relevant Dedalo's ranked explanations were according to the users.

Ranking Success. In this case, as in Chapter 6, we used the Normalised Discounted Cumulative Gain to compare Dedalo's ranked explanations before and after the Web-based filtering using the ranks provided by the users in Task 3 (first evaluation). The Cumulative Gain for a path is defined as in Equation 6.1; however, since users did not provide a rel_m

[10] http://linkedu.eu/dedalo/eval/relFinder

score for the paths, we assessed it based on the rank m that the explanation had in the set R:

$$rel_m = |R| - m \tag{7.4}$$

As in Chapter 6, we then normalised each path's CG based on the ideal ranks the users had given, and averaged them into $avg(P_i)$ as in Equation 6.2. Finally, we estimated the $\overline{\text{RKS}}$ for a given trend as:

$$\overline{\text{RKS}} = \frac{\sum_{i=1}^{j} avg(P_i)}{j} \tag{7.5}$$

with j being the number of pairs for a given trend. The closer to 1 $\overline{\text{RKS}}$ is, the better Dedalo was in ranking explanations.

Pathfinding Success. We also used the Normalised Discounted Cumulative Gain to measure how good the Linked Data pathfinding cost-function of Equation 6.5 was in ranking contexts compared to the rankings provided by the users during the second evaluation. Since users were asked in this case to focus on the first three strongest relationships, we defined the path relevance score rel_m assessed by the users as:

$$rel_m = |4 - m| \tag{7.6}$$

The Pathfinding Success is then evaluated on each trend as above:

$$\overline{\text{PFS}} = \frac{\sum_{i=1}^{j} avg(P_i)}{j} \tag{7.7}$$

7.3.4 User Agreement

The second evaluation was designed in order to assess the validity of Dedalo's rankings and participants were not asked to provide personal details. As said, we limited the number of users of a survey (3 in total) to 10, for an overall participation of 30 people.

We then used Fleiss Kappa to evaluate how much users agreed on ranking the sets of alternative paths in the second evaluation. In Table 7.5, we show the results collected, namely the pairs presented to the users, the number of alternative paths between a pair of entities, and the rating agreement of the 10 judges. Although there is no standardised interpretation of the kappa's values, the higher those are, the stronger the users agreed between each other.

[11] redirection from the db:Hurricane entity.

[12] Note that the first entity refers to the novel, while the second to the topic within the DBpedia taxonomy.

Table 7.5 Pairs presented in the second evaluation, number of alternative paths $|P_i|$ and Fleiss Kappa k agreement.

Pair		$\|P_i\|$	k
db:Brazil	dbc:FIFAWorldCupTournaments	7	0.471
db:TropicalCyclone[11]	dbc:AtlanticHurricaneSeasons	6	0.433
db:Brazil	dbc:2014FIFAWorldCup	9	0.328
db:BarackObama	dbc:UnitedStatesPresidentialElectionsInDelaware	8	0.259
db:BarackObama	dbc:UnitedStatesPresidentialElectionsInMissouri	7	0.249
db:DanielRadcliffe	dbc:HarryPotter	10	0.246
db:TropicalCyclone	dbc:TropicalCycloneMeteorology	14	0.243
db:Wembley	dbc:UEFAChampionsLeagueFinals	11	0.189
db:BarackObama	dbc:UnitedStatesPresidentialElectionsInWashington(state)	7	0.181
db:DanielRadcliffe	dbc:21st-centuryBritishChildren'sLiterature	12	0.178
db:TropicalCyclone	dbc:TropicalCyclonesByStrength	4	0.175
db:BarackObama	db:HistoryOfTheUnitedStates(1991-present)	20	0.172
db:BarackObama	dbc:UnitedStatesPresidentialElectionsInMassachusetts	6	0.166
db:BarackObama	dbc:UnitedStatesPresidentialElectionsInIllinois	8	0.163
db:ASongOfIceAndFire	dbc:GameOfThrones(TVseries)	7	0.157
db:Wembley	dbc:FootballLeaguePlay-offs	3	0.142
db:ASongOfIceAndFire	dbc:GameOfThrones(TV series)(episodes)	5	0.129
db:Italy	dbc:AustraliaAtTheFIFAWorldCup	10	0.120
db:ASongOfIceAndFire	dbc:ASongOfIceAndFire[12]	20	0.104
db:TropicalCyclone	dbc:HurricanesInFlorida	5	0.098
db:TropicalCyclone	dbc:HurricanesInTheDominicanRepublic	17	0.072
db:Wembley	dbc:FootballLeaguePlay-offs-Finals	5	0.037
db:TropicalCyclone	dbc:HurricanesInHaiti	17	0.034
db:Germany	dbc:FIFAWorldCupTournaments	4	0.029
db:HowIMetYourMother	dbc:TelevisionSeriesSetInThe2030s	20	-0.008
db:TropicalCyclone	dbc:HurricanesInTheUnitedStatesByState	7	-0.023
db:Rugby	dbc:InternationalRugbyBoardCompetitions	4	-0.036

What we notice here is that better agreements were reached with more popular topics such as "Brazil", "Hurricane" or "Barack Obama", even when the number of paths $|P_i|$ to rank was large. On the contrary, when the trend topic was less trivial, the more paths were to be ranked, the less agreement was reached. The nature of the set of paths that was presented to the users also affects the agreement: using a breath-first search, in fact, many of the paths were equally relevant. For instance, in the example of Figure 7.11, each of the paths ranked from 2 to 5 are related to one specific episode of Game of Thrones, and making a real difference between them is not an easy task.

7.3.5 Results, Discussion and Error Analysis

Here, we evaluate Dedalo's ranking and pathfinding success using the two indicators presented in Section 7.3.3.

On the success in ranking explanations. First, we analysed how Dedalo performed in ranking the explanations in Task 3 (see Section 7.2.2), by comparing the $\overline{\text{RKS}}$ score before and after the web-based filtering of irrelevant explanations. We additionally compare the explanation rankings obtained with the Fuzzy F-Measure to the F-Measure one. Results are reported in Table 7.6.

Table 7.6 Dedalo's ranking success $\overline{\text{RKS}}$ before (B) and after (A) Web-based filtering.

Trend	Fuzzy F-Measure		F-Measure	
	$\overline{\text{RKS}}$ (B)	$\overline{\text{RKS}}$ (A)	$\overline{\text{RKS}}$ (B)	$\overline{\text{RKS}}$ (A)
A Song of Ice and Fire	0.271	0.738	0.249	0.684
Brazil	0.956	0.979	0.608	0.806
Dakar	0.907	0.986	0.693	0.781
Daniel Radcliffe	0.054	0.325	0.183	0.568
Germany	0.946	0.980	0.952	0.999
How I met your Mother	0.289	1,000	0.310	1,000
Hurricane	0.468	0.528	0.418	0.546
Italy	0.892	0.929	0.208	0.224
Obama	0.492	0.566	0.858	0.888
Rugby	0.701	0.726	0.307	0.361
Taylor	0.526	0.770	0.061	0,000
Turkey	0.187	0.331	0.134	0.317
Wembley	0.742	0.794	0.881	0.889

First, by comparing the F-Measure and Fuzzy F-Measure before the filtering (B), we observe how the F-Measure ranking success is lower than the Fuzzy F-Measure one in most of the cases, meaning that the best 20 explanations provided by the F-Measure were irrelevant when compared to user-provided explanations. Those results are an additional confirmation that the Fuzzy F-Measure is required to obtain more precise explanations. As for the few exceptions to it (namely "Daniel Radcliffe", "How I met your Mother", "Obama" and "Wembley"), a plausible explanation might be the unrelated events that happened at the same time of the events actually related to the pattern (e.g. the "Lacrosse Championship" happening at the same time than the events at the Wembley stadium) and this, along with a less clearly defined pattern, might have induced the Fuzzy F-Measure to introduce more noise. In other cases, e.g. for "Daniel Radcliffe", the user-based gold-standard included misleading explanations, such as events related to the TV series *Rome* or to the category "Television series set in Italy".

We then compared the results before (B) and after (A) the web-relatedness filtering. In most of the cases, the web-relatedness helps in improving the relevance of the induced explanations. In those cases in which results were more ambiguous, such as "A Song of

Ice and Fire" and "How I met your Mother", we observe how the web-relatedness filtering considerably helped in reducing the noise, reaching a perfect agreement in the second case. As said in Section 7.2.6, both "Dakar" and "Taylor" were misunderstood by the users, and therefore those results are not to be taken into consideration.

On the success in ranking contexts. The final analysis consists in evaluating whether the pathfinding cost-function was good in assessing the strongest paths in the Google Trends scenario. In Table 7.7, we report the $\overline{\text{PFS}}$ of each trend.

Table 7.7 Pathfinding success on each trend.

Trend	$\overline{\text{PFS}}$	Trend	$\overline{\text{PFS}}$
Brazil	1	Germany	0.571
Daniel Radcliffe	1	Italy	0.531
Rugby	0.834	Obama	0.388
Wembley	0.604	Hurricane	0.264
A Song of Ice and Fire	0.593	How I met your Mother	0

**Dakar, Taylor,Turkey - N/A

What we observe from here is that some trends were more successful than others, as in the case of "Brazil" and "Daniel Radcliffe". Here, the agreement between users and Dedalo was perfect (or almost), which further demonstrates how the Web of Data structural features that we learnt using Genetic Programming are beneficial in a large and cross-domain graph search space such as in the Google Trends scenario. Acceptable results are also obtained for "Wembley", "A Song of Ice and Fire", "Germany" "Italy", which present a lower $\overline{\text{PFS}}$ score mostly because of the quality of the rankings that were presented to the users – as previously said, many alternative paths were equivalent and making a real difference might be hard. A similar explanation justifies the less satisfying performance on "Obama" and "Hurricane": as reported in Table 7.5, we observed that the number of paths to choose was in general higher and, by consequence, users tended to disagree much more. Similarly, since there was no user agreement on "How I met your Mother", the function was not able to properly rank the relationships. As for the trend "Rugby", for which the $\overline{\text{PFS}}$ score was surprisingly high when compared to its kappa user agreement, we believe that the relationships to be ranked being structurally equivalent, brought both Dedalo and the users into erroneous conclusions. Finally, no pairs for "Dakar", "Taylor" and "Turkey" could be found by the bi-directional search due to Linked Data lacking information, and therefore evaluating those three trends in our setting was not possible.

7.4 Final Discussion and Conclusions

This chapter presented the final step of our work, in which we evaluated the proposed approach empirically in a user-study where users were asked to assess the validity of Dedalo's results – namely, the induced explanations and their contextual relationship with the pattern. We described how we designed the evaluation upon a real-world scenario, i.e. the popularity increase of certain web searches according to the Google Trends tool, and then thoroughly presented the two evaluations that we conducted. For each study, we presented how data, tasks and interfaces were prepared; we showed which aspects we intended to evaluate; we described the users' engagement; and finally we presented the results that we obtained, and discussed reasons for errors and misinterpretations.

Rather than presenting the limitations, that we discuss in detail in Chapter 8, we present here the conclusions we could draw from the empirical study.

The first conclusion concerns the general process of explaining a pattern. We showed that the explanation cannot be easily achieved when not being a domain expert: the majority of the users in the first study, in fact, employed external knowledge to support the explanations they had given, which also demonstrates that expertise is not sufficient to explain a multi-faceted pattern.

The second one concerns the feasibility of generating candidate explanations automatically with the Web of Data. We showed that it is indeed possible to automatically induce explanations, provided that patterns are clear (in the case of the trends, recurrent) and enough information about them is available within Linked Data. In those cases, the Fuzzy F-Measure is the factor allowing our system to identify the right explanations, by giving more importance to data points that are more representative of the pattern.

The third one is on the importance of the contextual information. We showed that the Fuzzy F-Measure is affected by coincidences with patterns of a more heterogeneous nature, and that Web-based filtering can be used to reduce such misunderstanding. On the other hand, we saw how contextual information has to play a role in the process to automatically generate explanations, because it further qualifies them, and we demonstrated how the cross-domain connected knowledge of the Web of Data can be exploited for this purpose.

Generally, this chapter validated our hypothesis that we could use the knowledge from the Web of Data to automatically generate explanations that are as meaningful as the ones human evaluators would give in many cases, which makes us conclude that the approach presented in this thesis is, to a certain extent, feasible. In the following chapter, we will discuss the limitations of our approach and some future work.

Chapter 8

Discussion and Conclusions

In this final chapter, we present the conclusions to our work. After the introduction of Section 8.1, we summarise the approach in Section 8.2, highlighting how our research questions have been answered and which are our contributions. The following Section 8.3 discusses the limitations of the approach, and some extensions that we aim to investigate in the future. The chapter closes with Section 8.4 with final remarks and some lessons that we have learnt.

8.1 Introduction

In the previous chapter we presented an extensive analysis of Dedalo, based on the empirical user-study that was performed during the third year of our research. For that, we used a real-world scenario, in which the focus for the participants (both Dedalo and the users) was to explain the popularity of terms from Google Trends. In short, the empirical study consisted in showing to users the explanations that we automatically produced with Dedalo, and whose soundness was then validated against the users' judgement. One of our objectives was to show that the process of interpreting patterns is indeed strenuous and time-consuming for humans, because it requires knowledge that one might lack, or that might not be promptly accessible, especially when various domains are involved.

We showed that it is possible to automatically retrieve knowledge from the Web of Data in order to produce explanations for the phenomena that were encoded into patterns. Secondly, we showed that those explanations are as meaningful as the ones given by humans. Finally, we demonstrated that the Web of Data acts as a support for the interpretation process, because it contains the knowledge that one might lack. This also supports our hypothesis that the structured knowledge and the techniques associated to the Web of Data can in fact improve the Knowledge Discovery process.

The scope of this chapter is to give the reader a conclusive overview of our research work. We analyse here what and how much we have done, and we compare this with the objectives that we originally set. For that, we summarise the work achieved in our thesis and how the research questions that we initially identified were answered. Out of those, we can then discuss which limitations our approach presents – we are particularly interested in finding some plausible reasons causing them. The investigation of those can also help us in defining and proposing new solutions and extensions of our work, that we will undertake in the future. Some of them, as we will see, have already partly investigated and have resulted in new publications.

8.2 Summary, Answers and Contributions

The goal of this thesis was to show that it is possible to automatically explain data patterns using background knowledge from the Web of Data. The main motivation behind that was to show how the interconnected and cross-domain knowledge of the Web of Data could be exploited to assist experts (and non-experts) in the process of explaining their results, and therefore to improve the process of discovering knowledge.

Based on this main research hypothesis, i.e.

the process of pattern explanation can be improved by using background knowledge from the Web of (Linked) Data

our work focused on setting up a process in which we could obtain sound explanations for a pattern under observation, given some background knowledge derived from the Web of Data. This process, of course, had to be completely human-independent, both in producing the explanations and in finding the background knowledge it required. Multiple research questions arose from setting such an objective, namely:

- what do we mean by explanation? (Section 1.3.1)
- how do we automatically get the relevant knowledge from the Web of Data? (Section 1.3.2)
- how do we know which knowledge is required for an explanation? (Section 1.3.2)
- how do we automatically generate explanations? (Section 1.3.3)
- how do we know that our explanations are good? (Section 1.3.4)
- how do we know that our process is good? (Section 1.3.4)

On the basis of those research questions, we can now analyse the answers and contributions we did bring.

8.2.1 Definition of an Explanation

Did we define what is an explanation?

Considering that the concept of explanation was at the core of our research hypothesis, the first question that we focused on was on how to formally define it. In fact, there was a need for us to identify the elements acting in an explanation, so that the process that we would build could know what to look for.

The challenge here was to prove that we were not making assumptions of why a phenomenon happens, nor identifying some potentially correlated events, whose validity could not be proved. Identifying those elements and how they interact in an explanation was therefore the key for stating that we were able to automatically produce meaningful and complete explanations for some phenomena. The challenge, then, became to find those elements, and once they were found, to use them to formally define an explanation.

The methodology we chose to answer this question was to conduct a survey in Cognitive Science (Section 2.1), where we studied what is important in an explanation according to the experts of the different disciplines (philosophy, psychology, neuroscience, computer science, sociology and linguistics). The result of the study was that disciplines in Cognitive Science structurally do see explanations in the same way. Besides the contextual or lexical differences that they can have, explanations are always composed by the same four components: an event (that we called anterior event or explanans) that happens first, a second event (posterior event or explanandum) which follows, a set of circumstances (or context) that links the two events, and a law (or theory) that governs their occurrence. We modelled those components in a small ontology that we called the Explanation Ontology and presented in detail in Section 2.1.2. This ontology, as well as the survey in Cognitive Science, represented our main contribution for this research question.

The formal model for an explanation revealed exactly which were the tasks that we needed to complete in our process. Given a posterior phenomenon encoded in a pattern, our ultimate target was to create an automatic process, which would use the knowledge from the Web of Data to derive the three other components: the anterior event, the context and the theory.

8.2.2 Detection of the Background Knowledge

Did we find a way to detect the background knowledge in the Web Data?

The next question we targeted was how to detect in the Web of Data the background knowledge that was needed to derive the components of an explanation. This question required us to investigate how to use the structured and cross-domain nature of Linked Data to reveal such knowledge.

Being determined to use Linked Data, the major challenge for us was to avoid falling into computational issues due to the large space of analysis. What was required was to focus on how to strategically find in Linked Data only the right piece of background knowledge. This research question raised two problems at the same time: on the one hand, the one of automatically finding and choosing the right datasets in the large choice offered by Linked Data (inter-dataset problem); on the other, the one of finding in a selected dataset the piece of information required for the explanation (intra-dataset problem).

The first contribution we brought here was to show that a combination Linked Data Traversal and some graph search strategies were the key to discover knowledge with little computational effort. In both Chapter 4, focused on the detection of the anterior events participating into an explanation, and Chapter 6, focused on the detection of the context, we showed that using URI dereferencing and the native links between entities was the key to agnostically access datasets on-the-fly, to efficiently access only the necessary portion of data, and to avoid data crawling and indexing – which would have implied, at the same time, introducing a priori knowledge about the problem and increasing the computational costs.

With that said, our second contribution for this question was to show which are the features of Linked Data that help in finding the right piece of information. In Chapter 4, we showed that an ad-hoc adaptation of the Shannon's Entropy measure was a successful solution for the detection of the right candidate anterior events. This was achieved after a comparative study on different graph search heuristics, where we showed that Entropy was the best measure in accessing the required information in an efficient (in time and search space) way. Analogously, in Chapter 6, we showed that specific, low-level entities that were richly described were the important Linked Data structural characteristics to be taken into account in order to efficiently identify the context in which the pattern and its candidate explanans happen.

8.2.3 Generation of the Explanations

Did we find a way to automatically generate explanations using background knowledge from the Web of Data?

Once detected the right background knowledge, the next problem was how to use it to automatically generate complete explanations. The particular challenge in this context was to be able to emulate the human process of "generating understandable explanations using one's own background knowledge", without running into the computational issues that the data deluge would bring.

Following the idea of using logical inference and reasoning, we showed that Inductive Logic Programming was an interesting solution to the problem. The main insight here was to recognise that Inductive Logic Programming frameworks, that combine features from Machine Learning and Logic Programming, are able to automatically derive (first-order clausal) hypotheses based on some reasoning upon a set of positive and negative examples and some background knowledge about them. The first contribution we brought was to show, in Chapter 3, how we could use ILP to generate pattern explanations with background knowledge from the Web of Data. With this scope in mind, we redesigned our problem as an ILP one, where: (1) items in the pattern to explain played the role of the positive examples from which we would learn; (2) the hypotheses to be derived were the anterior event explaining the pattern; and (3) the background knowledge upon which we would have based our reasoning was composed by Linked Data statements.

Once shown the potential of ILP in generating potentially useful anterior events for a pattern, we proposed in Chapter 4 to extend and adapt the process to make it more suitable for Linked Data, whose size and complexity could not be handled by the ILP frameworks, notoriously known for not being efficient or scalable. The approach we presented, defining Dedalo's initial design, was considered more adequate because it used an anytime process, in which the background knowledge was iteratively enlarged using the above mentioned graph Link Traversal-based search, which allowed to constantly find new and possibly better anterior events to a pattern. The design and implementation of this process represents our second contribution for this question.

8.2.4 Evaluation of the Explanations

Did we find a way to evaluate the explanations, and to evaluate the approach we propose?

The final question consisted in how to make sure that the induced explanations could be validated and evaluated as valid for a pattern. The major challenge consisted in defining what makes an explanation to be valid and how to measure it.

Here, the first part of our work was focused on evaluating the generated explanations, especially by finding which were the criteria to assess their interestingness. We began by using standard accuracy measures, i.e. the F-Measure and the Weighted Relative Accuracy, to statistically determine how much an induced anterior event would be correlated to a pattern. To do so, we used the anterior event as a classifier that was tested on the ground truth of the pattern. Inspired from cluster analysis approaches, we then extended the F-Measure to a weighted variant of it, that we called “fuzzy” F-Measure. Our choice, intentionally done to prioritise the most influential data within the pattern, gave a first contribution to the problem: in fact, we demonstrated that better explanations could be found when taking into account that items influence patterns in a different way. As a second contribution to the problem, we proposed in Chapter 5 to improve the accuracy of explanations by aggregating different explanantia, and used for that a trained Neural Network to predict which aggregations were more convenient. Looking for producing explanations more complete with respect to the Explanation Ontology, we finally focus on showing a pattern and an induced Linked Data explanation to be in the right context. We measured the pertinence of this context using the evaluation function learnt in Chapter 6. Our major contribution here was to reveal which were the features of the Web of Data topology to be taken into account when measuring entity relatedness.

The second part of this problem concerned how to evaluate our approach against domain experts and users. For that, we designed an empirical user-study based on a real-world scenario (the interpretation of Google Trends), where we evaluated Dedalo’s results against human judgment. Chapter 7 presented how we articulated this study, where users were required to provide their own explanations to a trend’s popularity as well as ranking contexts according to their strengths, that we then used to assess the validity of Dedalo’s results. Our results were finally measured in terms of how much knowledge we could automatically find, and how helpful an automatic process to produce explanation could be for non experts in the relevant domain. The general evaluation showed that our approach was in fact able to identify explanations that could be as interesting as the ones of the human evaluators, and possibly to bring new knowledge beyond what non expert users would already know.

The work we presented demonstrated that it is possible to use the Web of Data as a source of background knowledge to explain data patterns, and it showed some of the possible solutions to achieve it. Of course, this does not come without limitations and possibilities of extension, that we will discuss in the following section.

8.3 Limitations and Future Work

In this section, we will discuss the major limitations of Dedalo. We tried to divide them following the narrative of our thesis and, for each of them, we will present and discuss some ideas of future work that we could (or already started to) undertake.

8.3.1 Dedalo vs. the Explanation Completeness

During our work, we often recalled that the explanations, according to the Explanation Ontology defined in Section 2.1.2, had to be composed of four elements – namely, two events, the context that relates them and the theory which make those events happening. Given that, our work was intentionally focused on automatically deriving candidate anterior events to explain patterns and, consequently, on identifying the context relating those. This means that it can be easily argued that Dedalo is not able to produce explanations that are complete with respect to the model we designed, because it is missing a process to automatically derive the law (theory), which is the general principle that governs the event happenings. At the moment, deducing this theory is left to the users to whom explanations are presented. For Dedalo to be complete, what is needed therefore is to set up an automatic process to derive a theory, possibly using knowledge from the Web of Data.

To derive a theory behind a candidate explanation, ontologies are an interesting starting point. For example, the ontological schema of an extracted candidate explanation and of its context might reveal part of the theory, e.g. "A Song of Ice and Fire" *is a* novel, "Game of Thrones" *is a* TV series, and TV series *are based* on novels. However, assuming that all the required relationships are somewhere described in Linked Data might be too naïve. Moreover, the theory also consists in the assumptions behind which the pattern was created, such as "people search over the Web for what they are interested in". This sort of information might be found in top-level ontologies such as Cyc[1], but it is difficult to expect that each of the assumptions behind the creation of a pattern can be represented in common-sense knowledge bases. Contrary to the trend searches, in which case only one clear assumption lies behind them, others patterns might be built upon more composite assumptions, which might be more difficult to find in the Web of Data. Also, given that we have listed different components that take part in a theory, the first question that should be answered is: "What else might be needed for a theory to be complete?". Finally, even assuming the whole theory has been found, its validation would require several patterns of the same type, which might not be available. Although automatically deriving a theory is indeed a valuable future work that we considered, we believe that the efforts and challenges it would require might result in several years of research at different conceptual levels, and such research would go in a different direction of our current one.

With that said, a possibility of future work that we foresee is to start with the assumption that we will not be able to find in the Web of Data the general principles that we are looking

[1] http://www.cyc.com/kb/

for, and therefore we can only use Dedalo as a mean to bring the new (missing) knowledge in it. This work opens a plethora of challenges, i.e. how to find the knowledge we need for a given theory; how do we generate this knowledge; how do we combine it with that part of knowledge that it is already accessible (if any) in the Web of Data; how do we evaluate the generated theories against the Web of Data; and how can we make them useful for it. We believe, however, that such a work would contribute on creating a continuity between the Web of Data, which is mostly data-driven, and ontological knowledge, that is more conceptual, upon which a theory needs to be built.

A second way of direction can be to focus on the extension of the Explanation Ontology. One could focus on how to refine the model by introducing which qualities are necessary for an explanation to be complete according to the cognitive sciences as, for instance, validity, truth or novelty. This direction of work is particularly interesting because it would make the evaluation of the candidate explanations a richer process, which includes both quantitative and qualitative aspects. New perspectives going beyond the disciplines in Cognitive Science might also be explored: for example, the model might include the audience that the explanation is targeting (e.g. experts and non-experts might in fact need different components, or a difference vocabulary might be used) or which is its goal. This would mostly mean adapting the traversal approach, which at the moment is thought to be domain-agnostic, and possibly reconsidering some of the measures that drive it. Finally, we might foresee an evaluation of the model using several frameworks to automatically derive explanations.

8.3.2 Dedalo vs. the Knowledge Discovery Field

One of the main motivations behind this thesis was to promote Dedalo as a novel framework for discovering knowledge thanks to the "easily-accessible" information from the Web of Data. The general idea we carried forward was to show the Knowledge Discovery community how explanations could be induced with knowledge from the Web of Data, and how we could compare Dedalo's induction to the common reasoning approaches used in Logic Programming. While we repeatedly presented it as an ILP-inspired framework, Dedalo is clearly not ready to be compared to reasoning frameworks, and consequently to be soundly presented as a contribution in the long-tradition Knowledge Discovery field.

The main argument from the Logics perspective is that Dedalo relies on the graph structure of Linked Data, and does not resort to any form of inference, as the ILP algorithms generally do. Also, in conformity with the Semantic Web's open-world assumption, Dedalo's inductive process only relies on the *assumed-to-be-true* statements found in Linked Data. This makes Knowledge Discovery experts, for whom true facts are only so if they can be asserted also based on some false evidence known in the knowledge base, more reluctant in accepting Dedalo.

Given this point, in a future work we intend to investigate a deep comparison between Dedalo and different machine learning techniques, and more precisely some Inductive Logic Programming ones, in order to identify the features that we are missing for Dedalo to be acceptable in a different community, but also to show that logics methods might not be effective in reasoning at a Web-scale (due to the abundance of uncertain and heterogeneous information). In accordance to what was discussed in Section 8.3.1, such a comparison might benefit from the inclusion of the process to derive the theory based upon reasoning over the ontological structure of the explanations. This would require us to thoroughly compare Dedalo and standard Machine Learning methods, in order to show how much more computational effort (in time, resources and knowledge accessibility) is required by those to derive the same results. Also, we could express Dedalo's output explanations in one of the Semantic Web standard rule languages, and compare them to common Knowledge Discovery patterns (e.g. association rules). Such a comparison would, on the one hand, give a sound justification to our choices, convincing that Dedalo's target problem is not an ad-hoc one but it is motivated by the nature of the Web of Data; on the other, it will also be a step forward in getting two communities (the Knowledge Discovery and the Web of Data one) closer to each other.

8.3.3 Dedalo vs. the Explanation Evaluation

Our work largely discussed the evaluation of Dedalo's explanations, that we eventually based on the candidate event statistical accuracy with respect to a pattern, combined with an evaluation function to measure the strength of the context between them. The issue that can be argued here is that the chosen evaluation strategy is highly sensitive, i.e. it can be easily affected by external factors which bring noisy results (as we have frequently seen).

One of the main reasons for such a sensitivity is that Dedalo was designed with the idea that explanations could be derived independently from the quality of the cluster, under the (naïve) assumption that datasets are naturally balanced. This means that the data in a pattern are considered to be equally important, and so are the patterns in the dataset. Also, Dedalo's strategy to induce explanations relies on the assumption that a hard clustering was performed on the dataset, i.e. that data are divided into distinct patterns. Under these assumptions, the choice of the F-Measure seemed the most valuable. With that said, balance in real-world scenarios is rare, and we foresee the need of extending Dedalo's evaluation so to take into account those aspects.

A possible way to improve the explanation evaluation is to focus on improving the F-Measure, which might not be adequate to our context the way it is designed right now. This would mean weighting the explanation score based on factors such as the balance in the dataset (e.g. using the pattern sizes) so that to enshrine their natural difference, or assessing the statistical significance of explanations based on some distribution probabilities (e.g. how can we estimate discrepancies in the ground truth?). More fine-grained evaluation measures could also be considered, namely the Mean Average Precision or the R-Precision (which averages the precision of a result over a set of different points), Accuracy (which also takes into account the True Negatives rate of a classification), and cluster quality measurements such as Sum of Squared Error, Purity, Normalised Mutual Information or Rand Index.

A second, possible scenario that is more Linked Data-oriented, is to improve the explanation evaluation based on assessing the quality of the collected Linked Data, namely by assessing how trustworthy is their provenance. A large portion of the Linked Data community is focused on proposing methodologies and metrics to assess the quality of datasets in Linked Data, based on their accessibility, content, relevance and trust, and those could be used to filter out explanations which seem less relevant.

A final work that we foresee, in alignment with the discussions of Chapter 6, is to study and analyse the topology of the Linked Data Cloud more deeply, namely the datasets aggregation, distribution and connectivity, so to detect what makes some explanations better than others, and therefore making a step forward towards understanding "what the represented data tell us", rather that understanding "how to use them".

8.3.4 Dedalo vs. the Linked Data Bias

The previous section also bring us to discuss the last limitation of our approach, that was already investigated in a preliminary study. While Linked Data quality measurements can help in assessing how trustful the information in the Web of Data is, it is arguable that Dedalo's results are also dependent on the *amount* of information therein declared. In fact, one of the major assumptions behind it is that the knowledge that is represented in the traversed Linked Data is naturally balanced and distributed: in other words, "we can trust the results we obtain, since no information is missing in there". This is a rather optimistic view, that was refuted in several parts of the work – namely, when our results were affected because the knowledge in the traversed datasets was not evenly distributed. We call(ed) this issue the "Linked Data bias".

The original problem stemmed from the Open University student enrolment dataset #OU.S, for which we were looking at explaining why the Health and Social Care faculty was more popular around Manchester than in other places of the U.K. (see Figure A.5, Appendix A.1). The best explanation we could obtain for that was that the faculty was popular because it was settled in the Northern Hemisphere, and the reason that we found was that DBpedia had incomplete information regarding places and locations, and such an uneven distribution was somehow aligned (to an extent) with the cluster, i.e. it was creating a bias in our results. In this scenario, an interesting area of study for future work is to be able to quantitatively and qualitatively identify such a bias, so to make Dedalo, and in general any Linked Data application, bias-aware, i.e. able to compensate the produced results.

Quantifying the bias. As a preliminary work, we studied and presented in [Tiddi *et al.*, 2014a] a process to numerically assess the bias in linked datasets. The challenge we had to face here was that, in order to quantify a bias in a dataset, the distribution of its information should be compared to an unbiased (ground truth) one, which is however not available. What we proposed in this work, then, was to approximate such a comparison using the information carried through the dataset's connections. More specifically, we suggested to measure the bias of one dataset using its projection into a dataset connecting to it through equivalence statements. The general idea was to compare the property values' distribution of the entities in the dataset with the one of the entities in its projection. The approach we proposed is briefly presented below.

1. *Problem formalisation.* Given two datasets A and B, we call $B' \subseteq B$ the projection of the dataset A in B, i.e. the set of entities in B' that are linked to an entity in B through an equivalence statement (e.g. *rdfs:seeAlso*, *skos:exactMatch*, *owl:sameAs*). For example, in Figure 8.1, where we try to establish the bias in LMDB (A) using its projection in DBpedia (B), we have $B' = \{$db:Stagecoach, db:TheGreatDictator, db:CityLights$\}$ and $B = B' \cup \{$ db:CaptainAmerica, db:X-Men, db:Godzilla$\}$. In order to measure the bias in A, we need to compare the property distribution of the entities in B with the property distribution of entities in B'.
To identify these distributions, given a dataset D , we say:

- $classes(D)$ is the set of the types for the entities in D.
- $properties(c, D)$ is the set of properties that apply to instances of the class c in D.
- $values(c, p, D)$ is the set of possible values (entities or literals) of the property p for the class c in D.

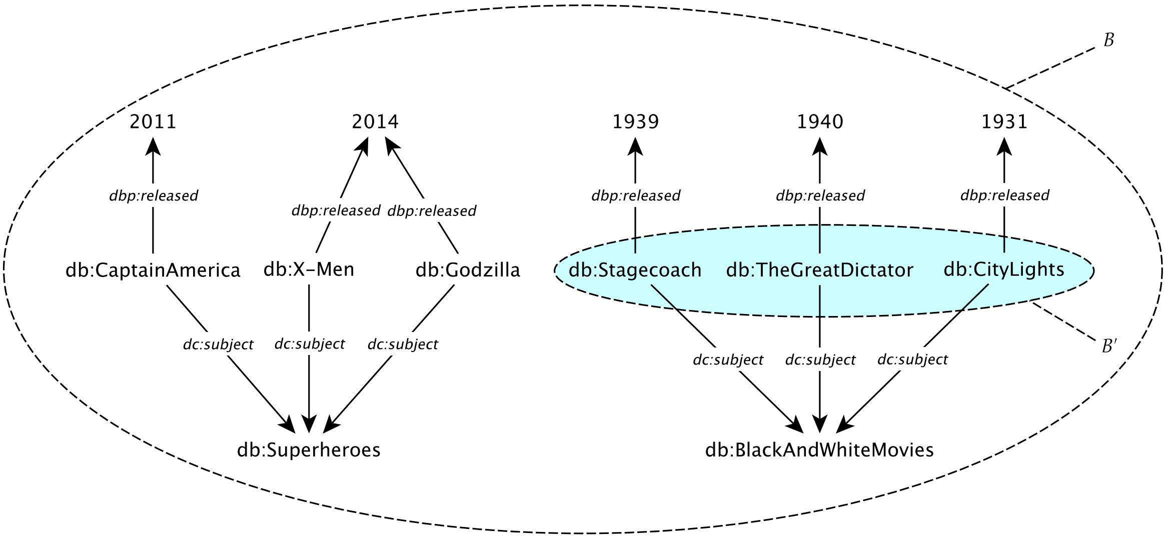


Figure 8.1 Identifying LMDB bias. B' is the projection of LMDB in DBpedia (B).

To identify towards which information A is biased, we statistically compare the distributions of the values in $values(c, p, B')$ with the ones in $values(c, p, B)$, for each of the properties $p_i \in properties(c_i, B')$ and for each of the classes $c_i \in classes(B')$. The expectation is that, when comparing B' to B, if B reveals that A is biased (noted $B \rightsquigarrow A$), then there will be a significant difference in some of the distributions of the values in B' and B.

If we take the simplified example of Figure 8.1, where we analyse the LMDB dataset using DBpedia (DBP⇝LMDB), we have $classes(B') = \{$db:Movie$\}$, because the entities $e_i \in B'$ are only movies, and consequently $properties($db:Movie, $B') = \{$dc:subject, dbp:released$\}$. We can see that all the entities of B' are categorised as black and white movies, while the movies of B are also about superheroes. From comparing the distribution of $values($db:Movie, *dc:subject*, $B) = \{$db:BlackAndWhiteMovies, db:SuperHeroes$\}$, with the one of $values($db: Movie, *dc:subject*, $B') = \{$db:BlackAndWhiteMovies$\}$, we can say that LMDB is biased towards black and white movies. Similarly, the average of the values of *dbp:released* in B (i.e., $values($db:Movie, *dbp:released*, $B) = \{$1931, 1939, 1940, 2011, 2014$\}$ is higher than the average in $values($db:Movie, *dbp:released*, $B') = \{$1931, 1939, 1940$\}$, so we can say that LMDB is biased towards older movies.

2. *Proposed Approach.* Given this formalisation, we proposed a process to identify bias in datasets consisting in five steps, described as follows.

In a first instance, we identify the projection B' of the dataset A. To do that, we retrieve all the entities `entB` in B that have an equivalent entity `entA` in A, using the SPARQL query

```
select ?entA ?entB where { ?entA ?_equivalentProp ?entB }
```

with `?_equivalentProp` being one of *rdfs:seeAlso, owl:sameAs, skos:exactMatch.* Entities in `entB` form the set B'. Next, we build the information related to the entities in B'. More specifically, we retrieve the set of classes $classes(B')$ the entities in B' are instance of and, for each of them, the set of properties $properties(c, B')$ and the set of values $values(c, p, B')$. In order to extract them, we use the SPARQL query

```
select ?c ?p ?val where { ?_ent a ?c. ?_ent ?p ?value. }
```

where `?_ent` is an entity $e_i \in B'$. We filter out classes with a low frequency, by setting a threshold $\delta = 5\%$, i.e. for a class c to be considered, at least 5% of the entities in B' should be of type c.

Using each of the obtained classes (`_c`) and properties (`_p`), we then extract values $values($`_c`$,$`_p`$,\ B)$ in the full dataset B. This is achieved using a third SPARQL query

```
select ?value where { ?ent a ?_c. ?ent ?_p ?value } LIMIT 10000
```

We limit the values to 10,000 elements, considering this amount sufficient to demonstrate the bias. Also, because the distributions of properties having a very large number of values (e.g. inverse-functional properties, titles, IDs, labels, etc.) would not be comparable, we define a second threshold ϕ such that properties whose number of possible values is above ϕ are discarded. ϕ has been set to 90% of the number of entities in B'.

Thus, for each considered property p, we compare the two value sets $values(c, p,\ B)$ and $values(c, p, B')$. Because what we have obtained at this stage is two populations of a same observation (the specific RDF property p), and we aim to prove whether there is a significant difference between their distributions, we use the statistical t-tests to demonstrate (or deny) that "the property p for the class c is biased", i.e. there is a statistically significant difference between *values*(c, p, B') and *values*(c, p, B). In order to reveal towards which values the property p was biased, we extract the mean of the $values(c, p, B)$ and $values(c, p, B')$ if values of p are numerical; otherwise, we calculate which values have the largest absolute differences in the two datasets. Finally, given the set of properties and their estimated bias, we rank them according to their p-value (from 0.01 to 0.1), so to reveal the most biased ones. Properties whose p-value was more 0.1 are discarded and considered non-biased.

3. *Experiments.* The process was trailed on a wide range of Linked Data datasets differing in size and domains from the Data Hub. We invite the reader to refer to Appendix B for a more detailed view of those experiments, e.g. the dataset characteristics, size or time taken to

compute the t-tests and ranking of all the properties (Table B.1), and for some more results (Table B.2). Here, we limit ourselves to show a few examples of biased datasets.

We ordered the examples of Table 8.1 using their “degree of expectability”, i.e. from the ones we mostly expected (marked ①) to the ones we could not imagine (marked ④). In the case ① of the geographical datasets, for instance, results are not especially surprising: the Finnish National library (NLFi) is biased towards places in Finland, because its projection in DBpedia (DBP) reveals that there is a significantly uneven distribution of the value db:CitiesAndTownsInFinland for the property *dc:subject*. Note that this bias is also confirmed by the latitude and longitude average values that we show in Table B.2, which roughly correspond to the coordinates of Finland. Similarly, the New York Times (NYT) is biased towards places and people of the U.S. To reveal that, we used the projection of the NYT dataset in Geonames (GN) and in DBpedia. In the case ② of more general-domain datasets, e.g. the National Library of Spain (NLEs), we were of course expecting some bias toward Spain, but it also turned out that the dataset focuses particularly on artists and writers; similarly, a comparison between the Reading Experience Database ontology and DBpedia revealed that RED describes more scientists from the UK or writers from the 19th century.

Table 8.1 Some bias examples.

$B \rightsquigarrow A$	c	p	value	p-value
①				
DBP⇝NLFi	db:Place	dc:subject	db:CitiesAndTownsInFinland	$p < 1.00 \times 10^{-15}$
GN⇝NYT	gn:Feature	gn:parentFeature	gn:US	$p < 1.00 \times 10^{-15}$
DBP⇝NYT	db:Agent	dbp:country	db:UnitedStates	$p < 3.87 \times 10^{-4}$
②				
DBP⇝NLEs	db:MusicalArtist	dbp:birthPlace	db:Spain	$p < 1.13 \times 10^{-13}$
DBP⇝NLEs	db:Writer	dbp:nationality	db:Spanish	$p < 4.64 \times 10^{-3}$
DBP⇝RED	db:Scientist	db:birthPlace	db:England	$p < 1.00 \times 10^{-15}$
DBP⇝RED	db:Scientist	db:birthPlace	db:Scotland	$p < 1.00 \times 10^{-15}$
DBP⇝RED	db:Writer	db:birthDate	*avg*: 1809, *stdev*: 96.21	$p < 1.00 \times 10^{-15}$
③				
UP⇝Bio2RDF	up:Protein	up:isolatedFrom	uptissue:Brain	$p < 9.29 \times 10^{-4}$
UP⇝BioPAX	up:Protein	up:isolatedFrom	uptissue:Brain	$p < 6.79 \times 10^{-4}$
UP⇝DgB	up:Protein	up:isolatedFrom	uptissue:Brain	$p < 1.33 \times 10^{-4}$
④				
DBP⇝NLEs	db:Writer	db:genre	db:MisteryFiction	$p < 4.28 \times 10^{-11}$
DBP⇝NLEs	db:MusicalArtist	dbp:instrument	db:Piano	$p < 2.73 \times 10^{-4}$
DBP⇝RED	db:Agent	db:genre	db:Novel, db:Poetry	$p < 1.00 \times 10^{-15}$
DBP⇝RED	db:Agent	db:movement	db:Naturalism, db:Romantism	$p < 1.00 \times 10^{-15}$
DBP⇝RED	db:Agent	db:deathCause	db:Suicide	$p < 1.00 \times 10^{-15}$

An interesting aspect is that detecting a bias also helps in understanding datasets one might not be familiar with, such as the datasets in the biomedical domain ③. For example, we discovered that UniProt (UP) is always biased towards cerebral tissues, no matter which projection we used (Bio2RDF, BioPAX and DrugBank – DgB). As for cases ④ of a complete unexpected biases, the BNEs is focused on musical artists that are pianists, and on fictional writers, while the RED dataset is focused on novelists and poets belonging to the 19th century literary movements who, eventually, committed suicide.

Compensating the Bias. Once proven that Linked Data introduce a bias, the future work and challenges consist in identifying strategies to make an application able to compensate the biased results. This might mean, for instance, that F-Measure could be redesigned as a more bias-aware measure, e.g. we could rebalance the distribution of the values V_i for a given path $\overrightarrow{p_i}$ between the items in and outside the pattern, if we encounter a significant difference of the path's frequency.

8.4 Conclusions

The work presented in this thesis is the result of four years of research in which we were put in front of a considerable number of issues to be solved. In an attempt to validate the research hypothesis behind this thesis, i.e. that we could exploit knowledge from the Web of Data to automatically explain patterns, we were brought to analyse problems and propose solutions that were spanning across areas ranging from the more "conceptually-oriented" Cognitive Science, Knowledge Representation and Inference, to the more "data-oriented" disciplines as Artificial Intelligence, Machine Learning and Statistics. This research has demonstrated that the Web of Data can indeed facilitate the deployment of more semantically-aware and human-independent applications, and especially that some of the features in the original vision of the Semantic Web can and should be exploited for this reason. The results of Dedalo, very far from being perfect, suggest however that an approach to blindly discover knowledge can be successfully applied to a wide range of domains, including the ones we have been playing with, and this is especially thanks to the cross-domain knowledge that is connected in the Web of Data. We believe that this is the most important message that our work should transmit.

Afterthoughts

Our achievements were not earned without sweat and tears. Here are some of the lessons that we have learnt while conducting this research.

No matter how much you try, data are biased. The bias in data is natural, and inevitable. The bias comes naturally when using data that somebody else has published, because the likelihood of the problem being seen in the same way is very low. And when accessing knowledge across domains (and datasets), this bias will be even more emphasised. This means that the only solution for Semantic Web research is to be aware of the existence of a bias in data, and to create applications so that they are able to compensate it.

Believe in Link Traversal. The research that works on Linked Data mostly exploits datasets in a standalone way. Few areas make an explicit use of the links between datasets, and even less traverse Linked Data for the purpose of automatically discovering knowledge. We showed how Link Traversal is a technique rather useful for the reasons that it allows to relieve the computational costs, to build domain-agnostic applications and, more importantly, to discover new knowledge serendipitously.

Keep It Simple and Stupid (and Context-aware). There is no need to create complex applications able to deal with any kind of data. During those years of research, we have manipulated and analysed a wide range of (messy) data and domains, and one of the major oversights from us was to look at the problems thinking that we needed a "global solution", something that would always work, while it would have been sufficient to understand the context in which we were moving. Data are nothing but flows of raw information, and context is what makes an application effective.

Linked Data are messy, and you need to deal with that. Considering that the Semantic Web was promoted as a mean to access knowledge more easily than in the Web of Documents, it is easy to naïvely assume that the "only effort" that has to be done is to create our applications. Developing Dedalo as an application based on Linked Data was however a relatively easy task, when compared to the amount of effort we had to put into dealing with Linked Data. In our research we learnt that, while much knowledge is indeed nowadays represented in Linked Data, the costs of cleaning data, re-organising them and dealing with exceptions, dead-ends or inaccessibility are still very high. We believe that much more effort deserves to be put in thinking about why (and how) we are publishing our data, and not on proposing new powerful tools to speed up data consumption.

References

Abedjan, Z. and Naumann, F. (2013). Improving RDF data through Association Rule Mining. *Datenbank-Spektrum*, **13**(2), 111–120.

Abiteboul, S., Quass, D., McHugh, J., Widom, J., and Wiener, J. L. (1997). The LOREL query language for semistructured data. *International journal on digital libraries*, **1**(1), 68–88.

Acosta, M. and Vidal, M.-E. (2015). Networks of Linked Data eddies: An adaptive Web query processing engine for RDF data. In *The Semantic Web–ISWC 2015*, pages 111–127. Springer.

Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., and Ruckhaus, E. (2011). Anapsid: An adaptive query processing engine for SPARQL endpoints. In *The Semantic Web–ISWC 2011*, pages 18–34. Springer.

Aggarwal, C. C. and Wang, H. (2010). Graph data management and mining: A survey of algorithms and applications. In *Managing and Mining Graph Data*, pages 13–68. Springer.

Agrawal, R., Borgida, A., and Jagadish, H. V. (1989). *Efficient management of transitive relationships in large data and knowledge bases*, volume 18. ACM.

Agrawal, R., Srikant, R., *et al.* (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

Akim, N. M., Dix, A., Katifori, A., Lepouras, G., Shabir, N., and Vassilakis, C. (2011). Spreading activation for web scale reasoning: Promise and problems.

Alavi, M. and Leidner, D. E. (1999). Knowledge management systems: Issues, challenges, and benefits. *Communications of the AIS*, **1**(2es), 1.

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, **40**(1), 1.

Anusha, P. and Reddy, G. S. (2012). Interactive postmining of association rules by validating ontologies. *International Journal of Electronics and Computer Science Engineering*.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). *DBpedia: A nucleus for a web of Open Data*. Springer.

Auer, S., Demter, J., Martin, M., and Lehmann, J. (2012). Lodstats – An extensible framework for high-performance dataset analytics. In *Knowledge Engineering and Knowledge Management*, pages 353–362. Springer.

Basse, A., Gandon, F., Mirbel, I., and Lo, M. (2010). DFS-based frequent graph pattern extraction to characterize the content of RDF triple stores. In *Web Science Conference 2010 (WebSci10)*.

Bechtel, W. and Wright, C. (2009). What is psychological explanation. *Routledge companion to the philosophy of psychology*, pages 113–130.

Beek, W., Rietveld, L., Bazoobandi, H. R., Wielemaker, J., and Schlobach, S. (2014). LOD Laundromat: A uniform way of publishing other people's dirty data. In *The Semantic Web–ISWC 2014*, pages 213–228. Springer.

Berners-Lee, T. (1996). WWW: Past, present, and future. *Computer*, **29**(10), 69–77.

Berners-Lee, T. (2006). Linked Data http://www. w3. org/designissues. *LinkedData. html.*

Berners-Lee, T. and Connolly, D. (1995). Hypertext Markup Language-2.0.

Berners-Lee, T., Fielding, R., and Masinter, L. (1998). Uniform Resource Identifier (URI): Generic syntax.

Berners-Lee, T., Hendler, J., Lassila, O., *et al.* (2001). The Semantic Web. *Scientific american*, **284**(5), 28–37.

Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., and Sudarshan, S. (2002). Keyword searching and browsing in databases using banks. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 431–440. IEEE.

Bicer, V., Tran, T., and Gossen, A. (2011). Relational kernel machines for learning from graph-structured RDF data. In *The Semantic Web: Research and Applications*, pages 47–62. Springer.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009a). DBpedia – A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web*, **7**(3), 154–165.

Bizer, C., Heath, T., and Berners-Lee, T. (2009b). Linked Data – The story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, **5**(3), 1–22.

Bloehdorn, S. and Sure, Y. (2007). *Kernel methods for mining instance data in ontologies*. Springer.

Bloehdorn, S., Haase, P., Sure, Y., and Voelker, J. (2006). Ontology Evolution. *Semantic web technologies: trends and research in ontology-based systems*, pages 51–70.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Bollegala, D., Matsuo, Y., and Ishizuka, M. (2007). Measuring semantic similarity between words using Web search engines. *www*, **7**, 757–766.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph theory with applications*, volume 290. Macmillan London.

Bouquet, P., Ghidini, C., and Serafini, L. (2009). Querying the Web of Data: A formal approach. In *The Semantic Web*, pages 291–305. Springer.

Brent, E., Thompson, A., and Vale, W. (2000). Sociology a computational approach to sociological explanations. *Social Science Computer Review*, **18**(2), 223–235.

Bringmann, B. and Nijssen, S. (2008). What is frequent in a single graph? In *Advances in Knowledge Discovery and Data Mining*, pages 858–863. Springer.

Brisson, L. and Collard, M. (2008). How to semantically enhance a Data Mining process?. In *ICEIS*, volume 19, pages 103–116. Springer.

Brisson, L., Collard, M., and Pasquier, N. (2005). Improving the Knowledge Discovery process using ontologies. In *IEEE MCD'2005 international workshop on Mining Complex Data*, pages 25–32.

Buil-Aranda, C., Arenas, M., and Corcho, O. (2011). Semantics and optimization of the SPARQL 1.1 federation extension. In *The Semantic Web: Research and Applications*, pages 1–15. Springer.

Buil-Aranda, C., Hogan, A., Umbrich, J., and Vandenbussche, P.-Y. (2013). SPARQL Web-querying infrastructure: Ready for action? In *The Semantic Web–ISWC 2013*, pages 277–293. Springer.

Buil-Aranda, C., Polleres, A., and Umbrich, J. (2014). Strategies for executing federated queries in SPARQL1. 1. In *The Semantic Web–ISWC 2014*, pages 390–405. Springer.

Buneman, P., Fernandez, M., and Suciu, D. (2000). UnQL: A query language and algebra for semistructured data based on structural recursion. *The VLDB Journal – The International Journal on Very Large Data Bases*, **9**(1), 76–110.

Calhoun, C. (2002). *Dictionary of the social sciences*. Oxford University Press on Demand.

Callahan, A., Cruz-Toledo, J., Ansell, P., and Dumontier, M. (2013). Bio2RDF release 2: Improved coverage, interoperability and provenance of life science Linked Data. In *The semantic web: semantics and big data*, pages 200–212. Springer.

Campbell, J. (2008). Causation in psychiatry. *Philosophical issues in psychiatry*, pages 196–215.

Carter, J. R. (1998). Description is not explanation: A methodology of comparison. *Method & Theory in the Study of Religion*, **10**(2), 133–148.

Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent systems*, (1), 20–26.

Chartrand, G. (2006). *Introduction to graph theory*. Tata McGraw-Hill Education.

Chen, C., Lin, C. X., Fredrikson, M., Christodorescu, M., Yan, X., and Han, J. (2009). Mining graph patterns efficiently via randomized summaries. *Proceedings of the VLDB Endowment*, **2**(1), 742–753.

Chen, H.-H., Lin, M.-S., and Wei, Y.-C. (2006). Novel association measures using Web search with double checking. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1009–1016. Association for Computational Linguistics.

Cheng, W., Kasneci, G., Graepel, T., Stern, D., and Herbrich, R. (2011). Automated feature generation from structured knowledge. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1395–1404. ACM.

Cilibrasi, R. L. and Vitanyi, P. (2007). The Google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on*, **19**(3), 370–383.

Clayton, P. (1989). Explanation from physics to the philosophy of religion. *International journal for philosophy of religion*, **26**(2), 89–108.

Cohen, E., Halperin, E., Kaplan, H., and Zwick, U. (2003). Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, **32**(5), 1338–1355.

Conklin, D. and Glasgow, J. (2014). Spatial analogy and subsumption. In *Machine Learning: Proceedings of the Ninth International Conference ML (92)*, pages 111–116.

Conte, D., Foggia, P., Sansone, C., and Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, **18**(03), 265–298.

Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. (2004). A (sub)graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **26**(10), 1367–1372.

Correndo, G., Salvadores, M., Millard, I., Glaser, H., and Shadbolt, N. (2010). SPARQL query rewriting for implementing data integration over Linked Data. In *Proceedings of the 2010 EDBT/ICDT Workshops*, page 4. ACM.

Crestani, F. (1997). Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, **11**(6), 453–482.

Culicover, P. W. and Jackendoff, R. (2005). *Simpler syntax*. Oxford University Press Oxford.

Cummins, R. (2000). "How does it work?" versus "What are the laws?": Two conceptions of psychological explanation. *Explanation and cognition*, pages 117–144.

Daga, E., d'Aquin, M., Gangemi, A., and Motta, E. (2014). From datasets to datanodes: An ontology pattern for networks of data artifacts. *Semantic Web Journal (submitted), http://semantic-web-journal. net/content/datasets-datanodes-ontology-pattern-networks-data-artifacts*.

Daga, E., Panziera, L., and Pedrinaci, C. (2015). A BASILar approach for building Web APIs on top of SPARQL endpoints. In *CEUR Workshop Proceedings*, volume 1359, pages 22–32.

d'Amato, C., Fanizzi, N., and Esposito, F. (2006). A dissimilarity measure for $\mathcal{ALC}$ concept descriptions. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1695–1699. ACM.

d'Amato, C., Staab, S., and Fanizzi, N. (2008). On the influence of Description Logics ontologies on conceptual similarity. In *Knowledge Engineering: Practice and Patterns*, pages 48–63. Springer.

d'Amato, C., Bryl, V., and Serafini, L. (2012). Data-driven logical reasoning. In *URSW*, pages 51–62. Citeseer.

Dantzig, G. B. (1998). *Linear Programming and extensions*. Princeton university press.

d'Aquin, M. and Jay, N. (2013). Interpreting Data Mining results with Linked Data for Learning Analytics: Motivation, case study and directions. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, pages 155–164. ACM.

d'Aquin, M., Kronberger, G., and del Carmen Suárez-Figueroa, M. (2012). Combining Data Mining and Ontology Engineering to enrich ontologies and Linked Data. In *KNOW@ LOD*, pages 19–24.

Darden, L. and Maull, N. (1977). Interfield theories. *Philosophy of Science*, pages 43–64.

Davidson, D. (1990). The structure and content of truth. *The journal of philosophy*, pages 279–328.

De Raedt, L. and Kramer, S. (2001). The levelwise version space algorithm and its application to molecular fragment finding. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 853–862. Citeseer.

De Vocht, L., Coppens, S., Verborgh, R., Vander Sande, M., Mannens, E., and Van de Walle, R. (2013). Discovering meaningful connections between resources in the web of data. In *LDOW*.

Dehaspe, L. and Toivonen, H. (1999). Discovery of frequent Datalog patterns. *Data Mining and Knowledge Discovery*, **3**(1), 7–36.

Dehmer, M. and Mowshowitz, A. (2011). Generalized graph entropies. *Complexity*, **17**(2), 45–50.

Denning, P. J. (1981). ACM president's letter – Performance analysis: Experimental computer science as its best. *Communications of the ACM*, **24**(11), 725–727.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, **1**(1), 269–271.

Dividino, R., Gottron, T., Scherp, A., and Gröner, G. (2014). From changes to dynamics: Dynamics analysis of Linked Open Data sources. In *PROFILES'14: Proceedings of the Workshop on Dataset ProfiIling and Federated Search for Linked Data*.

Dividino, R., Gottron, T., and Scherp, A. (2015). Strategies for efficiently keeping local Linked Open Data caches up-to-date. In *International Semantic Web Conference*. Springer.

Dividino, R. Q., Scherp, A., Gröner, G., and Grotton, T. (2013). Change-a-LOD: Does the schema on the Linked Data Cloud change or not? In *COLD*.

Dominguez-Sal, D., Urbón-Bayes, P., Giménez-Vañó, A., Gómez-Villamor, S., Martínez-Bazan, N., and Larriba-Pey, J.-L. (2010). Survey of graph database performance on the HPC scalable graph analysis benchmark. In *Web-Age Information Management*, pages 37–48. Springer.

Doran, J. E. and Michie, D. (1966). Experiments with the graph traverser program. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 294, pages 235–259. The Royal Society.

Durkheim, E. (2014). *The rules of sociological method, and selected texts on sociology and its method*. Simon and Schuster.

Eliassi-Rad, T. and Chow, E. (2005). Using ontological information to accelerate path-finding in large semantic graphs: A probabilistic approach. In *American Association for Artificial Intelligence*.

Elio, R., Hoover, J., Nikolaidis, I., Salavatipour, M., Stewart, L., and Wong, K. (2011). About computing science research methodology.

Erling, O. and Mikhailov, I. (2010). *Virtuoso: RDF support in a native RDBMS*. Springer.

Escovar, E. L., Yaguinuma, C. A., and Biajiz, M. (2006). Using fuzzy ontologies to extend semantically similar Data Mining. In *SBBD*, pages 16–30.

Euzenat, J., Shvaiko, P., *et al.* (2007). *Ontology Matching*, volume 333. Springer.

Fanizzi, N. and d'Amato, C. (2006). A declarative kernel for $\mathcal{ALC}$ concept descriptions. In *Foundations of Intelligent Systems*, pages 322–331. Springer.

Fanizzi, N., d'Amato, C., and Esposito, F. (2008). *Statistical learning for inductive query answering on OWL ontologies*. Springer.

Fanizzi, N., d'Amato, C., and Esposito, F. (2012). Mining Linked Open Data through semi-supervised learning methods based on self-training. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 277–284. IEEE.

Fann, K. T. (1970). *Peirce's theory of abduction*. Springer Science & Business Media.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI magazine*, **17**(3), 37.

Feigenbaum, L., Williams, G. T., Clark, K. G., and Torres, E. (2013). SPARQL 1.1 protocol. *Recommendation, W3C, March*.

Felner, A. (2011). Position paper: Dijkstra's algorithm versus Uniform Cost Search or a case against Dijkstra's algorithm. In *Fourth Annual Symposium on Combinatorial Search*.

Fiedler, M. and Borgelt, C. (2007). Support computation for mining frequent subgraphs in a single graph. In *MLG*. Citeseer.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol–HTTP/1.1.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Ph.D. thesis, University of California, Irvine.

Fionda, V., Gutierrez, C., and Pirró, G. (2012). Semantic navigation on the Web of Data: Specification of routes, Web fragments and actions. In *Proceedings of the 21st international conference on World Wide Web*, pages 281–290. ACM.

Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, **1**(4), 385–408.

Florescu, D., Levy, A. Y., and Mendelzon, A. O. (1998). Database techniques for the World-Wide Web: A survey. *SIGMOD record*, **27**(3), 59–74.

Freitas, A., Oliveira, J. G., O'Riain, S., Curry, E., and Da Silva, J. C. P. (2011a). Querying linked data using semantic relatedness: a vocabulary independent approach. In *International Conference on Application of Natural Language to Information Systems*, pages 40–51. Springer.

Freitas, A., Oliveira, J. G., Curry, E., O'Riain, S., and da Silva, J. C. P. (2011b). Treo: combining entity-search, spreading activation and semantic relatedness for querying linked data. In *Proc. of 1st Workshop on Question Answering over Linked Data (QALD-1) at the 8th Extended Semantic Web Conference (ESWC 2011).*

Friedenberg, J. and Silverman, G. (2011). *Cognitive Science: An introduction to the study of mind*. Sage.

Friedman, M. (1974). Explanation and scientific understanding. *The Journal of Philosophy*, pages 5–19.

Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based Explicit Semantic Analysis. In *IJCAI*, volume 7, pages 1606–1611.

Gangemi, A. and Mika, P. (2003). Understanding the Semantic Web through descriptions and situations. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 689–706. Springer.

Gangemi, A. and Presutti, V. (2009). Ontology Design Patterns. In *Handbook on ontologies*, pages 221–243. Springer.

Gardner, H. (2008). *The mind's new science: A history of the cognitive revolution*. Basic books.

Garriga, G. C., Ukkonen, A., and Mannila, H. (2008). Feature selection in taxonomies with applications to paleontology. In *Discovery Science*, pages 112–123. Springer.

Gärtner, T. (2003). A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, **5**(1), 49–58.

Ge, W., Chen, J., Hu, W., and Qu, Y. (2010). Object link structure in the Semantic Web. In *The Semantic Web: Research and Applications*, pages 257–271. Springer.

Geng, L. and Hamilton, H. J. (2006). Interestingness measures for Data Mining: A survey. *ACM Computing Surveys (CSUR)*, **38**(3), 9.

Getoor, L. (2007). *Introduction to Statistical Relational Learning*. MIT press.

Getoor, L. and Diehl, C. P. (2005). Link Mining: A survey. *ACM SIGKDD Explorations Newsletter*, **7**(2), 3–12.

Giblin, P. (2013). *Graphs, surfaces and homology: An introduction to algebraic topology*. Springer Science & Business Media.

Gil, R., García, R., and Delgado, J. (2004). Measuring the Semantic Web. *AIS SIGSEMIS Bulletin*, **1**(2), 69–72.

Görlitz, O. and Staab, S. (2011). Splendid: SPARQL endpoint federation exploiting void descriptions. *COLD*, **782**.

Gorski, P. S. (2004). The poverty of deductivism: A constructive realist model of sociological explanation. *Sociological Methodology*, **34**(1), 1–33.

Gottron, T. and Gottron, C. (2014). Perplexity of index models over evolving Linked Data. In *The Semantic Web: Trends and Challenges*, pages 161–175. Springer.

Grimnes, G. A., Edwards, P., and Preece, A. (2008). *Instance-based clustering of Semantic Web resources*. Springer.

Grobe, M. (2009). RDF, Jena, SPARQL and the Semantic Web. In *Proceedings of the 37th annual ACM SIGUCCS fall conference: communication and collaboration*, pages 131–138. ACM.

Grobelnik, M. and Mladenić, D. (2006). *Knowledge Discovery for ontology construction*. Wiley Online Library.

Groth, P., Loizou, A., Gray, A. J., Goble, C., Harland, L., and Pettifer, S. (2014). API-centric Linked Data integration: The open PHACTS discovery platform case study. *Web Semantics: Science, Services and Agents on the World Wide Web*, **29**, 12–18.

Guéret, C., Groth, P., Van Harmelen, F., and Schlobach, S. (2010). Finding the Achilles heel of the Web of Data: Using network analysis for link-recommendation. *The Semantic Web–ISWC 2010*, pages 289–304.

Gutierrez, C. (2011). Modeling the Web of Data (introductory overview). In *Reasoning Web. Semantic Technologies for the Web of Data*, pages 416–444. Springer.

Harris, S., Seaborne, A., and Prud'hommeaux, E. (2013). SPARQL 1.1 Query Language. *W3C Recommendation*, **21**.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, **4**(2), 100–107.

Harth, A. and Speiser, S. (2012). On completeness classes for query evaluation on Linked Data. In *AAAI*. Citeseer.

Hartig, O. (2011). Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In *The Semantic Web: Research and Applications*, pages 154–169. Springer.

Hartig, O. (2012). SPARQL for a Web of Linked Data: Semantics and computability. In *The Semantic Web: Research and Applications*, pages 8–23. Springer.

Hartig, O. (2013). An overview on execution strategies for Linked Data queries. *Datenbank-Spektrum*, **13**(2), 89–99.

Hartig, O. and Pirrò, G. (2015). A context-based semantics for SPARQL property paths over the Web. In *The Semantic Web. Latest Advances and New Domains*, pages 71–87. Springer.

Haslhofer, B. and Isaac, A. (2011). data.europeana.eu: The Europeana Linked Open Data pilot. In *International Conference on Dublin Core and Metadata Applications*, pages 94–104.

Haspelmath, M., Bibiko, H.-J., Hagen, J., and Schmidt, C. (2005). *The world atlas of language structures*, volume 1. Oxford University Press Oxford.

Hausenblas, M., Halb, W., Raimond, Y., and Heath, T. (2008). What is the size of the Semantic Web. *Proceedings of I-Semantics*, pages 9–16.

Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L., and Ayers, D. (2009). Scovo: Using statistics on the Web of Data. In *The Semantic Web: Research and Applications*, pages 708–722. Springer.

Hawkins, J. A. (1988). *Explaining language universals*. Blackwell.

He, H., Wang, H., Yang, J., and Yu, P. S. (2007). BLINKS: Ranked keyword searches on graphs. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 305–316. ACM.

Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., and Stegemann, T. (2009). Relfinder: Revealing relationships in rdf knowledge bases. In *International Conference on Semantic and Digital Media Technologies*, pages 182–187. Springer.

Hempel, C. G. and Oppenheim, P. (1948). Studies in the logic of explanation. *Philosophy of science*, pages 135–175.

Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., and Decker, S. (2012). An empirical survey of Linked Data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, **14**, 14–44.

Holder, L. B., Cook, D. J., Djoko, S., *et al.* (1994). Substucture discovery in the SUBDUE system. In *KDD workshop*, pages 169–180.

Hoser, B., Hotho, A., Jäschke, R., Schmitz, C., and Stumme, G. (2006). *Semantic network analysis of ontologies*. Springer.

Huan, J., Wang, W., and Prins, J. (2003). Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552. IEEE.

Hutten, E. H. (1956). On explanation in psychology and in physics. *The British Journal for the Philosophy of Science*, **7**(25), 73–85.

Ibragimov, D., Hose, K., Pedersen, T. B., and Zimányi, E. (2015). Processing aggregate queries in a federation of SPARQL endpoints. In *The Semantic Web. Latest Advances and New Domains*, pages 269–285. Springer.

Imielinski, T. and Mannila, H. (1996). A database perspective on Knowledge Discovery. *Communications of the ACM*, **39**(11), 58–64.

Inokuchi, A., Washio, T., and Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer.

Inokuchi, A., Washio, T., and Motoda, H. (2003). Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, **50**(3), 321–354.

Iordanov, B. (2010). Hypergraphdb: A generalized graph database. In *Web-Age information management*, pages 25–36. Springer.

Itkonen, E. (2013). On explanation in linguistics. In *article in the forum discussion of this issue: http://www. kabatek. de/energeia/E5discussion*.

Jagadish, H. (1990). A compression technique to materialize transitive closure. *ACM Transactions on Database Systems (TODS)*, **15**(4), 558–598.

Janowicz, K. (2006). SIM-DL: Towards a semantic similarity measurement theory for the Description Logic $\mathcal{ALCNR}$ in geographic Information Retrieval. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 1681–1692. Springer.

Janowicz, K., Keßler, C., Schwarz, M., Wilkes, M., Panov, I., Espeter, M., and Bäumer, B. (2007). Algorithm, implementation and application of the SIM-DL similarity server. In *GeoSpatial Semantics*, pages 128–145. Springer.

Jones, J., Kuhn, W., Keßler, C., and Scheider, S. (2014). Making the Web of Data available via Web feature services. In *Connecting a Digital Europe Through Location and Place*, pages 341–361. Springer.

Jonyer, I., Holder, L., and Cook, D. (2002). Concept formation using graph grammars. In *Proceedings of the KDD Workshop on Multi-Relational Data Mining*, volume 2, pages 19–43.

Joslyn, C., Adolf, B., al Saffar, S., Feo, J., Goodman, E., Haglin, D., Mackey, G., and Mizell, D. (2010). High performance semantic factoring of Giga-scale semantic graph databases. *Contribution to Semantic Web Challenge at ISWC*.

Józefowska, J., Ławrynowicz, A., and Łukaszewski, T. (2010). The role of semantics in mining frequent patterns from knowledge bases in Description Logics with rules. *Theory and Practice of Logic Programming*, **10**(03), 251–289.

Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., and Karambelkar, H. (2005). Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st international conference on Very large data bases*, pages 505–516. VLDB Endowment.

Käfer, T., Umbrich, J., Hogan, A., and Polleres, A. (2012). Towards a dynamic Linked Data observatory. *LDOW at WWW*.

Käfer, T., Abdelrahman, A., Umbrich, J., O'Byrne, P., and Hogan, A. (2013). Observing Linked Data dynamics. In *The Semantic Web: Semantics and Big Data*, pages 213–227. Springer.

Kashima, H. and Inokuchi, A. (2002). Kernels for graph classification. In *ICDM Workshop on Active Mining*, volume 2002. Citeseer.

Kasneci, G., Suchanek, F. M., Ifrim, G., Ramanath, M., and Weikum, G. (2008). Naga: Searching and ranking knowledge. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 953–962. IEEE.

Ketkar, N. S., Holder, L. B., and Cook, D. J. (2005). SUBDUE: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st international workshop on open source Data Mining: frequent pattern mining implementations*, pages 71–76. ACM.

Khan, M. A., Grimnes, G. A., and Dengel, A. (2010). Two pre-processing operators for improved learning from Semantic Web data. In *First RapidMiner Community Meeting And Conference (RCOMM 2010)*, volume 20.

Kilssen, N. J. (1971). Problem-solving methods in Artificial Intelligence.

Kitcher, P. (1981). Explanatory unification. *Philosophy of Science*, pages 507–531.

Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and abstract syntax. W3C Recommendation, 2004. *World Wide Web Consortium, http://w3c.org/TR/rdf-concepts*.

Kondor, R. I. and Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, pages 315–322.

Krompaß, D., Baier, S., and Tresp, V. (2015). Type-constrained representation learning in knowledge graphs. In *The Semantic Web–ISWC 2015*, pages 640–655. Springer.

Kuramochi, M. and Karypis, G. (2001). Frequent Subgraph Discovery. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 313–320. IEEE.

Kuramochi, M. and Karypis, G. (2005). Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, **11**(3), 243–271.

Larrosa, J. and Valiente, G. (2002). Constraint satisfaction algorithms for graph pattern matching. *Mathematical structures in computer science*, **12**(04), 403–422.

Lavrac, N. and Dzeroski, S. (1994). Inductive Logic Programming. In *WLP*, pages 146–160. Springer.

Lavrač, N., Flach, P., and Zupan, B. (1999). *Rule evaluation measures: A unifying view*. Springer.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2013). Prov-o: The prov ontology. *W3C Recommendation*, **30**.

Levinson, R. A. (1985). *A self-organizing retrieval system for graphs*. Ph.D. thesis, University of Texas at Austin.

Lisi, F. A. and Esposito, F. (2009). On ontologies as prior conceptual knowledge in Inductive Logic Programming. In *Knowledge discovery enhanced with semantic and social information*, pages 3–17. Springer.

Liu, J., Wang, W., and Yang, J. (2004). A framework for ontology-driven subspace clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 623–628. ACM.

Lloyd, J. W. (2012). *Foundations of Logic Programming*. Springer Science & Business Media.

Lösch, U., Bloehdorn, S., and Rettinger, A. (2012). Graph kernels for RDF data. In *The Semantic Web: Research and Applications*, pages 134–148. Springer.

Maali, F., Erickson, J., and Archer, P. (2014). Data catalog vocabulary (DCAT). *W3C Recommendation*.

Maaløe, E. (2007). *Modes of Explanation: From Rules to Emergence*. Handelshøjskolen, Aarhus Universitet, Institut for Ledelse.

Maduko, A., Anyanwu, K., Sheth, A., and Schliekelman, P. (2008). Graph summaries for subgraph frequency estimation. *The Semantic Web: Research and Applications*, pages 508–523.

Maedche, A. and Staab, S. (2004). Ontology Learning. In *Handbook on ontologies*, pages 173–190. Springer.

Marie, N., Corby, O., Gandon, F., and Ribière, M. (2013a). Composite interests' exploration thanks to on-the-fly linked data spreading activation. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 31–40. ACM.

Marie, N., Gandon, F., Ribière, M., and Rodio, F. (2013b). Discovery hub: on-the-fly linked data exploratory search. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 17–24. ACM.

Marinica, C. and Guillet, F. (2010). Knowledge-based interactive postmining of association rules using ontologies. *Knowledge and Data Engineering, IEEE Transactions on*, **22**(6), 784–797.

Marr, D. and Vision, A. (1982). A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*.

Marraffa, M. and Paternoster, A. (2012). Functions, levels, and mechanisms: Explanation in Cognitive Science and its problems. *Theory & Psychology*, page 0959354312451958.

Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., Nin, J., Sánchez-Martínez, M.-A., and Larriba-Pey, J.-L. (2007). Dex: High-performance exploration on large graphs for Information Retrieval. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 573–582. ACM.

McKay, B. D. (2007). Nauty user's guide (version 2.4). *Computer Science Dept., Australian National University*, pages 225–239.

Mencıa, E. L., Holthausen, S., Schulz, A., and Janssen, F. (2013). Using Data Mining on Linked Open Data for analyzing e-procurement information. In *Proceedings of the International Workshop on Data Mining on Linked Data, with Linked Data Mining Challenge collocated with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECMLPKDD*. Citeseer.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine Learning: An Artificial Intelligence approach*. Springer Science & Business Media.

Miller, W. and Myers, E. W. (1985). A file comparison program. *Software: Practice and Experience*, **15**(11), 1025–1040.

Mitchell, T. M. (1982). Generalization as search. *Artificial intelligence*, **18**(2), 203–226.

Montoya, G., Skaf-Molli, H., Molli, P., and Vidal, M.-E. (2015). Federated SPARQL queries processing with replicated fragments. In *The Semantic Web–ISWC 2015*, pages 36–51. Springer.

Moore, J. L., Steinke, F., and Tresp, V. (2011). A novel metric for information retrieval in semantic networks. In *Extended Semantic Web Conference*, pages 65–79. Springer.

Moore, J. L., Steinke, F., and Tresp, V. (2012). A novel metric for Information Retrieval in semantic networks. In *The Semantic Web: ESWC 2011 Workshops*, pages 65–79. Springer.

Mowshowitz, A. and Dehmer, M. (2012). Entropy and the complexity of graphs revisited. *Entropy*, **14**(3), 559–570.

Muggleton, S. and De Raedt, L. (1994). Inductive Logic Programming: Theory and methods. *The Journal of Logic Programming*, **19**, 629–679.

Muggleton, S., Otero, R., and Tamaddoni-Nezhad, A. (1992). *Inductive Logic Programming*, volume 168. Springer.

Mulwad, V., Finin, T., Syed, Z., and Joshi, A. (2010). Using Linked Data to interpret tables. *COLD*, **665**.

Myers, E. W. (1986). An O(ND) difference algorithm and its variations. *Algorithmica*, **1**(1-4), 251–266.

Nandhini, M., Janani, M., and Sivanandham, S. (2012). Association Rule Mining using swarm intelligence and domain ontology. In *Recent Trends In Information Technology (ICRTIT), 2012 International Conference on*, pages 537–541. IEEE.

Nebot, V. and Berlanga, R. (2012). Finding association rules in Semantic Web data. *Knowledge-Based Systems*, **25**(1), 51–62.

Newell, A., Shaw, J. C., and Simon, H. A. (1959). Report on a general problem-solving program. In *IFIP Congress*, pages 256–264.

Nijssen, S. and Kok, J. (2001). Faster association rules for multiple relations. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 891–896. Citeseer.

Nijssen, S. and Kok, J. N. (2004). A quick start in frequent structure mining can make a difference. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 647–652. ACM.

Nishioka, C. and Scherp, A. (2015). Temporal patterns and periodicity of entity dynamics in the Linked Open Data Cloud. In *Proceedings of the 8th International Conference on Knowledge Capture*, K-CAP 2015, pages 22:1–22:4, New York, NY, USA. ACM.

Novak, P. K., Vavpetic, A., Trajkovski, I., and Lavrac, N. (2009). Towards semantic Data Mining with G-SEGS. In *Proceedings of the 11th International Multiconference Information Society (IS 2009)*, volume 20.

Ogbuji, C. (2011). SPARQL 1.1 graph store HTTP protocol. *W3C Recommendation (March 21, 2013).*

Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., and Tummarello, G. (2008). Sindice.com: A document-oriented lookup index for Open Linked Data. *International Journal of Metadata, Semantics and Ontologies*, **3**(1), 37–52.

Özsu, M. T. and Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.

Páez, A. (2009). Artificial explanations: The epistemological interpretation of explanation in AI. *Synthese*, **170**(1), 131–146.

Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., *et al.* (2013). Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*, **45**(4), 42.

Paulheim, H. (2012). Generating possible interpretations for statistics from Linked Open Data. In *The Semantic Web: Research and Applications*, pages 560–574. Springer.

Paulheim, H. and Fümkranz, J. (2012). Unsupervised generation of Data Mining features from Linked Open Data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 31. ACM.

Paulheim, H. and Gangemi, A. (2015). Serving DBpedia with DOLCE – More than just adding a cherry on top. In *The Semantic Web–ISWC 2015*, pages 180–196. Springer.

Pedrinaci, C. and Domingue, J. (2010). Toward the next wave of services: Linked services for the Web of Data. *J. ucs*, **16**(13), 1694–1719.

Pirrò, G. (2015). Explaining and suggesting relatedness in knowledge graphs. In *International Semantic Web Conference*, pages 622–639. Springer.

Pohl, I. (1970). *Bi-directional search.* IBM TJ Watson Research Center.

Pohle, C. (2003). Integrating and updating domain knowledge with Data Mining. In *VLDB PhD Workshop*.

Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. (2008). *A field guide to Genetic Programming*. Lulu. com.

Psillos, S. (2007). Past and contemporary perspectives on explanation. *General philosophy of science. Focal issues*, pages 97–173.

Quilitz, B. and Leser, U. (2008). *Querying distributed RDF data sources with SPARQL.* Springer.

Quinlan, J. R. and Hunt, E. (1968). A formal deductive problem-solving system. *Journal of the ACM (JACM)*, **15**(4), 625–646.

Ramesh, C. R., Ramana, K., Rao, K. R., and Sastry, C. (2013). Interactive post mining association rules using cost complexity pruning and ontologies KDD. *International Journal of Computer Applications*, **68**(20).

Rettinger, A., Nickles, M., and Tresp, V. (2009). Statistical Relational Learning with formal ontologies. In *Machine Learning and Knowledge Discovery in Databases*, pages 286–301. Springer.

Rettinger, A., Lösch, U., Tresp, V., d'Amato, C., and Fanizzi, N. (2012). Mining the Semantic Web. *Data Mining and Knowledge Discovery*, **24**(3), 613–662.

Ristoski, P. (2015). Towards Linked Open Data enabled Data Mining. In *The Semantic Web. Latest Advances and New Domains*, pages 772–782. Springer.

Ristoski, P. and Paulheim, H. (2014). A comparison of propositionalization strategies for creating features from Linked Open Data. *Linked Data for Knowledge Discovery*, page 6.

Rothleder, N. J. (1996). Induction: Inference and process.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, **20**, 53–65.

Ruben, D.-H. (1990). *Explaining explanation*. Cambridge Univ Press.

Rumelhart, D. E. and Norman, D. A. (1983). Representation in memory. Technical report, DTIC Document.

Russ, T. A., Ramakrishnan, C., Hovy, E. H., Bota, M., and Burns, G. A. (2011). Knowledge engineering tools for reasoning with scientific observations and interpretations: A neural connectivity use case. *BMC bioinformatics*, **12**(1), 351.

Russell, S., Norvig, P., and Intelligence, A. (1995). Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, **25**.

Sahami, M. and Heilman, T. D. (2006). A Web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, pages 377–386. AcM.

Salmon, W. C. (1990). Scientific explanation: Causation and unification. *Critica: Revista Hispanoamericana de Filosofia*, pages 3–23.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, **1**(1), 27–64.

Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of the linked data best practices in different topical domains. In *International Semantic Web Conference*, pages 245–260. Springer.

Schuhmacher, M. and Ponzetto, S. P. (2014). Knowledge-based graph document modeling. In *Proceedings of the 7th ACM international conference on Web Search and Data Mining*, pages 543–552. ACM.

Schurz, G. (2008). Patterns of abduction. *Synthese*, **164**(2), 201–234.

Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M. (2011). Fedx: Optimization techniques for federated query processing on Linked Data. In *The Semantic Web–ISWC 2011*, pages 601–616. Springer.

Segen, J. (2014). Graph clustering and model learning by data compression. In *Proceedings of the Machine Learning Conference*, page 93.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, **5**(1), 3–55.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and brain sciences*, **11**(01), 1–23.

Southwick, R. W. (1991). Explaining reasoning: An overview of explanation in knowledge-based systems. *The knowledge engineering review*, **6**(01), 1–19.

Speicher, S., Arwe, J., and Malhotra, A. (2014). Linked Data Platform 1.0. *Proposed Recommendation, W3C*.

Srikant, R. and Agrawal, R. (1995). *Mining generalized association rules*. IBM Research Division.

Stankovic, M., Wagner, C., Jovanovic, J., and Laublet, P. (2010). Looking for experts? What can Linked Data do for you? In *LDOW*.

Strevens, M. (2006). Scientific explanation. *Encyclopedia of Philosophy, second edition. DM Borchert (ed.). Detroit: Macmillan Reference USA*.

Strube, M. and Ponzetto, S. P. (2006). WikiRelate! Computing semantic relatedness using Wikipedia. In *AAAI*, volume 6, pages 1419–1424.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). YAGO: A core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Theoharis, Y., Tzitzikas, Y., Kotzinos, D., and Christophides, V. (2008). On graph features of Semantic Web schemas. *Knowledge and Data Engineering, IEEE Transactions on*, **20**(5), 692–702.

Thompson, K. and Langley, P. (1991). Concept formation in structured domains. *Concept formation: Knowledge and experience in unsupervised learning*, pages 127–161.

Thor, A., Anderson, P., Raschid, L., Navlakha, S., Saha, B., Khuller, S., and Zhang, X.-N. (2011). Link prediction for annotation graphs using graph summarization. In *The Semantic Web–ISWC 2011*, pages 714–729. Springer.

Tiddi, I., d'Aquin, M., and Motta, E. (2014a). Quantifying the bias in data links. In *Knowledge Engineering and Knowledge Management*, pages 531–546. Springer.

Tiddi, I., d'Aquin, M., and Motta, E. (2014b). Walking Linked Data: A graph traversal approach to explain clusters. In *Proceedings of the 5th International Conference on Consuming Linked Data-Volume 1264*, pages 73–84. CEUR-WS. org.

Trajkovski, I., Lavrač, N., and Tolar, J. (2008). SEGS: Search for enriched gene sets in microarray data. *Journal of biomedical informatics*, **41**(4), 588–601.

Trißl, S. and Leser, U. (2007). Fast and practical indexing and querying of very large graphs. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 845–856. ACM.

Tseng, M.-C., Lin, W.-Y., and Jeng, R. (2005). Maintenance of generalized association rules under transaction update and taxonomy evolution. In *Data Warehousing and Knowledge Discovery*, pages 336–345. Springer.

Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., and Decker, S. (2010). Sig.ma: Live views on the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, **8**(4), 355–364.

Ukkonen, E. (1985). Algorithms for approximate string matching. *Information and control*, **64**(1), 100–118.

Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, **23**(1), 31–42.

Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., and Decker, S. (2010). Towards dataset dynamics: Change frequency of Linked Open Data sources.

Umbrich, J., Hogan, A., Polleres, A., and Decker, S. (2014). Link traversal querying for a diverse Web of Data. *Semantic Web Journal*.

Van Herwegen, J., Verborgh, R., Mannens, E., and Van de Walle, R. (2015). Query execution optimization for clients of triple pattern fragments. In *The Semantic Web. Latest Advances and New Domains*, pages 302–318. Springer.

Vandenbussche, P.-Y. and Vatant, B. (2011). Metadata recommendations for Linked Open Data vocabularies. *Version*, **1**, 2011–12.

Vandenbussche, P.-Y., Umbrich, J., Hogan, A., and Buil-Aranda, C. (2015). SPARQLES: Monitoring public SPARQL endpoints. *Semantic Web Journal*.

Vanetik, N., Gudes, E., and Shimony, S. E. (2002). Computing frequent graph patterns from semistructured data. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 458–465. IEEE.

Verborgh, R., Hartig, O., De Meester, B., Haesendonck, G., De Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., and Van de Walle, R. (2014a). Querying datasets on the Web with high availability. In *The Semantic Web–ISWC 2014*, pages 180–196. Springer.

Verborgh, R., Vander Sande, M., Colpaert, P., Coppens, S., Mannens, E., and Van de Walle, R. (2014b). Web-scale querying through Linked Data fragments. In *Proceedings of the 7th Workshop on Linked Data on the Web*.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: A free collaborative knowledge base. *Communications of the ACM*, **57**(10), 78–85.

Wang, C., Chen, H., Gu, P., and Yu, T. (2011). Relation discovery in medicine through semantic graph routing. In *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, volume 2, pages 640–644. IEEE.

Wang, H., He, H., Yang, J., Yu, P. S., and Yu, J. X. (2006). Dual labeling: Answering graph reachability queries in constant time. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 75–75. IEEE.

Washio, T. and Motoda, H. (2003). State of the art of graph-based Data Mining. *Acm Sigkdd Explorations Newsletter*, **5**(1), 59–68.

Weber, M. and Heydebrand, W. V. (1994). *Sociological writings*. Continuum International Publishing Group Ltd.

West, D. B. *et al.* (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

Whiting, P. and Hillier, J. (1960). A method for finding the shortest route through a road network. *OR*, pages 37–40.

Wilson, R. A. and Keil, F. C. (2001). *The MIT Encyclopedia of the cognitive sciences*. MIT press.

Witten, I. and Milne, D. (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30.

Wood, P. T. (2012). Query languages for graph databases. *ACM SIGMOD Record*, **41**(1), 50–60.

Woodward, J. (2003). *Making things happen: A theory of causal explanation*. Oxford University Press.

Woodward, J. (2008). Cause and explanation in psychiatry. *Philosophical issues in psychiatry. Explanation, phenomenology, and nosology*, pages 132–184.

Wooley, B. A. (1998). Explanation component of software system. *Crossroads*, **5**(1), 24–28.

Wu, S., Manber, U., Myers, G., and Miller, W. (1990). An O(NP) sequence comparison algorithm. *Information Processing Letters*, **35**(6), 317–323.

Yan, X. and Han, J. (2002). GSPAN: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE.

Yan, X., Yu, P. S., and Han, J. (2004). Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 335–346. ACM.

Yeh, E., Ramage, D., Manning, C. D., Agirre, E., and Soroa, A. (2009). WikiWalk: Random walks on Wikipedia for semantic relatedness. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, pages 41–49. Association for Computational Linguistics.

Yoshida, K., Motoda, H., and Indurkhya, N. (1994). Graph-based induction as a unified learning framework. *Applied Intelligence*, **4**(3), 297–316.

Zapilko, B., Harth, A., and Mathiak, B. (2011). Enriching and analysing statistics with Linked Open Data. In *Proceedings of the NTTS Conference*.

Zhang, H. (2008). The scale-free nature of Semantic Web ontology. In *Proceedings of the 17th international conference on World Wide Web*, pages 1047–1048. ACM.

Zhang, X., Zhao, C., Wang, P., and Zhou, F. (2012). Mining link patterns in Linked Data. In *Web-Age Information Management*, pages 83–94. Springer.

Zheng, Q., Chen, H., Yu, T., and Pan, G. (2009). Collaborative semantic association discovery from Linked Data. In *Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on*, pages 394–399. IEEE.

Appendix A

Datasets Details

In this part, we present the details of the datasets that were used for the experiments throughout this work. As said, since our work was focused on the explanation of the patterns and not on their creation, we intentionally left the details of how the datasets were obtained as a minor section in this thesis. Table A.1 summarises the details of the datasets that will be further presented: the size of the dataset R, the number of clusters C, the method that we used to obtain the patterns, and how difficult (complex) we estimate it would be to find the pattern explanations.

Table A.1 Detailed description of the datasets used for the experiments.

Dataset R	$\|R\|$	$\|C\|$	Method	Complexity
#Red	368	10	K-Means clustering	3
#KMi.A	92	6	Network partitioning clustering	1
#KMi.P	865	6	XK-Means clustering	3
#KMi.H	6969	10	K-Means clustering	2
#Lit	152	3	Rate distance	2
#OU.P	1,192	30	Network partitioning clustering	3
#Tre	559	2*	Distance from average clustering	3
#OU.S	380	4	K-Means clustering	3

*per 13 trends

#RED - Readers from the Reading Experience Database. The *Reading Experience Database*[1] is a record of people's reading experiences, including metadata regarding the reader, author, and book involved in a reader's experience as well as its date and location. It consists in almost 400 people clustered in 10 groups according to the topics of the books the people have read. What we were interested in finding out was what makes people reading similar things. To obtain the patterns, we followed the process described below.

[1] http://www.open.ac.uk/Arts/RED/index.html

1) we used the DBpedia SPARQL endpoint to retrieve the topic (revealed by the *dc:subject*) of the books that were read in each experience.

2) we built a clustering matrix where each row was a reader of one experience, and each column one of the retrieved book topics. Each rows was then a vector of 0 and 1 representing whether the reader had read a book of that given topic (1), or not (0). An example of that is shown in Table A.2.

Table A.2 Clustering matrix for #Red.

reader	authors' categories				cluster general topic
	Women writers	English poets	Political philosophers		
Lord Byron	{0,	1,	0,	...}	J.Milton
Lady Byron	{1,	0,	0,	...}	J.Austen
James Clunie	{0,	0,	1,	...}	Karl Marx

3) we clustered the readers using the in-build K-means implementation of the Weka tool[2].

4) we labelled the clusters with the most representative topic of the group (last column in Table A.2), that we used as a reference when producing Dedalo's explanations (e.g. "people read *X* because of...", where *X* is the label of the cluster). The resulting clusters details and labels have been illustrated in Table 3.4.

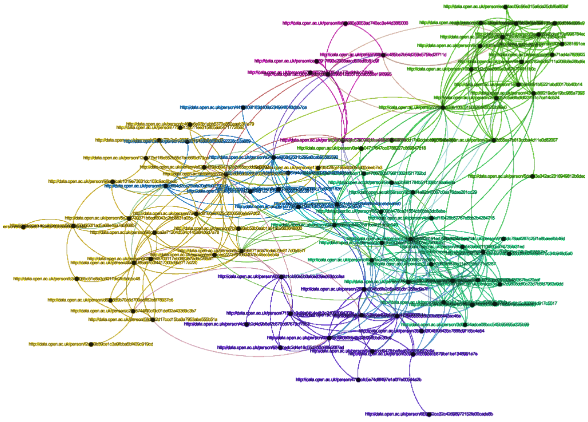


Figure A.1 #KMi.A. Each node is an author, and each colour is a group.

#KMi.A – The Knowledge Media Institute co-authorship. A dataset of 92 researchers from the Knowledge Media Institute that we clustered in 6 groups according to the papers they have written together. In this scenario, we were interested in explaining what makes researchers preferring some co-authors rather than others. Clusters where obtained as illustrated.

[2] http://www.cs.waikato.ac.nz/ml/weka/

1) we selected from the Open University dataset the identifiers for the researchers[3];

2) we built a graph where a researcher (a node) was connected to another only if the two researchers were co-authors;

3) we ran the Louvain Network Partitioning algorithm provided in Gephi, resulting in groups of co-authors as in Figure A.1.

4) we used our expertise (i.e. the knowledge about the department) to label the obtained clusters, as follows:

Table A.3 #KMiA: size of the cluster and attributed label.

Size	Label
19	People working on Semantic Web services
6	People working on Social Media Analysis
13	People working on Open Access
22	People working on Ontologies
23	People working on Learning Technologies
9	People working on Image Processing and Retrieval

#KMi.P – The Knowledge Media Institute publications. The dataset is similar to #KMi.A, except that here we clustered 865 research papers from the KMi department according to the content of their abstract. In this use-case we are looking into revealing the reasons of papers are similar. The dataset was obtained as illustrated:

1) we extracted the abstracts of each KMi paper using the Open University SPARQL endpoint[4];

2) we cleaned the texts using standard natural language techniques including normalisation, stopwords removal, and stemming;

3) we created a matrix where each row was a paper and each column was one of the terms, whose value for the row was the TF-IDF weight of the given word for the given paper;

4) we ran the X-Means algorithm using the Rapidminer tool[5];

5) we analysed the clusters and tried to define a general topic for them, as in Table A.4.

[3] e.g. Ilaria Tiddi's URI is http://data.open.ac.uk/page/person/cacb9bf9d6f500b32ecbd3751165bc53

[4] http://data.open.ac.uk/sparql

[5] https://rapidminer.com/

Table A.4 #KMiP: size of the cluster and attributed label.

Size	Label
18	Papers on Image Processing and Retrieval
5	Papers on Digital Storytelling
15	Papers on Educational Narrative Content
601	Papers on Learning Technologies and Human Computer Interaction
6	Papers on Semantic Web Services
220	Papers on Semantic Web, Ontologies, Knowledge and Smartcities

#KMi.H – The books borrowing observations. This is a dataset of 6,969 books that were borrowed by some university students, that we clustered according to the Faculty borrowers belong to. The original observations were provided by the University of Huddersfield dataset, from which we could extract the borrowed book and the students' faculties. We are interested in explaining what makes books being grouped together, e.g. they are about the same topics. We build the dataset as described below:

1) we extracted the books from the British National Library SPARQL endpoint;

2) we created a matrix where each row was a book and each column was one of the university faculties, and whose value was the number of students from that faculty that borrowed the book;

3) we used K-Means to obtain the clusters, that then we labeled as in Table 3.4.

#Lit – World literacy rate observations. The dataset consists in world countries partitioned in 3 groups according to the gender literacy rate. The idea for this use-case is to reveal what countries of a same rate have in common. To build the dataset, we followed the described steps.

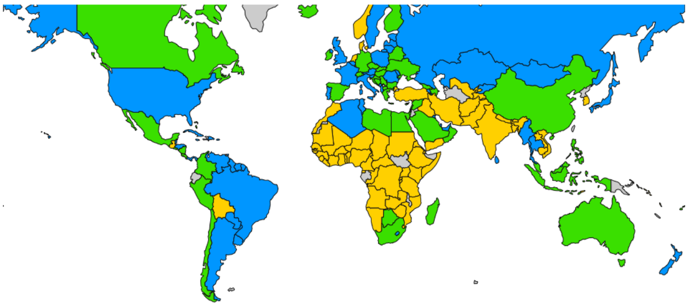
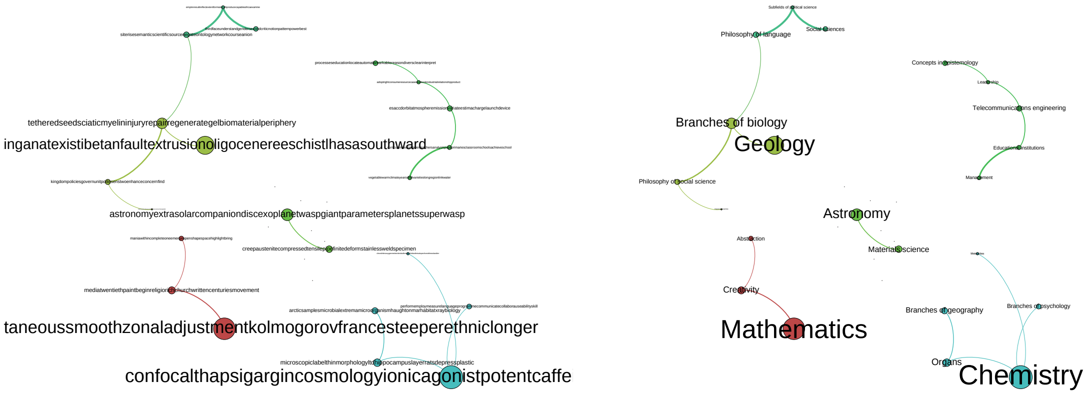


Figure A.2 #Lit – World countries grouped by their literacy rate.

1) We use a SPARQL query on the UNESCO dataset endpoint[6] to retrieve each country, the percentage of females and the percentage of males enrolled in secondary and tertiary education since the year 2000;
2) We calculated, for each country, the difference between the two obtained average rates
3) We created three groups accordingly: one of countries where men were more educated than women, one where women were more educated than men, and one where the education average was similar (the absolute difference between the two percentages is less than 2%), as in Figure A.2.

#OU.P – The Open University publications. This dataset consists in an extended version of #KMi.P, where 1,200 English words were extracted from ca. 17,000 paper abstracts from the Open University as in #KMi.P. For this use-case, we looked at explaining "why" words appear together, which indicates the topic of the papers clustered together. Words have been clustered into 30 communities according to their semantic similarity, using the following process.


siterisesemanticscientificsourcerelateontologynetworkcourseanion
thirdfaceunderstandgendersecondcriticnotionpatternpowerbest
processeseducationlocateautomatedremarkablereasondiversclearinterpret
adoptrightconsumeresourcecasebookindustrialrelationshipproduct
esaccdorbitatmospheremissionvariateestimachargelaunchdevice
tetheredseedsciaticmyelininjuryrepairregenerategelbiomaterialperiphery
inganatexistibetanfaultextrusionoligocenereeschistlhasasouthward
vegetablewarmclimateyearsvarietieslongregionlinkwater
kingdompoliciesgovernunitpoliticsenstwoenhanceconcernfind
astronomyextrasolarcompaniondiscexoplanetwaspgiantparametersplanetssuperwasp
maniawithincompleteoneemergeopenshapespacehighlightbring
creepaustenitecompressedtensilepipefinitedeformstainlessweldspecimen
mediatwentiethpaintbeginreligionchurchwrittencenturiesmovement
arcticsamplesmicrobialextremamicroorganismhaughtonmarhabitatxraybiology
performemploymeasurelanguageprogrammecommunicatecollaborauseabilityskill
taneoussmoothzonaladjustmentkolmogorovfrancesteeperethniclonger
microscopiclabelthinmorphologyltdhippocampuslayerratsdepressplastic
confocalthapsigargincosmologyionicagonistpotentcaffe

(a) Before Dedalo's labelling.

Subfields of political science
Philosophy of language
Social sciences
Concepts in epistemology
Leadership
Telecommunications engineering
Branches of biology
Geology
Educational institutions
Philosophy of social science
Management
Astronomy
Abstraction
Materials science
Creativity
Branches of geography
Branches of psychology
Mathematics
Organs
Chemistry

(b) After Dedalo's labelling.

Figure A.3 The Open University community network before and after Dedalo's labelling.

1) We pre-processed and cleaned the corpus of publication abstracts. We normalised the text to remove stopwords, numbers and punctuation and to reduce the words to their shortest possible raw form (stemming and stem completion). We also filtered out words whose length was too small (less than 3 characters).
2) We looked up in DBpedia for entities corresponding to the remaining words. If no correspondent was found the word was discarded.

[6] http://uis.270a.info/sparql

3) We applied the Latent Semantic Analysis (LSA) technique to derive the contextual similarity of the words. We built a TF-IDF weighted term-document matrix in which each column was a unique word of the cleaned corpus and each row a document of the corpus. Such matrix was then reduced into a lower dimensional space (the latent semantic space), using single value decomposition. This dimensions reduction collapsed the space in such a way that words occurring in similar contexts will appear with a greater (or lesser) estimated frequency.

4) We clustered words using the resulting space, and obtaining 30 communities of words. We then represented communities in a network connecting each community according to the distance between their centroids. In Figure A.3, we show the communities before having run Dedalo (where communities are still groups of words to be interpreted) and after.

#Tre – The Google Trends dataset. This dataset consists in 13 cases of Google Trend data[7], showing the graph of how often a particular search-term (a word, or a group of words) has been entered in the last 10 years. In this scenario we have only two groups: the moments in which the term is popular (high in the graph) and the moments in which it is not, and we are looking at discovering what makes a given term more popular than at other times with a specific regularity. To obtain the patterns, we proceeded as follows.

1) We retrieved some trends in a CSV format from the Google Trends page using a simple HTTP GET request. Each file included 559 weeks, from January 2004 to September 2014, and the search rate of the trend for that given week, normalised on the total number of searches[8]. We chose 30 trends divided into:

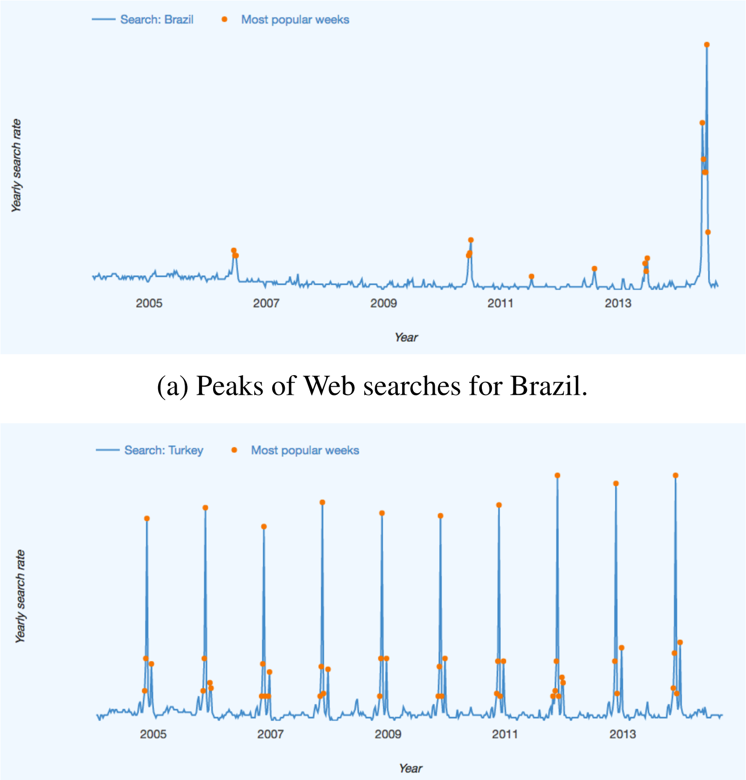
- *People*: Beppe (referred to Beppe Grillo, the Italian comedian and politician), Carla (referring to French-Italian singer and former French Première Dame Carla Bruni), Daniel Radcliffe, Obama, Taylor (referred to the American singer Taylor Swift);
- *Music*: Biffy, Carla, Pearl Jam, Taylor;
- *Places*: Brazil, France, Germany, Italy, MK Bowl (the Milton Keynes concert venue), Turkey, Wembley;
- *Movies/TV*: A Song of Ice and Fire, How I Met Your Mother, Sherlock, Star Wars, Winter is coming;
- *Sport events*: Dakar, Roland (the tennis tournament Roland Garros), Rugby, Tennis , US open, Wimbledon;

[7] http://www.google.com/trends/

[8] More information at https://support.google.com/trends/?hl=en#topic=4365599

- *Miscellaneous*: AT&T, Bitcoin, Hurricane, Vaccine, Volcano.

2) We partitioned weeks in a group of peaks and another group of non-peaks. A week was considered as a “peak" if its value was above the annual average (the 52 weeks around that week) plus an estimated threshold δ (to reduce noise). Each week was then weighted using the normalised distance between the real search value and the moving average. Figure A.4 shows the peaks for the trends Brazil and Turkey.



(a) Peaks of Web searches for Brazil.

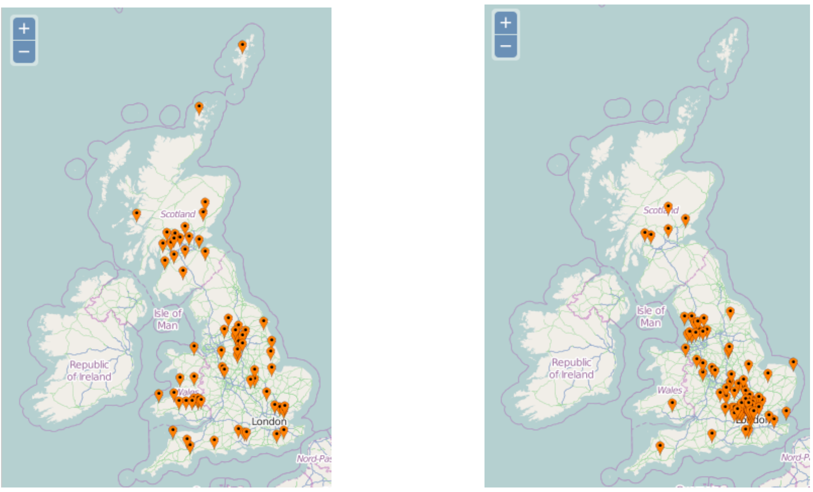
(b) Peaks of Web searches for Turkey.

Figure A.4 #Tre patterns.

3) We transformed the set of weeks in a RDF dataset by creating entities of type ddl:Week that were linked to events happened during that week. For that, we used a SPARQL query on DBpedia, where we selected any entity that had a declared date property between the `?_startDate` and the `?_finalDate` of the week:

```
select ?event where
  {?event ?happenedIn ?date.
  filter(?date >= "?_startDate"^^xsd:date &&
  ?date <= "?_finalDate"^^xsd:date)}
```

#OU.S – The Student Enrolment dataset. This dataset consists in observations about the Open University students and the courses they have enrolled. Since some faculties attract more students than others (e.g. Figure A.5 shows that students enrolling to the Health and Social Care Faculty (left) are concentrated around places such the Manchester area, while the Business and Law Faculty (right) attracts students from other places as the London districts), we are then looking into identifying a possible eco-demographic reason for that.



(a) Health and Social Care. (b) Business and Law Faculty.

Figure A.5 #OU.S patterns.

The dataset was built as follows:

1) From the Open University Linked Data dataset[9], we used a SPARQL query to extract OU students enrolled between 2005 and 2011, their address and the courses they have enrolled. We obtained 1,003,435 enrolments;
2) We identified students by their postcodes, and aggregated them using districts from the Ordnance Survey dataset[10]. We obtained 380 districts;
3) To reduce the number of courses, we aggregated them using the topic they are related to in the OU dataset. We obtained 14 topics;
4) We used the K-Means clustering to aggregated postcodes according to the topic of preference, and obtained 4 clusters.

[9] http://data.open.ac.uk

[10] http://data.ordnancesurvey.co.uk/

Appendix B

Linked Data Bias: Additional Results

In this Appendix we show the full experiments and results of the preliminary work conducted to quantify the bias in data links. The full work was published in [Tiddi *et al.*, 2014a]. All the experiments were run on a 2.9 GHz Intel Core i7 machine with 8GB of RAM.

First, Table B.1 shows the set of datasets for which we tried to identify the bias. Datasets were identified from the DataHub. For most of the datasets we chose, it was possible to calculate the bias in both directions. In cases where no SPARQL endpoint nor data dump was available, we marked the dataset as $D*$ and computed the bias only in the available direction. The Table include the dataset A for which we want to identify the bias, the dataset B that we used to project A, the the size $|B'|$ of the projection (in number of triples), and the time it took to run the full process.

The results showed that, on average, detecting the bias is not an expensive task. This, of course, is highly dependent on the resources one has to deal with: for very big linksets (in number of linked entities) or rich graphs (in number of classes, properties and values), the computation was naturally slower. Typically, this was the case when we ran the process on DBpedia, for instance, and on large biomedical data sources.

Table B.1 Bias detection. Computational costs in seconds.

A	B	$\|B'\|$	time”
LinkedGeoData	Linked Food	38	0.04
data.open.ac.uk	Unistats	82	0.11
DBpedia	Finnish National Library	336	0.40
DBpedia	DBTune	882	0.65
DBpedia	Hungarian National Library	500	0.74
New York Times	Geonames	1,787	0.74
DBpedia	Food and Agriculture Org.	215	0.79

AGROVOC	Food and Agriculture Org.	256	0.84
DBpedia	Reading Experience Database	6,549	0.93
DBpedia	New York Times	9,123	1.33
Freebase	New York Times	10,302	1.45
VIAF*	Spanish National Library	3,000	1.45
LOD-Ecosystems*	UniProt	17,355	3.83
Linked Food	LinkedGeoData	38	8.35
EUROVOC*	Arbeitsrecht	247	9.43
DrugBank	DBpedia	1,481	9.65
Unistats	data.open.ac.uk	82	11.92
LOD-Ecosystem	GeoSpecies	2,814	13.00
DBpedia	Org. Economic Coop. and Dev.	223	17.97
Food and Agriculture Org.	AGROVOC	256	21.03
DBpedia	DrugBank	1,481	32.39
UniProt	DrugBank	4,168	44.44
Finnish National Library	DBpedia	336	65.66
Org. Economic Coop. and Dev.	DBpedia	223	71.22
Eurostats*	LinkedGeoData	1,558	125.55
UniProt	Bio2RDF	18,997	127.21
BioPAX	UniProt	58,398	144.91
Food and Agriculture Org.	DBpedia	256	146.90
Eurostats*	Org. Economic Coop. and Dev.	3,488	301.57
DBTune	DBpedia	882	305.69
DBpedia	SW Dog Food	461	321.79
SW Dog Food	DBpedia	461	346.89
Geonames	New York Times	1,787	366.37
Hungarian National Library	DBpedia	500	484.40
New York Times	DBpedia	9,123	575.75
DBpedia	Spanish National Library	36,431	613.68
AGROVOC	DBpedia	11,014	657.60
Reading Experience Database	DBpedia	6,549	682.11
LOD-Ecosystems*	DBpedia	43,452	751.82
Spanish National Library	DBpedia	36,431	827.21
Open Energy Info	DBpedia	10,069	830.39
DBpedia	Open Energy Info	10,069	834.12
Unesco	Org. Eco. Coop. and Dev.	17,338	1,143.42
DBpedia	AGROVOC	11,014	1,270.46
Org. Economic Coop. and Dev.	Unesco	17,338	1,565.79
DBpedia	Linked Movie Database	13,758	1,579.36
New York Times	Freebase	10,302	1,587.00
Linked Movie Database	DBpedia	13,758	1,908.11
GeoSpecies	LOD-Ecosystem	2,814	1,944.32

DrugBank	UniProt	4,168	4,606.75
BioPAX	Entrez Gene	36,006	5,253.63
Entrez Gene	BioPAX	36,006	58,752.77
Bio2RDF	UniProt	18,997	60,459.88
UniProt	BioPAX	58,398	67,405.53

Finally, the second Table B.2 presents some more examples of detected bias, ordered by expectedness (from 1 to 4), as in Chapter 8.

$B \rightsquigarrow A$	c	p	value	p-value
(1) DBP⇝NLFi	db:Place	dc:subject	db:CitiesAndTownsInFinland	$p < 1.00 \times 10^{-15}$
(1) DBP⇝NLFi	db:Place	dbp:longd	*avg*: 24.6, *stdev*: 0.78	$p < 1.00 \times 10^{-15}$
(1) DBP⇝NLFi	db:Place	dbp:latd	*avg*: 40.5, *stdev*: 2.82	$p < 1.00 \times 10^{-15}$
(1) GN⇝NYT	gn:Feature	gn:parentFeature	gn:US	$p < 1.00 \times 10^{-15}$
(1) GN⇝NYT	skos:Concept	gn:inCountry	gn:US; gn:Mexico	$p < 3.34 \times 10^{-2}$
(1) DBP⇝NYT	db:Agent	dbp:country	db:UnitedStates	$p < 3.87 \times 10^{-4}$
(2) DBP⇝NLEs	db:MusicalArtist	dbp:birthPlace	db:England	$p < 1.13 \times 10^{-13}$
(2) DBP⇝NLEs	db:MusicalArtist	dbp:birthPlace	db:Spain	$p < 1.13 \times 10^{-13}$
(2) DBP⇝NLEs	db:Writer	dbp:nationality	db:French	$p < 4.64 \times 10^{-3}$
(2) DBP⇝NLEs	db:Writer	dbp:nationality	db:Spanish	$p < 4.64 \times 10^{-3}$
(2) DBP⇝RED	db:Scientist	db:almaMater	db:CambridgeUniversity	$p < 1.00 \times 10^{-15}$
(2) DBP⇝RED	db:Scientist	db:almaMater	db:UniversityOfEdinburgh	$p < 1.00 \times 10^{-15}$
(2) DBP⇝RED	db:Scientist	db:birthPlace	db:England	$p < 1.00 \times 10^{-15}$
(2) DBP⇝RED	db:Scientist	db:birthPlace	db:Scotland	$p < 1.00 \times 10^{-15}$
(2) DBP⇝RED	db:Writer	db:birthDate	*avg*: 1809, *stdev*: 96.21	$p < 1.00 \times 10^{-15}$
(2) DBP⇝RED	db:Writer	db:deathDate	*avg*: 1871, *stdev*: 99.19	$p < 1.00 \times 10^{-15}$
(2) RED⇝DBP	db:Scientist	db:birthPlace	db:UnitedStates	$p < 1.00 \times 10^{-15}$
(2) RED⇝DBP	db:Writer	db:birthDate	*avg*: 1916, *stdev*: 40.43	$p < 1.00 \times 10^{-15}$
(2) RED⇝DBP	db:Scientist	db:nationality	db:UnitedStates	$p < 4.28 \times 10^{-3}$
(3) UP⇝Bio2RDF	up:Protein	up:isolatedFrom	uptissue:Brain	$p < 9.29 \times 10^{-4}$
(3) UP⇝Bio2RDF	up:Protein	up:isolatedFrom	uptissue:Cervical	$p < 9.29 \times 10^{-4}$
(3) UP⇝BioPAX	up:Protein	up:isolatedFrom	uptissue:Brain	$p < 6.79 \times 10^{-4}$
(3) UP⇝DgB	up:Protein	up:isolatedFrom	uptissue:Brain	$p < 1.33 \times 10^{-4}$
(4) DBP⇝NLEs	db:Writer	db:genre	db:MisteryFiction	$p < 4.28 \times 10^{-11}$
(4) DBP⇝NLEs	db:MusicalArtist	dbp:instrument	db:Piano	$p < 2.73 \times 10^{-4}$
(4) DBP⇝RED	db:Agent	db:genre	db:Novel	$p < 1.00 \times 10^{-15}$
(4) DBP⇝RED	db:Agent	db:genre	db:Poetry	$p < 1.00 \times 10^{-15}$
(4) DBP⇝RED	db:Agent	db:movement	db:Romantism	$p < 1.00 \times 10^{-15}$
(4) DBP⇝RED	db:Agent	db:movement	db:Naturalism	$p < 1.00 \times 10^{-15}$
(4) DBP⇝RED	db:Agent	db:deathCause	db:Suicide	$p < 1.00 \times 10^{-15}$