JYVÄSKYLÄ STUDIES IN COMPUTING
237

Michael Cochez

Taming Big Knowledge Evolution

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 237

Michael Cochez

Taming Big Knowledge Evolution

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Delta-salissa
toukokuun 23. päivänä 2016 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Delta-hall, on May 23, 2016 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2016

Taming Big Knowledge Evolution

JYVÄSKYLÄ STUDIES IN COMPUTING 237

Michael Cochez

Taming Big Knowledge Evolution

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2016

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

Cover illustration by Michael Cochez based on a photo by David Iliff.
License: CC-BY-SA 3.0
https://creativecommons.org/licenses/by/3.0/deed.en

URN:ISBN:978-951-39-6649-2
ISBN 978-951-39-6649-2 (PDF)

ISBN 978-951-39-6648-5 (nid.)
ISSN 1456-5390

Copyright © 2016, by University of Jyväskylä

Jyväskylä University Printing House, Jyväskylä 2016

Ei menny niinku Strömsössä.

ABSTRACT

Cochez, Michael
Taming Big Knowledge Evolution
Jyväskylä: University of Jyväskylä, 2016, 56 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 237)
ISBN 978-951-39-6648-5 (nid.)
ISBN 978-951-39-6649-2 (PDF)
Finnish summary
Diss.

Information and its derived knowledge are not static. Instead, information is changing over time and our understanding of it evolves with our ability and willingness to consume the information. When compared to humans, current computer systems seem very limited in their ability to really understand the meaning of things. On the other hand, they are very powerful when it comes down to performing exact computations. One aspect which sets humans apart from machines when trying to understand the world is that we will often make mistakes, forget information, or choose what to focus on. To put this in another perspective, it seems like humans can behave somehow more randomly and still outperform machines in knowledge related tasks.

In computer science there is a branch of research concerned with allowing randomness or inaccuracy in algorithms, which are then called approximate algorithms. The main benefit of using these algorithms is that they are often much faster than their exact counterparts, at the cost of producing wrong or inexact results, once in a while. So, these algorithms could be used in contexts where erring once in while does not harm. If the chance of making a mistake is very slim, say lower than the chance of a memory error, then the expected precision will rival their exact counterparts. Furthermore, the input data to the algorithms often already contains a fair amount of uncertainty, such that the small error which the approximate algorithm introduces becomes more or less insignificant.

In this dissertation, the author investigates the use of familiar and new approximate algorithms to knowledge discovery and evolution. The main contributions of the dissertation are *a*) an abstract formulation of what it means for an ontology to be and stay optimal over time, *b*) a contribution to a vision paper regarding the future of evolving knowledge ecosystems, *c*) an investigation of the application of locality-sensitive hashing (LSH) in the context of ontology matching and semantic search, *d*) the twister tries algorithm which is a novel approximate hierarchical clustering approach with linear space and time constraints, and *e*) an extension on the twister tries algorithm which trades a longer, but adaptable running time for a likely improvement of the clustering result.

Keywords: Knowledge Evolution, Hierarchical Clustering, Information Retrieval

Author

M.Sc. Michael Cochez
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisors

Professor Vagan Terziyan
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Professor Ferrante Neri
Centre for Computational Intelligence, School of Computer Science and Informatics
De Montfort University
United Kingdom

Reviewers

Professor Ngoc Thanh Nguyen
Head of Information Systems Department
Wroclaw University of Technology
Poland

Associate Professor Francesco Guerra
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Italy

Opponent

Professor Evgeny Osipov
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Sweden

PREFACE

At the beginning of my studies (2012), I started to investigate what it would mean for an ontology or knowledge base to be optimal. The result was a definition of optimality described in function of an abstract system. The paper was published in the *Workshop on Dynamics and Evolution in Intelligent Systems* but remained very abstract. Further, I realized that for the time being it would not be feasible to solve this problem since there were too many unknowns and any attempt would hit the hard wall of time complexity.

After this realization, I joined other researchers who were investigating the use of biological models to mimic the evolution of knowledge. The result of this collaboration was published as a book chapter in the book "Big Data Computing", published by Chapman and Hall/CRC in early 2014. The core idea of this chapter is that the methods and mechanisms for the evolution of knowledge could be spotted from the ones enabling the evolution of living beings.

Besides, I started investigating the ontology alignment research field and realized that I should stop looking at ontologies as isolated entities. I became more interested in how to connect knowledge bases. Since small ontologies can be aligned fairly easily by hand, my main interest was the integration of large bodies of information. I studied techniques used in information retrieval and in particular document similarity and came up with a novel method for the alignment of large ontologies. This result was presented at the Web Intelligence Congress in 2014. The main benefit of the approach is that it is very fast. However, the price to pay for the fast answer is that the quality of the alignments is not very high.

Next, I wanted to try hierarchical clustering as an aid to improve the quality of the ontology alignment. However,traditional methods for hierarchical clustering do not scale well in the number of elements to be clustered. Hence, I started looking for a more scalable approach and came up with the idea of the twister tries algorithm. Using this algorithm, which uses locality-sensitive hashing, it becomes possible to approximately cluster very large datasets within reasonable time. Since the technique is wider applicable than just in my own research, I decided to publish it as a separate paper which was presented at the SIGMOD 2015 conference.

Then, my supervisor got me interested in the Keystone COST project in which research related to semantic keyword search is stimulated. We decided to take part in the conference and published a work showing how LSH tries can be used to dynamically align information tokens. This paper was then invited for extension to the LNCS Transactions on Computational Collective Intelligence journal.

In the mean time also the work related to hierarchical clustering advanced. Initial experimentation showed that it did not significantly increase the quality of ontology alignments, but experiments which enhance the twister tries algorithm with evolutionary algorithms turned out successful. This was reported in the 6th IEEE symposium on Computational Intelligence and Data Mining.

ACKNOWLEDGEMENTS

Eén op de zes mensen hebben [sic] vijf andere om zich heen.

– Harrie Jekkers

First and foremost, I would like to thank my partner, Petra Arminen, for tremendous amount of patience and support in this undertaking. Also my children, Emilia and Jasper have given me a fair amount of joy during these often stressful times. Next, I am grateful to my parents, Jacques Cochez and Anne Stevens for giving me an environment in which I could develop myself, allowing me to explore, believing in me at each step of my journey, and providing a safety net which I can always rely on. Besides them, I also have to mention my brothers and sisters: Maarten, Lara, Carla-Reina, Ada-Lina, Raphael, Emmanuel, and Lander, who are always ready to listen to my problems and about whom I know that they will always be ready to help at the full extend of their ability.

Before coming to the those who have had a direct influence on this dissertation, I would like to mention a couple of people who have had a long-term influence on my choices. First, there is Lode Van Puyvelde, my sixth grade teacher who allowed me to program instead of working on endless repetitions of the same sums and other pretty boring 'mathematics'. Then, there is Marc Verdonckt who was ICT coordinator at the secondary school I went to. He allowed me to work on computers during breaks and school holidays. Finally, there is Bruno Tourwé, my first boss, who gave me the chance to work as a summer intern at the ICT department of the Plantijn Hogeschool, Antwerp and employed me later during other school holidays as well. I learned a lot during these jobs and it shaped my decision to start studying computer science. These holiday jobs also gave enough savings to go on an Erasmus exchange during my bachelor studies.

During this exchange I attended the courses of Vagan Terziyan who later helped me to get a study place as a master student at the University of Jyväskylä. His courses had inspired me and a couple of months later I joined his research group, the Industrial Ontologies Group (IOG), trough the UBIWARE project . Then, a couple of years later, after acting as a supervisor of my master thesis, he accepted to be my guide on the path to defending my dissertation. For giving me all these opportunities, for much patience, standing my stubbornness, and for many interesting discussions, I would like to thank Vagan very much.

During my master studies I also met with Ferrante Neri, who was working as a researcher at the department. At the start of my doctoral studies he was planning to change affiliation, but still prepared to take the supervision. Him, I would especially like to thank for excellent advice and efficient collaboration.

Further, I would like to thank the other people of IOG, while I was there: Helena, Oleksiy, Michal, Timo, Sergiy, Joonas, Artem, Viljo, Jose, and Atte. I also had the honor of working on research papers (both dissertation related and not related) with several other people like Vadim, Hao Mou, Jiawen, Ville I., Ville T., Jonne, Sergey, Tapani, Horacio, Tero, Muhammad Zeeshan, Jacques P., Kyryl, and Rajendra. I thank all of them for fruitful collaboration.

I would also like to thank the reviewers and editor for their comments aimed at the improvement of this work. Already in advance I thank Evgeny Osipov for accepting the invitation to act as an opponent of my dissertation.

Further, I would like to acknowledge the support of the Faculty of Information technology; in particular the Faculty Office and the Department of Mathematical Information Technology. Another supporting environment was the Agora Center in which the UBIWARE, Cloud Software SHOK, and Need4Speed SHOK programs were hosted. These projects were all partially funded by the National Technology Agency of Finland (TEKES) and participating companies, like ABB, Hansa Ecuras, Fingrid, Inno-W, Metso Automation, Metso Shared Services, Nokia, and Tieto. From these projects, I want to highlight one company and person with whom I have had the pleasure of collaborating on several stages, namely Steeri Oy and their former Development Director Sami Helin (now Development Manager at Hospital District of Helsinki and Uusimaa (HUS)).

Another organization which affected my work has been CSC, the Finnish IT Center for Science, which has provided me with computational resources and a summer school about high performance computing. Also the local IT support at the Agora building deserves a special mention for reacting fast and professionally on my sometimes quite demanding requests.

During my studies I also had the opportunity to attend the summer school on modern computational science in Oldenburg, Germany for which I received an Erasmus international staff training grant. Further financial support for conference travel was provided by the ICTERI conference organizers, ACM SIGMOD, and the IEEE Computational Intelligence Society.

I also have to thank the Nokia Foundation for financially supporting my studies and the research council of the university of Jyväskylä for proving funding for my research visit in Galway, Ireland. This visit was in part made possible due to Stefan Decker, my host at the Insight Centre. I have to thank him for many interesting discussions and inspiring viewpoints.

One special group of people was essential for my work. The janitors and cleaning personnel at the university kept my work environment survivable despite the jungle-like disorder on my desk. I very much appreciate their work.

Getting closer to the end of this long list of people and organizations who have helped and inspired me, I cannot get around thanking all the teachers who contributed to my personal development as well as the many students who attended my courses and whom I supervised. The students often challenged me by asking the right questions. Hopefully, my answers were adequate and will help them on their path.

Writing this dissertation would have been a lot harder without the effort of Matthieu Weber and Antti-Juhani Kaijanaho, who created the LaTeX template and with whom I have had several interesting discussions. Last, but certainly not least, I would like to thank the many people I played floorball with and other friends I made along my journey.

Jyväskylä, May 2016
Michael Cochez

ACRONYMS

ABox	Assertion Box
AI	Artificial Intelligence
DDL	Data Definition Language (part of SQL)
DL	Description Logics
DML	Data Manipulation Language (part of SQL)
DNA	Deoxyribonucleic Acid
EA	Evolutionary Algorithms
EKE	Evolving Knowledge Ecosystem
ES	Evolution Strategies
f-RHH	fuzzy Random Hyperplane Hashing
GA	Genetic Algorithms
GP	Genetic Programming
ICT	Information and Communications Technology
IoT	the Internet of Things
KE	Knowledge Evolution
LSH	Locality-sensitive Hashing
NIST	National Institute of Standards and Technology (US)
OWL	Web Ontology Language
RDF	Resource Description Framework
RHH	Random Hyperplane Hashing
SHOK	Strategic Centres for Science, Technology and Innovation (Strategisen huippuosaamisen keskittymät) – a type of projects funded by TEKES
SQL	Structured Query Language
TBox	Terminological Box
TEKES	The Finnish Funding Agency for Innovation
W3C	The World Wide Web Consortium

LIST OF FIGURES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Michael Cochez and Vagan Terziyan. Quality of an ontology as a dynamic optimisation problem. *ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Proceedings of the 8th International Conference ICTERI 2012*, CEUR Workshop Proceedings (Vol-848). Aachen: RWTH Aachen, 2012.

PII Vadim Ermolayev, Rajendra Akerkar, Vagan Terziyan, and Michael Cochez. Toward evolving knowledge ecosystems for big data understanding. *R. Akerkar (Ed.), Big Data Computing (pp. 3-56). Boca Raton, FL: Taylor & Francis*, 2014.

PIII Michael Cochez. Locality-sensitive hashing for massive string-based ontology matching. *In D. Ślęzak, B. Dunin-Kęplicz, M. Lewis, & T. Terano (Eds.), 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 2014.

PIV Michael Cochez, Vagan Terziyan, and Vadim Ermolayev. Balanced Large Scale Knowledge Matching Using LSH Forest. *In J. Cardoso, et al. (Eds.), Semantic Keyword-based Search on Structured Data Sources : International KEYSTONE Conference, IKC 2015, Coimbra, Portugal*, Revised Selected Papers (pp. 36-50). Lecture Notes in Computer Science (9398). Springer International Publishing. This actual article was not included since PV is the extended version of this paper, 2015.

PV Michael Cochez, Vagan Terziyan, and Vadim Ermolayev. Large Scale Knowledge Matching with Balanced Efficiency-Effectiveness Using LSH Forest. *This paper is an extended version of PIV*, It is currently under review for publication in the LNCS TCCI Journal, 2016.

PVI Michael Cochez and Hao Mou. Twister Tries: Approximate Hierarchical Agglomerative Clustering for Average Distance in Linear Time.. *In T. Sellis, S. Davidson, & Z. Ives (Eds.), SIGMOD '15 : Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, New York: Association for Computing Machinery. This article received the Reproducible label http://db-reproducibility.seas.harvard.edu/papers/*, 2015.

PVII Michael Cochez and Ferrante Neri. Scalable Hierarchical Clustering : Twister Tries with a Posteriori Trie Elimination.. *In SSCI 2015 : Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence. Symposium CIDM 2015 : 6th IEEE Symposium on Computational Intelligence and Data Mining*, 2015.

1 INTRODUCTION

> It is very difficult to know what you take for granted. And the reason is that you take it for granted.
>
> *– Sir Ken Robinson*

For a long time people have tried to create machines whose capabilities equal or surpass human ability, creating what some would call strong[1] artificial intelligence (Searle, 1980). However, despite recent successes (Schaeffer et al., 2007; Tesauro et al., 2013; Lee, 2016) in creating specific algorithms to play games for which most would agree that a combination of intelligence and creativity are required, there is no machine with capabilities resembling general human intelligence to date. One of the reasons why human intelligence is hard to attain is that we are capable of adaptation to the environment in which we reside. The above game playing programs are tailored to a specific game, and will not adapt themselves to totally new tasks when asked to do so[2].

One of the tasks an intelligent machine needs to perform is keeping a representation of its environment. The environment is complex and large and hence also its representation will be. Moreover, this representation should be updated whenever the environment changes (Ermolayev et al., 2014). The currently available computing resources are unable to deal with a problem of such complexity in an exact fashion. Humans, however, seem to be able to perform this task, but are also prone to making mistakes and forgetting facts. Therefore, we might be able to build a machine with a competing level of intelligence if we also allow it to err once in while. This dissertation provides some parts of the puzzle by investigating approaches for connecting knowledge and managing its evolution, while allowing an amount of error to happen.

---

1 Most commonly the distinction between *strong* and *weak* AI is that the former really *understands* what it is doing, while the later only *simulates* this behavior. This distinction is sometimes used to argue that strong AI does not exist.

2 Obviously, parts of these engines are more general and can be used in other games or intelligent tasks. But, these systems require engineers who create the new set up. For general game AI, see http://www.gvgai.net/.

Research Questions

This dissertation contains several contributions related to the use of approximate algorithms to aid knowledge discovery and evolution. In particular, the following research questions are answered:

RQ1: What does it mean for an ontology to be optimal?
RQ2: Can min-hashing, as applied in information retrieval, help in the alignment of ontologies?
RQ3: Is the same min-hashing also helpful when sowing knowledge tokens in a knowledge ecosystem?
RQ4: Can a hierarchical clustering algorithm be adapted to large scale data if we allow an error?

Structure of the Dissertation

This dissertation is composed of several articles, which contain the contributions. These articles are preceded by this introduction, a foundations section, and overview of the contributions. The foundations section contains short introductory descriptions of concepts necessary for a fluent understanding of the articles. The reader is encouraged to skim trough these and read the explanation of topics which are unfamiliar. After the foundation section, the reader is expected to read the actual articles, which are the main contribution of this dissertation. Next, there is an overview of the research contributions organized by topic and a listing of the share of the work performed by the candidate. Finally, there is a conclusion and outlook on future research. Throughout the dissertation the included papers are cited using the letter P followed by the Roman numerals of the article. For instance, the sixth paper is cited using PVI.

2 FOUNDATIONS

> Souvenez-vous que dans les champs de l'observation le hasard ne favorise que les esprits préparés.
>
> *– Louis Pasteur*

In this chapter the foundations for the contributions are presented. The descriptions are intentionally left broad to give a wider view onto the field as strictly required, on the other hand many details are left out. The main goal of this chapter is to sketch a context for the articles, and the next contribution section. More detailed definitions, results, and focused related work is to be found from the included articles themselves.

2.1 Ontologies

Already in ancient times philosophers thought in abstract terms about concrete things in their environment. These abstract terms, also called concepts, and their relations is what some would call an ontology. A more recent, often quoted, definition by Gruber (Gruber, 1995) in the context of artificial intelligence states that "An ontology is an explicit specification of a conceptualization". Further, Gruber stresses that an ontology needs to be a formal specification, i.e., one which can be described using mathematical logic or, in other words, unambiguously.

In the past there have been several efforts to harness the problem of representing knowledge, however many have fallen out of favor. For example, Frame languages (see (Karp, 1993) for an overview) and several Description Logics (DL) systems (see (Baader, 2003)) have not achieved high levels of adoption. The main exception is the OWL, the Web Ontology Language, which is essentially a specific description logic (Horrocks et al., 2003). OWL (and its later incarnation OWL2) are standardized by the W3C (see van Harmelen and McGuinness (2004) and OWL Working Group (2012)). There are several differences between these two versions, but for the scope of this dissertation they are not essential. Therefore,

```
@prefix ex: <http://www.example.com/#> .
@prefix user: <http://users.jyu.fi/~> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

user:miselico a ex:teacher .
ex:teacher rdfs:subClassOf ex:employee .
user:miselico ex:name "Michael"
```

FIGURE 1 An example RDF document encoded in Turtle. The example shows an individual (`ex:miselico`) which is an instance of the class `ex:teacher`. Further, this teacher class is itself a subclass of the `ex:employee` class. The final triple states that individual has the string literal property `ex:name` with value "Michael"

we will refer to both as OWL and in cases were distinction is needed as OWL1 and OWL2.

A second framework which is commonly used for knowledge representation is Resource Description Framework (RDF) (Wood et al., 2014). At the core of RDF lies the *RDF Graph*, which is essentially a set of triples. Each triple consists of a subject, a predicate, and an object. The subject and object can be regarded as nodes in the graph and the predicate as a directed arc between them. The interpretation of such triple is that the relation represented by the predicate holds between the entities referred to by the subject and object. Information is encoded in RDF by declaring classes, the inheritance relation between the classes, instances of these classes, and other properties of the instances. An illustrative example in can be found in fig. 1. OWL ontologies are commonly encoded using RDF[1] and the main idea of OWL is that it defines an ontology for data which is encoded in RDF.

A word about terminology: in current research on knowledge representation the term ontology is often used in two somewhat different meanings. First, it is used to mean everything related to concepts, classes, and relations, but not instances. Another term used for this meaning is the so-called TBox. Second, it is used to *also* include the individuals. The term ABox is used to denote the individuals and their assertions and hence this second meaning would be the union of the ABox and TBox. For the remainder of this work, except where explicitly noted otherwise, we will be using the first meaning.

2.1.1 Ontology Matching

This short discussion on ontology matching is mainly based on the Ontology Matching book by Euzenat and Shvaiko (Euzenat and Shvaiko, 2013), which is considered a standard work in the field, one of their recent articles (Shvaiko and

[1] To be precise, OWL has both RDF-based Semantics and Direct Semantics based on description logics. These have differences, but for the scope of this dissertation they can be regarded as equivalent.

Euzenat, 2013), and the somewhat older work, on more general schema matching, by Rahm and Bernstein (Rahm and Bernstein, 2001). An ontology, as described above, is used to represent information about classes, their relations and properties. In practice, an ontology is created for a certain domain of interest. Ideally, everyone would publish their ontologies and reuse would be one of the main objectives when creating new ontologies. In that case all ontologies would be interlinked and duplication of classes would be rather rare. However, in practice ontologies are often created in isolation, i.e., not linked to any other ontology and hence not reusing concepts. The reason for this is that it would require a large upfront investment to research the existing ontologies and discover what can be reused. Furthermore, the creator of the original ontology might at a later point decide to make changes in his ontology leading to unexpected incompatibilities later on. This duplication of efforts leads to the situation where many ontologies exist for the same or overlapping domains of interest.

The idea of ontology matching is to solve this problem by finding out which entities (classes, relations, etc., but in some formulations also entities) from two different ontologies have a relation between them, like, for instance, the same semantic meaning. For example, two classes A and B have the same semantic meaning if all individuals from A are also in B and vice versa. In this case it is said that there is an equivalence relation between A and B. Also other relations are of interest, most commonly the 'less general' and 'more general' relations. Meaning that all entities belonging to A also belong to B, or that all entities belonging to B also belong to A, but not the other way around. When a relation r between two entities A and B has been found, it is said that a correspondence $\langle A, B, r \rangle$ exists between the ontologies. A set of these correspondences is called an alignment between the ontologies. Using this vocabulary, the goal of ontology matching is to find a complete and consistent alignment between ontologies.

After the alignment has been found it can then be used in several ways. Most commonly it is used for ontology merging (two or more ontologies are combined into a single one) or data translation (instances described according to the one ontology are transformed into instances according to the other ontology).

There are several ways to classify ontology matching algorithms. There are three main characteristics, namely *a*) the input of the algorithm, *b*) the matching process, and *c*) the output produced by the process. We will not give a detailed overview of the input and output formats since it is not essential to the dissertation. Most modern systems for ontology matching take RDF and/or OWL as their input and produce a list of correspondences between entities from both 'sides', often with an associated confidence level. Some systems will also utilize external resources as part of the process.

The matching process can be classified according to several facets.

- First, one could decide to find correspondences by investigating entities in an isolated fashion, essentially ignoring their relationships as well as possible information available about their instances. These *element-level* approaches are opposed to structure-level techniques where the relationships between entities, or in other words the structure of the RDF graph, is used

to detect relationships.

- Second, a distinction is made between semantic and non-semantic (also called syntactic) methods. Semantic techniques use formal semantics to obtain their results while syntactic methods do not.

The articles in this dissertation present syntactic, element-level approaches. Hence, we limit further discussion to this class. The most common techniques for discovering matches for element-level matching can be divided in string-based, language-based, constraint-based, and resource-based techniques (Euzenat and Shvaiko, 2013). Often a combination of these techniques is used to create a complete matcher.

The string-based techniques exploit the fact that if labels or descriptions of entities in two ontologies are similar, then it is likely that the entities are similar. In practice all sorts of techniques have been used to determine the similarity of strings. Most often a distance metric (see definition 1 for a precise definition) is used. This function maps two strings to a non-negative real number and the smaller the number, the more similar the strings. Examples of string distance metrics include edit distances (a measure of the number of changes needed to transform the one string into the other), the Jaccard distance between the set formed by the words (see also PIV), angular distance between the word vectors (see also PV), length of common prefix, etc. The label or description will often be preprocessed before the metric is applied (e.g., characters can be converted to lowercase, punctuation removed, etc.)

Language-based techniques use knowledge about the language in which the information about the entities is encoded to make a decision about the similarity. A canonical example is when two ontologies are described with labels in different languages, say English and German. In that case one might try to first translate the German information to English before attempting to use one of the string-based techniques mentioned above. This is fairly typical: often one first uses language-based techniques as a preprocessing step for a string-based metric. Other examples of language-based techniques include tokenization (segmenting into a sequence of tokens), lemmatization (replacing words by their roots), stemming (a simplified form of lemmatization in which words are replaced by their stems, which are formed by following a set of rewrite rules), and stop word elimination (removal of frequently used words, which are unlikely to add meaning to the comparison).

Constraint-based techniques take into account the restrictions which are placed on the members of the class in the ontology. The assumption is that similar entities will have similar constraints in both ontologies, or that if the constraints on a class A in the one ontology imply all constraints on a class B in the other one, then it is likely that B is a subclass of A, or using the terminology introduced above that A is more general than B. Examples of constraints include data-types, cardinalities, and other relational properties.

Resource-based techniques are used when also other resources are available for the matching task. The idea is that if both an entity $\hat{A}$ in ontology A and

an entity $\hat{B}$ of ontology B have a relation to similar external resources, then $\hat{A}$ and $\hat{B}$ are likely similar. For instance, imagine that the ontologies A and B are used to annotate images. Then, if an image of a cat gets annotated with class $\hat{A}$ of ontology A and another cat image gets annotated with class $\hat{B}$ of ontology B, then it is likely that A and B are similar. The images and their annotations are then examples of external resources. One can argue that this is an example of structural-level matching, since strictly speaking the images have an instance-of relation with their respective classes. Note, however, that we are not matching the individuals (images) but the classes. Other techniques in this category include those which use external ontologies to augment the matching quality.

2.2 The Case for Big Knowledge Evolution

When an ontology is combined with instances and looked upon as a whole, it is often called a knowledge base[2]. In this section, a short introduction of why knowledge bases are growing out of control will be presented. Further, there will be a discussion of their unavoidable evolution. Much of this section is inspired by the second article included in this dissertation (PII).

2.2.1 Big Knowledge

The near ubiquitous availability of the Internet and the explosive growth of the mobile device market, combined with the lowering prices of sensors, leads to the generation and availability of a vast amount of information. The volume of the data produced by this so-called Internet of Things (IoT) is expected to grow exponentially in the coming decades. Clearly, the production of data is surpassing the capabilities of current tools for processing and performing analyzes of the data. (Manyika et al., 2011)

This growth in data also causes a growth in the ontologies needed to describe it. The main cause of this is that not only the size of the data is growing, but also the variety of data is on the rise. It has also been observed that the more data, the more potentially interesting patterns can be found. And, this also means that the tools for detecting actually interesting patterns have to be more sophisticated.

All in all, it can be noticed that also knowledge bases will soon enough fulfill the properties of Big Data as put forward by the NIST Big Data working group (Grady and Chang, 2015), namely that the current state of knowledge base implementations cannot efficiently handle the new datasets (See also Urbani et al. (2009); Oren et al. (2009) and Ivanova et al. (2015) for examples related to reasoning, alignment, and query transformation). The NIST Big Data definition stipulates that the following characteristics are pressing the need for new architectures:

[2] The term knowledge base has many overloaded definitions. Common to these is that a knowledge base contains facts about entities.

a) Volume – the size of the ontology and the amount of instances, *b*) Velocity – the speed at which data is generated, *c*) Variety – data from multiple domains, *d*) and Variability – the change of characteristics of the data, which also causes a need to change the ontology (see also section 2.2.2).

Hence, to keep up with the growth and changes in the extracted information in a timely manner we will need to develop algorithms with a lower time complexity and likely have to content ourselves with approximate solutions or trust heuristics (see also section 2.3.4).

2.2.2 Knowledge Evolution

Knowledge evolution (sometimes called ontology evolution) observed in knowledge bases is corresponding to schema evolution in relational databases. Schema evolution has been studies for quite some time (see e.g., Bachman (1975); Ventrone (1991)) and is concerned with changes in the schema of relational databases needed to adapt the system to changing requirements of applications. In knowledge bases it is the ontology (the TBox) which needs to adapt to changes in the entities it is describing. It seems like schema evolution and ontology evolution are very similar. There are, however, some major differences (Hartung et al., 2011). First, in database schema there is a strict separation between the schema and the data. In fact SQL, the most common family of query languages for relational databases, does distinguish between the data definition language (DDL, involving the schema) and the data manipulation language (DML, involving language constructs for data). In knowledge based systems the ontology is often described in the same terms (using the same formalism) as the data and hence it can also be added to the data creating some form of self-descriptive data. Second, whereas schema are often designed with only the specific application in mind, ontologies are usually designed for reuse in other contexts in a decentralized fashion. Last, ontology models (like OWL, see section 2.1) have a larger set of tools available to describe the domain.

The causes of a need to adapt the ontology are either changes in the domain, changes in the shared conceptualization, or changes in the specification (Klein and Fensel, 2001). This is perhaps not surprising given the definition of ontologies in section 2.1. Changes in the domain are often occurring (Klein and Fensel, 2001) and obviously more often than changes in the other two aspects because the latter ones are more or less in control of the designers of the system; the most common reason to make changes in these is in order to make systems compatible. Usually ontologies are changed manually or semi-automatically and therefore they are often available in several revisions. One of the disadvantages of these revisions is that instances created according to the 'same' ontology are not by default compatible. Misinterpretations might be possible and also internal models used by applications might need adaptations to be able to deal with the new revision. The disconnection between the domain and the knowledge which it should reflect is one of the major obstacles for wider industrial adoption of semantic technologies (Hepp, 2007; Marshall and Shipman, 2003).

The whole concept of ontologies changing over time is captured by the term Knowledge Evolution. Knowledge Evolution is a dynamic and inherently active process which encompasses the (stimulation of) knowledge creation, the ability of knowledge sharing, and a way to remove knowledge (i.e, forget) when it becomes obsolete. (Maracine and Scarlat, 2009)

2.3 Locality-sensitive Hashing

When classes are being aligned as part of an ontology alignment one searches for correspondences such that the two classes (from different ontologies) are similar to each other. One way to do this is take a class $\hat{A}$ in ontology A and compare it with each class of ontology B. From B, one would then select $\hat{B}$ such that $\hat{B}$ is class most similar to $\hat{A}$. In this case we say that $\hat{B}$ is the nearest neighbor of $\hat{A}$. This step can be repeated for each class in A and works if the number of classes of ontology A and B (which we will call $|A|$ and $|B|$) are not too large. However, the number of comparisons needed is $O(|A| * |B|)$ and hence, when A and B are large, this becomes prohibitively many. Therefore, we need a faster way to find nearest neighbors for classes in A.

Locality-sensitive hashing (LSH) refers to an algorithm used for approximate near neighbor search. In this section we first introduce the nearest neighbor problem and a basic solution for it. Then, we introduce locality-sensitive hashing, which results in faster, but approximate near neighbors. Next, LSH forest is introduced, which is derived from standard LSH but has some specific beneficial properties. Finally, we look at approximate algorithms in general. LSH is one algorithm of that class.

2.3.1 Nearest Neighbor Search – Distance Metrics

Imagine a company planning to employ a new employee. The company has a dataset consisting of many profiles of potential employees and a job description. Now, they want to look for the 10 candidates who are as similar as possible to the said description to invite them for an interview. To solve this task, the employer can take the description and compare it with each profile, and rank the profiles by similarity to the profile. In the end the company can decide to interview the 10 candidates with the best ranking.

The previous scenario is incomplete and in order to create a formal algorithm to solve the company's problem there are two more questions to answer. First, what do the profiles and the description look like? Second, given a profile and a description, how do we measure their similarity?

When generalizing this problem, one could call a candidate's profile a *data point* and the job description a *query*. Then, we can say that given a *database* (i.e., a collection of data points) the task is to find the subset of k data points which are most similar to the query. This problem is known in the literature as the

k-nearest neighbor problem. This generalization has the same shortcomings as the example above. We did not yet specify what a data point consists of and how we decide upon the similarity. In general, any set of things and any kind of query, combined with a mathematical function, mapping a thing and a query to a completely ordered set, would be sufficient for this purpose. In practice, however, we will limit ourselves to queries and data points which are elements of a metric space. A metric space consists of a set of elements on which a distance metric is defined. In turn, this distance metric is a function which takes as its arguments two elements from the set and returns an element from its range, which we will limit to be a subset of the real numbers[3].

Let us look at another example where we are interested in finding the 5 monuments closest to the Eiffel Tower in France. First, we would need a database of the monuments, with their location. Then, we use the position of the Eiffel Tower as a query. The 5 closest monuments are then found by computing the Euclidean distance between the Eiffel Tower and all other monuments in the database, ranking them by distance, and returning the 5 monuments with the lowest distance. This example looks somehow different from the previous one. Instead of looking for data points with the highest similarity, we are looking for data points with the smallest distance to the query point. However, these two problems are practically the same and for many similarity measures (where a high number indicates similar things) there are equivalent distance metrics (where a number close to zero indicates similar things).

We will now define what a distance metric is and illustrate it with the Eiffel Tower example.

Definition 1 (distance metrics) *(adapted from (Leskovec et al., 2012, p92)) Given a set S, a distance metric d on this set is a function which maps two elements of the set to a real number. The function must satisfy the following conditions*

1. $\forall a, b \in S, d(a,b) \geq 0$. *Positive: no distance can be negative.*

2. $d(a,b) = 0 \iff a = b$. *The distance can only be zero for identical points. In the monument example, this means that only the Eiffel Tower itself has a distance of zero to the Eiffel Tower. Combined with property 1 this means that every other monument must have a strictly positive distance to the Eiffel Tower.*

3. $d(x,y) = d(y,x)$. *Symmetric. The distance from the Eiffel Tower to the Louvre is the same as the distance from the Louvre to the Eiffel tower.*

4. $d(x,y) \leq d(x,z) + d(z,y)$. *Triangle inequality. If the distance from The Eiffel Tower (x) to the Louvre (y) is $\hat{d}$, then it is impossible to find a third monument (z) such that the sum of the distance from x to z and the distance from z to y would be smaller than $\hat{d}$*

[3] In principle, any totally ordered set would work, but we avoid this because in practice most, if not all, distance metrics have a subset of the real numbers as their range. This limitation also saves on notation.

Definition 2 (Metric space) *A metric space M is a pair (S, d), where S a set and d a distance metric on S. When we talk about a metric space M with associated distance metric d, the existence of S is implicitly implied. Further, if we write $p \in M$, we mean $p \in S$.*

The Euclidean distance as used in the Eiffel Tower example above is a distance metric. Other distance metric include the angular distance (i.e. the angle between two vectors), the Jaccard distance (given two sets A and B, the Jaccard distance is defined as $d_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$), and the Hamming distance (number of places in which two vectors differ from each other).

2.3.2 Locality-sensitive Hashing

The examples in the previous section show how defining a distance metric is sufficient for solving the k-nearest neighbor problem. However, there is something somewhat unsatisfying about the approach. Namely that in order to find the near neighbors of a query, one has to investigate the distance of the query to all data points in the database. This becomes problematic if the database is large. Several techniques have been developed to find faster solutions for this problem. In this section Locality-sensitive Hashing (LSH) will be discussed, which solves the problem of finding approximate near neighbors. This discussion is inspired by Andoni and Indyk (2008). The attentive reader will have noticed that the problem name has changed slightly. Instead of looking for near**est** neighbors, we are looking for near neighbors. In the case of k-nearest neighbors we are looking for the k points which are closest to the query point, while in the case of near neighbors we try to find all points which are within a given radius R from the query point. Formally, the near neighbor problem is defined as follows:

Definition 3 (R-near neighbor) *Given a metric space M with associated distance metric d. Then, given two points $p, q \in M$ and a constant $R \in \mathbb{R}$, p is a near neighbor of q if $d(p, q) \leq R$.*

The near and nearest neighbor problems are closely related, but not exactly the same. It seems that if we find all near neighbors, we can obtain the nearest by applying the exact nearest neighbor finding approach described in the previous section on the near points. However, we are not guaranteed that any point will be returned at all to our near neighbor query with parameter R. To solve this problem, we will need to query with different, increasing values of R until we find enough points to return (see also section 2.3.3). Obviously one can also fix a limiting value of R above which we are not interested in the neighbors any longer. To continue the example from the previous section, we could for instance say that we are not interested in monuments near the Eiffel tower which are more than 10km. away, even if we have not found 5 monuments yet.

A more relaxed formulation of the near neighbor finding problem is the following:

Definition 4 (randomized R-near neighbor reporting problem) *(adapted from (Andoni and Indyk, 2008, def 2.2), but generalized to any distance metric) Given a set P of points in a metric space and parameters $R > 0, \delta > 0$, construct a data structure that, given any* query *point q, reports each R-near neighbor of q in P with probability $1 - \delta$.*

The new part is that there is a (in practice small) probability δ that points which are within the radius are not returned. It will turn out later that allowing this small probability of error enables us to create faster algorithms. Whether these errors are a problem depends on the use case in question. For our Eiffel Tower example it would not be a big problem if one of the monuments within the set radius is not included in the final results. Moreover, in this example, it would not even matter if a monument slightly beyond the set radius is included (i.e., a 'reasonable' false positive does not harm). Now, we will introduce locality-sensitive families, which are sets of functions which are used to later create data structures to solve the randomized R-near neighbor reporting problem. There are several slightly different definitions for locality-sensitive hash function (see e.g., Charikar (2002); Wang et al. (2014); Andoni and Indyk (2008)). We will use the following definition, which is most general:

Definition 5 (Locality-sensitive family) *(adapted from Andoni and Indyk (2008)) Let $\mathcal{H}$ be a family of hash functions mapping from a domain $\mathcal{D}$ to some universe $\mathcal{U}$ and d be a distance metric defined on $\mathcal{D}$. Then, given $d_1 < d_2$, $\mathcal{H}$ is called (d_1, d_2, p_1, p_2)-sensitive if for every two $p, q \in \mathcal{D}$ and every $h \in \mathcal{H}$*

$$\text{if } d(p,q) \leq d_1 \text{ then } \Pr_{\mathcal{H}}[h(p) = h(q)] \geq p_1$$
$$\text{if } d(p,q) \geq d_2 \text{ then } \Pr_{\mathcal{H}}[h(p) = h(q)] \leq p_2$$

where $p1 > p2$

This definition is somewhat bombastic and can best be explained with an example. However, to explain this with an example, we first have to introduce a specific LSH family. For this example, we selected a well known LSH function family for the Jaccard distance which was introduced in Broder (1997). This family of functions is called *Min-hash*

Definition 6 (Min-hash) *Min-hash is a family of functions $h_\pi(K) = \min\{\pi(k) | k \in K\}$, where π is a random permutation of the universe.*

In words, min-hash is a family of functions which maps a set to a natural number. Each function does so by *1*) defining a permutation π of the universe S from which the elements of the set K are chosen (each function has its own fixed permutation), *2*) computing the index of each element of K in this permutation, and *3*) selecting the lowest index as the outcome.

It can be shown that the min-hash family is $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive as is done by e.g., Leskovec et al. (2012). Now, using this family of functions, we can continue explaining what it means to be locality sensitive. Imagine two sets A and B containing letters chosen from the set $\{a, b, \ldots, z\}$. Their Jaccard similarity

is $d_J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$. In this case, the domain of the min-hash function is $2^{\{a,b,\ldots,z\}} \setminus \emptyset$ (i.e., the min-hash function can be applied on all possible subsets, except the empty set). The range of the min-hash function (the universe $\mathcal{D}$) can be any ordered set of 26 elements, for example the numbers 1 till 26. The locality-sensitive property now means that for a min-hash function h chosen at random (i.e., for a permutation π chosen at random) [4]

$$\text{if } d(A,B) \leq d_1 \text{ then } \Pr[h(A) = h(B)] \geq 1 - d_1$$
$$\text{if } d(A,B) \geq d_2 \text{ then } \Pr[h(A) = h(B)] \leq 1 - d_2$$

Or in words, if the Jaccard distance between A and B is small, then the probability that a randomly chosen min-hash function will map A and B to the same value is higher. If the distance between A and B is large, the probability of obtaining the same value is lower.

If we know a locality sensitive hash function for a distance metric, we can use it to find approximate near neighbors. First, prepare the data structure as follows: *1*) choose a hash function at random *2*) hash all elements in the database using this hash function *3*) store all these hashes in a hashtable together with the original elements keeping duplicate entries in the hash table (essentially creating a multimap).

Now, whenever we want to perform a near neighbor search for a query q, we do the following: *1*) hash q with the hash function, *2*) locate the points in the hash table with the same hash outcome, and *3*) depending on the strategy chosen *a*) return all elements found, *b*) interrupt the search after finding a fixed number of points, or *c*) find the closest element(s) using exact distance calculations (if we are looking for nearest neighbors). Note that it might happen that the returned points are not true R-near neighbors of the query, i.e., there could be false positives. If these are not tolerable, one can at the final stage also compute all distances to the retrieved points and guarantee that all found points are true R-near neighbors. (See also the discussion on randomized algorithms in section 2.3.4).

The worst case complexity of the algorithm is $\Theta(n)$, with n the number of points in the database. This is the same worst case complexity as would be found for the straightforward algorithm from section 2.3.1. The reason for this worst case complexity is that the algorithm might return all points in the dataset if they happen to be close to the query point. However, for real world datasets, with a reasonable choice for R, the *average* case query time will be sub-linear (Andoni and Indyk, 2008).

The idea behind the algorithm presented is that if points are close to q, then they will likely be in the same bucket in the hash table. However, sometimes the likelihood ($1 - d_1$ in the case of min-hash) is not as we would want it to be. To solve this issue, a technique called amplification is used (this version of amplification is due to Leskovec et al. (2012)). The idea of amplification is to create a new

[4] This simplifies to: if $d(A,B) = d_1$ then $\Pr[h(A) = h(B)] = 1 - d_1$, in the case of Jaccard distance. This is, however, not true for all LSH families. A notable exception are LSH functions for Euclidean distance.

LSH function family based on an existing one. Each function of the new family is created by combining multiple randomly chosen LSH functions from the original family. A first amplification strategy is concatenation. To do this, concatenate r hash functions into a single one, where the outcome of the combined hash function is the concatenation of the outcomes of the original hash function[5]. For the second strategy we can hash each element (and the query) with b hash functions, resulting in multiple hash tables. Two results of the combined hash function compare equal if the results of *any* of the elementary hash functions compares equal.

Now, it can be shown that the locality sensitive hash function family constructed using b hash functions (as above) consisting of r concatenated hash functions of a (d_1, d_2, p_1, p_2)-sensitive family results in a $(d_1, d_2, 1 - (1 - {p_1}^r)^b, 1 - (1 - {p_2}^r)^b)$- sensitive locality-sensitive family of functions. Here, b and r can be adapted to the application's requirements. The effect of these parameters, for the min-hash function, can be observed from figs. 2 and 3. What we observe is that a larger r results in a steeper curve, but also pulls the curve to the left. This means that the threshold for getting points to collide is more strict and that points have to be closer to have a chance to collide at all. A large b parameter also makes the curve steeper, but pulls it to the right, meaning that also points further apart have a chance to collide.

2.3.3 LSH Forest

The LSH structure described in the previous section is intended to solve the near neighbor finding problem. As described, it is possible to use it for the nearest neighbor problem too, if one used multiple radii R and hence creates multiple indexes. This, however, has a major problem: we must have an idea about the smallest and largest R we will ever need in order to make a system with good performance. If we don't choose the smallest radius small enough, we will have queries which result in too many points to investigate. From the other end, if we don't choose the largest radius large enough, we will end up without any nearest neighbor. So, in order to overcome this problem we will have to create a lot of superfluous indexes, many of which will rarely be used in practice. Another problem is that if the points in the indexes change over time (i.e. points are added or removed), then we might have to recreate indexes in case the data characteristics change. In this subsection the LSH forest, which does away with these issues, will be introduced. The LSH Forest is due to Bawa et al. (2005) and this description is based on their work.

LSH forest uses a prefix trees or tries. We will introduce this data structure first. A trie is similar to the well-known tree data structure. However, the objects which are added to the trie have an ordered list of values chosen from a predefined alphabet. And, opposed to many trees, the trie has labeled arcs (a label being one symbol of the alphabet) instead of labels at its nodes. An object

[5] Note, the concatenation of hash outcomes is not the same as the concatenation of strings. The best way to look at the concatenation of n outcomes is as an n-tuple. Hence, the concatenated outcome (5, 16) does not compare equal to (51, 6).

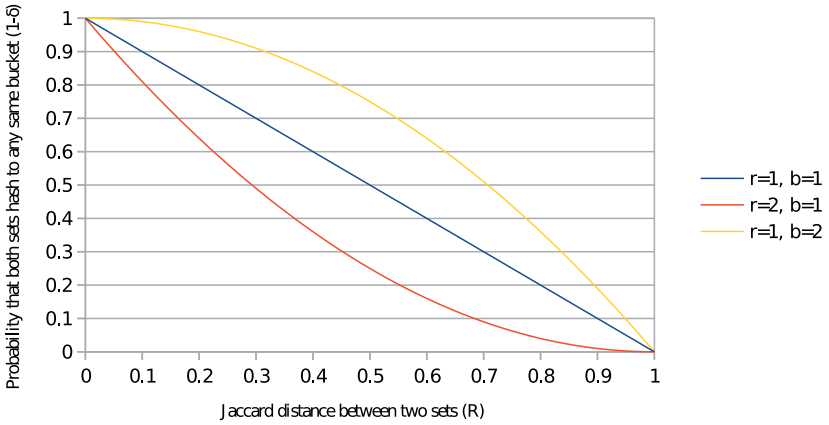


FIGURE 2 The effect of changing the r and b parameters on the probability of hashing to any same bucket $(1 - \delta)$, given a certain distance (R) for a low r and b.

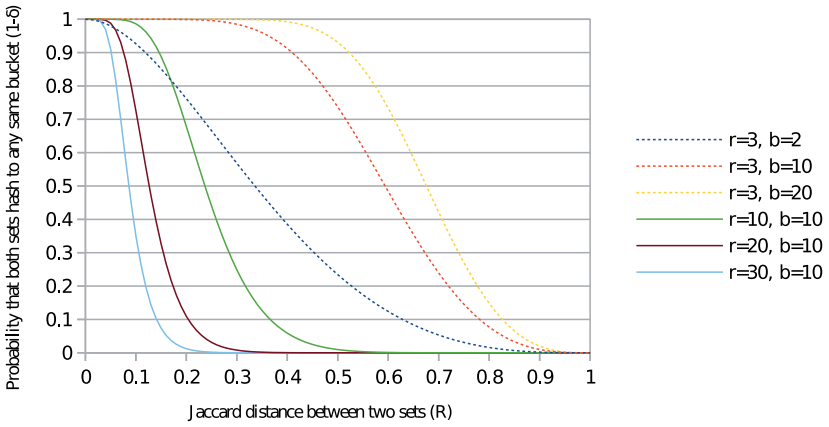


FIGURE 3 The effect of changing the r and b parameters on the probability of hashing to any same bucket $(1 - \delta)$, given a certain distance (R) for a high r and b.

is added to the trie at a leaf node, such that when following the path from the root node to the leaf, the labels on the arcs are exactly those in the list, in the same order. Further, the tree cannot have two arcs originating from the same node with the same label. An example of a trie can be found in fig. 4. In this case the alphabet is $\{a, b\}$ and objects with the following lists have been inserted: (a, a, a),(a, a, b, b),(a, b, b, a),and (b, a, b, a).

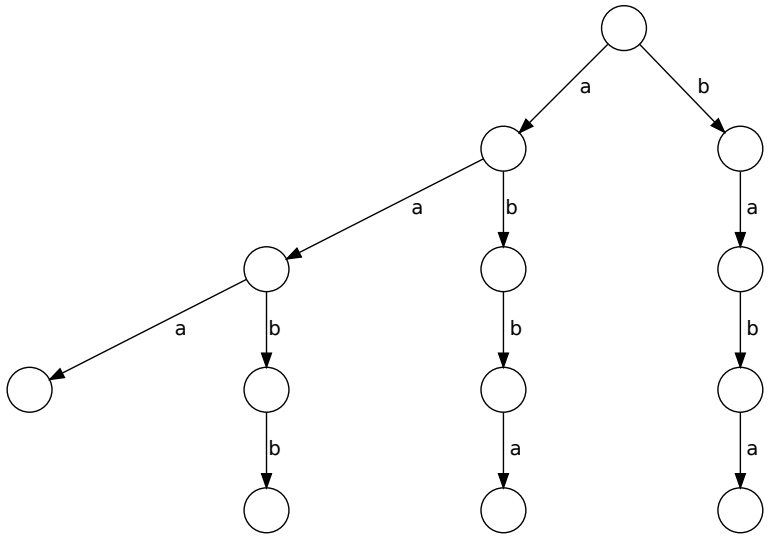


FIGURE 4 An example of a trie containing the following data : (a, a, a),(a, a, b, b),(a, b, b, a),and (b, a, b, a)

Note, that it can not be told from the illustration how many objects with, for example, list (a, a, a) have been inserted, however, we know that there is at least one. In an actual implementation the leaf node represents (and often contains a pointer to) the actual object(s). Further, the example seems to suggest that the alphabet would be limited to only two symbols. However, in theory this can be an infinite set.[6] Finally, note that the trie described in this section is somewhat specific to our needs. In general tries can have all sorts of other features like merging branches, loops, wildcards, etc. resulting in a data structure which is actually a more general directed graph.

Now, LSH forest will be introduced and we will see how it uses the trie data structure to store its actual index. LSH Forest uses the same locality-sensitive hash functions as standard LSH. However, the amplification of multiple hash functions is done differently, namely such that the length of the concatenated hash functions is dynamic and never larger than actually needed. LSH Forest maintains multiple tries (the forest) and a locality-sensitive hash function, ran-

[6] In practice, at any point in time, only a finite subset will be in use.

domly chosen from the family, is linked to each level in each of the tries, whose heights are limited to a certain maximum k_m. When a new element is to be inserted into the data structure it is inserted in each trie independently. The possible symbols on the arcs, i.e. the alphabet is chosen to be the outcome space of the LSH function used. In a trie, at each level needed, the linked hash function is evaluated and the arc with that label is either followed or created. Whether a level is needed depends on the other objects already in the trie: each of the elements, which is not at the bottom most level of the trie must be in its own leaf node. If two elements are in the same leaf node, then the linked hash function has to be evaluated for both and both will follow or create the arc of the outcome until both are in their own leaf or at the bottom of the trie.

The effect of this way of inserting data into the trie is that the concatenated hash outcome of a point can be found by concatenating the labels when following the path from the root to the leaf containing the data point. Hence, this path is only as long as needed to distinguish it from other points. This has two effects. First, it saves evaluation time since no unnecessary hashes are computed. Second, it saves memory space since only the needed hash values are stored and parts of the hash values are reused by multiple data points in case they share a common sub-path from the top of the trie.

Besides these benefits, and perhaps more importantly, the dynamic labels eliminate the need for multiple indexes since areas where points are dense will cause longer paths in the trie, while areas with low density will result in shorter paths.

In order to find near neighbors using LSH Forest, one will select these points in the tries which have the longest common prefix with the hash code computed for the query point, using the hash functions of the specific trie. For an efficient algorithm to find these points from the trie, see Bawa et al. (2005).

One issue with both the conventional LSH and LSH Forest is that when using hashing, it is not possible to distinguish between arbitrary close points. Therefore, LSH assumes a minimum distance between any two points and LSH Forest defines a maximum label length. This maximum label length is equal to the maximum height of the tree and is indicated as k_m, as mentioned above.

2.3.4 Approximate Algorithms

As mentioned in section 2.2, the size of the ontologies and the complexity of the processing will sometimes force us to accept inaccurate results from the algorithms used. Algorithms which do not always produce exact results are widely studied and the locality sensitive hashing algorithms introduced in this chapter are examples of such algorithms.

There seems to be no complete consensus on classification of algorithms which do not produce exact outcomes and some other terms get often mixed up in the discussion around these algorithms. Some definitions of terms used are the following.

- An *approximate* algorithm is an algorithm which does not necessarily pro-

duce the exact or correct answer. This term is, for example, used for algorithms with a likelihood of producing a correct result. For instance, an algorithm might produce a result and the only thing we know is that it is correct with a certain probability, say 80 percent of the time. Another example is the approximate near neighbor search algorithm, which does produce the results it is expected to with a given likelihood.

- A *Randomized* algorithm is an algorithm of which the working is influenced by a random number generator (Cormen et al., 2009, p116). Quicksort with random pivot selection is an example of a randomized algorithm (Motwani and Raghavan, 1995). In the context of optimization, these algorithms would often be called stochastic algorithms. These algorithms are not always approximate in the sense that sometimes they are designed such that they can only produce exact results. (See also the discussion about Las Vegas algorithms below.)

- An algorithm is an *approximation* if it does not necessarily provide an exact answer. But, the algorithm provides bounds on the error it can make and on its run-time (Talbi, 2009). Usually this error bound is a factor ϵ which indicates how far the reported answer can be from the real value. The algorithm is then called an ϵ-approximation. For example, an algorithm for measuring the distance between two cities might produce a result and guarantee it to be within a ten percent margin from the real value. An approximation is an approximate algorithm.

- *Heuristics* (based on Talbi (2009)) are designed to deliver satisfactory solutions within a shorter or reasonable time. There is not necessarily any guarantee on how good the solution will be nor any convergence proofs. The working of these techniques is usually demonstrated experimentally. For instance, imagine we know that the lists of numbers to be sorted are usually just the lists in reversed order. Then, we could include a first step in our algorithm which reverses the list and checks whether it is already sorted. If this is not the case then we apply a general sorting algorithm. Our algorithm will, for data with this characteristic, outperform general sorting algorithms. However, when used with data which does not have this characteristic our algorithm will waste time in this step and actually perform worse than just using the general sorting algorithm. A heuristic is an approximate algorithm.

  In optimization there is further classification into two families, namely *specific heuristics* and *metaheuristics*. The former ones are, as the name implies, specific for a certain problem. They are tailored for a specific purpose, by using known information about the problem. The latter class contains general-purpose algorithms . These can be used for a large range of optimization problems. Metaheuristics can be seen as more general guidelines for solving specific problems.

Note that an algorithm can have more than one of these properties at the same time. For example, a genetic algorithm, which we will discuss further in section 2.4, is a (meta)heuristic, approximate, and a randomized algorithm. It is, however, not an approximation because it does not provide any convergence guarantees. Another example: if we would use the near neighbor search algorithms devised in the previous sections with radii chosen in a way which we expect to work well for our data set at hand, we are creating an approximate algorithm which makes use of (problem specific) heuristics. Whether the algorithm is randomized depends on whether the selection of hash functions from the family is performed using a random number or deterministically.

In the work regarding randomized algorithms there are two sub types which often receive special attention, namely *Monte Carlo* and *Las Vegas* algorithms which are named after cities known for their casino culture. This discussion is based on Motwani and Raghavan (1995).

Monte Carlo algorithms sometimes produce results that are not correct, but a bound on the probability of getting an incorrect result is known. These algorithms are randomized and repeating the algorithm multiple times (with different random numbers) will increase the likelihood of obtaining a correct result. There are also Monte Carlo algorithms with even more interesting properties, especially these which solve decision problems (giving a binary yes – no answer). Some of these algorithms provide a one-sided error, meaning that for one of the outcomes, if it occurs, it is correct. For the sake of an example we could devise an algorithm based on a locality-sensitive hash functions as defined in definition 5. To decide whether two objects are equal we hash them with a hash function chosen at random from the family. If the hash outcomes are different, we can be hundred percent sure that the objects are not identical. Conversely, if the hash outcomes are the same we cannot yet make a conclusion. However, the more times we repeat the procedure, the more likely it becomes that objects which are different will be detected. If we would have two objects with a Jaccard distance of d, then the probability of detecting that they are different (i.e., the probability of obtaining a different hash outcome) using min-hash (see definition 6) would be d. If we would repeat this test k times, we have a likelihood of $1 - (1 - d)^k$ that we will detect that the objects are different. Concrete, with two sets having a distance of 0.05, and repeating the procedure 50 times, we will have a likelihood of 0.92 that we will detect that they are different. Applying the procedure 100 times results in a likelihood of 0.99.

Las Vegas algorithms, on the other hand, always produce the correct result. However, their running time is not constant and is exactly what is being studied. A simplified example of a Las Vegas algorithm is one which attempts to find an optimal element from a finite set when the function value of the optimal element is known. The algorithm could randomly select an element and check whether it found the right one. If so, the algorithm ends. If not, the algorithm repeats itself until it finds the solution. Obviously, the correct solution will always be found. However, the time this may take could be unbounded. This example also brings us to the next section, optimization.

2.4 Optimization

In general, optimization is the process to find the best solution(s) for a problem. Optimization problems appear often in everyday live. They could range from selecting the tasks to perform in order to finish a dissertation to finding the shortest path trough the supermarket. However, one also has to keep certain limitations in mind: besides the dissertation one also has to fulfill teaching duties and the path in the supermarket should be such that it passes trough all aisles where the needed supplies are stored. When solving optimization problems, the first thing to do is to make a model of the actual optimization problem. In most cases the model will be a simplification of the real problem at hand because the actual problem is too complicated to solve or there is just no need to solve the problem exactly. For example, in the supermarket problem it would be very hard to include information about other customers moving around into the model.

So, in short optimization is the process of finding an input to a model which results in an output with desired properties. The model captures as well as needed the setting to be optimized, including possible constraints. The nearest neighbor problem presented in section 2.3 is one example of an optimization problem. In this case we want to find the data point of the database (the input from an optimization perspective) which minimizes the distance to the query point according to the collection of data points with a given distance metric defined (the model).

Often, optimization problems get narrowed down to minimization (or maximization) problems. In mathematical terms this is formulated using an objective function f with a domain D and range R. The domain is often specified as a well known set, like the natural ($\mathbb{N}$) or real numbers ($\mathbb{R}$), and possibly constraints. In effect, the domain consists of the points from the set where the constraints are true. The range is a total ordered set, often $\mathbb{N}$ or $\mathbb{R}$.

Definition 7 (Optimal) *Given a function $f : D \to R$, an element $d \in D$ is optimal for $f \Leftrightarrow \forall\, e \in D : f(e) \leq f(d)$*

Often optimization problems are solved analytically and sometimes with brute force computation. The problem with analytical methods is, however, that these require knowledge about or put limitations on the function which is being optimized (e.g., the function has to be linear, convex, differentiable, or continuous). A brute force computation is than again limited to finite sets which do not have too many elements in them, such that computing all function values remains feasible.

In the remainder of this section we look at genetic algorithms to solve optimization problems. These methods have the benefit that they do not have the aforementioned requirements. The function does not have to be stated in analytical form nor does the domain have to be finite. On the flip side these algorithms apply heuristics which do not guarantee that the final solution will be the real global optimum of the function. It might be that a local optimum is found, and even that is not guaranteed.

For a description of multi-objective optimization, which is discussed and used in some of the articles, the reader is referred to PI (or Miettinen (1998) for a more thorough treatment).

Genetic Algorithms

Genetic algorithms (GA) are a specific group of algorithms within the wider research area of evolutionary computing. In this section we will provide a short introduction of evolutionary computing and then focus on Evolutionary Algorithms (EA), of which GA is a specific flavor. This section is based on Eiben and Smith (2003), a standard work in the field.

Evolutionary computing encompasses computing strategies which employ analogies to the Darwinian principle of survival of the fittest to find a solution for the problem at hand. Usually, the following analogies are made: the problem or model is thought of as the environment in which individual organisms (candidate solutions) reside. The quality of the solution is regarded as the fitness of the individual and is, in analogy with Darwinian principles, used as a criteria for reproduction and selection. Simulating this environment for some amount of time likely leads to more fit individuals in the population.

In nature the fitness of an organism is determined by its phenotypical appearance (i.e., the physical characteristics). These features directly interact with the environment and hence determine the chances of survival. The phenotype of an individual is determined by its genotype (i.e., its DNA). The genotype does not directly interact with the environment but, since it determines the phenotype, certain phenotypical traits will dominate others. These ideas are used in EA to find solutions for optimization problems. The way the mapping from the Darwinian theory to the actual algorithm happens depends on the particular algorithm used.

The general workings of a EA are depicted in fig. 5. At the start an *initial population* is generated. This collection of individuals is taken as the *population*. Next, the *termination criteria* are checked and the algorithm terminates if they are met. If the algorithm continues, *parents* are selected from the population. Then, two or more parent are *recombined* and subsequently *mutated* to form a member of the *offspring*. This selection, recombination, and mutation process continues until the offspring is large enough. In the end of the cycle the new population is *selected* from the offspring. In some algorithms also the individuals of the previous population can become part of the next population (the dashed line in the figure). With this new population the algorithm repeats.

The idea behind this algorithm is that over time the individuals would become closer and closer to the actual optimal of the function which is used as a fitness function. This is aided by the fact that better (closer to the unknown optimum) individuals are preferred over others.

The way individuals are represented, the recombination and mutation operators, and the parent and survivor selection are dependent on the EA flavor used. Three groups of EA are commonly identified: Genetic Algorithms (GA),

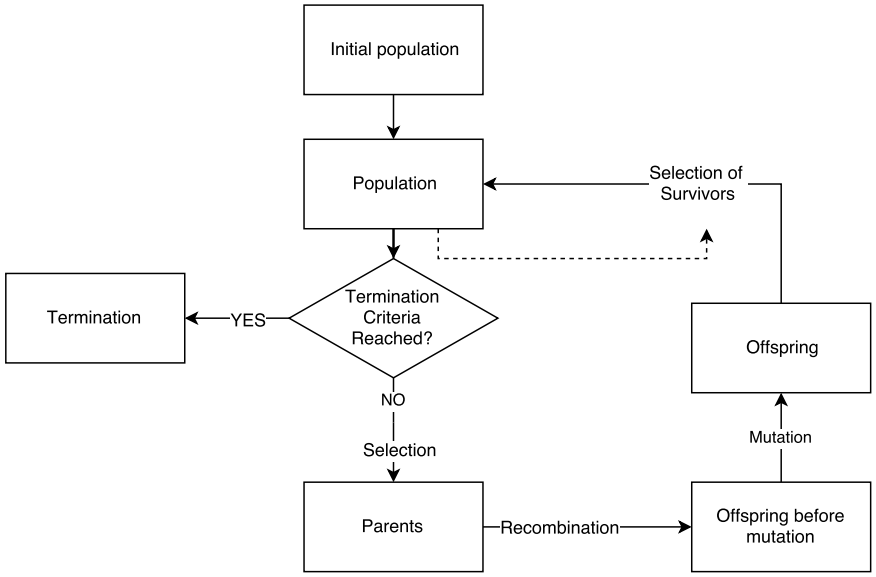


FIGURE 5 A flow-chart showing the workings of an Evolutionary Algorithm. This figure is loosely based on (Eiben and Smith, 2003, Fig. 2.2), but adds the option to select survivors directly from the previous population.

Evolution Strategies (ES), and Genetic Programming (GP). We will only discuss GA in detail since this is the one used in PVII. For the other algorithms we refer the reader to (Eiben and Smith, 2003, chapter 4 and 6). Note that often (and also in PVII) the algorithm is further tailored to the problem at hand. For this reason, the GA described here is called the *simple* or *canonical* GA in Eiben and Smith (2003).

In a simple GA, each individual's genome is represented by a bit-string (i.e., a series of zeros and ones) of a fixed length l. Parents are selected using fitness proportional selection, meaning that the higher the fitness value of the parent (i.e., the better its phenotype performs with regard to the fitness function), the more likely it will be selected. Two parents, say A and B, are combined using one point crossover. This procedure is performed by selecting an index i in the bit-string at random and creating two children. The first child receives the bits zero till $i-1$ from parent A and bits i till $l-1$ from parent B. The second child receives zero till $i-1$ from parent B and i till $l-1$ from parent A. The children created by this process then undergo a mutation, which flips bits according to a given mutation rate (the probability that a given bit flips). In the simple GA the whole population is replaced by the offspring. The termination criteria is often a fixed number of generations or a fixed number of fitness function evaluations (also called the budget).

2.5 Clustering

The general idea behind clustering is the identification of groups of objects where *a*) objects within each group have high mutual similarity, and *b*) the similarity between objects from distinct groups is low. These groups of points are called clusters. The reader might wonder whether the LSH techniques introduced in the previous section are in fact clustering algorithms. In the literature they are not considered clustering algorithms as such, mainly because they work more on the neighborhood of individual points instead of observing the dataset as whole. However, as will become clear in the articles attached to this dissertation, LSH can be used as a supporting tool for clustering.

Clustering is one of the so-called unsupervised learning techniques because it can be performed without any labeled information (i.e., training data) available to the algorithm. The field of clustering is vast and hence we will only present a limited set of concepts and techniques which are relevant to the papers in this dissertation. First, we will limit ourselves to distance-based algorithms, i.e., the clusters are created using information about the distance (see also definition 1) between the objects. These distance-based algorithms are subdivided in two large groups, based on the outcome produced. The first group of algorithms produces flat clusters, which is the same as a partition of the set of objects. A prototypical algorithm for this group would be k-Means. The second group of algorithms creates hierarchical clusterings. The result of this kind of clustering is a dendrogram as illustrated in fig. 6. Within this group two types of algorithms are distinguished depending on how the dendrogram was obtained. If a bottom-up approach is used (i.e., each object starts in its own cluster and two clusters are merged until only one cluster is left) then the approach is called an agglomerative clustering method. Alternatively, if the clusters are created top-down (i.e., starting from one large cluster, new clusters are created by slicing up a cluster into two sub-clusters until all objects are in their own cluster), then the approach is classified as a divisive method. Hierarchical clustering is discussed in more depth in PVI and PVII.

A recent treatment on clustering which presents the topic in great detail can be found from Aggarwal and Reddy (2013). This section is partially based on that text.

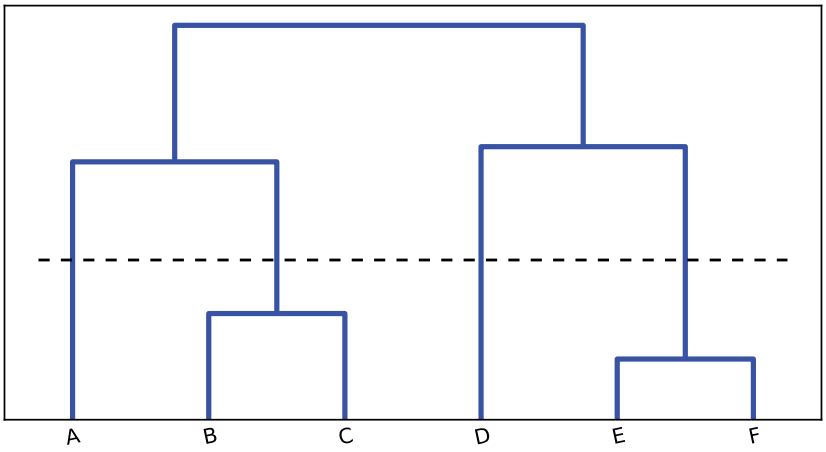


FIGURE 6 An example of a dendrogram. In this example E and F are merged together first, forming cluster EF. Then, B and C are merged, forming BC. Next, A is merged with BC, forming ABC. Next, D is merged with EF, resulting in DEF. Finally, ABC and DEF are merged together into one large cluster containing all the data points. The dashed line indicates a possible cut of the dendrogram which would results in the flat clustering consisting of the clusters A, BC, D, and EF.

3 RESEARCH CONTRIBUTIONS OF THE INCLUDED ARTICLES AND CONTRIBUTION BY THE DOCTORAL CANDIDATE

Denken, mama, dat moet ge aan mij overlaten – ik heb daar ne kop voor.

– Michael Cochez (3 years old, allegedly)

In this section an overview of the scientific contributions of the included articles is given. Further, a description of the contributions by the author of the dissertation is provided. The descriptions of the scientific contributions are intentionally left short and will not necessarily be self-evident after reading only the Foundations chapter. The reader is encouraged to first read trough the original articles, which are an integral part of the dissertation, and then return to this chapter.

3.1 Research Contributions of the Included Papers

The articles in this dissertation are all related to knowledge evolution. The Venn diagram in fig. 7 depicts the relation between the research topics introduced in the previous chapter and the papers. Note that it does not necessarily depict the relations between the research topics in a broader sense[1]. Each of the following sections describes the contributions to a topic from that figure.

Knowledge evolution

All papers in this dissertation contribute up to some extent to the knowledge evolution topic. The papers PI and PII, however, are completely focused on it. The first paper (PI) discusses about knowledge evolution from the perspective of keeping the ontology optimal. The idea is that the ideal ontology is not a static

[1] It is, for instance, not hard to find clustering research which does not belong to the LSH topic or ontology alignment research where optimization techniques are used.

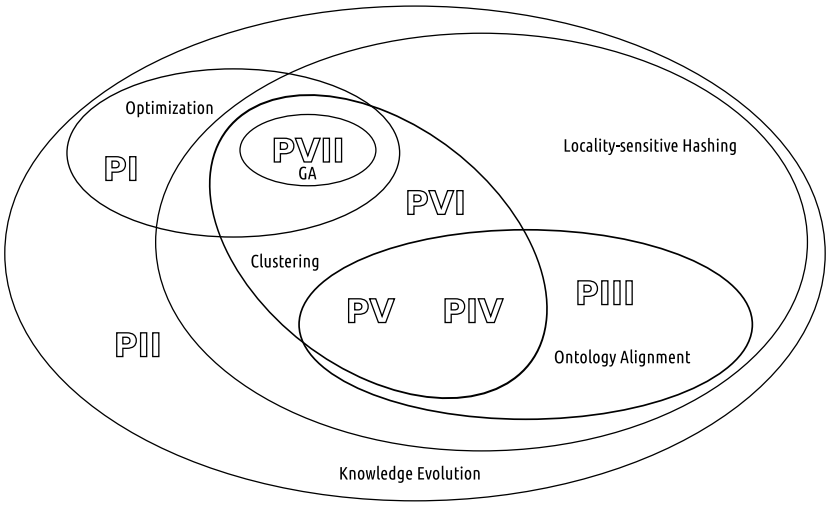


FIGURE 7 A Venn diagram visualizing how the papers relate to the topics introduced in the foundation section. A paper relates to a topic either if it is using techniques from the area , contributes new scientific knowledge to it, or both. This diagram does not depict the relation between the research topics in a broader sense. GA means Genetic Algorithms.

target, but rather a moving ideal which a system could attempt to reach. In the article we present factors which affect the quality of an ontology. Example factors include, price to build, re-usability, actual re-use, maintenance, performance in terms of memory allocation and speed, and the integration into other frameworks (see also Brewster et al. (2004) and Aruna et al. (2011)). Then, since we have the notion of quality, one can decide whether the one ontology is better than another one. However, quality is regarded as a multidimensional property with non-comparable dimensions. Hence, when defining the optimization problem, it is presented as a multi-objective one. Further, since the quality of a given ontology changes depending on the context, the optimization problem is refined to take these changes into account. We are not aware of any prior work which stated the problem of keeping ontologies optimal by means of a dynamic multi-objective optimization problem. This contribution answers research question **RQ1**.

The second article (PII), which is in the form of a book chapter provides a provocative viewpoint on how knowledge evolution could be dealt with when the amount of data handled by the system, and hence the amount of knowledge, becomes large. The whole idea of the article is new and the main contribution is that it opens a new branch of research. The crux of the paper is the following hypothesis about knowledge evolution:

> The mechanisms of knowledge evolution are very similar to the mechanisms of biological evolution. Hence, the methods and mechanisms for the evolution of knowledge could be spotted from the ones enabling the evolution of living beings.
>
> *– Ermolayev et al. (2014)*

Then, several properties of so-called knowledge organisms are presented and the analogy with natural evolution is emphasized. Another concept which is introduced in the work is knowledge tokens. These small ontological fragments are akin to food for the knowledge organisms and are the entities which carry the actual information.

The papers related to ontology matching and sowing knowledge tokens in environmental contexts (PIII, PIV PV) contribute to the knowledge evolution topic by providing ways to ingest information and connect it to the existing knowledge in the knowledge ecosystem, specifically the later two contribute by showing that LSH forest is a suitable data structure which adapts well to knowledge tokens streaming into the system. This contribution answers research question **RQ3**.

Also papers PVI and PVII can prove useful for knowledge evolution since they enable the creation of knowledge from data. Hierarchical clustering can further help in the problem of focusing which has been described in details in PII. Note also that the twister tries algorithm can be adapted to handle a stream of data. The already existing data points do not have to re-hashed to account for newly arriving data. Hence only hashes for new data (and potentially up to an equal number of colliding existing points) will have to be computed and the twisting phase will have to be executed again.

Locality-sensitive hashing

Random hyperplane hash (RHH) functions form a family of locality-sensitive hash functions. PV proposes a new schema for hashing based on the standard RHH scheme, which was named fuzzy Random Hyperplane Hashing or f-RHH. RHH only allows two outcomes for its hash function. Either a point is on the one side or on the other side of the randomly chosen hyperplane. f-RHH also allows a third outcome which is used whenever the point in question is very close to the hyperplane. Whenever a hash function gives this result, it is ignored when comparing the outcome with the outcome obtained for other points. In practice, using LSH Forest, the point is inserted in both sub-trees, which essentially means that the outcome will be ignored for all future comparisons. As part of this new hashing scheme, an efficient way of deciding whether a point is close to a hyperplane is devised and proven. This is necessary because it would be too expensive to compute the actual angle between the vector from the origin to the point in question and the hyperplane. Instead, it is shown that it suffices to compute the dot product of the normalized vector with the normal vector on the hyperplane and compare the obtained number with a predefined constant. This is remarkable since the computation of a dot product is needed in any case for the computation of standard RHH, and hence the only overhead for the computation of this new scheme is that the points have to be normalized. This normalization has to happen only once for all further hash evaluations.

A minor contribution to the LSH topic is made in PVI. We introduce a new hash function family based on existing locality-sensitive families and prove that this new family is locality-sensitive for the average distance. The family does at

least exist for average Jaccard, cosine, and Hamming distance computed between two sets of points.

Ontology Matching

PIII provides a new way of matching very large ontologies. The matching is performed using techniques borrowed from information retrieval and would classify among the syntactic, element-level approaches, and more specific among the string-based approaches. However, it is combined with fast language-based techniques, like tokenization, stemming, and stop word elimination. The experimental evaluation showed that this approach is fast, but a reduced quality of matching has to be taken in return. It was further noticed that it is better to keep the labels separate and match them as such instead of merging the contents of multiple labels together. This contribution answers research question **RQ2**

PIV and its extension PV apply the same technique as used in PIII, but use the more adaptive LSH forest instead of the standard LSH scheme. The experimental results shows similar results with regards to precision and recall, but the algorithm performs the matching tasks roughly seven times faster. This gain is mainly due to the LSH Forest which uses far fewer evaluations of the hash functions in comparison to the standard LSH algorithm. Also important is that the forest only used about 10% of the memory used by the standard LSH scheme.

Clustering

The largest contributions to the clustering research are to be found in papers PVI and PVII. The first article introduced the twister tries data structure and algorithm. This algorithm is able to perform an agglomerative hierarchical clustering in linear time using a linear amount of memory. The clusterings created by the algorithm are comparable to the exact AHC algorithms. This paper received the Reproducible label awarded by the reproducibility committee of the SIGMOD 2015 conference. See http://db-reproducibility.seas.harvard.edu/papers/ for more information.

The second paper related to twister tries provides a somewhat different kind of algorithm. Instead of proving a linear running time, the algorithm provides a likely better result the longer it runs. This is achieved by a genetic algorithm which attempts to improve the tries used in the original twister tries algorithm.

These two papers answer research question **RQ4**.

Another contribution of PVII is a proof that the Joining Distance Ratio (which was proposed by Kull and Vilo (2008)) has a time complexity of $O(n^2)$, showing that it would become infeasible to compute for very large instances and definitely too expensive as an evaluation function for the genetic algorithm (see also the next section on optimization and GAs).

PV (and the conference paper PIV) provide a marginal contribution to the field of clustering. Hence, the techniques for connecting knowledge tokens described in these papers could be applied as, very specific, clustering algorithms.

Optimization and Genetic Algorithms

PI provides a very minor contribution to the field of dynamic optimization. The idea of context dependent optimality is new, but should only be considered a marginal contribution since anyone in the field could come up with this sort of new notation.

PVII provides experiments showing that surrogates can be used instead of performing exact calculations when doing the evaluation of the objective function for the individuals in a genetic algorithm. This technique is not new as such, but the specific surrogates used are novel, hence this is only a small contribution.

3.2 Contributions by the Doctoral Candidate

Publications

PI: The contribution of the author of this dissertation to this article was major. He independently developed the concepts presented in the paper and did most of the writing work. The co-author was in a more passive role, providing feedback and suggestions.

PII: The first author of this book chapter (Vadim Ermolayev) clearly contributed the lion's share of the work. Also the contribution of Vagan Terziyan was sizable, and mainly focused around the 3Co+3F theme presented in the chapter. The contribution of the third author was limited to parts of section 1.4. The author of the dissertation mainly contributed to the fifth section (1.5) of the chapter about the evolutionary model and near-independently provided the section related to the fitness of Knowledge Organisms and Related Ontologies.

PIII: As a sole author of the paper, all contributions are made by the doctoral candidate.

PIV and PV: The candidate provided the main idea for the paper and wrote the first draft version of the paper. Both coauthors provided significant help in the writing of this paper. Vadim Ermolayev and Vagan Terziyan mainly focused on aspect related to Evolving Knowledge Ecosystems, while the candidate worked on the LSH Forest and experiments. The f-RHH which was added to the extended article was conceived of by the applicant based on discussions with Vagan Terziyan on algorithms intentionally making mistakes during their operations, in the hope to improve results.

PVI: The candidate invented the twister tries data structure and algorithm as well as the locality-sensitive family for average distance. Further, he came up with the optimizations used and performed the theoretical analysis (except for the correctness analysis which was contributed by Mou Hao). The candidate also performed the experimental evaluation, assisted by his coauthor who did the majority of the work for dendrogram comparisons.

PVII: The author of this dissertation was the main author of this paper. He conceived of the initial idea and planned the paper. He devised the different surrogates and performed all experimentation related to the paper. Also the proof of the JDR complexity is entirely his work. The main contribution by Ferrante Neri is the text about the used evolutionary algorithm and some ideas related to measuring surrogate quality.

Implementations

The candidate has written the majority of the source code needed in experiments for the publications. The only exception is the dendrogram evaluation code written by Mou Hao in the scope of PVI.

A rough estimation learns that about 13,000 lines where used for the evaluation code of the papers PIII, PIV and PV (large parts of the code were re-usable, part of this code consists of experiments which did not make it to any paper). Another 9,000 lines of code were written for the evaluations of the twister tries related articles PVI and PVII (again, some of the code was reusable). The code was mainly written in Java, but also bash, python, and gnuplot were used. Part of the code is available from the software page of the author at http://users.jyu.fi/~miselico/software/.

4 CONCLUSION AND OUTLOOK

> People who are right a lot of the time are people who often change their minds.
>
> *– Jeff Bezos*

Whenever the state of the environment changes, its description, used in context aware applications must change accordingly. However, since the environment and the rate of change are vast, it is impossible to represent the environment or process the changes in an exact way. Hence, we will have to accept that only approximate representations and approximate processing are a feasible alternative. In this dissertation we presented several new research directions and advances related to this timely topic.

A first contribution is the formulations of an optimization problem for keeping an ontology in optimal shape while the context in which it is used is changing over time. It turned out, however, that this problem is so complex that is cannot be solved exactly. Future work could point out whether parts of the objectives could be dealt with and what the practical result would be. One concept which would be enabled by this is a self-optimizing ontology, where the ontology becomes a proactive entity able to reconfigure itself to become more suitable for its environment. Because of the fact that the optimization problem cannot be solved exactly the idea of evolving knowledge ecosystems (EKE) is important and this is the second contribution of the dissertation. This idea, including the hypothesis that the evolution mechanisms of knowledge might be spotted from nature, opens a new direction of research and suggests what kind of models might be useful for knowledge representation in the future. The third related contribution showed that locality-sensitive hashing can be used for ontology alignment and specifically that LSH Forest would be a good candidate algorithm for knowledge token sowing in an EKE. The fourth major contribution is related to scalability of clustering algorithms. The dissertation reports on a new approach to scale hierarchical clustering with average linkage to large datasets.

Besides these, the dissertation also provided new scientific findings related to Locality-sensitive hashing. First, a it was shown how a family of LSH functions can be created for average distance between sets of objects between which the

distance can be measure using Jaccard, cosine, or Hamming distance. Second, a new LSH schema was developed which improves random hyperplane hashing in certain cases by assigning both possible hash outcomes to an object is it happens to be close to the random hyperplane. The research question are answered by the contributions in the research papers as follows:

RQ1: What does it mean for an ontology to be optimal?
An ontology is optimal if it maximizes its utility with regard to the context-dependent multi-objective problem as stated in PI. This context is determined by the system in which the ontology is used and changes over time.

RQ2: Can min-hashing, as applied in information retrieval, help in the alignment of ontologies?
Yes, for large ontologies min-hashing can significantly increase the speed of the matching. When combined with fast language-based techniques like tokenization, stemming, and stop word removal, a matcher using min-hash can provide results of reasonable quality.

RQ3: Is the same min-hashing also helpful when sowing knowledge tokens in a knowledge ecosystem?
Yes, at least when combined with the LSH Forest data structure. It was shown that the LSH Forest data structure provides the features needed for knowledge token sowing like focusing, filtering, and forgetting. Also the three other aspects (Contextualizing, compressing, and connecting) are covered.

RQ4: Can a hierarchical clustering algorithm be adapted to large scale data if we allow an error?
Yes. The twister tries algorithm is a hierarchical clustering algorithm which can cluster dataset much larger than what is possible with traditional exact algorithms. The main indication is its linear runtime and memory usage. The twister tries will often not create the same dendrogram as produced by an exact algorithm, but it has been shown that the outcome is significantly similar.

There are some obvious limitations to the findings presented. Regarding the idea of EKE, the main limitation is that all of this research is at a very initial stage. It is mainly a vision that this kind of system might be the next step towards something greater. A somewhat similar critique can be formulated regarding the optimization problem which is formulated in PI. Many would agree that if this problem is solvable, it would help us forward. But, the current state of affairs is that the problem is not solvable because it would require computational resources beyond our current capabilities. With regard to the ontology matching schemes and clustering algorithms one can bring in the argument that they have not been tested widely enough to declare them generally applicable. This is a fair point, but is partially countered by the theoretical analysis. The clustering approach presented is currently being tested in bio-informatics to see whether it results in semantically reasonable clusters, while it is still under investigation whether it would be possible to adapt the hashing schema to allow Euclidean distance to

be used. Another open question is whether it is possible to find a good way to evaluate the quality of large dendrograms, or at least perform a good comparison in reasonable time.

One question to be answered after reading this dissertation is whether Big Knowledge has been Tamed, as the title might suggest. The answer to this question would be a resounding "no". Some parts of the puzzle have been put in place, but many others are still not where they should be or we do not even know that they exist. Only at a later point will it become clear whether the evolving knowledge ecosystem ideas can stand the test of time and maybe some day it will be possible to prove or disprove its main hypothesis. One might well say that the knowledge in the field is still evolving and hence, if one wants to write a dissertation addressing all the challenges related to taming big knowledge evolution, one will never finish...

OTHER ACTIVITIES SUPPORTING THE DOCTORAL STUDIES

> Quidquid latine dictum sit, altum videtur.
>
> *– Nomen Nescio*

Besides the work presented in this dissertation and 60 ECTS credits[1], the candidate also performed the following other academic work during his doctoral and master study period:

Participation in Projects

The doctoral candidate participated in the following research projects during his studies.

UBIWARE project 2007-2010: The project's goal was the creation of an innovative middleware supporting complex self-managed industrial systems. The nature of the components managed by the system varies from smart sensors and actuators to web services and humans. A Multi-Agent System was used as a foundation where the beliefs, desires, intentions, and even the communication is performed using S-APL (the Semantic Agent Programming Language).

Cloud software program SHOK 2010-2013: A program directed towards the creation of new business models, lean software principles, and an open infrastructure for a cloud computing environment.

Need4Speed SHOK 2014-2015: An environment for experimenting with real-time business models based on customer insight.

Scientific publications not included in the dissertation

During his studies the author also worked on the following publications which are not included into the dissertation:

Chernov, S., Cochez, M. & Ristaniemi, T. 2015. Anomaly detection algorithms for the sleeping cell detection in LTE networks. In Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st. IEEE, 1–5.

Cochez, M., Helin, S. & Chen, J. 2013a. Developing cloud software : algorithms, applications, and tools. In I. Porres, T. Mikkonen & A. Ashraf (Eds.) Developing cloud software : algorithms, applications, and tools. TUCS Turku Center for Computer Science, general publication series.

[1] The equivalent of one year of full-time study

Cochez, M., Isomöttönen, V., Tirronen, V. & Itkonen, J. 2013b. How do computer science students use distributed version control systems? In V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky & G. Zholtkevych (Eds.) 9th International Conference, ICTERI 2013, Kherson, Ukraine, June 19-22, 2013, Revised Selected Papers. Cham: Springer International Publishing, 210–228. doi:10.1007/978-3-319-03998-5_11. ⟨URL:http://dx.doi.org/10.1007/978-3-319-03998-5_11⟩.

Cochez, M., Isomöttönen, V., Tirronen, V. & Itkonen, J. 2013c. The use of distributed version control systems in advanced programming courses. In ICTERI 2013 - ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Aachen: CEUR Workshop Proceedings (1000), 221–235. ⟨URL:http://ceur-ws.org/Vol-1000/ICTERI-2013-p-221-235.pdf⟩.

Cochez, M., Periaux, J., Terziyan, V., Kamlyk, K. & Tuovinen, T. 2014. Evolutionary cloud for cooperative UAV coordination. Reports of the Department of Mathematical Information Technology, Series C. Software and Computational Engineering, No. C 1.

Cochez, M. 2012. Semantic agent programming language: use and formalization. University of Jyväskylä. Master's Thesis, 92pp.

Isomöttönen, V., Tirronen, V. & Cochez, M. 2013. Issues with a course that emphasizes self-direction. In Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education. New York, NY, USA: ACM. ITiCSE '13, 111–116. doi:10.1145/2462476.2462495. ⟨URL:http://doi.acm.org/10.1145/2462476.2462495⟩.

Isomöttönen, V. & Cochez, M. 2014. Challenges and confusions in learning version control with git. In V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky & G. Zholtkevych (Eds.) 10th International Conference, ICTERI 2014, Kherson, Ukraine, June 9-12, 2013, Revised Selected Papers. Cham: Springer International Publishing, 178–193. doi:10.1007/978-3-319-13206-8_9. ⟨URL:http://dx.doi.org/10.1007/978-3-319-13206-8_9⟩.

Khriyenko, O. & Cochez, M. 2011. Open environment for collaborative cloud ecosystems. In CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization, 147–153.

Paggi, H. & Cochez, M. 2014. Use of a semantic language to reduce the indeterminacy in agents communication. In Mathematics and Computers in Sciences and in Industry (MCSI), 2014 International Conference on. IEEE, 281–287.

Paggi, H. & Cochez, M. 2015. Indeterminacy reduction in agent communication using a semantic language. WSEAS TRANSACTIONS on SYSTEMS 14, 77–89.

Terziyan, V., Nikitin, S., Nagy, M., Khriyenko, O., Kesäniemi, J., Cochez, M. & Pulkkis, A. 2010. UBIWARE Platform Prototype v 3.0. Agora Centre, University of Jyväskylä. Technical Report (Deliverable D3.3), 45pp. (UBIWARE Tekes Project).

Courses taught

The following courses were taught independently by the author:

Introduction to service oriented architectures (SOA) and cloud computing: 2013, 2014, 2015 – During this course the student got an introduction to technologies used in SOA and cloud computing settings.

Service oriented architectures and cloud computing for developers: 2013, 2014, 2015 – This course was a follow-up course of the TIES456 course. Students worked individually on more advanced tasks related to the topics from the basic course.

Service oriented architectures and cloud computing: 2012 – old form of the two courses above.

Big data engineering: 2014, 2015 – Multiple topics related to Big Data were be studied. Students will get acquainted to large data sets and streaming. Some storage and processing algorithms were studied and hardware related issues discussed. The gathered knowledge was then applied on real world data sets.

Agent technologies for developers: 2014, 2016 – The course is about practical use of distributed AI methods. More concretely of multi-agent technologies, for the development of complex cooperating software systems.

Design of agent-based systems, Part II: 2013 – old form of the course above.

Besides the classroom teaching activity, the author was also a main supervisor of three master thesis workers who already finalized their work. Several others are close to finalizing their master thesis.

Conference and Journal Review

The author was a programming committee member for the following scientific forums:

- DEIS workshop 2012
- ICTERI 2013, 2014, 2015, 2016
- WIMS 2014, 2015, 2016
- Science of Computer Programming Journal (Elsevier), reviewer for Special issue on Systems development by means of semantic technologies

Other Activities

The candidate has also held the following positions of trust during his years as a doctoral student:

- Member of the faculty council – 1.1.2014-31.12.2017
- Member of the planning group of the Web Intelligence and Service Engineering (WISE) master program – 2012-2016.
- Member of WISE selection committee – 2013, 2014, 2015, 2016.
- Member of the working groups for curriculum development (2014–2017) in the areas of computation (applied mathematics, data analysis, etc.) and technology (software engineering, mobile systems, sensor networks, games, and gamification) – 2013.

YHTEENVETO (FINNISH SUMMARY)

Informaatio ja siitä johdettu tieto eivät ole staattisia. Sen sijaan informaatio muuttuu jatkuvasti, ja meidän kyky ja halu sisäistää informaatiota vaikuttaa ymmärrykseemme siitä. Jos verrataan tietokonetta ja ihmistä, huomataan että tietokone ei pysty havainnoimaan samalla tavalla kuin ihminen. Vaikuttaa siltä, että kone ei voi oikeasti ymmärtää asioiden merkitystä. Toisaalta tietokoneet ovat ylivoimaisia, jos ajatellaan laskentakykyä. Yksi asia, joka erottaa ihmisen koneesta, on että silloin kun ihminen yrittää ymmärtää maailmaa, hän tekee usein virheitä, unohtaa asioita tai päättää keskittyä vain tiettyihin asioihin. Toisin sanoen vaikuttaa siltä, että ihminen voi suoriutua tietoon liittyvistä tehtävistä paremmin kuin kone, vaikka hän käyttäytyisi osittain satunnaisesti.

Eräs tietojenkäsittelytieteen tutkimushaara tutkii, mitä tapahtuu, jos algoritmeissa sallitaan satunnaisuutta tai epätarkkuutta. Englannin kielessä tällaisista algoritmeista käytetään termiä *approximate algorithms*. Näiden algoritmien etu on, että ne ovat usein nopeampia kuin tarkat algoritmit, mutta riskinä on, että ne tuottavat silloin tällöin vääriä vastauksia. Tästä syystä niiden käyttö on syytä rajoittaa vain tilanteisiin, joissa pienet virheet silloin tällöin eivät aiheuta suuria ongelmia. Toisaalta, jos väärän vastauksen todennäköisyys on pienempi kuin tietokoneella tapahtuvien muistivirheiden todennäköisyys, silloin nämä algoritmit ovat yhtä käyttökelpoisia kuin tarkatkin algoritmit. On myös havaittu, että algoritmeille syötetyt parametrit ja muu data yleensä sisältävät jonkin verran epätarkkuuksia. Tällöin lisävirhe, joka aiheutuu epätarkan algoritmin käytöstä, voi olla käytännössä merkityksetön.

Väitöskirjassa tutkitaan, miten sekä tunnettuja että uusia epätarkkoja algoritmeja käytetään tiedon löytämiseen ja sen evoluutioon. Väitöskirjan tärkeimmät kontribuutiot liittyvät ontologioiden optimaalisuuteen ja yhteensovittamiseen, tiedon ekosysteemeihin ja hierarkkiseen klusterointiin.

REFERENCES

Aggarwal, C. C. & Reddy, C. K. 2013. Data clustering: algorithms and applications. CRC Press.

Andoni, A. & Indyk, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51 (1), 117–122. doi:10.1145/1327452.1327494. ⟨URL:http://doi.acm.org/10.1145/1327452.1327494⟩.

Aruna, T., Saranya, K. & Bhandari, C. 2011. A survey on ontology evaluation tools. In Process Automation, Control and Computing (PACC), 2011 International Conference on, 1 -5. doi:10.1109/PACC.2011.5978931.

Baader, F. 2003. The description logic handbook: Theory, implementation and applications. Cambridge university press.

Bachman, C. W. 1975. Trends in database management: 1975. In Proceedings of the May 19-22, 1975, National Computer Conference and Exposition. New York, NY, USA: ACM. AFIPS '75, 569–576. doi:10.1145/1499949.1500065. ⟨URL:http://doi.acm.org/10.1145/1499949.1500065⟩.

Bawa, M., Condie, T. & Ganesan, P. 2005. LSH forest: self-tuning indexes for similarity search. In Proceedings of the 14th international conference on World Wide Web. ACM, 651–660.

Brewster, C., Alani, H., Dasmahapatra, S. & Wilks, Y. 2004. Data driven ontology evaluation. In International Conference on Language Resources and Evaluation (LREC 2004). ⟨URL:http://oro.open.ac.uk/20045/⟩.

Broder, A. Z. 1997. On the resemblance and containment of documents. In Compression and Complexity of Sequences 1997. Proceedings. IEEE, 21–29.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing. New York, NY, USA: ACM. STOC '02, 380–388. doi:10.1145/509907.509965. ⟨URL:http://doi.acm.org/10.1145/509907.509965⟩.

Chernov, S., Cochez, M. & Ristaniemi, T. 2015. Anomaly detection algorithms for the sleeping cell detection in LTE networks. In Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st. IEEE, 1–5.

Cochez, M., Helin, S. & Chen, J. 2013a. Developing cloud software : algorithms, applications, and tools. In I. Porres, T. Mikkonen & A. Ashraf (Eds.) Developing cloud software : algorithms, applications, and tools. TUCS Turku Center for Computer Science, general publication series.

Cochez, M., Isomöttönen, V., Tirronen, V. & Itkonen, J. 2013b. How do computer science students use distributed version control systems? In V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky & G. Zholtkevych

(Eds.) 9th International Conference, ICTERI 2013, Kherson, Ukraine, June 19-22, 2013, Revised Selected Papers. Cham: Springer International Publishing, 210–228. doi:10.1007/978-3-319-03998-5_11. ⟨URL:http://dx.doi.org/10.1007/978-3-319-03998-5_11⟩.

Cochez, M., Isomöttönen, V., Tirronen, V. & Itkonen, J. 2013c. The use of distributed version control systems in advanced programming courses. In ICTERI 2013 - ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Aachen: CEUR Workshop Proceedings (1000), 221–235. ⟨URL:http://ceur-ws.org/Vol-1000/ICTERI-2013-p-221-235.pdf⟩.

Cochez, M. & Mou, H. 2015. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In Proceedings of the 2015 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM. SIGMOD '15. doi:10.1145/2723372.2751521.

Cochez, M. & Neri, F. 2015. Scalable hierarchical clustering: Twister tries with a posteriori trie elimination. In Computational Intelligence, 2015 IEEE Symposium Series on. IEEE, 756–763. doi:10.1109/SSCI.2015.12.

Cochez, M., Periaux, J., Terziyan, V., Kamlyk, K. & Tuovinen, T. 2014. Evolutionary cloud for cooperative UAV coordination. Reports of the Department of Mathematical Information Technology, Series C. Software and Computational Engineering, No. C 1.

Cochez, M., Terziyan, V. & Ermolayev, V. 2015. Balanced large scale knowledge matching using LSH forest. In J. Cardoso, F. Guerra, G.-J. Houben, M. A. Pinto & Y. Velegrakis (Eds.) Semantic Keyword-based Search on Structured Data Sources: First COST Action IC1302 International KEYSTONE Conference, IKC 2015, Coimbra, Portugal, September 8-9, 2015. Revised Selected Papers, Vol. 9398. Springer International Publishing. Lecture Notes in Computer Science, 36–50. doi:10.1007/978-3-319-27932-9_4. ⟨URL:http://dx.doi.org/10.1007/978-3-319-27932-9_4⟩.

Cochez, M., Terziyan, V. & Ermolayev, V. 2016. Large Scale Knowledge Matching with Balanced Efficiency-Effectiveness Using LSH Forest. (Submitted to LNCS Transactions on Computational Collective Intelligence Journal).

Cochez, M. & Terziyan, V. 2012. Quality of an ontology as a dynamic optimisation problem. In V. E. et. al (Ed.) ICTERI 2012 - ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Aachen: CEUR Workshop Proceedings (848), 249–256. ⟨URL:http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-DEIS-paper-1-p-249-256.pdf⟩.

Cochez, M. 2012. Semantic agent programming language: use and formalization. University of Jyväskylä. Master's Thesis, 92pp.

Cochez, M. 2014. Locality-sensitive hashing for massive string-based ontology matching. In Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on, Vol. 1. IEEE, 134–140.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. 2009. Introduction to algorithms (Third Edition edition). MIT press Cambridge.

Eiben, A. E. & Smith, J. E. 2003. Introduction to evolutionary computing (1st edition). Springer. Natural Computing Series.

Ermolayev, V., Akerkar, R., Terziyan, V. & Cochez, M. 2014. Big data computing. In R. Akerkar (Ed.) Big Data Computing. Taylor & Francis group - Chapman and Hall/CRC, 3–55.

Euzenat, J. & Shvaiko, P. 2013. Ontology matching. Springer.

Grady, N. & Chang, W. 2015. NIST Big Data Interoperability Framework. National Institute of Standards and Technology. ⟨URL:http://dx.doi.org/10.6028/NIST.SP.1500-1⟩. (retrieved march 2016).

Gruber, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing? International Journal of Human-Computer Studies 43 (5–6), 907 - 928. doi:http://dx.doi.org/10.1006/ijhc.1995.1081. ⟨URL:http://www.sciencedirect.com/science/article/pii/S1071581985710816⟩.

van Harmelen, F. & McGuinness, D. 2004. OWL Web Ontology Language Overview. W3C. W3C Recommendation. (http://www.w3.org/TR/2004/REC-owl-features-20040210/).

Hartung, M., Terwilliger, J. & Rahm, E. 2011. Recent advances in schema and ontology evolution. In Schema matching and mapping. Springer, 149–190.

Hepp, M. 2007. Possible ontologies: How reality constrains the development of relevant ontologies. Internet Computing, IEEE 11 (1), 90–96.

Horrocks, I., Patel-Schneider, P. F. & van Harmelen, F. 2003. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1 (1), 7–26. ⟨URL:download/2003/HoPH03a.pdf⟩.

Isomöttönen, V., Tirronen, V. & Cochez, M. 2013. Issues with a course that emphasizes self-direction. In Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education. New York, NY, USA: ACM. ITiCSE '13, 111–116. doi:10.1145/2462476.2462495. ⟨URL:http://doi.acm.org/10.1145/2462476.2462495⟩.

Isomöttönen, V. & Cochez, M. 2014. Challenges and confusions in learning version control with git. In V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky & G. Zholtkevych (Eds.) 10th International Conference, ICTERI 2014,

Kherson, Ukraine, June 9-12, 2013, Revised Selected Papers. Cham: Springer International Publishing, 178–193. doi:10.1007/978-3-319-13206-8_9. ⟨URL:http://dx.doi.org/10.1007/978-3-319-13206-8_9⟩.

Ivanova, V., Lambrix, P. & Åberg, J. 2015. Requirements for and evaluation of user support for large-scale ontology alignment. In The Semantic Web. Latest Advances and New Domains. Springer, 3–20.

Karp, P. D. 1993. The Design Space of Frame Knowledge Representation Systems. SRI International Artificial Intelligence.

Khriyenko, O. & Cochez, M. 2011. Open environment for collaborative cloud ecosystems. In CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization, 147–153.

Klein, M. C. & Fensel, D. 2001. Ontology versioning on the semantic web. In SWWS, 75–91.

Kull, M. & Vilo, J. 2008. Fast approximate hierarchical clustering using similarity heuristics. BioData mining 1 (1), 9. doi:10.1186/1756-0381-1-9.

Lee, D. 2016. Artificial intelligence: Google's AlphaGo beats Go master Lee Sedol. ⟨URL:http://www.bbc.com/news/technology-35785875⟩.

Leskovec, J., Rajaraman, A. & Ullman, J. D. 2012. Mining of massive datasets, chapter 3 Finding Similar Items. Cambridge University Press, 71–128. ⟨URL:http://infolab.stanford.edu/~ullman/mmds.html⟩.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C. & Hung Byers, A. 2011. Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute. ⟨URL:http://www.mckinsey.com/business-functions/business-technology/our-insights/big-data-the-next-frontier-for-innovation⟩. (retrieved march 2016).

Maracine, V. & Scarlat, E. 2009. Dynamic knowledge and healthcare knowledge ecosystems. Electronic Journal of Knowledge Management 7 (1), 99–110.

Marshall, C. C. & Shipman, F. M. 2003. Which semantic web? In Proceedings of the fourteenth ACM conference on Hypertext and hypermedia. ACM, 57–66.

Miettinen, K. 1998. Nonlinear multiobjective optimization, Vol. 12. Springer US. International Series in Operations Research & Management Science. doi:10.1007/978-1-4615-5563-6. ⟨URL:http://www.springer.com/gp/book/9780792382782⟩.

Motwani, R. & Raghavan, P. 1995. Randomized algorithms. Cambridge University Press.

Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A. & van Harmelen, F. 2009. Marvin: Distributed reasoning over large-scale semantic web data. Web Semantics: Science, Services and Agents on the World Wide Web 7 (4), 305–316.

Paggi, H. & Cochez, M. 2014. Use of a semantic language to reduce the indeterminacy in agents communication. In Mathematics and Computers in Sciences and in Industry (MCSI), 2014 International Conference on. IEEE, 281–287.

Paggi, H. & Cochez, M. 2015. Indeterminacy reduction in agent communication using a semantic language. WSEAS TRANSACTIONS on SYSTEMS 14, 77–89.

Rahm, E. & Bernstein, P. A. 2001. A survey of approaches to automatic schema matching. the VLDB Journal 10 (4), 334–350.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P. & Sutphen, S. 2007. Checkers is solved. science 317 (5844), 1518–1522.

Searle, J. R. 1980. Minds, brains, and programs. Behavioral and Brain Sciences 3, 417–424. doi:10.1017/S0140525X00005756. ⟨URL:http://journals.cambridge.org/article_S0140525X00005756⟩.

Shvaiko, P. & Euzenat, J. 2013. Ontology matching: state of the art and future challenges. Knowledge and Data Engineering, IEEE Transactions on 25 (1), 158–176.

Talbi, E.-G. 2009. Metaheuristics: from design to implementation, Vol. 74. John Wiley & Sons.

Terziyan, V., Nikitin, S., Nagy, M., Khriyenko, O., Kesäniemi, J., Cochez, M. & Pulkkis, A. 2010. UBIWARE Platform Prototype v 3.0. Agora Centre, University of Jyväskylä. Technical Report (Deliverable D3.3), 45pp. (UBIWARE Tekes Project).

Tesauro, G., Gondek, D. C., Lenchner, J., Fan, J. & Prager, J. M. 2013. Analysis of watson's strategies for playing jeopardy! Journal of Artificial Intelligence Research 47, 205-251. doi:10.1613/jair.3834. ⟨URL:http://dx.doi.org/10.1613/jair.3834⟩.

Urbani, J., Kotoulas, S., Oren, E. & Harmelen, F. 2009. Scalable distributed reasoning using mapreduce. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta & K. Thirunarayan (Eds.) The Semantic Web - ISWC 2009: 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg. ISWC 2009, 634–649. doi:10.1007/978-3-642-04930-9_40. ⟨URL:http://dx.doi.org/10.1007/978-3-642-04930-9_40⟩.

Ventrone, V. 1991. Semantic heterogeneity as a result of domain evolution. ACM SIGMOD Record 20 (4), 16–20.

Wang, J., Shen, H. T., Song, J. & Ji, J. 2014. Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927.

Wood, D., Lanthaler, M. & Cyganiak, R. 2014. RDF 1.1 Concepts and Abstract Syntax. W3C. W3C Recommendation. (http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/).

OWL Working Group 2012. OWL 2 Web Ontology Language Document Overview. W3C. W3C Recommendation. ⟨URL:http://www.w3.org/TR/2012/REC-owl2-overview-20121211/⟩.

ORIGINAL PAPERS

PI

QUALITY OF AN ONTOLOGY AS A DYNAMIC OPTIMISATION PROBLEM

by

Michael Cochez and Vagan Terziyan 2012

ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Proceedings of the 8th International Conference ICTERI 2012, CEUR Workshop Proceedings (Vol-848). Aachen: RWTH Aachen

Quality of an Ontology as a Dynamic Optimisation Problem

Michael Cochez and Vagan Terziyan

Department of Mathematical Information Technology
University of Jyväskylä
P.O. Box 35 (Agora), 40014, Jyväskylä, Finland
firstname.lastname@jyu.fi

Abstract. The Semantic Web is a proposal from the World Wide Web Consortium aimed at solving problems like data integration and application interoperability. To reach these goals several languages for the representation of semantic data have been proposed. One of the essential concepts behind semantic data is that the data is according to a certain ontology. However, the goals of the semantic web seem challenged because it seems essential for its working that ontologies are agreed upon and shared. This work-in-progress paper describes a first step to solving these problems. When an ontology is missing or only partially known a system might try to make an approximation of the missing part of the ontology. The quality of this estimated ontology will depend on the context of the application. This paper proposes the solving of a dynamic multi-objective and context sensitive optimisation problem as a way to evalute the quality of the ontology.

Keywords: Semantic Web, Ontology features, Ontology quality, Contextual optimisation
Key Terms: KnowledgeEvolution, Intelligence, SemanticWebService

1 Introduction

The World Wide Web Consortium (W3C) regards the Semantic Web as a web of data. Currently, data is created at extremely high rate and is not available enough to end users. The Semantic Web aims "To do for machine processable information (application data) what the World Wide Web has done for hypertext" by supporting the creation of interoperable and linked data. [1] Linking data mostly happens by using shared identifiers and linking concepts by using shared ontologies.

The paper "Which Semantic Web?" [2] gives quite a critical view on many concepts of the Semantic Web. It states for instance that "Agreeing on a cataloging scheme for Semantic Web documents is a prerequisite for any sharing of semantic knowledge." and "It is easy enough for computers to exchange data about computational abstractions such as filenames, sizes, usernames, passwords, etc. It is much harder for computers to exchange information about human-oriented

concepts such as happiness and beauty.". These statements indicate that the Semantic Web might actually fail in its basic ideas of making decentralised information management possible. This work-in-progress paper describes the idea behind one possible approach to overcome these problems.

In order to address the first problem, it would be necessary to create a system which can make use of semantic data without having knowledge about the ontology used by the system which generated it. Therefore, it would be needed to derive the meaning from the data which another application generated. The approach proposed in this paper is that there would be an optimisation procedure which yields an ontology for an application processing semantic data. The second problem is that computer systems might not be suitable to describe human-oriented concepts. This problem has been tackled in the past by using fuzzy logic. One approach is described in [3], where a computational model which maps events and observations to an emotional state uses a fuzzy-logic representation. This paper is keeping the option of using an ontology based on fuzzy logic as one possible way of finding an ontology.

One important application of the proposed approach can be found in self-managed systems. The 'executable reality' approach as described in [4] is one example of a system which would benefit from the approach described in this article. 'Executable reality' is described in as "an extension of the (Mobile) Mixed Reality concept". The described extension replaces part of the 'static' retrieval of information by computation of data using context sensitive business intelligence. When a device with such system is used at a location close by the sea concepts related to shores, harbours and seashells might become part of the active ontology. When the device is at a later time point used in a mountainous area the sea related concepts become partly redundant. If the device has a small storage capacity, the most optimal ontology will not contain these concepts any longer. A device which has, on the contrary, an abundant storage capacity but low processing power should keep the concepts stored to avoid computationally expensive changes in the ontology.

2 Optimisation

Optimisation problems are in general statements of problems where a best solution is to be found according to certain criteria and restrictions. The first paragraphs describe a few classes of global optimisation problems in order to introduce the Context-dependant Dynamic Multi-objective optimisation class in the last paragraph.

Static single-objective optimisation is considered a basic type of optimisation problems. The problem statement consists of a function which will be called f with domain D and range R. The domain of the function can be given explicitly as a set or described using constraints on a set. The set used for the range should have a total order relation $\leqslant$ defined on its elements, i.e. there is a transitive, antisymmetric, and total relation on the elements of R.

Definition 1. *An element $d \in D$ is optimal for a function f, i.e. $d \in opt(f) \Leftrightarrow \forall\ e \in D : f(e) \leqslant f(d)$*

The class of optimisation problems described in the previous definition can be extended to a multi-objective variant by allowing the range of the function to be a set of vectors of dimension n. The range of the function f is thus $R_1 \times R_2 \times \cdots \times R_n$ where we require a (strict) total order relation $<_i$ to be defined on each R_i. The domain of the function is a set D. Note that all single objective optimisation problems are multi-objective problems.

Definition 2. *An element $d \in D$ is multi-objective optimal for a function f, i.e. $d \in opt(f) \Leftrightarrow \forall\ e \in D \setminus \{d\} : (\exists\ i \in [1, n] : f(e)_i <_i f(d)_i)$*

Dynamic optimisation is essentially not different from solving a series of (multi-objective) static optimisation problems. The dynamism in the series is to be found in the way the function which is optimised or the constraints on the domain of the function being optimised are changing. The series can be represented by a function F, which maps the natural numbers to a set of pairs of a function and its domain. The functions in the tuples have a domain which is a subset of the total domain D the optimisation problem is working on. One way of solving the dynamic optimisation problem is by finding the optimal value for the functions for all possible values in $\mathbb{N}$ and then selecting the maximum value. We denote $opt'(F(t))$ to be the optimisation problem with function $F(t)_1$ and constraints $F(t)_2$.

Definition 3. *An element $d \in D$ is considered optimal for the dynamic optimisation of the function $F : \mathbb{N} \to (functions, constraints)$ if $\forall t \in \mathbb{N} : opt'(F(t)) \leq d$.*

Another possible definition follows:

Definition 4. *A function sol is the solution of the dynamic optimisation problem if $sol : \mathbb{N} \to D$ and $sol(t) = opt'(F(t))$*

In other words, a solution is such function which gives the optimal solution for each possible $t \in \mathbb{N}$.

Context-dependant dynamic multi-objective optimisation problems are an extension of the dynamic optimisation problems defined in the previous paragraph. In this case the domain of the function F is a series of what will be called 'contexts' instead of the natural numbers. The solution of the optimisation problem can be stated in a similar way as the definitions in the previous paragraphs. A solution of this type of problem indicates (multi-objective) optimal solutions in particular contexts, or analog optimal solutions over the range of all possible contexts.

3 Features of an Ontology

An application needs schemas or ontologies in order to give meaning to the semantic data it is processing. Different definitions of ontologies have been proposed. Gruber [5] defined, for instance, that "An ontology is an explicit specification of a conceptualization.", which allows for a very broad interpretation. A

more concrete definition for description of ontologies is OWL2 endorsed by the World Wide Web Consortium. [6] There is a close correspondence (and sometimes even compatibility) between ontologies and description logic. OWL2 is for instance compatible with the description logic SROIQ. [7]

For this article, the concrete syntax of the ontology used is not of mayor importance. More important is the fact that an ontology has certain features. Examples of features of an ontology include coverage, cohesion and coupling. [8] In this article the properties which an ontology has independent of any context will be called the 'features' of the ontology. Some literature calls these properties 'quality' factors. The name quality will however be used in one of the following sections to describe the effect of features in a context. Other methods for measuring features of an ontology have been proposed by Burton-Jones et al. [9] and Yao et al. [10] and many other feature sets can be found in the literature.

4 Fuzzy Ontologies

For the representation of not exactly defined sets, fuzzy sets as described in "Fuzzy Sets" [11] can be used. The elements of a fuzzy set are members of the set according to a given membership function. This function defines for each element of the considered universe a grade of membership in the set. The grade of membership is a real value in the interval $[0, 1]$, where a value of 0 means that the element is not in the set and a value of 1 indicates that the element is in the set. Any value in between indicates up to which extend the element is a member of the set.

Calegari and Giucci used the concept of fuzzy sets to describe what they call 'Fuzzy Ontologies', 'Fuzzy Description Logics' and 'Fuzzy-OWL'. [12] This research showed it to be possible to describe ontologies which are not exactly known by using membership functions. Bobillo and Straccia [13] did a similar work, but used OWL 2, and proposed a concrete XML syntax for the extension.

5 Ontology Evolution

In research on databases it has been noticed that the schemas which are used change over time. Ontologies have similar properties. Changes in the domain, the conceptualisation and the specification are unavoidable and the ontology has to be changed accordingly. The domain changes because the real world changes, the conceptualisation because the perspective changes and the specification changes when an ontology is to be represented in a language with different semantics and expressiveness. The whole of these changes is called ontology evolution. Ontologies and database schemas are different concepts. Firstly, the ontology itself can also be part of the data and this way the data becomes self-descriptive. Secondly, ontologies are explicitly designed for reuse in other context as the initial context of creation and are, moreover, decentralised by nature. Lastly, ontology models have, in general, a richer set of properties available for describing the

domain and quite often the border between the schema and instance data is blurry.[14]

The same article also describes concrete effects of ontology evolution on the data set. For instance removal of a class causes instances to have a less specific type, declaring classes disjoint makes instances that are in both classes invalid and defining a class as a subclass of another one adds new possible properties on the subclass.

Despite their differences, a more recent article by Hartung et al. [15] claims that ontology evolution has similar requirements as changes in database schemas evolution. Ontology evolution requires, for instance, "support for a rich set of changes, expressive mappings, update propagation to instances and dependant schemas/ontologies, versioning and user-friendly tools". The same article compares several approaches for managing and tracking ontology evolution in terms of the above mentioned criteria.

6 Context and Quality of an Ontology

In order to talk about the quality of an ontology, one needs to take the context within which it is used into account. This statement can be supported by an example. Imagine for instance that one would say that an ontology which contains more concepts, is better than one with less concepts. This could be true in certain situations. However, if one imagines a system which only uses the concepts which are mentioned in the ontology with less concepts, the smaller ontology might be even better because it uses less memory space.

One way of defining ontological quality is described in "Data Driven Ontology Evaluation" [16]. This method uses a combination of a corpus and the ontology to evaluate the quality of the ontology. The corpus is a textual description of the ontology, which form the basis of different approaches of measuring ontologies described. The paper further elaborates on the fact that there is more as one quality aspect with regard to ontologies. Factors like price to build, maintenance and re-use are highlighted as very influential. Furthermore, the article mentions that the quality might be subjective to time, location and other contextual factors.

A recent survey on ontology evaluation tools was performed by Aruna et al. [17] This survey puts a stress on more technical demands of an ontology in a working system. Technical properties surveyed are interoperability, turn around ability, performance, memory allocation, scalability, integration into frameworks and interfaces. The ontological properties are limited to language conformity and consistency.

Research on improving the quality of an ontology by transformation operations on an existing ontology is described in [18]. The idea is that certain quality criteria can be fulfilled better by a transformation of the existing ontology into a new, but equally valid ontology. Concrete, several transformations are described to improve the ontology in terms of homogeneity, totality of properties, stability over time, and explicitness (as opposed to inferred) and uniformity in proper-

ties. The size of the ontology and simplicity of queries is, according to the same article, only assessable in a context.

One way of measuring the quality of an ontology might be to compare the ontology with one or more sets of data which should be described by the ontology. This comparison can be done using either the open-world assumption which is typically made in the semantic web or a closed-world assumption.

Quality does not have to be a one dimensional property. The quality of an ontology in a context can thus be a multidimensional property. The more complete the context for optimisation is, the fewer dimensions the quality will have. A 'complete' context will lead to a one dimensional quality.

7 Quality of Ontology in a Context as a Dynamic Multi-objective Optimisation Problem

As argued in the previous section, it is only reasonable to make statements about the quality of an ontology in a given context. If for a certain context the quality for two ontologies is given, then it is possible to make a comparison between the two ontologies in that particular context. However, quality will not always be one-dimensional and hence analog to the multi-objective class of optimisation problems, it is not always possible to say that either the first or the second ontology is better.

When it is possible to compare the quality of ontologies in a context, then it is also possible to define what it means for an ontology to be optimal in a given context. Because of the fact that there is no total order on the quality of an ontology in a context, it will be necessary to consider the search for an optimal ontology for a given context as a multi-objective optimisation problem.

Let C be the set of contexts, O be the set of ontologies and Q be the set of qualities. Now we can define the optimisation as:

Definition 5. *The function sol is the solution of the context-dependant dynamic multi-objective problem of finding an optimal ontology for a given context* $\Leftrightarrow$ $sol : C \rightarrow O$ *and* $sol(c) = opt'(F(c))$

Where $F\,(C) \rightarrow (O \rightarrow Q)$ a function which maps the context c to a function which incorporates the context when evaluating the quality of an ontology and its associated domain.

It makes most sense to use the definition of context-dependant optimisation with a function since the system will be used in a real life setting and it is impossible to predict the optimal ontology over the whole life-time of the system at any point. It should also be noted that in any real system, the function *sol* can only be known partially and most likely by approximation. It is only known partially because the context is changing all the time and the future contexts are still unknown. Only an approximation can be found because no real system can react quick enough to all changes in a complex context in order to compute a new optimal ontology.

8 Future Research Directions

As said in the previous section, it is important to note that the context of the system in which the ontology is optimised will change dynamically over time. One important aspect of the context is the history of the system. The reason is that taking a new ontology into use causes a certain replacement overhead. A new ontology causing a big overhead has a lower quality in the context of the system. Cuenca Grau et al. [19] tried to reduce the cost of consistency checking of a new ontology by using the previously used ontology, i.e. the history of the system.

In this article we did not specify any concrete ontology notation to be used for this type of system. There is a need to evaluate the different possible classes of ontologies and see which properties of ontologies are affected in this type of optimisation. Depending on the type of ontology chosen, the system has different options for the development of its ontology. If the ontology would be for instance a fuzzy ontology, it might even be feasible to change only membership functions to adapt to changes in the context.

It is an open question how the system should search for an optimal ontology for a given context. Furthermore, this article refrained from defining a concrete definition of context and how it should be incorporated in the functions for optimisation. One popular choice for the incorporation of context is the use of weighted sum based methods. More research is needed to link particular features of an ontology to quality aspects, as well as how the context influences this.

Next, it seems reasonable to look whether it is possible to notice trends in the evolution of the ontology. A system could then take the trends into account when assessing a new ontology or predict resource consumption in the future.

Lastly, it is interesting to consider what would happen when two separate systems have their isolated evolutions of the ontology. Questions in that kind of situation include what should be done to align these ontologies, what to do with discrepancy between the ontologies, whether these systems can interact if they are using different ontologies, etc...

9 Conclusion

The aim of this article was to show initial findings to solve the problems of interoperability in the Semantic web in case ontologies are not shared among applications and how to allow these applications to work with more 'humanised' concepts. The first problem was reduced to a formulation of the finding of an optimal ontology in a given context in section 7. This context includes for instance the data processed by the application, the past used ontologies and constraints related to the system. The challenge of allowing humanised concepts is attempted by allowing fuzzy logic to be used in this kind of system.

This article was financially supported by the 'Cloud Software Program' of TiViT Oy. We would like to thank 'Intelligent Precision Solutions and Services Oy' and in particular Sami Helin for proposing the initial 'Cloud Communication Channel' business case in which this research idea was elaborated.

References

1. Carroll, J.J., Klyne, G.: Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C (February 2004) http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.
2. Marshall, C., Shipman, F.: Which semantic web? In: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, ACM (2003) 57–66
3. El-Nasr, M.S., Yen, J., Ioerger, T.R.: Flame—fuzzy logic adaptive model of emotions. Autonomous Agents and Multi-Agent Systems **3** (2000) 219–257 10.1023/A:1010030809960.
4. Terziyan, V., Kaykova, O.: Towards" executable reality": Business intelligence on top of linked data. In: BUSTECH 2011, The First International Conference on Business Intelligence and Technology. (2011) 26–33
5. Gruber, T., et al.: A translation approach to portable ontology specifications. Knowledge acquisition **5**(2) (1993) 199–220
6. Group, W.O.W.: OWL 2 web ontology language document overview. Technical report, W3C (October 2009) http://www.w3.org/TR/2009/REC-owl2-overview-20091027/.
7. Patel-Schneider, P.F., Motik, B., Grau, B.C.: OWL 2 web ontology language direct semantics. W3C recommendation, W3C (October 2009) http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/.
8. Ouyang, L., Zou, B., Qu, M., Zhang, C.: A method of ontology evaluation based on coverage, cohesion and coupling. In: FSKD. (2011) 2451–2455
9. Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P.: A semiotic metrics suite for assessing the quality of ontologies. Data Knowl. Eng. **55**(1) (October 2005) 84–102
10. Yao, H., Orme, A.M., Etzkorn, L.: Cohesion Metrics for Ontology Design and Application. Journal of Computer Science **1**(1) (2005) 107–113
11. Zadeh, L.: Fuzzy sets. Information Control **8** (1965) 338–353
12. Calegari, S., Ciucci, D.: Fuzzy ontology, fuzzy description logics and fuzzy-owl. In: Proceedings of the 7th international workshop on Fuzzy Logic and Applications: Applications of Fuzzy Sets Theory. WILF '07, Berlin, Heidelberg, Springer-Verlag (2007) 118–126
13. Bobillo, F., Straccia, U.: Fuzzy ontology representation using owl 2. CoRR **abs/1009.3391** (2010)
14. Noy, N., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems **6**(4) (2004) 428–440
15. Hartung, M., Terwilliger, J., Rahm, E.: Recent advances in schema and ontology evolution. Schema Matching and Mapping (2011) 149–190
16. Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y.: Data driven ontology evaluation. In: International Conference on Language Resources and Evaluation (LREC 2004). (2004)
17. Aruna, T., Saranya, K., Bhandari, C.: A survey on ontology evaluation tools. In: Process Automation, Control and Computing (PACC), 2011 International Conference on. (july 2011) 1 –5
18. Mostowfi, F., Fotouhi, F.: Improving quality of ontology: An ontology transformation approach. In: Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on, IEEE (2006) 61–61
19. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. The Semantic Web (2007) 183–196

PII

TOWARD EVOLVING KNOWLEDGE ECOSYSTEMS FOR BIG DATA UNDERSTANDING

by

Vadim Ermolayev, Rajendra Akerkar, Vagan Terziyan, and Michael Cochez 2014

R. Akerkar (Ed.), Big Data Computing (pp. 3-56). Boca Raton, FL: Taylor & Francis

Reproduced with kind permission of Taylor & Francis/CRC Press.

1

Toward Evolving Knowledge Ecosystems for Big Data Understanding

Vadim Ermolayev, Rajendra Akerkar, Vagan Terziyan, and Michael Cochez

CONTENTS

© 2014 by Taylor & Francis Group, LLC

Introduction

Big Data is a phenomenon that leaves a rare information professional negligent these days. Remarkably, application demands and developments in the context of related disciplines resulted in technologies that boosted data generation and storage at unprecedented scales in terms of volumes and rates. To mention just a few facts reported by Manyika et al. (2011): a disk drive capable of storing all the world's music could be purchased for about US $600; 30 billion of content pieces are shared monthly only at Facebook (facebook.com). Exponential growth of data volumes is accelerated by a dramatic increase in social networking applications that allow nonspecialist users create a huge amount of content easily and freely. Equipped with rapidly evolving mobile devices, a user is becoming a nomadic gateway boosting the generation of additional real-time sensor data. The emerging Internet of Things makes every thing a data or content, adding billions of additional artificial and autonomic sources of data to the overall picture. Smart spaces, where people, devices, and their infrastructure are all loosely connected, also generate data of unprecedented volumes and with velocities rarely observed before. An expectation is that valuable information will be extracted out of all these data to help improve the quality of life and make our world a better place.

Society is, however, left bewildered about how to use all these data efficiently and effectively. For example, a topical estimate for the number of a need for data-savvy managers to take full advantage of Big Data in the United States is 1.5 million (Manyika et al. 2011). A major challenge would be finding a balance between the two evident facets of the whole Big Data adventure: (a) the more data we have, the more potentially useful patterns it may include and (b) the more data we have, the less the hope is that any machine-learning algorithm is capable of discovering these patterns in an acceptable time frame. Perhaps because of this intrinsic conflict, many experts consider that this Big Data not only brings one of the biggest challenges, but also a most exciting opportunity in the recent 10 years (cf. Fan et al. 2012b)

The avalanche of Big Data causes a conceptual divide in minds and opinions. Enthusiasts claim that, faced with massive data, a scientific approach "… hypothesize, model, test—is becoming obsolete. … Petabytes allow us to say: 'Correlation is enough.' We can stop looking for models. We can analyze the data without hypotheses about what it might show. We can throw the numbers into the biggest computing clusters the world has ever seen and let statistical algorithms find patterns …" (Anderson 2008). Pessimists, however, point out

© 2014 by Taylor & Francis Group, LLC

that Big Data provides "… destabilising amounts of knowledge and information that lack the regulating force of philosophy" (Berry 2011). Indeed, being abnormally big does not yet mean being healthy and wealthy and should be treated appropriately (Figure 1.1): a diet, exercise, medication, or even surgery (philosophy). Those data sets, for which systematic health treatment is ignored in favor of correlations, will die sooner—as useless. There is a hope, however, that holistic integration of evolving algorithms, machines, and people reinforced by research effort across many domains will guarantee required fitness of Big Data, assuring proper quality at right time (Joseph 2012).

Mined correlations, though very useful, may hint about an answer to a "what," but not "why" kind of questions. For example, if Big Data about Royal guards and their habits had been collected in the 1700s' France, one could mine today that all musketeers who used to have red Burgundy regularly for dinners have not survived till now. Pity, red Burgundy was only one of many and a very minor problem. A scientific approach is needed to infer real reasons—the work currently done predominantly by human analysts.

Effectiveness and efficiency are the evident keys in Big Data analysis. Cradling the gems of knowledge extracted out of Big Data would only be effective if: (i) not a single important fact is left in the burden—which means completeness and (ii) these facts are faceted adequately for further inference—which means expressiveness and granularity. Efficiency may be interpreted as the ratio of spent effort to the utility of result. In Big Data analytics, it could be straightforwardly mapped to timeliness. If a result is not timely, its utility (Ermolayev et al. 2004) may go down to zero or even far below in seconds to milliseconds for some important industrial applications such as technological process or air traffic control.

Notably, increasing effectiveness means increasing the effort or making the analysis computationally more complex, which negatively affects efficiency.

FIGURE 1.1
Evolution of data collections—dimensions (see also Figure 1.3) have to be treated with care. (Courtesy of Vladimir Ermolayev.)

© 2014 by Taylor & Francis Group, LLC

Finding a balanced solution with a sufficient degree of automation is the challenge that is not yet fully addressed by the research community.

One derivative problem concerns knowledge extracted out of Big Data as the result of some analytical processing. In many cases, it may be expected that the knowledge mechanistically extracted out of Big Data will also be big. Therefore, taking care of Big Knowledge (which has more value than the source data) would be at least of the same importance as resolving challenges associated with Big Data processing. Uplifting the problem to the level of knowledge is inevitable and brings additional complications such as resolving contradictory and changing opinions of everyone on everything. Here, an adequate approach in managing the authority and reputation of "experts" will play an important role (Weinberger 2012).

This chapter offers a possible approach in addressing the problem of "understanding" Big Data in an effective and efficient way. The idea is making adequately grained and expressive knowledge representations and fact collections evolve naturally, triggered by new tokens of relevant data coming along. Pursuing this way would also imply conceptual changes in the Big Data Processing stack. A refined semantic layer has to be added to it for providing adequate interfaces to interlink horizontal layers and enable knowledge-related functionality coordinated in top-down and bottom-up directions.

The remainder of the chapter is structured as follows. The "Motivation and Unsolved Issues" section offers an illustrative example and the analysis of the demand for understanding Big Data. The "State of Technology, Research, and Development in Big Data Computing" section reviews the relevant research on using semantic and related technologies for Big Data processing and outlines our approach to refine the processing stack. The "Scaling with a Traditional Database" section focuses on how the basic data storage and management layer could be refined in terms of scalability, which is necessary for improving efficiency/effectiveness. The "Knowledge Self-Management and Refinement through Evolution" section presents our approach, inspired by the mechanisms of natural evolution studied in evolutionary biology. We focus on a means of arranging the evolution of knowledge, using knowledge organisms, their species, and populations with the aim of balancing efficiency and effectiveness of processing Big Data and its semantics. We also provide our preliminary considerations on assessing fitness in an evolving knowledge ecosystem. Our conclusions are drawn in the "Some Conclusions" section.

Motivation and Unsolved Issues

Practitioners, including systems engineers, Information Technology architects, Chief Information and Technology Officers, and data scientists, use

© 2014 by Taylor & Francis Group, LLC

the phenomenon of Big Data in their dialog over means of improving sense-making. The phenomenon remains a constructive way of introducing others, including nontechnologists, to new approaches such as the Apache Hadoop (hadoop.apache.org) framework. Apparently, Big Data is collected to be analyzed. "Fundamentally, big data analytics is a workflow that distills terabytes of low-value data down to, in some cases, a single bit of high-value data. … The goal is to see the big picture from the minutia of our digital lives" (cf. Fisher et al. 2012). Evidently, "seeing the big picture" in its entirety is the key and requires making Big Data healthy and understandable in terms of effectiveness and efficiency for analytics.

In this section, the motivation for understanding the Big Data that improves the performance of analytics is presented and analyzed. It begins with presenting a simple example which is further used throughout the chapter. It continues with the analysis of industrial demand for Big Data analytics. In this context, the major problems as perceived by industries are analyzed and informally mapped to unsolved technological issues.

Illustrative Example

Imagine a stock market analytics workflow inferring trends in share price changes. One possible way of doing this is to extrapolate on stock price data. However, a more robust approach could be extracting these trends from market news. Hence, the incoming data for analysis would very likely be several streams of news feeds resulting in a vast amount of tokens per day. An illustrative example of such a news token is:

> Posted: Tue, 03 Jul 2012 05:01:10-04:00
>
> LONDON (Reuters)
>
> U.S. planemaker Boeing hiked its 20-year market forecast, predicting demand for 34,000 new aircraft worth $4.5 trillion, on growth in emerging regions and as airlines seek efficient new planes to counter high fuel costs.*

Provided that an adequate technology is available,† one may extract the knowledge pictured as thick-bounded and gray-shaded elements in Figure 1.2.

This portion of extracted knowledge is quite shallow, as it simply interprets the source text in a structured and logical way. Unfortunately, it does

* topix.com/aircraft/airbus-a380/2012/07/boeing-hikes-20-year-market-forecast (accessed July 5, 2012).

† The technologies for this are under intensive development currently, for example, wit.istc.cnr.it/stlab-tools/fred/ (accessed October 8, 2012).

© 2014 by Taylor & Francis Group, LLC

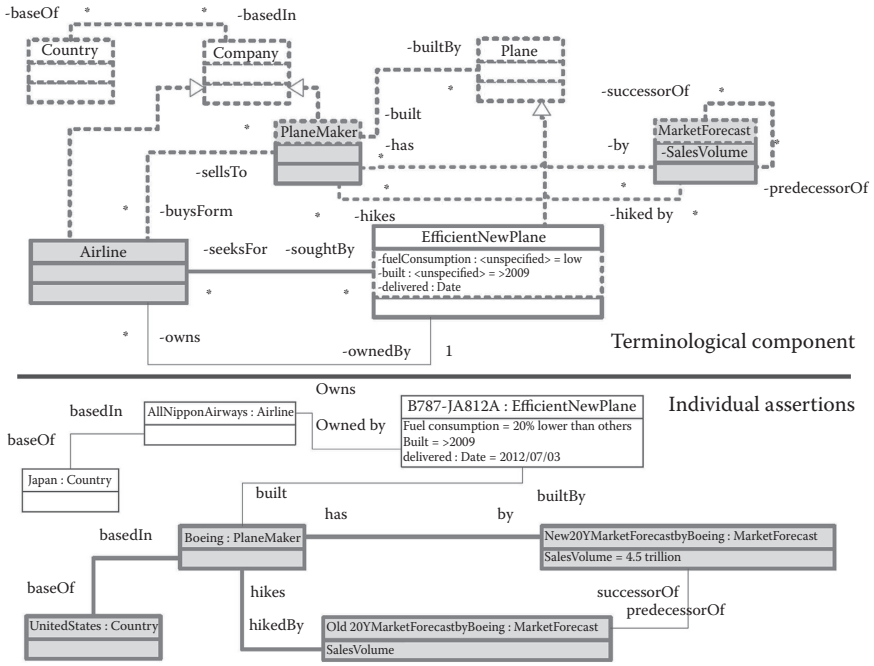


FIGURE 1.2
Semantics associated with a news data token.

not answer several important questions for revealing the motives for Boeing to hike their market forecast:

Q1. What is an efficient new plane? How is efficiency related to high fuel costs to be countered?

Q2. Which airlines seek for efficient new planes? What are the emerging regions? How could their growth be assessed?

Q3. How are plane makers, airlines, and efficient new planes related to each other?

In an attempt to answering these questions, a human analyst will exploit his commonsense knowledge and look around the context for additional relevant evidence. He will likely find out that Q1 and Q3 could be answered using commonsense statements acquired from a foundational ontology, for example, CYC (Lenat 1995), as shown by dotted line bounded items in Figure 1.2.

Answering Q2, however, requires looking for additional information like: the fleet list of All Nippon Airways* who was the first to buy B787 airplanes

* For example, at airfleets.net/flottecie/All%20Nippon%20Airways-active-b787.htm (accessed July 5, 2012).

© 2014 by Taylor & Francis Group, LLC

from Boeing (the rest of Figure 1.2); and a relevant list of emerging regions and growth factors (not shown in Figure 1.2). The challenge for a human analyst in performing the task is low speed of data analysis. The available time slot for providing his recommendation is too small, given the effort to be spent per one news token for deep knowledge extraction. This is one good reason for growing demand for industrial strength technologies to assist in analytical work on Big Data, increase quality, and reduce related efforts.

Demand in Industry

Turning available Big Data assets into action and performance is considered a deciding factor by today's business analytics. For example, the report by Capgemini (2012) concludes, based on a survey of the interviews with more than 600 business executives, that Big Data use is highly demanded in industries. Interviewees firmly believe that their companies' competitiveness and performance strongly depend on the effective and efficient use of Big Data. In particular, on average,

- Big Data is already used for decision support 58% of the time, and 29% of the time for decision automation
- It is believed that the use of Big Data will improve organizational performance by 41% over the next three years

The report by Capgemini (2012) also summarizes that the following are the perceived benefits of harnessing Big Data for decision-making:

- More complete understanding of market conditions and evolving business trends
- Better business investment decisions
- More accurate and precise responses to customer needs
- Consistency of decision-making and greater group participation in shared decisions
- Focusing resources more efficiently for optimal returns
- Faster business growth
- Competitive advantage (new data-driven services)
- Common basis for evaluation (one true starting point)
- Better risk management

Problems in Industry

Though the majority of business executives firmly believe in the utility of Big Data and analytics, doubts still persist about its proper use and the availability of appropriate technologies. As a consequence, "We no longer

© 2014 by Taylor & Francis Group, LLC

speak of the Knowledge Economy or the Information Society. It's all data now: Data Economy and Data Society. This is a confession that we are no longer in control of the knowledge contained in the data our systems collect" (Greller 2012).

Capgemini (2012) outlines the following problems reported by their interviewees:

- *Unstructured data are hard to process at scale*. Forty-two percent of respondents state that unstructured content is too difficult to interpret. Forty percent of respondents believe that they have too much unstructured data to support decision-making.
- *Fragmentation is a substantial obstacle*. Fifty-six percent of respondents across all sectors consider organizational silos the biggest impediment to effective decision-making using Big Data.
- *Effectiveness needs to be balanced with efficiency in "cooking" Big Data*. Eighty-five percent of respondents say the major problem is the lack of effective ability to analyze and act on data in real time.

The last conclusion by Capgemini is also supported by Bowker (2005, pp. 183–184) who suggests that "raw data is both an oxymoron and a bad idea; to the contrary, data should be cooked with care." This argument is further detailed by Bollier (2010, p. 13) who stresses that Big Data is a huge "mass of raw information." It needs to be added that this "huge mass" may change in time with varying velocity, is also noisy, and cannot be considered as self-explanatory. Hence, an answer to the question whether Big Data indeed represent a "ground truth" becomes very important—opening pathways to all sorts of philosophical and pragmatic discussions. One aspect of particular importance is interpretation that defines the ways of cleaning Big Data. Those ways are straightforwardly biased because any interpretation is subjective.

As observed, old problems of data processing that are well known for decades in industry are made even sharper when data becomes Big. Boyd and Crawford (2012) point out several aspects to pay attention to while "cooking" Big Data, hinting that industrial strength technologies for that are not yet in place:

- *Big Data changes the way knowledge is acquired and even defined*. As already mentioned above (cf. Anderson 2008), correlations mined from Big Data may hint about model changes and knowledge representation updates and refinements. This may require conceptually novel solutions for evolving knowledge representation, reasoning, and management.
- *Having Big Data does not yet imply objectivity, or accuracy, on time*. Here, the clinch between efficiency and effectiveness of Big Data interpretation and processing is one of the important factors. Selecting

© 2014 by Taylor & Francis Group, LLC

a sample of an appropriate size for being effective may bring bias, harm correctness, and accuracy. Otherwise, analyzing Big Data in source volumes will definitely distort timeliness.

- Therefore, *Big Data is not always the best option*. A question that requires research effort in this context is about the appropriate sample, size, and granularity to best answer the question of a data analyst.
- Consequently, *taken off-context Big Data is meaningless in interpretation*. Indeed, choosing an appropriate sample and granularity may be seen as contextualization—circumscribing (Ermolayev et al. 2010) the part of data which is potentially the best-fitted sample for the analytical query. Managing context and contextualization for Big Data at scale is a typical problem and is perceived as one of the research and development challenges.

One more aspect having indirect relevance to technology, but important in terms of socio-psychological perceptions and impact on industries, is *ethics* and *Big Data divide*. Ethics is concerned with legal regulations and constraints of allowing a Big Data collector interpreting personal or company information without informing the subjects about it. Ethical issues become sharper when used for competition and lead to the emergence of and separation to Big Data rich and poor implied by accessibility to data sources at required scale.

Major Issues

Applying Big Data analytics faces different issues related with the characteristics of data, analysis process, and also social concerns. Privacy is a very sensitive issue and has conceptual, legal, and technological implications. This concern increases its importance in the context of big data. Privacy is defined by the International Telecommunications Union as the "right of individuals to control or influence what information related to them may be disclosed" (Gordon 2005). Personal records of individuals are increasingly being collected by several government and corporate organizations. These records usually used for the purpose of data analytics. To facilitate data analytics, such organizations publish "appropriately private" views over the collected data. However, privacy is a double-edged sword—there should be enough privacy to ensure that sensitive information about the individuals is not disclosed and at the same time there should be enough data to perform the data analysis. Thus, privacy is a primary concern that has widespread implications for someone desiring to explore the use of Big Data for development in terms of data acquisition, storage, preservation, presentation, and use.

Another concern is the access and sharing of information. Usually private organizations and other institutions are reluctant to share data about their

© 2014 by Taylor & Francis Group, LLC

clients and users, as well as about their own operations. Barriers may include legal considerations, a need to protect their competitiveness, a culture of confidentiality, and, largely, the lack of the right incentive and information structures. There are also institutional and technical issues, when data are stored in places and ways that make them difficult to be accessed and transferred.

One significant issue is to rethink security for information sharing in Big Data use cases. Several online services allow us to share private information (i.e., facebook.com, geni.com, linkedin.com, etc.), but outside record-level access control we do not comprehend what it means to share data and how the shared data can be linked.

Managing large and rapidly increasing volumes of data has been a challenging issue. Earlier, this issue was mitigated by processors getting faster, which provide us with the resources needed to cope with increasing volumes of data. However, there is a fundamental shift underway considering that data volume is scaling faster than computer resources. Consequently, extracting sense of data at required scale is far beyond human capability. So, we, the humans, increasingly "... require the help of automated systems to make sense of the data produced by other (automated) systems" (Greller 2012). These instruments produce new data at comparable scale—kick-starting a new iteration in this endless cycle.

In general, given a large data set, it is often necessary to find elements in it that meet a certain criterion which likely occurs repeatedly. Scanning the entire data set to find suitable elements is obviously impractical. Instead, index structures are created in advance to permit finding the qualifying elements quickly.

Moreover, dealing with new data sources brings a significant number of analytical issues. The relevance of these issues will vary depending on the type of analysis being conducted and on the type of decisions that the data might ultimately inform. The big core issue is to analyze what the data are really telling us in an entirely transparent manner.

State of Technology, Research, and Development in Big Data Computing

After giving an overview of the influence of Big Data on industries and society as a phenomenon and outlining the problems in Big Data computing context as perceived by technology consumers, we now proceed with the analysis of the state of development of those technologies. We begin with presenting the overall Big Data Processing technology stack and point out how different dimensions of Big Data affect the requirements to technologies, having understanding—in particular, semantics-based processing—as a primary focus. We continue with presenting a selection of Big Data

© 2014 by Taylor & Francis Group, LLC

research and development projects and focus on what they do in advancing the state-of-the-art in semantic technologies for Big Data processing. Further, we summarize the analysis by pointing out the observed complications and overheads in processing Big Data semantics. Finally, we outline a high-level proposal for the refinement of the Big Data semantics layer in the technology stack.

Big Data Processing—Technology Stack and Dimensions

At a high level of detail, Driscoll (2011) describes the Big Data processing technology stack comprising three major layers: foundational, analytics, and applications (upper part of Figure 1.3).

The foundational layer provides the infrastructure for storage, access, and management of Big Data. Depending on the nature of data, stream processing solutions (Abadi et al. 2003; Golab and Tamer Ozsu 2003; Salehi 2010), distributed persistent storage (Chang et al. 2008; Roy et al. 2009; Shvachko et al. 2010), cloud infrastructures (Rimal et al. 2009; Tsangaris et al. 2009; Cusumano 2010), or a reasonable combination of these (Gu and Grossman 2009; He et al. 2010; Sakr et al. 2011) may be used for storing and accessing data in response to the upper-layer requests and requirements.

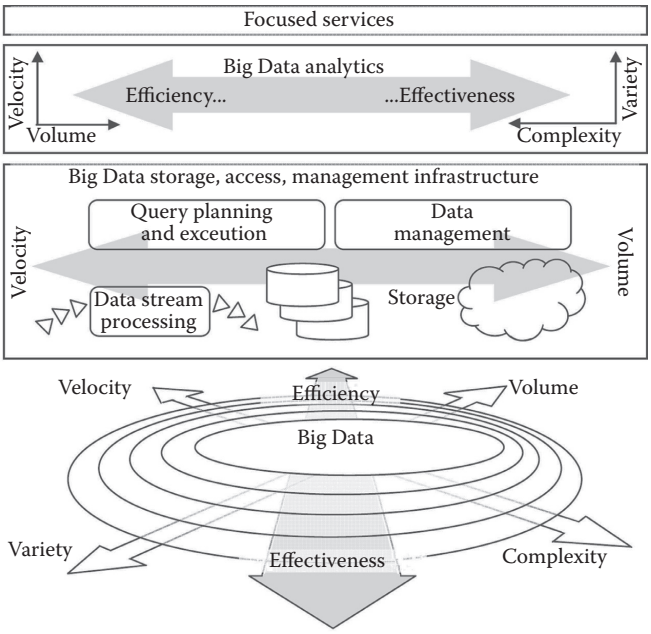


FIGURE 1.3
Processing stack, based on Driscoll (2011), and the four dimensions of Big Data, based on Beyer et al. (2011), influencing efficiency and effectiveness of analytics.

© 2014 by Taylor & Francis Group, LLC

The middle layer of the stack is responsible for analytics. Here data warehousing technologies (e.g., Nemani and Konda 2009; Ponniah 2010; Thusoo et al. 2010) are currently exploited for extracting correlations and features (e.g., Ishai et al. 2009) from data and feeding classification and prediction algorithms (e.g., Mills 2011).

Focused applications or services are at the top of the stack. Their functionality is based on the use of more generic lower-layer technologies and exposed to end users as Big Data products.

Example of a startup offering focused services is BillGuard (billguard.com). It monitors customers' credit card statements for dubious charges and even leverages the collective behavior of users to improve its fraud predictions. Another company called Klout (klout.com/home) provides a genuine data service that uses social media activity to measure online influence. LinkedIn's *People you may know* feature is also a kind of focused service. This service is presumably based on graph theory, starting exploration of the graph of your relations from your node and filtering those relations according to what is called "homophily." The greater the homophily between two nodes, the more likely two nodes will be connected.

According to its purpose, the foundational layer is concerned about being capable of processing as much as possible data (*volume)* and as soon as possible. In particular, if streaming data are used, the faster the stream is (*velocity*), the more difficult it is to process the data in a stream window. Currently available technologies and tools for the foundational level are not equally well coping with volume and velocity dimensions which are, so to say, anticorrelated due to their nature. Therefore, hybrid infrastructures are in use for balancing processing efficiency aspects (Figure 1.3)—comprising solutions focused on taking care of volumes, and, separately, of velocity. Some examples are given in "Big Data in European Research" section.

For the analytics layer (Figure 1.3), *volume* and *velocity* dimensions (Beyer et al. 2011) are also important and constitute the facet of efficiency—big volumes of data which may change swiftly have to be processed in a timely fashion. However, two more dimensions of Big Data become important—*complexity* and *variety*—which form the facet of effectiveness. *Complexity* is clearly about the adequacy of data representations and descriptions for analysis. *Variety* describes a degree of syntactic and semantic heterogeneity in distributed modules of data that need to be integrated or harmonized for analysis. A major conceptual complication for analytics is that efficiency is anticorrelated to effectiveness.

Big Data in European Research

Due to its huge demand, Big Data Computing is currently on the hype as a field of research and development, producing a vast domain of work. To keep the size of this review observable for a reader, we focus on the batch of the running 7th Framework Programme (FP7) Information and Communication

© 2014 by Taylor & Francis Group, LLC

Technology (ICT; cordis.europa.eu/fp7/ict/) projects within this vibrant field. Big Data processing, including semantics, is addressed by the strategic objective of Intelligent Information Management (IIM; cordis.europa.eu/fp7/ict/content-knowledge/projects_en.html). IIM projects funded in frame of FP7 ICT Call 5 are listed in Table 1.1 and further analyzed below.

SmartVortex [Integrating Project (IP); smartvortex.eu] develops a technological infrastructure—a comprehensive suite of interoperable tools, services, and methods—for intelligent management and analysis of massive data streams. The goal is to achieving better collaboration and decision-making in large-scale collaborative projects concerning industrial innovation engineering.

Legend: AEP, action extraction and prediction; DLi, data linking; DM, data mining; DS, diversity in semantics; DV, domain vocabulary; FCA, formal concept analysis; IE, information extraction; Int, integration; KD, knowledge discovery; M-LS, multi-lingual search; MT, machine translation; O, ontology; OM, opinion mining; QL, query language; R, reasoning; SBI, business intelligence over semantic data; SDW, semantic data warehouse (triple store); SUM, summarization.

LOD2 (IP; lod2.eu) claims delivering: industrial strength tools and methodologies for exposing and managing very large amounts of structured information; a bootstrap network of multidomain and multilingual ontologies from sources such as Wikipedia (wikipedia.org) and OpenStreetMap (openstreetmap.org); machine learning algorithms for enriching, repairing, interlinking, and fusing data from Web resources; standards and methods for tracking provenance, ensuring privacy and data security, assessing information quality; adaptive tools for searching, browsing, and authoring Linked Data.

Tridec (IP; tridec-online.eu) develops a service platform accompanied with the next-generation work environments supporting human experts in decision processes for managing and mitigating emergency situations triggered by the earth (observation) system in complex and time-critical settings. The platform enables "smart" management of collected sensor data and facts inferred from these data with respect to crisis situations.

First [Small Targeted Research Project (STREP); project-first.eu] develops an information extraction, integration, and decision-making infrastructure for financial domain with extremely large, dynamic, and heterogeneous sources of information.

iProd (STREP; iprod-project.eu) investigates approaches of reducing product development costs by efficient use of large amounts of data comprising the development of a software framework to support complex information management. Key aspects addressed by the project are handling heterogeneous information and semantic diversity using semantic technologies including knowledge bases and reasoning.

Teleios (STREP; earthobservatory.eu) focuses on elaborating a data model and query language for Earth Observation (EO) images. Based on

© 2014 by Taylor & Francis Group, LLC

TABLE 1.1

FP7 ICT Call 5 Projects and their Contributions to Big Data Processing and Understanding

Acronym	IIM Cluster[a]			Domain(s)/Industry(ies)	Contribution to Coping with Big Data dimensions[b]				Contribution to Big Data Processing Stack Layers[c]			Contribution to Big Data Understanding
	Online Content, Interactive and Social Media	Reasoning and Information Exploitation	Knowledge Discovery and Management		Volume	Velocity	Variety	Complexity	i. Fast Access to/Management of Data at Scale	ii. Fast Analytics of Data at Scale	iii. Focused Services	
SmartVortex	X	X	X	Industrial innovation engineering	X	X			X	X		
LOD2		X	X	Media and publishing, corporate data intranets, eGovernment	X		X	X	X	X		O, ML
Tridec		X	X	Crisis/emergency response, government, oil and gas	X	X		X	X	X		R
First		X	X	Market surveillance, investment management, online retail banking and brokerage	X	X	X		X	X		IE
iProd		X	X	Manufacturing: aerospace, automotive, and home appliances	X		X	X	X			R, Int
Teleios		X	X	Civil defense, environmental agencies. Use cases: a virtual observatory for TerraSAR-X data; real-time fire monitoring	X			X	X	X		DM, QL, KD

© 2014 by Taylor & Francis Group, LLC

Khresmoi		X		X		Medical imaging in healthcare, biomedicine	X			X		X		X	X	IE, DLi, M-LS, MT
Robust	X		X			Online communities (internet, extranet and intranet) addressing: customer support; knowledge sharing; hosting services	X	X			X					AEP
Digital.me	X		X			Personal sphere			X		X					
Fish4Knowledge	X		X			Marine sciences, environment	X				X		X			DV (fish), SUM
Render	X		X			Information management (wiki), news aggregation (search engine), customer relationship management (telecommunications)	X		X		X					DS
PlanetData			X			Cross-domain			X							
LATC			X			Government			X		X				X	
Advance			X			Logistics	X	X			X		X			
Cubist			X			Market intelligence, computational biology, control centre operations			X		X		X			SDW, SBI, FCA
Promise					X	Cross-domain			X	X	X		X			
Dicode					X	Clinico-genomic research, healthcare, marketing	X			X	X		X			DM, OM

[a] IIM clustering information has been taken from the Commission's source cordis.europa.eu/fp7/ict/content-knowledge/projects_en.html.
[b] As per the Gartner report on extreme information management (Gartner 2011).
[c] The contributions of the projects to the developments in the Big Data Stack layers have been assessed based on their public deliverables.

© 2014 by Taylor & Francis Group, LLC

these, a scalable and adaptive environment for knowledge discovery from EO images and geospatial data sets, and a query processing and optimization technique for queries over multidimensional arrays and EO image annotations are developed and implemented on top of the MonetDB (monetdb.org) system.

Khresmoi (IP; khresmoi.eu) develops an advanced multilingual and multimodal search and access system for biomedical information and documents. The advancements of the Khresmoi comprise: an automated information extraction from biomedical documents reinforced by using crowd sourcing, active learning, automated estimation of trust level, and target user expertise; automated analysis and indexing for 2-, 3-, 4D medical images; linking information extracted from unstructured or semistructured biomedical texts and images to structured information in knowledge bases; multilingual search including multiple languages in queries and machine-translated pertinent excerpts; visual user interfaces to assist in formulating queries and displaying search results.

Robust (IP; robust-project.eu) investigates models and methods for describing, understanding, and managing the users, groups, behaviors, and needs of online communities. The project develops a scalable cloud and stream-based data management infrastructure for handling the real-time analysis of large volumes of community data. Understanding and prediction of actions is envisioned using simulation and visualization services. All the developed tools are combined under the umbrella of the risk management framework, resulting in the methodology for the detection, tracking, and management of opportunities and threats to online community prosperity.

Digital.me (STREP; dime-project.eu) integrates all personal data in a personal sphere at a single user-controlled point of access—a user-controlled personal service for intelligent personal information management. The software is targeted on integrating social web systems and communities and implements decentralized communication to avoid external data storage and undesired data disclosure.

Fish4Knowledge (STREP; homepages.inf.ed.ac.uk/rbf/Fish4Knowledge/) develops methods for information abstraction and storage that reduce the amount of video data at a rate of 10×10^{15} pixels to 10×10^{12} units of information. The project also develops machine- and human-accessible vocabularies for describing fish. The framework also comprises flexible data-processing architecture and a specialized query system tailored to the domain. To achieve these, the project exploits a combination of computer vision, video summarization, database storage, scientific workflow, and human–computer interaction methods.

Render (STREP; render-project.eu) is focused on investigating the aspect of diversity of Big Data semantics. It investigates methods and techniques, develops software, and collects data sets that will leverage diversity as a source of innovation and creativity. The project also claims providing enhanced support for feasibly managing data on a very large scale and for

© 2014 by Taylor & Francis Group, LLC

designing novel algorithms that reflect diversity in the ways information is selected, ranked, aggregated, presented, and used.

PlanetData [Network of Excellence (NoE); planet-data.eu] works toward establishing a sustainable European community of researchers that supports organizations in exposing their data in new and useful ways and develops technologies that are able to handle data purposefully at scale. The network also facilitates researchers' exchange, training, and mentoring, and event organization based substantially on an open partnership scheme.

LATC (Support Action; latc-project.eu) creates an in-depth test-bed for data-intensive applications by publishing data sets produced by the European Commission, the European Parliament, and other European institutions as Linked Data on the Web and by interlinking them with other governmental data.

Advance (STREP; advance-logistics.eu) develops a decision support platform for improving strategies in logistics operations. The platform is based on the refinement of predictive analysis techniques to process massive data sets for long-term planning and cope with huge amounts of new data in real time.

Cubist (STREP; cubist-project.eu) elaborates methodologies and implements a platform that brings together several essential features of Semantic Technologies and Business Intelligence (BI): support for the federation of data coming from unstructured and structured sources; a BI-enabled triple store as a data persistency layer; data volume reduction and preprocessing using data semantics; enabling BI operations over semantic data; a semantic data warehouse implementing FCA; applying visual analytics for rendering, navigating, and querying data.

Promise (NoE; promise-noe.eu) establishes a virtual laboratory for conducting participative research and experimentation to carry out, advance, and bring automation into the evaluation and benchmarking of multilingual and multimedia information systems. The project offers the infrastructure for access, curation, preservation, re-use, analysis, visualization, and mining of the collected experimental data.

Dicode (STREP; dicode-project.eu) develops a workbench of interoperable services, in particular, for: (i) scalable text and opinion mining; (ii) collaboration support; and (iii) decision-making support. The workbench is designed to reduce data intensiveness and complexity overload at critical decision points to a manageable level. It is envisioned that the use of the workbench will help stakeholders to be more productive and concentrate on creative activities.

In summary, the contributions to Big Data understanding of all the projects mentioned above result in the provision of different functionality for a semantic layer—an interface between the Data and Analytics layers of the Big Data processing stack—as pictured in Figure 1.4.

However, these advancements remain somewhat insufficient in terms of reaching a desired balance between efficiency and effectiveness, as outlined

© 2014 by Taylor & Francis Group, LLC

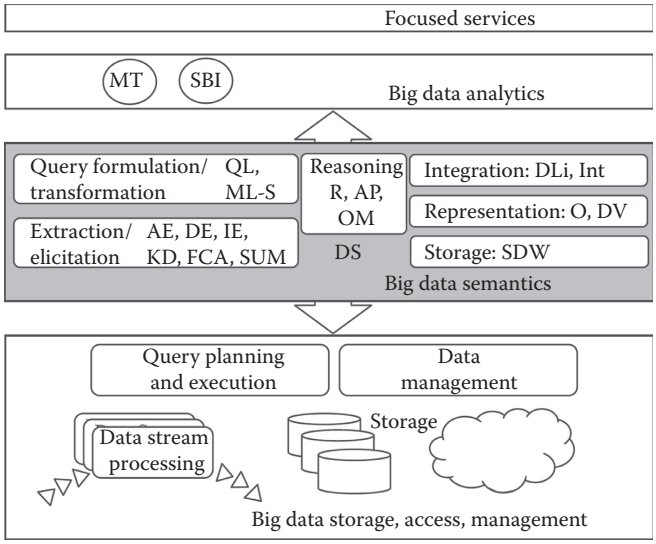


FIGURE 1.4
Contribution of the selection of FP7 ICT projects to technologies for Big Data understanding. Abbreviations are explained in the legend to Table 1.1.

in the introduction of this chapter. Analysis of Table 1.1 reveals that no one of the reviewed projects addresses all four dimensions of Big Data in a balanced manner. In particular, only two projects—*Trydec* and *First*—claim contributions addressing Big Data velocity and variety-complexity. This fact points out that the clinch between efficiency and effectiveness in Big Data processing still remains a challenge.

Complications and Overheads in Understanding Big Data

As observed, the mankind collects and stores data through generations, without a clear account of the utility of these data. Out of data at hand, each generation extracts a relatively small proportion of knowledge for their everyday needs. The knowledge is produced by a generation for their needs—to an extent they have to satisfy their "nutrition" requirement for supporting decision-making. Hence, knowledge is "food" for data analytics. An optimistic assumption usually made here is that the next generation will succeed in advancing tools for data mining, knowledge discovery, and extraction. So the data which the current generation cannot process effectively and efficiently is left as a legacy for the next generation in a hope that the ancestors cope better. The truth, however, is that the developments of data and knowledge-processing tools fail to keep pace with the explosive growth of data in all four dimensions mentioned above. Suspending understanding Big Data until an advanced next-generation capability is at hand is therefore an illusion of a solution.

© 2014 by Taylor & Francis Group, LLC

Do today's state-of-the-art technologies allow us to understand Big Data with an attempt to balance effectiveness and efficiency?—probably not. Our brief analysis reveals that Big Data computing is currently developed toward more effective versus efficient use of semantics. It is done by adding the semantics layer to the processing stack (cf. Figures 1.3 and 1.4) with an objective of processing all the available data and using all the generated knowledge. Perhaps, the major issue is the attempt to eat all we have on the table. Following the metaphor of "nutrition," it has to be noted that the "food" needs to be "healthy" in terms of all the discussed dimensions of Big Data.

Our perceptions of the consequences of being not selective with respect to consuming data for understanding are as follows.

The major problem is the introduction of a new interface *per se* and in an improper way. The advent of semantic technologies aimed at breaking down data silos and simultaneously enabling efficient knowledge management at scale. Assuming that databases describe data using multiple heterogeneous labels, one might expect that annotating these labels using ontology elements as semantic tags enables virtual integration and provides immediate benefits for search, retrieval, reasoning, etc. without a need to modify existing code, or data. Unfortunately, as noticed by Smith (2012), it is now too easy to create "ontologies." As a consequence, myriads of them are being created in *ad hoc* ways and with no respect to compatibility, which implies the creation of new semantic silos and, further bringing something like a "Big Ontology" challenge to the agenda. According to Smith (2012), the big reason is the lack of a rational (monetary) incentive for investing in reuse. Therefore, it is often accepted that a new "ontology" is developed for a new project. Harmonization is left for someone else's work—in the next generation. Therefore, the more semantic technology simplifying ontology creation is successful, the more we fail to achieve our goals for interoperability and integration (Smith 2012).

It is worth noting here that there is still a way to start doing things correctly which, according to Smith (2012), would be "to create an incremental, evolutionary process, where what is good survives, and what is bad fails; create a scenario in which people will find it profitable to reuse ontologies, terminologies and coding systems which have been tried and tested; silo effects will be avoided and results of investment in Semantic Technology will cumulate effectively."

A good example of a collaborative effort going in this correct direction is the approach used by the Gene Ontology initiative (geneontology.org) which follows the principles of the OBO Foundry (obofoundry.org). The Gene Ontology project is a major bioinformatics initiative with the aim of standardizing the representation of gene and gene product attributes across species and databases. The project provides a controlled vocabulary of terms for describing gene product characteristics and gene product annotation data, as well as tools to access and process this data. The mission of OBO Foundry is to support community members in developing and publishing

© 2014 by Taylor & Francis Group, LLC

fully interoperable ontologies in the biomedical domain following common evolving design philosophy and implementation and ensuring a gradual improvement of the quality of ontologies.

Furthermore, adding a data semantics layer facilitates increasing effectiveness in understanding Big Data, but also substantially increases the computational overhead for processing the representations of knowledge—decreasing efficiency. A solution is needed that harmonically and rationally balances between the increase in the adequacy and the completeness of Big Data semantics, on the one hand, and the increase in computational complexity, on the other hand. A straightforward approach is using scalable infrastructures for processing knowledge representations. A vast body of related work focuses on elaborating this approach (e.g., Broekstra et al. 2002; Wielemaker et al. 2003; Cai and Frank 2004; DeCandia et al. 2007).

The reasons to qualifying this approach only as a mechanistic solution are

- Using distributed scalable infrastructures, such as clouds or grids, implies new implementation problems and computational overheads.
- Typical tasks for processing knowledge representations, such as reasoning, alignment, query formulation and transformation, etc., scale hardly (e.g., Oren et al. 2009; Urbani et al. 2009; Hogan et al. 2011)—more expressiveness implies harder problems in decoupling the fragments for distribution. Nontrivial optimization, approximation, or load-balancing techniques are required.

Another effective approach to balance complexity and timeliness is maintaining history or learning from the past. A simple but topical example in data processing is the use of previously acquired information for saving approximately 50% of comparison operations in sorting by selection (Knuth 1998, p. 141). In Distributed Artificial Intelligence software, agent architectures maintaining their states or history for more efficient and effective deliberation have also been developed (cf. Dickinson and Wooldridge 2003). In Knowledge Representation and Reasoning, maintaining history is often implemented as inference or query result materialization (cf. Kontchakov et al. 2010; McGlothlin and Khan 2010), which also do not scale well up to the volumes characterizing real Big Data.

Yet another way to find a proper balance is exploiting incomplete or approximate methods. These methods yield results of acceptable quality much faster than approaches aiming at building fully complete or exact, that is, ideal results. Good examples of technologies for incomplete or partial reasoning and approximate query answering (Fensel et al. 2008) are elaborated in the FP7 LarKC project (larkc.eu). Remarkably, some of the approximate querying techniques, for example, Guéret et al. (2008), are based on evolutionary computing.

© 2014 by Taylor & Francis Group, LLC

Refining Big Data Semantics Layer for Balancing Efficiency Effectiveness

As one may notice, the developments in the Big Data semantics layer are mainly focused on posing and appropriately transforming the semantics of queries all the way down to the available data, using networked ontologies.

At least two shortcomings of this, in fact, unidirectional* approach need to be identified:

1. *Scalability overhead implies insufficient efficiency*. Indeed, executing queries at the data layer implies processing volumes at the scale of stored data. Additional overhead is caused by the query transformation, distribution, and planning interfaces. Lifting up the stack and fusing the results of these queries also imply similar computational overheads. A possible solution for this problem may be sought following a supposition that the volume of knowledge describing data adequately for further analyses is substantially smaller than the volume of this data. Hence, down-lifting queries for execution need to be stopped at the layer of knowledge storage for better efficiency. However, the knowledge should be consistent enough with the data so that it can fulfill completeness and correctness requirements specified in the contract of the query engine.
2. *Having ontologies inconsistent with data implies effectiveness problems*. Indeed, in the vast majority of cases, the ontologies containing knowledge about data are not updated consistently with the changes in data. At best, these knowledge representations are revised in a sequence of discrete versions. So, they are not consistent with the data at an arbitrary point in time. This shortcoming may be overcome only if ontologies in a knowledge repository evolve continuously in response to data change. Ontology evolution will have a substantially lower overhead because the volume of changes is always significantly lower than the volume of data, though depends on data velocity (Figure 1.3).

To sum up, relaxing the consequences of the two outlined shortcomings and, hence, balancing efficiency and effectiveness may be achievable if a bidirectional processing approach is followed. Top-down query answering has to be complemented by a bottom-up ontology evolution process, which meet at the knowledge representation layer. In addition to a balance between efficiency and effectiveness, such an approach of processing huge data sets may help us "... find and see dynamically changing ontologies without hav-

* Technologies for information and knowledge extraction are also developed and need to be regarded as bottom-up. However, these technologies are designed to work off-line for updating the existing ontologies in a discrete manner. Their execution is not coordinated with the top-down query processing and data changes. So, the shortcomings outlined below persist.

© 2014 by Taylor & Francis Group, LLC

ing to try to prescribe them in advance.* Taxonomies and ontologies are things that you might discover by observation, and watch evolve over time" (cf. Bollier 2010).

Further, we focus on outlining a complementary bottom-up path in the overall processing stack which facilitates existing top-down query answering frameworks by providing knowledge evolution in line with data change—as pictured in Figure 1.5. In a nutshell, the proposed bottom-up path is characterized by:

- Efficiently performing simple scalable queries on vast volumes of data or in a stream window for extracting facts and decreasing volumes (more details could be found in the "Scaling with a Traditional Database" section)
- Adding extracted facts to a highly expressive persistent knowledge base allowing evolution of knowledge (more details on that could be seen in Knowledge Self-Management and Refinement through Evolution)
- Assessing fitness of knowledge organisms and knowledge representations in the evolving knowledge ecosystem (our approach to that is also outlined in the "Knowledge Self-Management and Refinement through Evolution" section)

This will enable reducing the overheads of the top-down path by performing refined inference using highly expressive and complex queries over

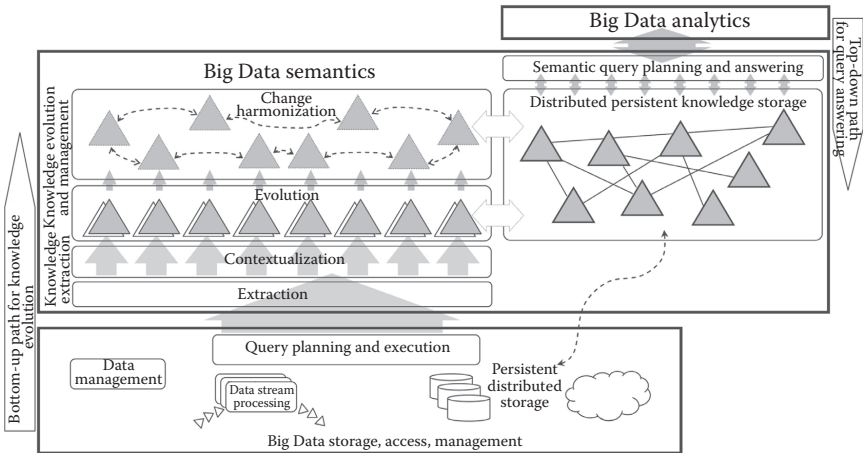


FIGURE 1.5
Refining Big Data semantics layer for balancing efficiency and effectiveness.

* Underlined by the authors of this chapter.

© 2014 by Taylor & Francis Group, LLC

evolving (i.e., consistent with data) and linked (i.e., harmonized), but reasonably small fragments of knowledge. Query results may also be materialized for further decreasing computational effort.

After outlining the abstract architecture and the bottom-up approach, we will now explain at a high level how Big Data needs to be treated along the way. A condensed formula for this high-level approach is "3F + 3Co" which is unfolded as

3F: Focusing-Filtering-Forgetting

3Co: Contextualizing-Compressing-Connecting

Notably, both 3F and 3Co are not novel and used in parts extensively in many domains and in different interpretations. For example, an interesting interpretation of 3F is offered by Dean and Webb (2011) who suggest this formula as a "treatment" for senior executives (CEOs) to deal with information overload and multitasking. Executives are offered to cope with the problem by focusing (doing one thing at a time), filtering (delegating so that they do not take on too many tasks or too much information), and forgetting (taking breaks and clearing their minds).

Focusing

Following our Boeing example, let us imagine a data analyst extracting knowledge tokens from a business news stream and putting these tokens as missing bits in the mosaic of his mental picture of the world. A tricky part of his work, guided by intuition or experience in practice, is choosing the order in which the facts are picked up from the token. Order of focusing is very important as it influences the formation and saturation of different fragments in the overall canvas. Even if the same input tokens are given, different curves of focusing may result in different knowledge representations and analysis outcomes.

A similar aspect of proper focusing is of importance also for automated processing of Big Data or its semantics. One could speculate whether a processing engine should select data tokens or assertions in the order of their appearance, in a reversed order, or anyhow else. If data or assertions are processed in a stream window and in real time, the order of focusing is of lesser relevance. However, if all the data or knowledge tokens are in a persistent storage, having some intelligence for optimal focusing may improve processing efficiency substantially. With smart focusing at hand, a useful token can be found or a hidden pattern extracted much faster and without making a complete scan of the source data. A complication for smart focusing is that the nodes on the focusing curve have to be decided upon on-the-fly because generally the locations of important tokens cannot be known in advance. Therefore, the processing of a current focal point should not only yield what

© 2014 by Taylor & Francis Group, LLC

is intended directly of this portion of data, but also hint about the next point on the curve.

A weak point in such a "problem-solving" approach is that some potentially valid alternatives are inevitably lost after each choice made on the decision path. So, only a suboptimal solution is practically achievable. The evolutionary approach detailed further in section "Knowledge Self-Management and Refinement through Evolution" follows, in fact, a similar approach of smart focusing, but uses a population of autonomous problem-solvers operating concurrently. Hence, it leaves a much smaller part of a solution space without attention, reduces the bias of each choice, and likely provides better results.

Filtering

A data analyst who receives dozens of news posts at once has to focus on the most valuable of them and filter out the rest which, according to his informed guess, do not bring anything important additionally to those in his focus. Moreover, it might also be very helpful to filter out noise, that is, irrelevant tokens, irrelevant dimensions of data, or those bits of data that are unreadable or corrupted in any other sense. In fact, an answer to the question about what to trash and what to process needs to be sought based on the understanding of the objective (e.g., was the reason for Boeing to hike their market forecast valid?) and the choice of the proper context (e.g., should we look into the airline fleets or the economic situation in developing countries?).

A reasonable selection of features for processing or otherwise a rational choice of the features that may be filtered out may essentially reduce the volume as well as the variety/complexity of data which result in higher efficiency balanced with effectiveness.

Quite similar to focusing, a complication here is that for big heterogeneous data it is not feasible to expect a one-size-fits-all filter in advance. Even more, for deciding about an appropriate filtering technique and the structure of a filter to be applied, a focused prescan of data may be required, which implies a decrease in efficiency. The major concern is again how to filter in a smart way and so as to balance the intentions to reduce processing effort (efficiency) and keep the quality of results within acceptable bounds (effectiveness).

Our evolutionary approach presented in the section "Knowledge Self-Management and Refinement through Evolution" uses a system of environmental contexts for smart filtering. These contexts are not fixed but may be adjusted by several independent evolutionary mechanisms. For example, a context may become more or less "popular" among the knowledge organisms that harvest knowledge tokens in them because these organisms may migrate freely between contexts in search for better, more appropriate, healthier knowledge to collect. Another useful property we propose for knowledge organisms is their resistance to sporadic mutagenic factors, which may be helpful for filtering out noise.

© 2014 by Taylor & Francis Group, LLC

Forgetting

A professional data analyst always keeps a record of data he used in his work and the knowledge he created in his previous analyses. The storage for all these gems of expertise is, however, limited, so it has to be cleaned periodically. Such a cleaning implies trashing potentially valuable things, though never or very rarely used, but causing doubts and further regrets about the lost. Similar thing happens when Big Data storage is overflown—some parts of it have to be trashed and so "forgotten." A question in this respect is about which part of a potentially useful collection may be sacrificed. Is forgetting the oldest records reasonable?—perhaps not. Shall we forget the features that have been previously filtered out?—negative again. There is always a chance that an unusual task for analysis pops up and requires the features never exploited before. Are the records with minimal potential utility the best candidates for trashing?—could be a rational way to go, but how would their potential value be assessed?

Practices in Big Data management confirm that forgetting following straightforward policies like fixed lifetime for keeping records causes regret almost inevitably. For example, the Climate Research Unit (one of the leading institutions that study natural and anthropogenic climate change and collect climate data) admits that they threw away the key data to be used in global warming calculations (Joseph 2012).

A better policy for forgetting might be to extract as much as possible knowledge out of data before deleting these data. It cannot be guaranteed, however, that future knowledge mining and extraction algorithms will not be capable of discovering more knowledge to preserve. Another potentially viable approach could be "forgetting before storing," that is, there should be a pragmatic reason to store anything. The approach we suggest in the section "Knowledge Self-Management and Refinement through Evolution" follows exactly this way. Though knowledge tokens are extracted from all the incoming data tokens, not all of them are consumed by knowledge organisms, but only those assertions that match to their knowledge genome to a sufficient extent. This similarity is considered a good reason for remembering a fact. The rest remains in the environment and dies out naturally after the lifetime comes to end as explained in "Knowledge Self-Management and Refinement through Evolution".

Contextualizing

Our reflection of the world is often polysemic, so a pragmatic choice of a context is often needed for proper understanding. For example, "taking a mountain hike" or "hiking a market forecast" are different actions though the same lexical root is used in the words. An indication of a context: recreation or business in this example would be necessary for making the statement explicit. To put it even broader, not only the sense of statements, but

© 2014 by Taylor & Francis Group, LLC

also judgments, assessments, attitudes, and sentiments about the same data or knowledge token may well differ in different contexts. When it goes about data, it might be useful to know:

1. The "context of origin"—the information about the source; who organized and performed the action; what were the objects; what features have been measured; what were the reasons or motives for collecting these data (transparent or hidden); when and where the data were collected; who were the owners; what were the license, price, etc.
2. The "context of processing"—formats, encryption keys, used preprocessing tools, predicted performance of various data mining algorithms, etc.; and
3. The "context of use"—potential domains, potential or known applications, which may use the data or the knowledge extracted from it, potential customers, markets, etc.

Having circumscribed these three different facets of context, we may say now that data contextualization is a transformation process which decontextualizes the data from the context of origin and recontextualizes it into the context of use (Thomason 1998), if the latter is known. This transformation is performed via smart management of the context of processing.

Known data mining methods are capable of automatically separating the so-called "predictive" and "contextual" features of data instances (e.g., Terziyan 2007). A predictive feature stands for a feature that directly influences the result of applying to data a knowledge extraction instrument—knowledge discovery, prediction, classification, diagnostics, recognition, etc.

RESULT = INSTRUMENT(Predictive Features).

Contextual features could be regarded as arguments to a meta-function that influences the choice of appropriate (based on predicted quality/performance) instrument to be applied to a particular fragment of data:

INSTRUMENT = CONTEXTUALIZATION(Contextual Features).

Hence, a correct way to process each data token and benefit of contextualization would be: (i) decide, based on contextual features, which would be an appropriate instrument to process the token; and then (ii) process it using the chosen instrument that takes the predictive features as an input. This approach to contextualization is not novel and is known in data mining and knowledge discovery as a "dynamic" integration, classification, selection, etc. Puuronen et al. (1999) and Terziyan (2001) proved that the use of dynamic contextualization in knowledge discovery yields essential quality improvement compared to "static" approaches.

© 2014 by Taylor & Francis Group, LLC

Compressing

In the context of Big Data, having data in a compact form is very important for saving storage space or reducing communication overheads. Compressing is a process of data transformation toward making data more compact in terms of required storage space, but still preserving either fully (lossless compression) or partly (lossy compression) the essential features of these data—those potentially required for further processing or use.

Compression, in general, and Big Data compression, in particular, are effectively possible due to a high probability of the presence of repetitive, periodical, or quasi-periodical data fractions or visible trends within data. Similar to contextualization, it is reasonable to select an appropriate data compression technique individually for different data fragments (clusters), also in a dynamic manner and using contextualization. Lossy compression may be applied if it is known how data will be used, at least potentially. So that some data fractions may be sacrificed without losing the facets of semantics and the overall quality of data required for known ways of its use. A relevant example of a lossy compression technique for data having quasi-periodical features and based on a kind of "meta-statistics" was reported by Terziyan et al. (1998).

Connecting

It is known that nutrition is healthy and balanced if it provides all the necessary components that are further used as building blocks in a human body. These components become parts of a body and are tightly connected to the rest of it. Big Data could evidently be regarded as nutrition for knowledge economy as discussed in "Motivation and Unsolved Issues". A challenge is to make this nutrition healthy and balanced for building an adequate mental representation of the world, which is Big Data understanding. Following the allusion of human body morphogenesis, understanding could be simplistically interpreted as connecting or linking new portions of data to the data that is already stored and understood. This immediately brings us about the concept of linked data (Bizer et al. 2009), where "linked" is interpreted as a sublimate of "understood." We have written "a sublimate" because having data linked is not yet sufficient, though necessary for further, more intelligent phase of building knowledge out of data. After data have been linked, data and knowledge mining, knowledge discovery, pattern recognition, diagnostics, prediction, etc. could be done more effectively and efficiently. For example, Terziyan and Kaykova (2012) demonstrated that executing business intelligence services on top of linked data is noticeably more efficient than without using linked data. Consequently, knowledge generated out of linked data could also be linked using the same approach, resulting in the linked knowledge. It is clear from the Linking Open Data Cloud Diagram by Richard Cyganiak and Anja Jentzsch (lod-cloud.net) that knowledge

© 2014 by Taylor & Francis Group, LLC

(e.g., RDF or OWL modules) represented as a linked data can be relatively easily linked to different public data sets, which creates a cloud of linked open semantic data.

Mitchell and Wilson (2012) argue that the key to extract value from Big Data lies in exploiting the concept of linked. They believe that linked data potentially creates ample opportunities from numerous data sources. For example, using links between data as a "broker" brings more possibilities of extracting new data from the old, creating insights that were previously unachievable, and facilitating exciting new scenarios for data processing.

For developing an appropriate connection technology, the results are relevant from numerous research and development efforts, for example, Linking Open Data (LOD; w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData) project, DBpedia (dbpedia.org), OpenCyc (opencyc.org), FOAF (foaf-project.org), CKAN (ckan.org), Freebase (freebase.com), Factual (factual.com), and INSEMTIVES (insemtives.eu/index.php). These projects create structured and interlinked semantic content, in fact, mashing up the features from Social and Semantic Web (Ankolekar et al. 2007). One strength of their approach is that collaborative content development effort is propagated up the level of the data-processing stack which allows creating semantic representations collaboratively and in an evolutionary manner.

Autonomic Big Data Computing

The treatment offered in the "Refining Big Data Semantics Layer for Balancing Efficiency-Effectiveness" section requires a paradigm shift in Big Data computing. In seeking for a suitable approach to building processing infrastructures, a look into Autonomic Computing might be helpful. Started by International Business Machines (IBM) in 2001, Autonomic Computing refers to the characteristics of complex computing systems allowing them to manage themselves without direct human intervention. A human, in fact, defines only general policies that constrain self-management process. According to IBM,* the four major functional areas of autonomic computing are: (i) *self-configuration*—automatic configuration of system components; (ii) *self-optimization*—automatic monitoring and ensuring the optimal functioning of the system within defined requirements; (iii) *self-protection*—automatic identification and protection from security threats; and (iv) *self-healing*—automatic fault discovery and correction. Other important capabilities of autonomic systems are: *self-identity* in a sense of being capable of knowing itself, its parts, and resources; *situatedness and self-adaptation*—sensing the influences from its environment and acting accordingly to what happens in the observed environment and a particular context; being *non-proprietary* in a sense of not constraining itself to a closed world but being capable of functioning in a heterogeneous word of open standards; and *anticipatory* in

* research.ibm.com/autonomic/overview/elements.html (accessed October 10, 2012).

© 2014 by Taylor & Francis Group, LLC

a sense of being able to automatically anticipate needed resources and seamlessly bridging user tasks to their technological implementations hiding complexity.

However, having an autonomic system for processing Big Data semantics might not be sufficient. Indeed, even such a sophisticated entity system may once face circumstances which it would not be capable of reacting to by reconfiguration. So, the design objectives will not be met by such a system and it should qualify itself as not useful for further exploitation and die. A next-generation software system will then be designed and implemented (by humans) which may inherit some valid features from the ancestor system but shall also have some principally new features. Therefore, it needs to be admitted that it is not always possible for even an autonomic system to adapt itself to a change within its lifetime. Consequently, self-management capability may not be sufficient for the system to survive autonomously—humans are required for giving birth to ancestors. Hence, we are coming to the necessity of a self-improvement feature which is very close to evolution. In that we may seek for inspiration in bio-social systems. Nature offers an automatic tool for adapting biological species across generations named genetic evolution. An evolutionary process could be denoted as the process of proactive change of the features in the populations of (natural or artificial) life forms over successive generations providing diversity at every level of life organization. Darwin (1859) put the following principles in the core of his theory:

- Principle of variation (variations of configuration and behavioral features);
- Principle of heredity (a child inherits some features from its parents);
- Principle of natural selection (some features make some individuals more competitive than others in getting needed for survival resources).

These principles may remain valid for evolving software systems, in particular, for Big Data computing. Processing knowledge originating from Big Data may, however, imply more complexity due to its intrinsic social features.

Knowledge is a product that needs to be shared within a group so that survivability and quality of life of the group members will be higher than those of any individual alone. Sharing knowledge facilitates collaboration and improves individual and group performance. Knowledge is actively consumed and also left as a major inheritance for future generations, for example, in the form of ontologies. As a collaborative and social substance, knowledge and cognition evolve in a more complex way for which additional facets have to be taken into account such as social or group focus of attention, bias, interpretation, explicitation, expressiveness, inconsistency, etc.

In summary, it may be admitted that Big Data is collected and supervised by different communities following different cultures, standards,

© 2014 by Taylor & Francis Group, LLC

objectives, etc. Big Data semantics is processed using naturally different ontologies. All these loosely coupled data and knowledge fractions in fact "live their own lives" based on very complex processes, that is, evolve following the evolution of these cultures, their cognition mechanisms, standards, objectives, ontologies, etc. An infrastructure for managing and understanding such data straightforwardly needs to be regarded as an ecosystem of evolving processing entities. Below we propose treating ontologies (a key for understanding Big Data) as genomes and bodies of those knowledge processing entities. For this, basic principles by Darwin are applied to their evolution aiming to get optimal or quasi-optimal (according to evolving definition of the quality) populations of knowledge species. These populations represent the evolving understanding of the respective islands of Big Data in their dynamics. This approach to knowledge evolution will require interpretation and implementation of concepts like "birth," "death," "morphogenesis," "mutation," "reproduction," etc., applied to knowledge organisms, their groups, and environments.

Scaling with a Traditional Database

In some sense, "Big data" is a term that is increasingly being used to describe very large volumes of unstructured and structured content—usually in amounts measured in terabytes or petabytes—that enterprises want to harness and analyze.

Traditional *relational* database management technologies, which use indexing for speedy data retrieval and complex query support, have been hard pressed to keep up with the data insertion speeds required for big data analytics. Once a database gets bigger than about half a terabyte, some database products' ability to rapidly accept new data start [start is to database products] to decrease.

There are two kinds of scalability, namely vertical and horizontal. Vertical scaling is just adding more capacity to a single machine. Fundamentally, every database product is vertically scalable to the extent that they can make good use of more central processing unit cores, random access memory, and disk space. With a horizontally scalable system, it is possible to add capacity by adding more machines. Beyond doubt, most database products are *not* horizontally scalable.

When an application needs more write capacity than they can get out of a single machine, they are required to shard (partition) their data across multiple database servers. This is how companies like Facebook (facebook.com) or Twitter (twitter.com) have scaled their MySQL installations to massive proportions. This is the closest to what one can get into horizontal scalability with database products.

© 2014 by Taylor & Francis Group, LLC

Sharding is a client-side affair, that is, the database server does not do it for user. In this kind of environment, when someone accesses data, the data access layer uses consistent hashing to determine which machine in the cluster a precise data should be written to (or read from). Enhancing capacity to a sharded system is a process of *manually* rebalancing the data across the cluster. The database system itself takes care of rebalancing the data and guaranteeing that it is adequately replicated across the cluster. This is what it means for a database to be horizontally scalable.

In many cases, constructing Big Data systems on premise provides better data flow performance, but requires a greater capital investment. Moreover, one has to consider the growth of the data. While many model linear growth curves, interestingly the patterns of data growth within Big Data systems are more exponential. Therefore, model both technology and costs to match up with sensible growth of the database so that the growth of the data flows.

Structured data transformation is the traditional approach of changing the structure of the data found within the source system to the structure of the target system, for instance, a Big Data system. The advantage of most Big Data systems is that deep structure is not a requirement; without doubt, structure can typically be layered in after the data arrive at the goal. However, it is a best practice to form the data within the goal. It should be a good abstraction of the source operational databases in a structure that allows those who analyze the data within the Big Data system to effectively and efficiently find the data required. The issue to consider with scaling is the amount of latency that transformations cause as data moves from the source(s) to the goal, and the data are changed in both structure and content. However, one should avoid complex transformations as data migrations for operational sources to the analytical goals. Once the data are contained within a Big Data system, the distributed nature of the architecture allows for the gathering of the proper result set. So, transformations that cause less latency are more suitable within Big Data domain.

Large Scale Data Processing Workflows

Overall infrastructure for many Internet companies can be represented as a pipeline with three layers: Ingestion, Storage & Processing, and Serving. The most vital among the three is the Storage & Processing layer. This layer can be represented as a multisub-layer stack with a scalable file system such as Google File System (Ghemawat et al. 2003) at the bottom, a framework for distributed sorting and hashing, for example, Map-Reduce (Dean and Ghemawat 2008) over the file system layer, a dataflow programming framework over the map-reduce layer, and a workflow manager at the top.

Debugging large-scale data, in the Internet firms, is crucial because data passes through many subsystems, each having different query interface, different metadata representation, different underlying models (some have files, some have records, some have workflows), etc. Thus, it is hard

© 2014 by Taylor & Francis Group, LLC

to maintain consistency and it is essential to factor out the debugging from the subsystems. There should be a self-governing system that takes care of all the metadata management. All data-processing subsystems can dispatch their metadata to such system which absorbs all the metadata, integrates them, and exposes a query interface for all metadata queries. This can provide a uniform view to users, factors out the metadata management code, and decouples metadata lifetime from data/subsystem lifetime.

Another stimulating problem is to deal with different data and process granularity. Data granularity can vary from a web page, to a table, to a row, to a cell. Process granularity can vary from a workflow, to a map-reduce program, to a map-reduce task. It is very hard to make an inference when the given relationship is in one granularity and the query is in other granularity and therefore it is vital to capture provenance data across the workflow. While there is no one-size-fits-all solution, a good methodology could be to use the best granularity at all levels. However, this may cause a lot of overhead and thus some smart domain-specific techniques need to be implemented (Lin and Dyer 2010; Olston 2012).

Knowledge Self-Management and Refinement through Evolution

World changes—so do the beliefs and reflections about it. Those beliefs and reflections are the knowledge humans have about their environments. However, the nature of those changes is different. The world just changes in events. Observation or sensing (Ermolayev et al. 2008) of events invokes generation of data—often in huge volumes and with high velocities. Humans evolve—adapt themselves to become better fitted to the habitat.

Knowledge is definitely a product of some processes carried out by conscious living beings (for example, humans). Following Darwin's (1859) approach and terminology to some extent, it may be stated that knowledge, both in terms of scope and quality, makes some individuals more competitive than others in getting vital resources or at least for improving their quality of life. The major role of knowledge as a required feature for survival is decision-making support. Humans differ in fortune and fate because they make different choices in similar situations, which is largely due to their possession of different knowledge. So, the evolution of conscious beings noticeably depends on the knowledge they possess. On the other hand, making a choice in turn triggers the production of knowledge by a human. Therefore, it is natural to assume that knowledge evolves triggered by the evolution of conscious beings, their decision-making needs and taken decisions, quality standards, etc. To put both halves in one whole, knowledge evolves in support of and to support the proactive needs of the owners more effectively, for

© 2014 by Taylor & Francis Group, LLC

example, to better interpret or explain the data generated when observing events, corresponding to the diversity and complexity of these data. This observation leads us to a hypothesis about the way knowledge evolves:

> The mechanisms of knowledge evolution are very similar to the mechanisms of biological evolution. Hence, the methods and mechanisms for the evolution of knowledge could be spotted from the ones enabling the evolution of living beings.

In particular, investigating the analogies and developing the mechanisms for the evolution of formal knowledge representations—specified as ontologies—is of interest for the Big Data semantics layer (Figure 1.5). The triggers for ontology evolution in the networked and interlinked environments could be external influences coming bottom-up from external and heterogeneous information streams.

Recently, the role of ontologies as formal and consensual knowledge representations has become established in different domains where the use of knowledge representations and reasoning is an essential requirement. Examples of these domains range from heterogeneous sensor network data processing through the Web of Things to Linked Open Data management and use. In all these domains, distributed information artifacts change sporadically and intensively in reflection of the changes in the world. However, the descriptions of the knowledge about these artifacts do not evolve in line with these changes.

Typically, ontologies are changed semiautomatically or even manually and are available in a sequence of discrete revisions. This fact points out a serious disadvantage of ontologies built using state-of-the-art knowledge engineering and management frameworks and methodologies: expanding and amplified distortion between the world and its reflection in knowledge. It is also one of the major obstacles for a wider acceptance of semantic technologies in industries (see also Hepp 2007; Tatarintseva et al. 2011).

The diversity of domain ontologies is an additional complication for proper and efficient use of dynamically changing knowledge and information artifacts for processing Big Data semantics. Currently, the selection of the best suiting one for a given set of requirements is carried out by a knowledge engineer using his/her subjective preferences. A more natural evolutionary approach for selecting the best-fitting knowledge representations promises enhancing robustness and transparency, and seems to be more technologically attractive.

Further, we elaborate a vision of a knowledge evolution ecosystem where agent-based software entities carry their knowledge genomes in the form of ontology schemas and evolve in response to the influences percieved from their environments. These influences are thought of as the tokens of Big Data (like news tokens in the "Illustrative Example" section) coming into the species' environments. Evolution implies natural changes in the ontologies which

© 2014 by Taylor & Francis Group, LLC

reflect the change in the world snap-shotted by Big Data tokens. Inspiration and analogies are taken from evolutionary biology.

Knowledge Organisms, their Environments, and Features

Evolving software entities are further referred to as individual *Knowledge Organisms* (KO). It is envisioned (Figure 1.6) that a KO:

1. Is *situated in its environment* as described in "Environment, Perception (Nutrition), and Mutagens"
2. Carries its individual *knowledge genome* represented as a schema or Terminological Box (TBox; Nardi and Brachman 2007) of the respective ontology (see "Knowledge Genome and Knowledge Body")
3. Has its individual *knowledge body* represented as an assertional component (ABox; Nardi and Brachman 2007) of the respective ontology (see "Knowledge Genome and Knowledge Body")
4. Is capable of *perceiving* the influences from the environment in the form of knowledge tokens (see "Environment, Perception (Nutrition), and Mutagens") that may cause the changes in the genome (see "Mutation") and body (see "Morphogenesis")—the *mutagens*
5. Is capable of *deliberating* about the *affected parts* of its genome and body (see "Morphogenesis" and "Mutation")

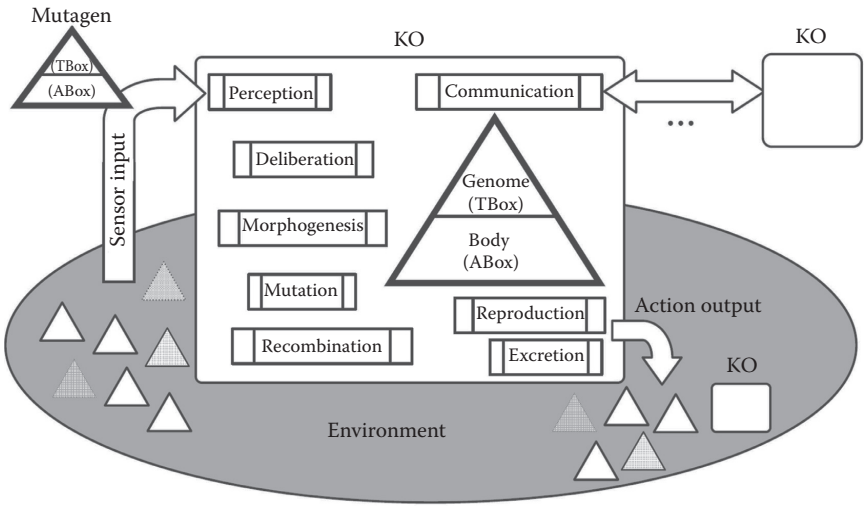


FIGURE 1.6
A Knowledge Organism: functionality and environment. Small triangles of different transparency represent knowledge tokens in the environment—consumed and produced by KOs. These knowledge tokens may also referred to as mutagens as they may trigger mutations.

© 2014 by Taylor & Francis Group, LLC

6. Is capable of consuming some parts of a mutagen for: (a) *morphogenesis* changing only the body (see "Morphogenesis"); (b) *mutation* changing both the genome and body (see "Mutation"); or (c) *recombination*—a mutual enrichment of several genomes in a group of KO which may trigger *reproduction*—recombination of body replicas giving "birth" to a new KO (see "Recombination and Reproduction")
7. Is capable of *excreting* the unused parts of mutagens or the "dead" parts of the body to the environment

The results of mainstream research in distributed artificial intelligence and semantic technologies suggest the following basic building blocks for developing a KO. The features of situatedness (Jennings 2000) and deliberation (Wooldridge and Jennings 1995) are characteristic to intelligent software agents, while the rest of the required functionality could be developed using the achievments in Ontology Alignment (Euzenat and Shvaiko 2007). Recombination involving a group of KOs could be thought of based on the known mechanisms for multiissue negotiations on semantic contexts (e.g., Ermolayev et al. 2005) among software agents—the members of a reproduction group.

Environment, Perception (Nutrition), and Mutagens

An environmental context for a KO could be thought of as an arial of its habitat. Such a context needs to be able to provide nutrition that is "healthy" for particular KO species, that is, matching their genome noticeably. The food for nutrition is provided by Knowledge Extraction and Contextualization functionality (Figure 1.7) in a form of knowledge tokens. Hence, several

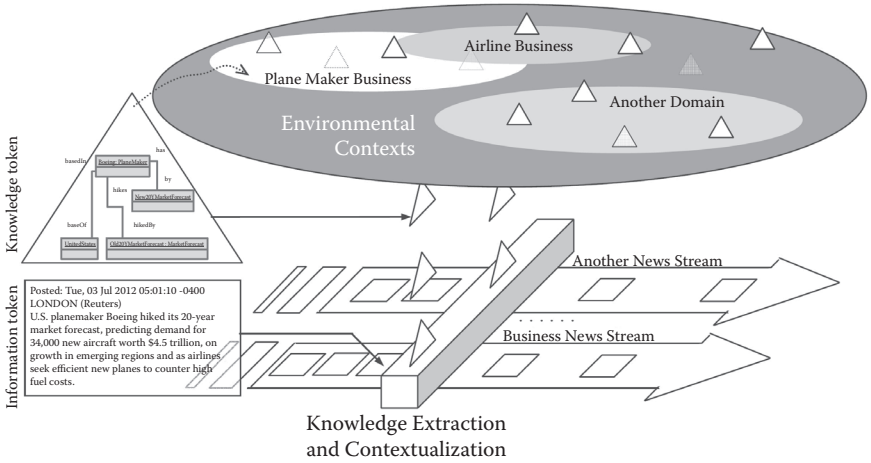


FIGURE 1.7
Environmental contexts, knowledge tokens, knowledge extraction, and contextualization.

© 2014 by Taylor & Francis Group, LLC

and possibly overlapping environmental contexts need to be regarded in a hierarchy which corresponds to several subject domains of intetrest and a foundational knowledge layer. By saying this, we subsume that there is a single domain or foundational ontology module schema per environmental context. Different environmental contexts corresponding to different subject domais of interest are pictured as ellipses in Figure 1.7.

Environmental contexts are sowed with knowledge tokens that correspond to their subject domains. It might be useful to limit the lifetime of a knowledge token in an environment—those which are not consumed dissolve finally when their lifetime ends. Fresh and older knowledge tokens are pictured with different transparency in Figure 1.7.

KOs inhabit one or several overlapping environmental contexts based on the nutritional healthiness of knowledge tokens sowed there, that is, the degree to which these knowledge tokens match to the genome of a particular KO. KOs use their perceptive ability to find and consume knowledge tokens for nutrition. A KO may decide to migrate from one environment to another based on the availability of healthy food there. Knowledge tokens that only partially match KOs' genome may cause both KO body and genome changes and are thought of as mutagens. Mutagens, in fact, deliver the information about the changes in the world to the environments of KOs.

Knowledge tokens are extracted from the information tokens either in a stream window or from the updates of the persistent data storage and further sawn in the appropriate environmental context. The context for placing a newly coming KO is chosen by the contextualization functionality (Figure 1.7) based on the match ratio to the ontology schema characterizing the context in the environment. Those knowledge tokens that are not mapped well to any of the ontology schemas are sown in the environment without attributing them to any particular context.

For this, existing shallow knowledge extraction techniques could be exploited, for example, Fan et al. (2012a). The choice of appropriate techniques depends on the nature and modality of data. Such a technique would extract several interrelated assertions from an information token and provide these as a knowledge token coded in a knowledge representation language of an appropriate expressiveness, for example, in a tractable subset of the Web Ontology Language (OWL) 2.0 (W3C 2009). Information and knowledge* tokens for the news item of our Boeing example are pictured in Figure 1.7.

* Unified Modeling Language (UML) notation is used for picturing the knowledge token in Figure 1.7 because it is more illustrative. Though not shown in Figure 1.7, it can be straightforwardly coded in OWL, following, for example, Kendall et al. (2009).

© 2014 by Taylor & Francis Group, LLC

Knowledge Genome and Knowledge Body

Two important aspects in contextualized knowledge representation for an outlined knowledge evolution ecosystem have to be considered with care (Figure 1.8):

- A knowledge genome etalon for a population of KOs belonging to one species
- An individual knowledge genome and body for a particular KO

A knowledge genome etalon may be regarded as the schema (TBox) of a distinct ontology module which represents an outstanding context in a subject domain. In our proposal, the etalon genome is carried by a dedicated Etalon KO (EKO; Figure 1.8) to enable alignments with individual genomes and other etalons in a uniform way. The individual assertions (ABox) of this ontology module are spread over the individual KOs belonging to the corresponding species—forming their individual bodies.

The individual genomes of those KOs are the recombined genomes of the KOs who gave birth to this particular KO. At the beginning of times, the individual genomes may be replicas of the etalon genome. Anyhow, they evolve independently in *mutations* or because of *morphogenesis* of an individual KO, or because of *recombinations* in reproductive groups.

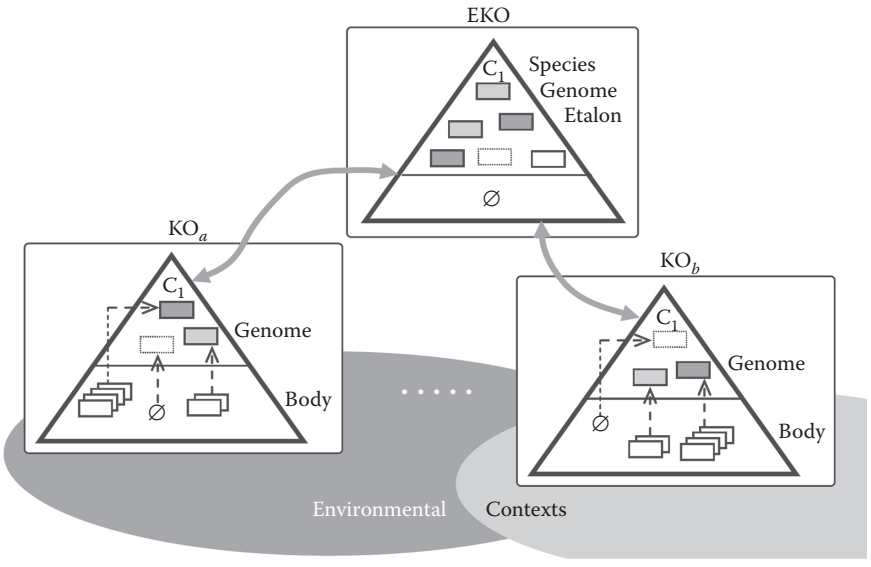


FIGURE 1.8
Knowledge genomes and bodies. Different groups of assertions in a KO body are attributed to different elements of its genome, as shown by dashed arrows. The more assertions relate to a genome element, the more dominant this element is as shown by shades of gray.

© 2014 by Taylor & Francis Group, LLC

Different elements (concepts, properties, axioms) in a knowledge genome may possess different strengths, that is, be dominant or recessive. For example (Figure 1.8) concept C_1 in the genome of KO_a is quite strong because it is reinforced by a significant number of individual assertions attributed to this concept, that is, dominant. On the contrary, C_1 in the genome of KO_b is very weak—that is, recessive—as it is not supported by individual assertions in the body of KO_b. Recessivness or dominance values may be set and altered using techniques like spreading activation (Quillian 1967, 1969; Collins and Loftus 1975) which also appropriately affect the structural contexts (Ermolayev et al. 2005, 2010) of the elements in focus.

Recessive elements may be kept in the genome as parts of the genetic memory, but until they do not contradict any dominant elements. For example, if a dominant property of the *PlaneMaker* concept in a particular period of time is *PlaneMaker–hikes–MarketForecast*, then a recessive property *PlaneMaker–lessens–MarketForecast* may die out soon with high probability, as contradictory to the corresponding dominant property.

The etalone genome of a species evolves in line with the evolution of the individual genomes. The difference, however, is that EKO has no direct relationship (situatedness) to any environmental context. So, all evolution influences are provided to EKO by the individual KOs belonging to the corresponding species via communication. If an EKO and KOs are implemented as agent-based software entities, the techniques like agent-based ontology alignment are of relevance for evolving etalon genomes. In particular, the alignment settings are similar to a Structural Dynamic Uni-directional Distributed (SDUD) ontology alignment problem (Ermolayev and Davidovsky 2012). The problem could be solved using multiissue negotiations on semantic contexts, for example, following the approach of Ermolayev et al. (2005) and Davidovsky et al. (2012). For assuring the consistency in the updated ontology modules after alignment, several approaches are applicable: incremental updates for atomic decompositions of ontology modules (Klinov et al. 2012); checking correctness of ontology contexts using ontology design patterns approach (Gangemi and Presutti 2009); evaluating formal correctness using formal (meta-) properties (Guarino and Welty 2001).

An interesting case would be if an individual genome of a particular KO evolves very differently to the rest of KOs in the species. This may happen if such a KO is situated in an environmental context substantially different from the context where the majority of the KOs of this species are collecting knowledge tokens. For example, the dominancy and recessiveness values in the genome of KO_b (Figure 1.8) differ noticeably from those of the genomes of the KOs similar to KO_a. A good reason for this may be: KO_b is situated in an environmental context different to the context of KO_a—so the knowledge tokens KO_b may consume are different to the food collected by KO_a. Hence, the changes to the individual genome of KO_b will be noticeably different to those of KO_a after some period of time. Such a *genetic drift* may cause that the structural difference in individual genomes goes beyond a threshold within

© 2014 by Taylor & Francis Group, LLC

which recombination gives ontologically viable posterity. A new knowledge genome etalon may, therefore, emerge if the group of the KOs with genomes drifted in a similar direction reaches a critical mass—giving birth to a new species.

The following are the features required to extend an ontology representation language for to cope with the mentioned evolutionary mechanisms:

- A temporal extension that allows representing and reasoning about the lifetime and temporal intervals of validity of the elements in knowledge genomes and bodies. One relevant extension and reasoning technique is OWL-MET (Keberle 2009).
- An extension that allows assigning meta-properties to ontological elements for verifying formal correctness or adherence to relevant design patterns. Relevant formalisms may be sought following Guarino and Welty (2001) or Gangemi and Presutti (2009).

Morphogenesis

Morphogenesis in a KO could be seen as a process of developing the shape of a KO body. In fact, such a development is done by adding new assertions to the body and attributing them to the correct parts of the genome. This process could be implemented using ontology instance migration technique (Davidovsky et al. 2011); however, the objective of morphogenesis differs from that of ontology instance migration. The task of the latter is to ensure correctness and completeness, that is, that, ideally, all the assertions are properly aligned with and added to the target ontology ABox. Morphogenesis requires that only the assertions that fit well to the TBox of the target ontology are consumed for shaping it out. Those below the fitness threshold are excreted. If, for example, a mutagen perceived by a KO is the one of our Boeing example presented in Figures 1.2 or 1.7, then the set of individual assertions will be*

{AllNipponAirways:Airline, B787-JA812A:EfficientNewPlane, Japan:Country, Boeing:PlaneMaker, New20YMarketForecastbyBoeing:MarketForecast, United States:Country, Old20YMarketForecastbyBoeing:MarketForecast}. (1.1)

Let us now assume that the genome (TBox) of the KO contains only the concepts represented in Figure 1.2 as grey-shaded classes—{Airline, PlaneMaker, MarketForecast} and thick-line relationships—{seeksFor–soughtBy}. Then only the bolded assertions from (1.1) could be consumed for morphogenesis by this KO and the rest have to be excreted back to the environment. Interestingly, the ratio of mutagen ABox consumption may be used as a good

* The syntax for representing individual assertions is similar to the syntax in UML for compatibility with Figure 1.2: ⟨assertion-name⟩:⟨concept-name⟩.

© 2014 by Taylor & Francis Group, LLC

metric for a KO in deliberations about: its resistance to mutations; desire to migrate to a different environmental context, or to start seeking for reproduction partners.

Another important case in a morphogenesis process is detecting a contradiction between a newly coming mutagenic assertion and the assertion that is in the body of the KO. For example, let us assume that the body already comprises the property *SalesVolume* of the assertion named *New20YMarketForecastbyBoeing* with the value of 2.1 million. The value of the same property coming with the mutagen equals to 4.5 million. So, the KO has to resolve this contradiction by: either (i) deciding to reshape its body by accepting the new assertion and excreting the old one; or (ii) resisting and declining the change. Another possible behavior would be collecting and keeping at hand the incoming assertions until their dominance is not proved by the quantity. Dominance may be assessed using different metrics. For example, a relevant technique is offered by the Strength Value-Based Argumentation Framework (Isaac et al. 2008).

Mutation

Mutation of a KO could be understood as the change of its genome caused by the environmental influences (mutagenic factors) coming with the consumed knowledge tokens. Similar to the biological evolution, a KO and its genome are resistent to mutagenic factors and do not change at once because of any incoming influence, but only because of those which could not be ignored because of their strength. Different genome elements may be differently resistant. Let us illustrate different aspects of mutation and resistance using our Boeing example. As depicted in Figure 1.9, the change of the *AirPlaneMaker* concept name (to *PlaneMaker*) in the genome did not happen though a new assertion had been added to the body as a result of morphogenesis (*Boeing*: (*PlaneMaker*) *AirPlaneMaker*[*]). The reason *AirPlaneMaker* concept resisted this mutation was that the assertions attributed to the concept of *PlaneMaker* were in the minority—so, the mutagenic factor has not yet been strong enough. This mutation will have a better chance to occur if similar mutagenic factors continue to come in and the old assertions in the body of the KO die out because their lifetime periods come to end. More generally, the more individual assertions are attributed to a genome element at a given point in time—the more strong this genome element is to mutations.

In contrast to the *AirPlaneMaker* case, the mutations brought by *hikes—hikedBy* and *successorOf—predecessorOf* object properties did happen (Figure 1.9) because the KO did not possess any (strong) argument to resist

[*] UML syntax is used as basic. The name of the class from the knowledge token is added in brackets before the name of the class to which the assertion is attributed in the KO body. This is done for keeping the information about the occurrences of a different name in the incoming knowledge tokens. This historical data may further be used for evaluating the strength of the mutagenic factor.

© 2014 by Taylor & Francis Group, LLC

them. Indeed, there were no contradictory properties both in the genome and the body of the KO before it accepted the grey-shaded assertions as a result of morphogenesis.

Not all the elements of an incoming knowledge token could be consumed by a KO. In our example (Figure 1.9), some of the structural elements (*AirLine*, *EfficientNewPlane*, *seeks—soughtBy*) were

- Too different to the genome of this particular KO, so the similarity factor was too low and the KO did not find any match to its TBox. Hence, the KO was not able to generate any replacement hypotheses also called propositional substitutions (Ermolayev et al. 2005).
- Too isolated from the elements of the genome—having no properties relating them to the genome elements. Hence, the KO was not able to generate any merge hypotheses.

These unused elements are excreted (Figure 1.9) back to the environment as a knowledge token. This token may further be consumed by another

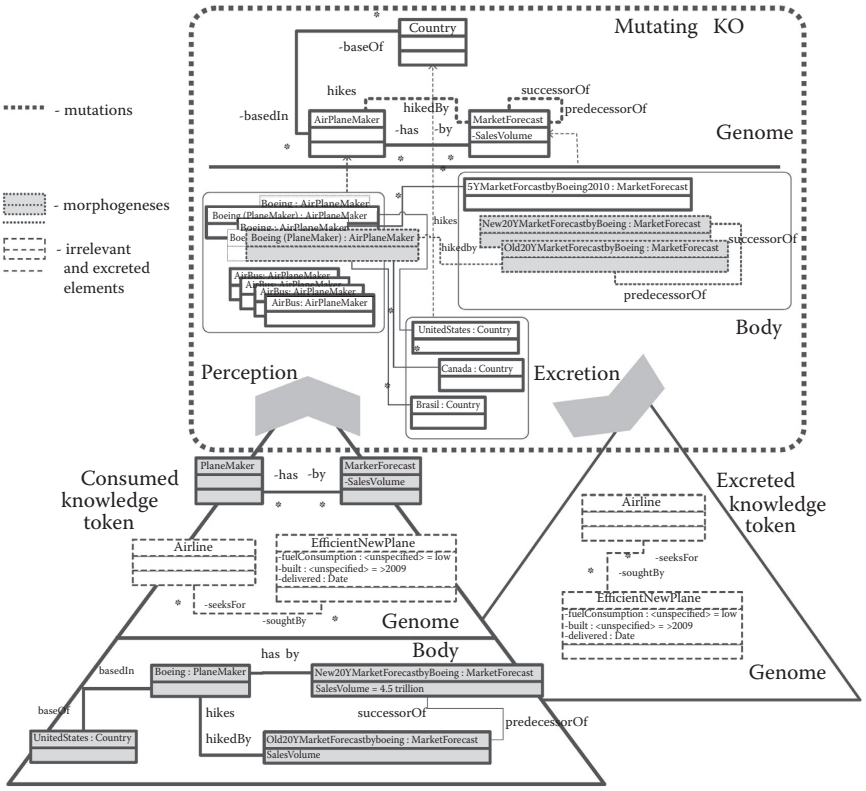


FIGURE 1.9
Mutation in an individual KO illustrated by our Boeing example.

© 2014 by Taylor & Francis Group, LLC

KO with different genome comprising matching elements. Such a KO may migrate from a different environmental context (e.g., Airlines Business).

Similar to morphogenesis, mutation may be regarded as a subproblem of ontology alignment. The focus is, however, a little bit different. In contrast to morphogenesis which was interpreted as a specific ontology instance migration problem, mutation affects the TBox and is therefore structural ontology alignment (Ermolayev and Davidovsky 2012). There is a solid body of related work in structural ontology alignment. Agent-based approaches relevant to our context are surveyed, for example, in Ermolayev and Davidovsky (2012).

In addition to the requirements already mentioned above, the following features extending an ontology representation language are essential for coping with the mechanisms of mutation:

- The information of the attribution of a consumed assertion to a particular structural element in the knowledge token needs to be preserved for future use in possible mutations. An example is given in Figure 1.8—*Boeing***:** *(PlaneMaker) AirPlaneMaker*. The name of the concept in the knowledge token (*PlaneMaker*) is preserved and the assertion is attributed to the *AirPlaneMaker* concept in the genome.

Recombination and Reproduction

As mutation, recombination is a mechanism of adapting KOs to environmental changes. Recombination involves a group of KOs belonging to one or several similar species with partially matching genomes. In contrast to mutation, recombination is triggered and performed differently. Mutation is invoked by external influences coming from the environment in the form of mutagens. Recombination is triggered by a conscious intention of a KO to make its genome more resistant and therefore better adapted to the environment in its current state. Conscious in this context means that a KO first analyzes the strength and adaptation of its genome, detects weak elements, and then reasons about the necessity of acquiring external reinforcements for these weak elements. Weaknesses may be detected by:

- Looking at the proportion of consumed and excreted parts in the perceived knowledge tokens—reasoning about how healthy is the food in its current environmental context. If not, then new elements extending the genome for increasing consumption and decreasing excretion may be desired to be acquired.
- Looking at the resistance of the elements in the genome to mutations. If weaknesses are detected, then it may be concluded that the assertions required for making these structural elements stronger are either nonexistent in the environmental context or are not consumed. In the latter case, a structural reinforcement by acquiring

© 2014 by Taylor & Francis Group, LLC

new genome elements through recombination may be useful. In the former case (nonexistence), the KO may decide to move to a different environmental context.

Recombination of KOs as a mechanism may be implemented using several available technologies. Firstly, a KO needs to reason about the strengths and weaknesses of the elements in its genome. For this, in addition to the extra knowledge representation language features mentioned above, it needs a simple reasoning functionality (pictured in Figure 1.6 as Deliberation). Secondly, a KO requires a means for getting in contact with the other KOs and checking if they have similar intentions to recombine their genomes. For this, the available mechanisms for communication (e.g., Labrou et al. 1999; Labrou 2006), meaning negotiation (e.g., Davidovsky et al. 2012), and coalition formation (e.g., Rahwan 2007) could be relevant.

Reproduction is based on recombination mechanism and results and goes further by combining the replicas of the bodies of those KOs who take part in the recombination group resulting in the production of a new KO. A KO may intend to reproduce itself because his lifetime period comes to an end or because of the other individual or group stimuli that have to be researched.

Populations of Knowledge Organisms

KOs may belong to different *species*—the groups of KOs that have similar genomes based on the same etalon carried by the EKO. KOs that share the same areal of habitat (environmental context) form the *population* which may comprise the representatives of several species. Environmental contexts may also overlap. So, the KOs of different species have possibilities to interact. With respect to species and populations, the mechanisms of (i) *migration*, (ii) *genetic drift*, (iii) *speciation*, and (iv) *breeding* for evolving knowledge representations are of interest.

Migration is the movement of KOs from one environmental context to another context because of different reasons mentioned in the "Knowledge Organisms, their Environments, and Features" section. Genetic drift is the change of genomes to a degree beyond the species tolerance (similarity) threshold caused by cumulative efffect of a series of mutations as explained in the "Knowledge Genome and Knowledge Body" section. Speciation effect occurs if genetic drift results in a distinct group of KOs capable of reproducing themselves with their recombined genomes.

If knowledge evolves in a way similar to biological evolution, the outcome of this process would best-fit KOs desires of environmental mimicry, but perhaps not the requirements of ontology users. Therefore, for ensuring human stakeholders' commitment to the ontology, it might be useful to keep the evolution process under control. For this, constraints, or restrictions in another form, may be introduced for relevant environmental contexts and fitness measurement functions so as to guide the evolution toward a desired

© 2014 by Taylor & Francis Group, LLC

goal. This artificial way of control over the natural evolutionary order of things may be regarded as breeding—a controlled process of sequencing desired mutations that causes the emergence of a species with the required genome features.

Fitness of Knowledge Organisms and Related Ontologies

It has been repeatedly stated in the discussion of the features of KOs in "Knowledge Organisms, their Environments, and Features" that they exhibit proactive behavior. One topical case is that a KO would rather migrate away from the current environmental context instead of continuing consuming knowledge tokens which are not healthy for it in terms of structural similarity to its genome. It has also been mentioned that a KO may cooperate with other KOs to fulfill its evolutionary intentions. For instance, KOs may form cooperative groups for recombination or reproduction. They also interact with their EKOs for improving the etalon genome of the species. Another valid case, though not mentioned in "Knowledge Organisms, their Environments, and Features", would be if a certain knowledge token is available in the environment and two or more KOs approach it concurrently with an intention to consume. If those KOs are cooperative, the token will be consumed by the one which needs it most—so that the overall "strength" of the species is increased. Otherwise, if the KOs are competitive, as it often happens in nature, the strongest KO will get the token. All these cases require a quantification of the strength, or fitness, of KOs and knowledge tokens. Fitness is, in fact, a complex metric having several important facets.

Firstly, we summarize what fitness of a KO means. We outline that their fitness is inseparable from (in fact, symmetric to) the fitness of the knowledge tokens that KOs consume from and excrete back to their environmental contexts. Then, we describe several factors which contribute to fitness. Finally, we discuss how several dimensions of fitness could be used to compare different KOs.

To start our deliberations about fitness, we have to map the high-level understanding of this metric to the requirements of Big Data processing as presented in the "Motivation and Unsolved Issues" and "State of Technology, Research, and Development in Big Data Computing" sections in the form of the processing stack (Figures 1.3 through 1.5). The grand objective of a Big Data computing system or infrastructure is providing a capability for data analysis with balanced effectiveness and efficiency. In particular, this capability subsumes facilitating decision-making and classification, providing adequate inputs to software applications, etc. An evolving knowledge ecosystem, comprising environmental contexts populated with Kos, is introduced in the semantics processing layer of the overall processing stack. The aim of introducing the ecosystem is to ensure seamless and balanced connection between a user who operates the system at the upper layers and the lower layers that provide data.

© 2014 by Taylor & Francis Group, LLC

Ontologies are the "blood and flesh" of the KOs and the whole ecosystem as they are both the code registering a desired evolutionary change and the result of this evolution. From the data-processing viewpoint, the ontologies are consensual knowledge representations that facilitate improving data integration, transformation, and interoperability between the processing nodes in the infrastructure. A seamless connection through the layers of the processing stack is facilitated by the way ontologies are created and changed. As already mentioned above in the introduction of the "Knowledge Self-Management and Refinement through Evolution" section, ontologies are traditionally designed beforehand and further populated by assertions taken from the source data. In our evolving ecosystem, ontologies evolve in parallel to data processing. Moreover, the changes in ontologies are caused by the mutagens brought by the incoming data. Knowledge extraction subsystem (Figure 1.7) transforms units of data to knowledge tokens. These in turn are sown in a corresponding environmental context by a contextualization subsystem and further consumed by KOs. KOs may change their body or even mutate due to the changes brought by consumed mutagenic knowledge tokens. The changes in the KOs are in fact the changes in the ontologies they carry. So, ontologies change seamlessly and naturally in a way to best suite the substance brought in by data. For assessing this change, the judgments about the value and appropriateness of ontologies in time are important. Those should, however, be formulated accounting for the fact that an ontology is able to self-evolve.

A degree to which an ontology is reused is one more important characteristic to be taken into account. Reuse means that data in multiple places refers to this ontology and when combined with interoperability it implies that data about similar things is described using the same ontological fragments. When looking at an evolving KO, having a perfect ontology would mean that if new knowledge tokens appear in the environmental contexts of an organism, the organism can integrate all assertions in the tokens, that is, without a need to excrete some parts of the consumed knowledge tokens back to the environment. That is to say, the ontology which was internal to the KO before the token was consumed was already prepared for the integration of the new token. Now, one could turn the viewpoint by saying that the information described in the token was already described in the ontology which the KO had and thus that the ontology was reused in one more place. This increases the value, that is, the fitness of the ontology maintained by the KO.

Using similar argumentation, we can conclude that if a KO needs to excrete a consumed knowledge token, the ontology fits worse to describing the fragment of data to which the excreted token is attributed. Thus, in conclusion, we could say that the fitness of a KO is directly dependent on the proportion between the parts of knowledge tokens which it: (a) is able to consume for morphogenesis and possibly mutation; versus (b) needs to excrete back to the environment. Additionally, the age of the assertions which build up the current knowledge body of a KO influences its quality. If the proportion

© 2014 by Taylor & Francis Group, LLC

of very young assertions in the body is high, the KO might be not resistant to stochastic changes, which is not healthy. Otherwise, if only long-living assertions form the body, it means that the KO is either in a wrong context or too resistant to mutagens. Both are bad as no new information is added, the KO ignores changes, and hence the ontology it carries may become irrelevant. Therefore, a good mix of young and old assertions in the body of a KO indicates high fitness—KO's knowledge is overall valid and evolves appropriately.

Of course stating that fitness depends only on the numbers of used and excreted assertions is an oversimplification. Indeed, incoming knowledge tokens that carry assertions may be very different. For instance, the knowledge token in our Boeing example contains several concepts and properties in its TBox: a Plane, a PlaneMaker, a MarketForecast, an Airline, a Country, SalesVolume, seeksFor—soughtBy, etc. Also, some individuals attributed to these TBox elements are given in the ABox: UnitedStates, Boeing, New20YMarketForecastByBoeing, 4.5 trillion, etc. One can imagine a less complex knowledge token which contains less information. In addition to size and complexity, a token has also other properties which are important to consider. One is the source where the token originates from. A token can be produced by knowledge extraction from a given channel or can be excreted by a KO. When the token is extracted from a channel, its value depends on the quality of the channel, relative to the quality of other channels in the system (see also the context of origin in the "Contextualizing" section). The quality of knowledge extraction is important as well, though random errors could be mitigated by statistical means. Further, a token could be attributed to a number of environmental contexts. A context is important, that is, adds more value to a token in the context if there are a lot of knowledge tokens in that context or more precisely there have appeared many tokens in the context recently. Consequently, a token becomes less valuable along its lifetime in the environment.

Till now, we have been looking at different fitness, value, and quality factors in insulation. The problem is, however, that there is no straightforward way to integrate these different factors. For this, an approach to address the problem of assessing the quality of an ontology as a dynamic optimization problem (Cochez and Terziyan 2012) may be relevant.

Some Conclusions

For all those who use or process Big Data a good mental picture of the world, dissolved in data tokens, may be worth of petabytes of raw information and save weeks of analytic work. Data emerge reflecting a change in the world. Hence, Big Data is a fine-grained reflection of the changes around

© 2014 by Taylor & Francis Group, LLC

us. Knowledge extracted from these data in an appropriate and timely way is an essence of adequate understanding of the change in the world. In this chapter, we provided the evidence that numerous challenges stand on the way of understanding the sense, the trends dissolved in the petabytes of Big Data—extracting its semantics for further use in analytics. Among those challenges, we have chosen the problem of balancing between effectiveness and efficiency in understanding Big Data as our focus. For better explaining our motivation and giving a reader the key that helps follow how our premises are transformed into conclusions, we offered a simple walkthrough example of a news token.

We began the analysis of Big Data Computing by looking at how the phenomenon influences and changes industrial landscapes. This overview helped us figure out that the demand in industries for effective and efficient use of Big Data, if properly understood, is enormous. However, this demand is not yet fully satisfied by the state-of-the-art technologies and methodologies. We then looked at current trends in research and development in order to narrow the gaps between the actual demand and the state of the art. The analysis of the current state of research activities resulted in pointing out the shortcomings and offering an approach that may help understand Big Data in a way that balances effectiveness and efficiency.

The major recommendations we elaborated for achieving the balance are: (i) devise approaches that intelligently combine top-down and bottom-up processing of data semantics by exploiting "3F + 3Co" in dynamics, at run time; (ii) use a natural incremental and evolutionary way of processing Big Data and its semantics instead of following a mechanistic approach to scalability.

Inspired by the harmony and beauty of biological evolution, we further presented our vision of how these high-level recommendations may be approached. The "Scaling with a Traditional Database" section offered a review of possible ways to solve scalability problem at data processing level. The "Knowledge Self-Management and Refinement through Evolution" section presented a conceptual level framework for building an evolving ecosystem of environmental contexts with knowledge tokens and different species of KOs that populate environmental contexts and collect knowledge tokens for nutrition. The genomes and bodies of these KOs are ontologies describing corresponding environmental contexts. These ontologies evolve in line with the evolution of KOs. Hence they reflect the evolution of our understanding of Big Data by collecting the refinements of our mental picture of the change in the world. Finally, we found out that such an evolutionary approach to building knowledge representations will naturally allow assuring fitness of knowledge representations—as the fitness of the corresponding KOs to the environmental contexts they inhabit.

We also found out that the major technological components for building such evolving knowledge ecosystems are already in place and could be effectively used, if refined and combined as outlined in the "Knowledge Self-Management and Refinement through Evolution" section.

© 2014 by Taylor & Francis Group, LLC

Acknowledgments

This work was supported in part by the "Cloud Software Program" managed by TiViT Oy and the Finnish Funding Agency for Technology and Innovation (TEKES).

References

Abadi, D. J., D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. 2003. Aurora: A new model and architecture for data stream management. *VLDB Journal* 12(2): 120–139.

Anderson, C. 2008. The end of theory: The data deluge makes the scientific method obsolete. Wired Magazine 16:07 (June 23). http://www.wired.com/science/discoveries/magazine/16–07/pb_theory.

Ankolekar, A., M. Krotzsch, T. Tran, and D. Vrandecic. 2007. The two cultures: Mashing up Web 2.0 and the Semantic Web. In *Proc Sixteenth Int Conf on World Wide Web (WWW'07)*, 825–834. New York: ACM.

Berry, D. 2011. The computational turn: Thinking about the digital humanities. Culture Machine 12 (July 11). http://www.culturemachine.net/index.php/cm/article/view/440/470.

Beyer, M. A., A. Lapkin, N. Gall, D. Feinberg, and V. T. Sribar. 2011. 'Big Data' is only the beginning of extreme information management. Gartner Inc. (April). http://www.gartner.com/id=1622715 (accessed August 30, 2012).

Bizer, C., T. Heath, and T. Berners-Lee. 2009. Linked data—The story so far. *International Journal on Semantic Web and Information Systems* 5(3): 1–22.

Bollier, D. 2010. The promise and peril of big data. Report, Eighteenth Annual Aspen Institute Roundtable on Information Technology, the Aspen Institute. http://www.aspeninstitute.org/sites/default/files/content/docs/pubs/The_Promise_and_Peril_of_Big_Data.pdf (accessed August 30, 2012).

Bowker, G. C. 2005. *Memory Practices in the Sciences*. Cambridge, MA: MIT Press.

Boyd, D. and K. Crawford. 2012. Critical questions for big data. *Information, Communication & Society* 15(5): 662–679.

Broekstra, J., A. Kampman, and F. van Harmelen. 2002. Sesame: A generic architecture for storing and querying RDF and RDF schema. In *The Semantic Web—ISWC 2002*, eds. I. Horrocks and J. Hendler, 54–68. Berlin, Heidelberg: Springer-Verlag, LNCS 2342.

Cai, M. and M. Frank. 2004. RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network. In Proc Thirteenth Int Conf World Wide Web *(WWW'04)*, 650–657. New York: ACM.

Capgemini. 2012. The deciding factor: Big data & decision making. Report. http://www.capgemini.com/services-and-solutions/technology/business-information-management/the-deciding-factor/ (accessed August 30, 2012).

Chang, F., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. 2008. Bigtable: A distributed storage

© 2014 by Taylor & Francis Group, LLC

system for structured data. *ACM Transactions on Computer Systems* 26(2): article 4.

Cochez, M. and V. Terziyan. 2012. Quality of an ontology as a dynamic optimisation problem. In *Proc Eighth Int Conf ICTERI 2012*, eds. V. Ermolayev et al., 249–256. CEUR-WS vol. 848. http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-DEIS-paper-1-p-249-256.pdf.

Collins, A. M. and E. F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review* 82(6): 407–428.

Cusumano, M. 2010. Cloud computing and SaaS as new computing platforms. *Communications of the ACM* 53(4): 27–29.

Darwin, C. 1859. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London: John Murrey.

Davidovsky, M., V. Ermolayev, and V. Tolok. 2011. Instance migration between ontologies having structural differences. *International Journal on Artificial Intelligence Tools* 20(6): 1127–1156.

Davidovsky, M., V. Ermolayev, and V. Tolok. 2012. Agent-based implementation for the discovery of structural difference in OWL DL ontologies. In *Proc. Fourth Int United Information Systems Conf (UNISCON 2012)*, eds. H. C. Mayr, A. Ginige, and S. Liddle, Berlin, Heidelberg: Springer-Verlag, LNBIP 137.

Dean, J. and S. Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Communications of the ACM* 51(1): 107–113.

Dean, D. and C. Webb. 2011. Recovering from information overload. *McKinsey Quarterly*. http://www.mckinseyquarterly.com/Recovering_from_information_overload_2735 (accessed October 8, 2012).

DeCandia, G., D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. 2007. Dynamo: Amazon's highly available key-value store. In *21st ACM Symposium on Operating Systems Principles*, eds. T. C. Bressoud and M. Frans Kaashoek, 205–220. New York: ACM.

Dickinson, I. and M. Wooldridge. 2003. Towards practical reasoning agents for the semantic web. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 827–834. New York: ACM.

Driscoll, M. 2011. Building data startups: Fast, big, and focused. *O'Reilly Radar* (9). http://radar.oreilly.com/2011/08/building-data-startups.html (accessed October 8, 2012).

Ermolayev, V. and M. Davidovsky. 2012. Agent-based ontology alignment: Basics, applications, theoretical foundations, and demonstration. In *Proc. Int Conf on Web Intelligence, Mining and Semantics (WIMS 2012)*, eds. D. Dan Burdescu, R. Akerkar, and C. Badica, 11–22. New York: ACM.

Ermolayev, V., N. Keberle, O. Kononenko, S. Plaksin, and V. Terziyan. 2004. Towards a framework for agent-enabled semantic web service composition. *International Journal of Web Services Research* 1(3): 63–87.

Ermolayev, V., N. Keberle, W.-E. Matzke, and V. Vladimirov. 2005. A strategy for automated meaning negotiation in distributed information retrieval. In *Proc 4th Int Semantic Web Conference (ISWC'05)*, eds. Y. Gil et al., 201–215. Berlin, Heidelberg: Springer-Verlag, LNCS 3729.

Ermolayev, V., N. Keberle, and W.-E. Matzke. 2008. An ontology of environments, events, and happenings, computer software and applications, 2008. *COMPSAC '08. 32nd Annual IEEE International*, pp. 539, 546, July 28, 2008–Aug. 1, 2008. doi: 10.1109/COMPSAC.2008.141

© 2014 by Taylor & Francis Group, LLC

Ermolayev, V., C. Ruiz, M. Tilly, E. Jentzsch, J.-M. Gomez-Perez, and W.-E. Matzke. 2010. A context model for knowledge workers. In *Proc Second Workshop on Content, Information, and Ontologies (CIAO 2010)*, eds. V. Ermolayev, J.-M. Gomez-Perez, P. Haase, and P. Warren, CEUR-WS, vol. 626. http://ceur-ws.org/Vol-626/regular2.pdf (online).

Euzenat, J. and P. Shvaiko. 2007. *Ontology Matching*. Berlin, Heidelberg: Springer-Verlag.

Fan, W., A. Bifet, Q. Yang, and P. Yu. 2012a. Foreword. In *Proc First Int Workshop on Big Data, Streams, and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, eds. W. Fan, A. Bifet, Q. Yang, and P. Yu, New York: ACM.

Fan, J., A. Kalyanpur, D. C. Gondek, and D. A. Ferrucci. 2012b. Automatic knowledge extraction from documents. *IBM Journal of Research and Development* 56(3.4): 5:1–5:10.

Fensel, D., F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. Della Valle, F. Fischer et al. 2008. Towards LarKC: A platform for web-scale reasoning, *Semantic Computing, 2008 IEEE International Conference on*, pp. 524, 529, 4–7 Aug. 2008. doi: 10.1109/ICSC.2008.41.

Fisher, D., R. DeLine, M. Czerwinski, and S. Drucker. 2012. Interactions with big data analytics. *Interactions* 19(3):50–59.

Gangemi, A. and V. Presutti. 2009. Ontology design patterns. In *Handbook on Ontologies*, eds. S. Staab and R. Studer, 221–243. Berlin, Heidelberg: Springer-Verlag, International Handbooks on Information Systems.

Ghemawat, S., H. Gobioff, and S.-T. Leung. 2003. The Google file system. *In Proc Nineteenth ACM Symposium on Operating Systems Principles (SOSP'03)*, 29–43. New York: ACM.

Golab, L. and M. Tamer Ozsu. 2003. Issues in data stream management. *SIGMOD Record* 32(2): 5–14.

Gordon, A. 2005. Privacy and ubiquitous network societies. In *Workshop on ITU Ubiquitous Network Societies*, 6–15.

Greller, W. 2012. Reflections on the knowledge society. http://wgreller.wordpress.com/2010/11/03/big-data-isnt-big-knowledge-its-big-business/ (accessed August 20, 2012).

Gu, Y. and R. L. Grossman. 2009. Sector and sphere: The design and implementation of a high-performance data cloud. *Philosophical Transactions of the Royal Society* 367(1897): 2429–2445.

Guarino, N. and C. Welty. 2001. Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering* 39(1): 51–74.

Guéret, C., E. Oren, S. Schlobach, and M. Schut. 2008. An evolutionary perspective on approximate RDF query answering. In *Proc Int Conf on Scalable Uncertainty Management*, eds. S. Greco and T. Lukasiewicz, 215–228. Berlin, Heidelberg: Springer-Verlag, LNAI 5291.

He, B., M. Yang, Z. Guo, R. Chen, B. Su, W. Lin, and L. Zhou. 2010. Comet: Batched stream processing for data intensive distributed computing, In *Proc First ACM symposium on Cloud Computing (SoCC'10)*, 63–74. New York: ACM.

Hepp, M. 2007. Possible ontologies: How reality constrains the development of relevant ontologies. *IEEE Internet Computing* 11(1): 90–96.

Hogan, A., J. Z. Pan, A. Polleres, and Y. Ren. 2011. Scalable OWL 2 reasoning for linked data. In Lecture Notes for the Reasoning Web Summer School, Galway, Ireland (August). http://aidanhogan.com/docs/rw_2011.pdf (accessed October 18, 2012).

© 2014 by Taylor & Francis Group, LLC

Isaac, A., C. Trojahn, S. Wang, and P. Quaresma. 2008. Using quantitative aspects of alignment generation for argumentation on mappings. In *Proc ISWC'08 Workshop on Ontology Matching*, ed. P. Shvaiko, J. Euzenat, F. Giunchiglia, and H. Stuckenschmidt, CEUR-WS Vol-431. http://ceur-ws.org/Vol-431/om2008_Tpaper5.pdf (online).

Ishai, Y., E. Kushilevitz, R. Ostrovsky, and A. Sahai. 2009. Extracting correlations, *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium on*, pp. 261, 270, 25–27 Oct. 2009. doi: 10.1109/FOCS.2009.56.

Joseph, A. 2012. A Berkeley view of big data. Closing keynote of Eduserv Symposium 2012: Big Data, Big Deal? http://www.eduserv.org.uk/newsandevents/events/2012/symposium/closing-keynote (accessed October 8, 2012).

Keberle, N. 2009. Temporal classes and OWL. In *Proc Sixth Int Workshop on OWL: Experiences and Directions (OWLED 2009)*, eds. R. Hoekstra and P. F. Patel-Schneider, CEUR-WS, vol 529. http://ceur-ws.org/Vol-529/owled2009_submission_27.pdf (online).

Kendall, E., R. Bell, R. Burkhart, M. Dutra, and E. Wallace. 2009. Towards a graphical notation for OWL 2. In *Proc Sixth Int Workshop on OWL: Experiences and Directions (OWLED 2009)*, eds. R. Hoekstra and P. F. Patel-Schneider, CEUR-WS, vol 529. http://ceur-ws.org/Vol-529/owled2009_submission_47.pdf (online).

Klinov, P., C. del Vescovo, and T. Schneider. 2012. Incrementally updateable and persistent decomposition of OWL ontologies. In *Proc OWL: Experiences and Directions Workshop*, ed. P. Klinov and M. Horridge, CEUR-WS, vol 849. http://ceur-ws.org/Vol-849/paper_7.pdf (online).

Kontchakov, R., C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. 2010. The combined approach to query answering in DL-Lite. In *Proc Twelfth Int Conf on the Principles of Knowledge Representation and Reasoning (KR 2010)*, eds. F. Lin and U. Sattler, 247–257. North America: AAAI.

Knuth, D. E. 1998. *The Art of Computer Programming. Volume 3: Sorting and Searching*. Second Edition, Reading, MA: Addison-Wesley.

Labrou, Y. 2006. Standardizing agent communication. In *Multi-Agent Systems and Applications*, eds. M. Luck, V. Marik, O. Stepankova, and R. Trappl, 74–97. Berlin, Heidelberg: Springer-Verlag, LNCS 2086.

Labrou, Y., T. Finin, and Y. Peng. 1999. Agent communication languages: The current landscape. *IEEE Intelligent Systems* 14(2): 45–52.

Lenat, D. B. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11): 33–38.

Lin, J. and C. Dyer. 2010. Data-Intensive Text Processing with MapReduce. Morgan & Claypool Synthesis Lectures on Human Language Technologies. http://lintool.github.com/MapReduceAlgorithms/MapReduce-book-final.pdf.

Manyika, J., M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Hung Byers. 2011. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute (May). http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation (accessed October 8, 2012).

McGlothlin, J. P. and L. Khan. 2010. Materializing inferred and uncertain knowledge in RDF datasets. In *Proc Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 1951–1952. North America: AAAI.

Mills, P. 2011. Efficient statistical classification of satellite measurements. *International Journal of Remote Sensing* 32(21): 6109–6132.

© 2014 by Taylor & Francis Group, LLC

Mitchell, I. and M. Wilson. 2012. Linked Data. Connecting and Exploiting Big Data. Fujitsu White Paper (March). http://www.fujitsu.com/uk/Images/Linked-data-connecting-and-exploiting-big-data-(v1.0).pdf.

Nardi, D. and R. J. Brachman. 2007. An introduction to description logics. In *The Description Logic Handbook*, eds. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. New York: Cambridge University Press.

Nemani, R. R. and R. Konda. 2009. A framework for data quality in data warehousing. In *Information Systems: Modeling, Development, and Integration*, eds. J. Yang, A. Ginige, H. C. Mayr, and R.-D. Kutsche, 292–297. Berlin, Heidelberg: Springer-Verlag, LNBIP 20.

Olston, C. 2012. Programming and debugging large scale data processing workflows. In *First Int Workshop on Hot Topics in Cloud Data Processing (HotCDP'12)*, Switzerland.

Oren, E., S. Kotoulas, G. Anadiotis, R. Siebes, A. ten Teije, and F. van Harmelen. 2009. Marvin: Distributed reasoning over large-scale Semantic Web data. *Journal of Web Semantics* 7(4): 305–316.

Ponniah, P. 2010. *Data Warehousing Fundamentals for IT Professionals*. Hoboken, NJ: John Wiley & Sons.

Puuronen, S., V. Terziyan, and A. Tsymbal. 1999. A dynamic integration algorithm for an ensemble of classifiers. In *Foundations of Intelligent Systems: Eleventh Int Symposium ISMIS'99*, eds. Z.W. Ras and A. Skowron, 592–600. Berlin, Heidelberg: Springer-Verlag, LNAI 1609.

Quillian, M. R. 1967. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science* 12(5): 410–430.

Quillian, M. R. 1969. The teachable language comprehender: A simulation program and theory of language. *Communications of the ACM* 12(8): 459–476.

Rahwan, T. 2007. Algorithms for coalition formation in multi-agent systems. PhD diss., University of Southampton. http://users.ecs.soton.ac.uk/nrj/download-files/lesser-award/rahwan-thesis.pdf (accessed October 8, 2012).

Rimal, B. P., C. Eunmi, and I. Lumb. 2009. A taxonomy and survey of cloud computing systems. In *Proc Fifth Int Joint Conf on INC, IMS and IDC*, 44–51. Washington, DC: IEEE CS Press.

Roy, G., L. Hyunyoung, J. L. Welch, Z. Yuan, V. Pandey, and D. Thurston. 2009. A distributed pool architecture for genetic algorithms, *Evolutionary Computation, 2009. CEC '09. IEEE Congress on,* pp. 1177, 1184, 18–21 May 2009. doi: 10.1109/CEC.2009.4983079

Sakr, S., A. Liu, D.M. Batista, and M. Alomari. 2011. A survey of large scale data management approaches in cloud environments. *IEEE Communications Society Surveys & Tutorials* 13(3): 311–336.

Salehi, A. 2010. *Low Latency, High Performance Data Stream Processing: Systems Architecture. Algorithms and Implementation*. Saarbrücken: VDM Verlag.

Shvachko, K., K. Hairong, S. Radia, R. Chansler. 2010. The Hadoop distributed file system, *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp.1,10, 3–7 May 2010. doi: 10.1109/MSST.2010.5496972.

Smith, B. 2012. Big data that might benefit from ontology technology, but why this usually fails. In *Ontology Summit 2012, Track 3 Challenge: Ontology and Big Data*. http://ontolog.cim3.net/file/work/OntologySummit2012/2012-02-09_BigDataChallenge-I-II/Ontology-for-Big-Data—BarrySmith_20120209.pdf (accessed October 8, 2012).

© 2014 by Taylor & Francis Group, LLC

Tatarintseva, O., V. Ermolayev, and A. Fensel. 2011. Is your Ontology a burden or a Gem?—Towards Xtreme Ontology engineering. In *Proc Seventh Int Conf ICTERI 2011*, eds. V. Ermolayev et al., 65–81. CEUR-WS, vol. 716. http://ceur-ws.org/Vol-716/ICTERI-2011-CEUR-WS-paper-4-p-65–81.pdf (online).

Terziyan, V. 2001. Dynamic integration of virtual predictors. In *Proc Int ICSC Congress on Computational Intelligence: Methods and Applications (CIMA'2001)*, eds. L. I. Kuncheva et al., 463–469. Canada: ICSC Academic Press.

Terziyan, V. 2007. Predictive and contextual feature separation for Bayesian metanetworks. In *Proc KES-2007/WIRN-2007*, ed. B. Apolloni et al., 634–644. Berlin, Heidelberg: Springer-Verlag, LNAI 4694.

Terziyan, V. and O. Kaykova. 2012. From linked data and business intelligence to executable reality. *International Journal on Advances in Intelligent Systems* 5(1–2): 194–208.

Terziyan, V., A. Tsymbal, and S. Puuronen. 1998. The decision support system for telemedicine based on multiple expertise. *International Journal of Medical Informatics* 49(2): 217–229.

Thomason, R. H. 1998. Representing and reasoning with context. In *Proc Int Conf on Artificial Intelligence and Symbolic Computation (AISC 1998)*, eds. J. Calmet and J. Plaza, 29–41. Berlin, Heidelberg: Springer-Verlag, LNAI 1476.

Thusoo, A., Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. S. Sarma, R. Murthy, and H. Liu. 2010. Data warehousing and analytics infrastructure at Facebook. In *Proc 2010 ACM SIGMOD Int Conf on Management of Data*, 1013–1020. New York: ACM.

Tsangaris, M. M., G. Kakaletris, H. Kllapi, G. Papanikos, F. Pentaris, P. Polydoras, E. Sitaridi, V. Stoumpos, and Y. E. Ioannidis. 2009. Dataflow processing and optimization on grid and cloud infrastructures. *IEEE Data Engineering Bulletin* 32(1): 67–74.

Urbani, J., S. Kotoulas, E. Oren, and F. van Harmelen. 2009. Scalable distributed reasoning using MapReduce. In Proc Eighth Int Semantic Web Conf (ISWC'09), eds. A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, 634–649. Berlin, Heidelberg: Springer-Verlag.

W3C. 2009. OWL 2 web ontology language profiles. W3C Recommendation (October). http://www.w3.org/TR/owl2-profiles/.

Weinberger, D. 2012. *Too Big to know. Rethinking Knowledge now that the Facts aren't the Facts, Experts are Everywhere, and the Smartest Person in the Room is the Room*. First Edition. New York, NY: Basic Books.

Wielemaker, J., G. Schreiber, and B. Wielinga. 2003. Prolog-based infrastructure for RDF: Scalability and performance. In *The Semantic Web—ISWC 2003*, eds. D. Fensel, K. Sycara, and J. Mylopoulos, 644–658. Berlin, Heidelberg: Springer-Verlag, LNCS 2870.

Wooldridge, M. and N. R. Jennings. 1995. Intelligent agents: Theory and practice. *The Knowledge Engineering Review* 10(2): 115–152.

© 2014 by Taylor & Francis Group, LLC

PIII

LOCALITY-SENSITIVE HASHING FOR MASSIVE STRING-BASED ONTOLOGY MATCHING

by

Michael Cochez 2014

In D. Ślęzak, B. Dunin-Kęplicz, M. Lewis, & T. Terano (Eds.), 2014
IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and
Intelligent Agent Technologies (IAT)

Reproduced with kind permission of IEEE Computer Society.

PV

LARGE SCALE KNOWLEDGE MATCHING WITH BALANCED EFFICIENCY-EFFECTIVENESS USING LSH FOREST

by

Michael Cochez, Vagan Terziyan, and Vadim Ermolayev 2016

This paper is an extended version of PIV, It is currently under review for publication in the LNCS TCCI Journal

Large Scale Knowledge Matching with Balanced Efficiency-Effectiveness Using LSH Forest

Michael Cochez[1], Vagan Terziyan[1], and Vadim Ermolayev[2]

[1] University of Jyvaskyla,
Department of Mathematical Information Technology
P.O. Box 35 (Agora),
FI-40014 University of Jyväskylä, Finland
michael.cochez@jyu.fi, vagan.terziyan@jyu.fi
[2] Zaporozhye National University,
Department of IT
66, Zhukovskogo st.,
UA-69063, Zaporozhye, Ukraine
vadim@ermolayev.com

Abstract. Evolving Knowledge Ecosystems were proposed recently to approach the Big Data challenge, following the hypothesis that knowledge evolves in a way similar to biological systems. Therefore, the inner working of the knowledge ecosystem can be spotted from natural evolution. An evolving knowledge ecosystem consists of Knowledge Organisms, which form a representation of the knowledge, and the environment in which they reside. The environment consists of contexts, which are composed of so-called knowledge tokens. These tokens are ontological fragments extracted from information tokens, in turn, which originate from the streams of information flowing into the ecosystem. In this article we investigate the use of LSH Forest (a self-tuning indexing schema based on locality-sensitive hashing) for solving the problem of placing new knowledge tokens in the right contexts of the environment. We argue and show experimentally that LSH Forest possesses required properties and could be used for large distributed set-ups. Further, we show experimentally that for our type of data minhashing works better than random hyperplane hashing. This paper is an extension of the paper "Balanced Large Scale Knowledge Matching Using LSH Forest" presented at the International Keystone Conference 2015.

Keywords: Evolving Knowledge Ecosystems, Locality-sensitive Hashing, LSH Forest, Minhash, Random Hyperplane Hashing, Big Data

1 Introduction

Semantic keyword search attempts to find results close to the intent of the user, i.e., it attempts to find out the meaning behind the keywords provided. Perhaps, one of the biggest problems when attempting this is that the search system needs knowledge that is evolving in line with the world it serves. In other words, only

if the search system has an up-to-date representation of the domain of interest of the user will it be possible to interpret the real world meaning of the keywords provided. However, this problem becomes very challenging given the wide range of possible search queries combined with the explosion in the volume of data available, its complexity, variety and rate of change.

Recently a conceptual approach to attack this challenging problem has been proposed [1]. The core of that proposal is the understanding that the mechanisms of knowledge evolution could be spotted from evolutionary biology. These mechanisms are enabled in an *Evolving Knowledge Ecosystem* (EKE) populated with *Knowledge Organisms* (KO). Individual KOs carry their fragments of knowledge — similarly to different people having their individual and potentially dissimilar perceptions and understanding of their environment. The population of KOs, like a human society, possesses the entire knowledge representation of the world, or more realistically — a subject domain. Information tokens flow into such an ecosystem, are further transformed into the knowledge tokens, and finally sown there. The KOs collect the available knowledge tokens and consume these as nutrition. Remarkably, the constitution of an EKE, allows natural scaling in a straightforward way. Indeed, the fragment of knowledge owned by an individual KO and the knowledge tokens consumed by KOs are small. Therefore, a well scalable method of sowing the knowledge tokens is under demand to complete a scalable knowledge feeding pipeline into the ecosystem. This paper reports on the implementation and evaluation of our knowledge token sowing solution based on the use of LSH Forest [2]. We demonstrate that: (i) the method scales very well for the volumes characteristic to big data processing scenarios, (ii) using random hyperplane hashing (RHH) for angular distance between knowledge tokens results in poor precision and recall, while (iii) Jaccard distance yields results with sufficiently good precision and recall. As a minor result we would like to highlight the f-RHH method which does not require more computations than standard RHH, but still improves the results. The rest of the paper is structured as follows. Section 2 sketches out the concept of EKE and also explains how knowledge tokens are sown in the environments. Section 3 presents the basic formalism of Locality Sensitive hashing (LSH) and LSH Forest and introduces the distance metrics. Finally, it outlines our arguments for using LSH Forest as an appropriate method. Section 4 describes the settings for our computational experiments whose results are presented in section 5. The paper is concluded and plans for future work are outlined in section 6.

2 Big Knowledge — Evolving Knowledge Ecosystems

Humans make different decisions in similar situations, thus taking different courses in their lives. This is largely due to the differences in their knowledge. So, the evolution of conscious beings noticeably depends on the knowledge they possess. On the other hand, making a choice triggers the emergence of new knowledge. Therefore, it is natural to assume that knowledge evolves because of the evolution of humans, their decision-making needs, their value systems, and

the decisions made. Hence, knowledge evolves to support the intellectual activity of its owners, e.g., to interpret the information generated in event observations — handling the diversity and complexity of such information. Consequently, Ermolayev et al. [1] hypothesize that the mechanisms of knowledge evolution are very similar to (and could be spotted from) the mechanisms of the evolution of humans. Apart from the societal aspects, these are appropriately described using the metaphor of biological evolution.

A biological habitat is in fact an ecosystem that frames out and enables the evolution of individual organisms, including humans. Similarly, a knowledge ecosystem has to be introduced for enabling and managing the evolution of knowledge. As proposed in [1], such EKE should scale adequately to cope with realistic and increasing characteristics of data/information to be processed and balance the efficiency and effectiveness while extracting knowledge from information and triggering the changes in the available knowledge.

2.1 Efficiency Versus Effectiveness

Effectiveness and efficiency are the important keys for big data processing and for the big knowledge extraction. Extracting knowledge out of big data would be effective only if: (i) not a single important fact is left unattended (completeness); and (ii) these facts are faceted adequately for further inference (expressiveness and granularity). Efficiency in this context may be interpreted as the ratio of the utility of the result to the effort spent.

In big knowledge extraction, efficiency could be naturally mapped to timeliness. If a result is not timely the utility of the resulting knowledge will drop. Further, it is apparent that increasing effectiveness means incrementing the effort spent on extracting knowledge, which negatively affects efficiency. In other words, if we would like to make a deeper analysis of the data we will have a less efficient system.

Finding a solution, which is balanced regarding these clashes, is challenging. In this paper we use a highly scalable method to collect the increments of incoming knowledge using a 3F+3Co approach, which stand for Focusing, Filtering, and Forgetting + Contextualizing, Compressing, and Connecting (c.f. [1] and section 3.2).

2.2 Evolving Knowledge Ecosystems

An environmental context for a KO could be thought of as its habitat. Such a context needs to provide nutrition that is "healthy" for particular KO species — i.e. matching their genome noticeably. The nutrition is provided by Knowledge Extraction and Contextualization functionality of the ecosystem [1] in a form of *knowledge tokens*. Hence, several and possibly overlapping environmental contexts need to be regarded in a hierarchy which corresponds to several subject domains of interest and a foundational knowledge layer. Environmental contexts are sowed with knowledge tokens that correspond to their subject domains. It is useful to limit the lifetime of a knowledge token in an environment – those

which are not consumed dissolve finally when their lifetime ends. KOs use their perceptive ability to find and consume knowledge tokens for nutrition. Knowledge tokens that only partially match KOs' genome may cause both KO body and genome changes and are thought of as mutagens. Mutagens in fact deliver the information about the changes in the world to the environment. Knowledge tokens are extracted from the information tokens either in a stream window, or from the updates of the persistent data storage and further sown in the appropriate environmental context. The context for placing a newly coming knowledge token is chosen by the contextualization functionality. In this paper we present a scalable solution for sowing these knowledge tokens in the appropriate environmental contexts.

3 Locality-Sensitive Hashing

The algorithms for finding nearest neighbors in a dataset were advanced in the work by Indyk and Motwani, who presented the seminal work on Locality-sensitive hashing (LSH) [3]. They relaxed the notion of a nearest neighbor to that of an approximate one, allowing for a manageable error in the found neighbors. Thanks to this relaxation, they were able to design a method which can handle queries in sub-linear time. To use LSH, one has to create a database containing outcomes of specific hash functions. These hash functions have to be independent and likely to give the same outcome when hashed objects are similar and likely to give different outcomes when they are dissimilar. Once this database is built one can query for nearest neighbors of a given query point by hashing it with the same hash functions. The points returned as approximate near neighbors are the objects in the database which got hashed to the same buckets as the query point. [4] If false positives are not acceptable, one can still filter these points.

Formally, to apply LSH we construct a family $\mathcal{H}$ of hash functions which map from a space $\mathcal{D}$ to a universe $\mathcal{U}$.

Let $d_1 < d_2$ be distances according to a distance measure d on a space $\mathcal{D}$. The family $\mathcal{H}$ is (d_1, d_2, p_1, p_2)-sensitive if for any two points $p, q \in \mathcal{D}$ and $h \in \mathcal{H}$:

- if $d(p,q) \leq d_1$ then $\Pr[h(p) = h(q)] \geq p_1$
- if $d(p,q) \geq d_2$ then $\Pr[h(p) = h(q)] \leq p_2$

where $p1 > p2$.

The probabilities p_1 and p_2 might be close to each other and hence only one function from $\mathcal{H}$ giving an equal result for two pints might not be sufficient to trust that these points are similar. Amplification is used to remedy this problem. This is achieved by creating b functions g_j, each consisting of r hash functions chosen uniformly at random from $\mathcal{H}$. The function g_j is the concatenation of r independent basic hash functions. The symbols b and r stand for *bands* and *rows*. These terms come from the representation of data. One could collect all outcomes of the hash functions in a two-dimensional table. This table can be divided in b bands containing r rows each. (See also [5].) The concatenated hash function g_j

maps points p and q to the same bucket if all hash functions it is constructed from hashes the points to the same buckets. If for any j, the function g_j maps p and q to the same bucket, p and q are considered close. The amplification creates a new locality sensitive family which is $\left(d_1, d_2, 1-(1-{p_1}^r)^b, 1-(1-{p_2}^r)^b\right)$ sensitive.

3.1 LSH Forest

The standard LSH algorithm is somewhat wasteful with regards to the amount of memory is uses. Objects always get hashed to a fixed length band, even if that is not strictly needed to decide whether points are approximate near neighbors. LSH Forest (introduced by Bawa et al. [2]) introduces variable length bands and stores the outcomes of the hashing in a prefix tree data structure.

The length of the band is reduced by only computing the hash functions if there is more than one point which is hashed to the same values. Put another way, in LSH the function g_j maps two points to the same bucket if all functions it is constructed from do so as well. LSH Forest potentially reduces the number of evaluations by only computing that much of g_j as needed to distinct between the different objects. Alternatively, one can view this as assigning a unique label with a dynamic length to each point. In the prefix tree the labels on the edges are the values of the sub-hash functions of g_j.

Hashing and quantization techniques have a limitation when considering very close points. If points are arbitrarily close to each other, then there is no number of hash functions which can tell them apart. This limitation applies to both traditional LSH and the Forest variant. Therefore, LSH assumes a minimum distance between any two points and LSH Forest defines a maximum label length equal to the maximum height of the tree (indicated as k_m).

3.2 Sowing Knowledge Tokens Using LSH Forest

The first requirement for knowledge token sowing is that similar tokens get sown close to each other. This is achieved by adding knowledge tokens to the forest. Similar ones will get placed such that they are more likely to show up when the trees are queried for such tokens. Further requirements come from the 3F+3Co [1] aspects. When using LSH Forest:

Focusing is achieved by avoiding deep analysis when there are no similar elements added to the trees.

Filtering is done by just not adding certain data to the tree.

Forgetting is achieved by removing data from the tree. Removal is supported by the Forest and is an efficient operation.

Contextualizing happens when different parts of the token are spread over the trees. A token may therefore belong to several contexts simultaneously.

Compressing the tree compresses data in two different ways. Firstly, it only stores the hashes computed from the original data and, secondly, common prefixes are not duplicated but re-used. Note that it is possible to store the actual data on a secondary storage and keep only the index in memory.

Connecting the Forest is a body which grows incrementally. Since representations of different tokens can reside together in disparate parts of the trees, they can be considered connected. However, the real connection of these parts will be the task of the KOs which will consume the knowledge tokens which are sown in a tree.

In the next section we will introduce our experiments. In the first experiment series we show that the Forest is able to fulfill the *focusing* requirement. The second one shows that the forest is able to aid the KO to *connect* concepts together. Finally, the last series shows that the data structure has desirable spacial and temporal properties, demonstrating that the tree is able to *compress* data meanwhile offering an appropriate efficiency — effectiveness trade-off.

3.3 Distance Metrics and Locality-Sensitive Hash functions

In our previous work [6] we only used Jaccard distance to evaluate the use of LSH Forests. Typical metrics used in the literature for distance between textual documents are Jaccard and angular distance. In this work we will also use the later one and compare their performance.

The Jaccard distance is defined on sets A and B as $d(A, B) = 1 - sim(A, B)$. Here, sim (also referred to as the Jaccard similarity) is defined as the number of elements the sets have in common divided by the total number of elements in the sets (i.e, $sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$). In the case of text documents the elements in the set are the words of the text (or are derived from the words in the text). The angular distance between texts is defined as the angle between vectors where each dimension encodes the frequency of a specific word (or derivation).

For example, if we have two texts $\hat{A}$ = "the cat sits on the table" and $\hat{B}$ = "the black cat sits with the other cats". Then, a preprocessing step could reduce these texts to "cat sit table" and "black cat sit cat" (removing common words and stemming, see also the next section). For the Jaccard distance, these texts will then be converted into sets $A = \{cat, sit, table\}$ and $B = \{black, sit, cat\}$ resulting in a Jaccard distance of $1 - \frac{2}{4} = 0.5$. For the angular distance we obtain vectors $A = [1, 1, 1, 0]$ and $B = [2, 1, 0, 1]$ where the dimensions encode the frequencies of the words cat, sit, table, and black, respectively. The resulting angular distance (the angle between A and B) is 0.785.

For both distance metrics Locality-Sensitive Hash functions are known. The LSH function family used for Jaccard distance is minhash from Broder [7]. The outcome of this hash function on a set is the lowest index (counting from 0) any of the elements in the set has in a permutation of the whole universe of elements. In our example from above with two documents the universe consists of only 4 words. One possible permutation is $[black, cat, sit, table]$ leading to an outcome of 1 for set A (the word in A with lowest index in the permutation is cat) and 0 for set B. The range of the outcome space is as large as the size of the universe. One could in principle first determine the size of the universe and then decide upon the permutations. However, measuring the size of the universe beforehand and performing actual permutations would be unpractical. Instead,

we use a normal hash function to perform the permutation by mapping each original index to a target index. Hence, the outcome space is limited to the range of that hash function.

For the angular distance we use random hyperplane hashing (RHH) [8]. The core idea is to project the frequency vector onto a random vector. The result of the hash function is 1 if the projection is a positive multiple of the random vector and -1, otherwise. In practice this comes down to finding the sign of the dot product between the frequency vector and the random vector. Another way of looking at this is that we are deciding whether the vector in question is above or below[3] the hyperplane on which the random vector is a normal vector. An intuitive proof for the correctness of both minhash and RHH can be found from [5].

When using RHH the LSH forest will place the element in the one subtree if the hash outcome is 1. On the contrary, an outcome of -1 will cause it to direct the element to the other subtree. However, sometimes this decision seems too harsh. If the projected vector is only a very small multiple of the random vector the element is very close to the hyperplane and the binary decision which is made could cause nearest neighbors to be hashed to different subtrees.

To alleviate this problem, we investigate a slightly different approach which we will call fuzzy random hyperplane hashing or f-RHH. Instead of only allowing a binary decision, the hash function can also report that it is unable to decide well enough on which side of the hyperplane the given vector is (i.e., the outcome of the projection is small). The result of the hashing can thus be 1, -1, or both. When the result is both, then we will place the element in both subtrees essentially ignoring the outcome of the hash function completely.

What we need to perform f-RHH is a way to decide whether a frequency vector is close to the hyperplane. Moreover, this method has to be efficiently implementable. A first attempt could be to compute the angle between the vector and the hyperplane. This is a feasible but relatively expensive computation (especially because it has to happen for all vector-hyperplane pairs). However, observe that the angle between the vector and the hyperplane is $\frac{\pi}{2}-$ 'the angle between the vector and the normal'. If we call the vector a and the normal n, then given an angle k [4], a will get assigned both hash outcomes if

$$\frac{\pi}{2} - \widehat{an} = \frac{\pi}{2} - \arccos\left(\frac{a \cdot n}{\|a\|\|n\|}\right) < k$$

Which can be rewritten as:

$$\arcsin\left(\frac{a \cdot n}{\|a\|\|n\|}\right) < k$$

[3] Above can be defined as on the same side as the normal vector; below is then the other side of the hyperplane.

[4] the maximumum angle between a vector and the hyperplane for a to be assigned both hash outcomes

In this expression $\|n\|$ is essentially a positive constant[5] which we will call R. If we normalize the vector a before we compute the angle, the angle will remain the same. We will cal this normalized vector $\bar{a}$ where $\|\bar{a}\| = 1$. Using these facts, the previous expression can be rewritten as:

$$\arcsin\left(\frac{\bar{a} \cdot n}{R}\right) < k$$

Which can be rearranged to:

$$|\bar{a} \cdot n| < \sin(k) * R = C$$

What this expression tells us is that if the angle between a vector a and the hyperplane is smaller than k, then the absolute value of the dot product of the normalized vector $\bar{a}$ and the normal vector is smaller than a given constant number C.

This last expression can be implemented very efficiently. In fact, besides the normalization of each frequency vector (which has to happen only once), the dot product computation is exactly the same as what we would be computing anyway for the random hyperplane hashing.

To illustrate the effect of f-RHH, we present a two dimensional example in fig. 1. The figure shows a random vector $\overrightarrow{n}$ and the hyperplane H on which $\overrightarrow{n}$ is a normal vector. The red shaded area contains all vectors for which the hash outcome will be negative. Conversely, vectors in the blue area will get the value $+1$ assigned. All vectors which are in the overlap between the red and blue are will get both values assigned; causing the hyperplane to not cut the space sharply in two. In other words, the hyperplane does not strictly subdivide the space into two subspaces. Instead it creates an overlapping boundary between the two subspaces in which points are in both of the subspaces at the same time.

One question which remains to be answered is the value of the constant C. In order to find a reasonable value, we ran several preliminary experiments and found that a reasonably well working value was 10^{14}. Note that our normal vector n has its components sampled from the range $\left[-2^{63}, 2^{63} - 1\right]$. We cautiously assume that this constant value is data and case dependent. Hence, this constant should not be taken as a general recommendation.

[5]The norm of a specific random vector, will be the same for all angle computations. Moreover, since this very high dimensional vector and the each dimension of the vector is sampled from a uniform distribution, the expected norm of the random vectors is constant. In any case, the values are most likely different but will be in the same ballpark.

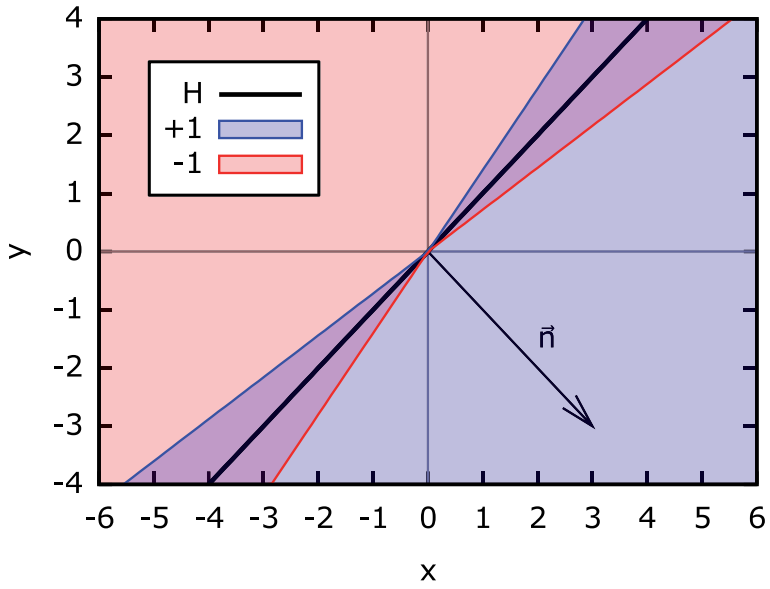


Fig. 1: An illustration of fuzzy random hyperplane hashing. Vectors which are in the area where -1 and 1 overlap have both hash outcomes at the same time.

4 Evaluation

The experiments are designed so that we start from a fairly simple set-up and more complexity is added in each following experiment. In the first series of experiments, we feed knowledge tokens created from three different data sources into an LSH tree and present measure how they are spread over the tree. In the following series, we use two and later three data sources and measure how the LSH Forest classifies the tokens and how it is capable of connecting the knowledge tokens. In that same series we compare the performance of the different hash functions. Finally, in the third series we add dynamism to the experiment by sampling the knowledge tokens in different ways and measure how the memory usage and processing time evolve.

Finding a suitable dataset for the experiment is not obvious. What we need are small pieces of information (i.e., the knowledge tokens) about which we know how they should be connected (i.e., a gold standard). Further, the dataset should be sufficiently large to conduct the experiments. We solved this issue by selecting three large ontologies for which a so-called alignment [9] has been created. These particular ontologies are large and have a fairly simple structure. Further,by using only the labels of the ontology a reasonable alignment can be

found [10]. Therefore, we extract the labels from these ontologies and use them as knowledge tokens. This is a relaxation of the knowledge token concept. In the earlier work [1] a knowledge token has an internal structure.

Datasets The *Large Biomed Track* of the Ontology Alignment Evaluation initiative[6] is the source of the datasets used in our evaluation. The FMA ontology[7], which contains 78,989 classes is the first dataset. The FMA ontology only contains classes and non-hierarchical datatype properties, i.e., no object or datatype properties nor instances. Secondly, there is the NCI ontology[8] containing 66,724 classes, and finally a fragment of 122,464 classes of the SNOMED ontology[9]. The NCI ontology contains classes, non-hierarchical datatype and hierarchical object properties. The classes of all ontologies are structured in a tree using *owl:SubClassOf* relations. The UMLS-based reference alignments as prepared for OAEI [10] are used as a gold standard. From these reference alignments we only retain the equal correspondences, with the confidence levels set to one.

Preprocessing We preprocess the ontologies by computing as many representations for each class as it has labels in the ontology. The preprocessing is very similar to the second strategy proposed in [10]. According to this strategy, for each label of each class, a set of strings is created as follows: the label is converted to lowercase and then split in strings using all the whitespace and punctuation marks as a delimiter. If this splitting created strings of 1 character, they are concatenated with the string that came before it. In addition to these steps, we also removed possessive suffixes from the substrings and removed the 20 most common English language words according to the Oxford English Dictionary[11]. This preprocessing results in 133628, 175698, and 122505 knowledge tokens, i.e., sets of strings for the FMA, NCI, and SNOMED ontology, respectively.

Implementation The implementation of our evaluation code heavily uses parallelism to speed up the computation. From the description of the LSH algorithm, it can be noticed that the hashing of the objects happens independent of each other. Therefore they can be computed in parallel using a multi-core system.

For the implementation of the minhash algorithm, we use Rabin fingerprints as described by Broder [11] instead of computing a real permutation of the universe. An improvement over earlier work [10] where Rabin hashing was also used is due to the fact that we invert the bits of the input to the hashing function. We noticed that small inputs gave a fairly high number of collisions using the functions normally, while the inverted versions do hardly cause any.

For the random hyperplane hashing we use a hash function to imitate an infinite random vector. The way this works is that we interpret each word as a number, which we then take to be the index (in the vector) representing the

[6] `http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2013/`

[7] `http://sig.biostr.washington.edu/projects/fm/`

[8] `http://www.obofoundry.org/cgi-bin/detail.cgi?id=ncithesaurus`

[9] `http://www.ihtsdo.org/index.php?id=545`

[10] `http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2013/oaei2013_umls_reference.html`

[11] `http://www.oxforddictionaries.com/words/the-oec-facts-about-the-language`

frequency of the word. Then, to find the value of the random vector for that index, we hash the index with the hash function. This has the practical implication that there is no need to store a random vector in its entirety, nor is there a need to know all words of the corpus beforehand. As a hash function we use murmur3 [12]. This choice is made because the hash function is fast, it provides reasonable mixing of the input bits, and has a close to uniform output range.

The outcome of RHH is binary and the trie used will be a binary tree as well (as opposed to the n-ary trie used for minhash). Because of this difference we can easily afford checking newly added data for exact duplicates. So, when we insert a knowledge token using RHH (or f-RHH) we check in the leaf nodes whether the already existing token has the same source concept and the same representation we ignore it immediately. This is as opposed to double insertions which happen in the case of minhash (see also the results in section 5.1).

The experiments are performed on hardware with two Intel Xeon E5-2670 processors (totaling 16 hyper-threaded cores) and limited to use a maximum of 16 GB RAM.

4.1 Single data source — Single Tree

In this series of experiments, we use only one LSH tree and knowledge tokens from a single dataset. First, the ontology is parsed and all its concepts are tokenized as described above. The resulting knowledge tokens are hashed (with the different hash functions — minhash, RHH, and f-RHH) and then fed into an LSH tree. We then analyze the distribution of the knowledge tokens in the tree obtained for each hashing option. Concrete, we observe how deep the knowledge tokens are located in the tree and how many siblings the leaves in the tree have. Further, for the case of minhash, we investigate chains of nodes which are only there because of a low number of tokens at the bottom of the tree.

4.2 Connecting Knowledge Tokens using LSH Forest, i.e. Matching

The objective of our the first experiment in this second series is to show how the ontology matching using LSH Forest compares to standard LSH. Besides the change in data structure we use the experimental set-up similar to what was used for testing standard LSH in our earlier research work [10]. In that work only Jaccard distance and minhashing were used and the best result for matching the SNOMED and NCI ontologies was obtained using 1 band of 480 rows which corresponds to 1 tree of maximum height $k_m = 480$. To keep the results comparable, we also do not use the reduced collision effect from inverting before hashing (see **Implementation** above). It needs to be noted, however, that we use a slightly different approach for selecting near neighbors compared to the standard LSH Forest approximate nearest neighbor querying. Since we are not interested in neighbors if they are to far away, we only take the siblings of each leaf into account when searching for related concepts. Further, we ignore

[12] `https://code.google.com/p/smhasher/wiki/MurmurHash3`

concepts if they their similarity is less than 0.8. Next to the traditional ontology matching measures of precision, recall, and F-measure, the potential memory and processing power savings are evaluated.

In the second part of this series we use the properties of the tree and also experiment with RHH and f-RRH. For minhashing we use our improved version, applying the inversion before hashing. We also incorporate the knowledge from the previous experiments to test how LSH Forest can perform when connecting knowledge tokens using a shorter tree. We measure both runtime performance and quality metrics for different number of trees.

In the last part we use the fact that there is no reason to limit ourselves to only using two data sources. Hence, we demonstrate scalability of the system by feeding all knowledge tokens created for all three datasets. We also analyze the time saving compared to performing three separate alignment tasks when pairs of datasets are used.

4.3 Adding Dynamics

In the final series of experiments we observe how the tree reacts to dynamic insertion of concepts. In the basic case, we select 10^6 knowledge tokens (from the three sets) using a uniform distribution. These are then one by one inserted into the tree. After every 10^4 insertions we measure number of hash operations used to measure the time complexity. The cumulative memory consumption is measured as the number of edges used in the trees. We also measure the real elapsed time after the insertion of every 10^5 knowledge tokens.

On an average system some knowledge tokens will be added much more frequently than others. This is due to the fact that the information or queries which the system processes are somehow focused on a certain domain. This also means that the tokens would not arrive according to a uniform distribution. A more plausible scenario is that certain concepts are very likely to occur, while others do hardly occur at all. We model this phenomena by using a so-called Zipf distribution with exponent 1 which causes few concepts to be inserted frequently while most are inserted seldom. Using this set-up we perform the same measurements as made for the uniform distribution.

It has to be noted that we need to make a minor change to the way our trees process the tokens. When a token already exists at a node, the standard implementation would build a chain which can only end at k_m. This is related to our above remark about the minimal distance between any two points. To solve this problem, the lowest internal nodes check whether the newly added representation is already existing and if so, it will ignore the representation. We shortly analyzed the effect of this change using the same set-up as in the second experiment series and noticed that this check does hardly affect runtime performance. The main effect is visible in the number of edges and hash operations which both drop by about 30 %. Further, a marginal decrease of the precision and a marginal increase of the recall is observable.

5 Results

5.1 Single Data Source — Single Tree

For the first series of experiments, we look at the characteristics of the LSH tree for the distance metrics and hash functions. We start with the cosine distance, RHH and f-RHH (the variant described above) because the outcome of the hashing is binary. This binary tree makes it somewhat easier to analyze.

Cosine Distance — RHH, f-RHH When measuring the frequencies of the depths of the leafs in the tree we obtain the results shown in fig. 2. To obtain this figure we placed all knowledge tokens from a given dataset into a tree with $k_m = 80$ after hashing them using RHH and f-RHH, respectively. Then we measure the number of leaves at a given height. From the figure it can be seen that there are only slight differences between the way the different datasets are spread over the tree. From the exect numbers we observed that the fRHH histograms are slightly skewed to the right when compared to their RHH counterparts. This is as expected since fRHH will insert extra elements into the tree whenever the outcome of the hashing has both values at the same time. The tail of the histogram decays pretty fast for all data sets indicating that the tree is able to differentiate between the majority of the tokens after about 40 hashings.

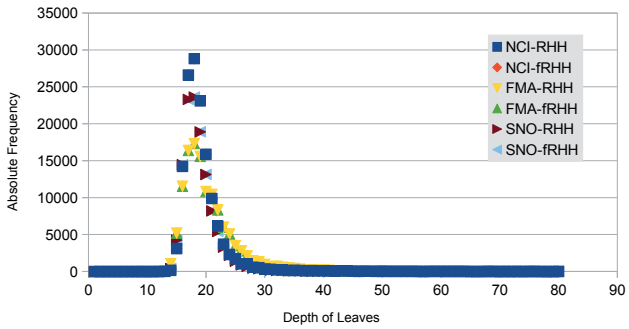
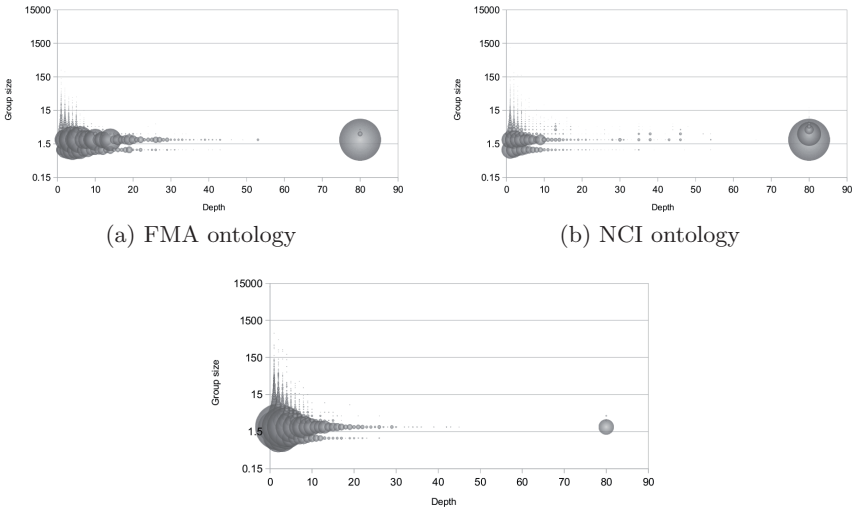


Fig. 2: The frequency of a leaf occurring at a given height for the knowledge tokens derived from the different data sets.

Jaccard Distance — Minhash After feeding the minhashed knowledge tokens of each data set into their own single LSH Tree with $k_m = 80$, we find clusters of leaves as shown in fig. 3. The figure shows how often a group of n siblings occurs as a function of the depth in the tree. Note that this figure is more complex than the figure we obtained for the (f-)RHH case. The reason for this complexity is that we are not dealing with a binary, but an n-ary tree.



(a) FMA ontology

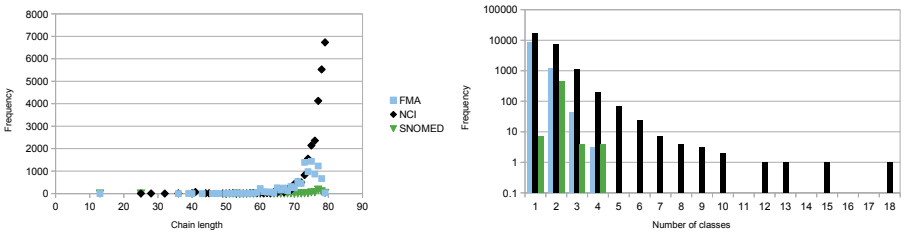
(b) NCI ontology

(c) SNOMED ontology

Fig. 3: Frequency of sibling groups of a given size at a given level in one LSH Tree. Note the logarithmic scale.

What we notice in the figures is that most of the concepts are fairly high up in the tree. After roughly 30 levels all the concepts, except these residing at the bottom of the tree, are placed. It is also visible that most knowledge tokens are located in the leaves which either have very few siblings or are located high up in the tree. This indicates that the tree is able to distinguish between the representations fairly fast. In both the FMA and NCI ontologies, we notice a high amount of knowledge tokens at the bottom of the tree, i.e., at level $k_m = 80$. We noticed that the same amount of concepts end up at the bottom of the tree even if k_m is chosen to be 1000, which indicates that hashing might be incapable to distinguish between the representations, i.e., they are so close that their hashes virtually always look the same. After further investigation, we found that the Jaccard similarities between the sibling concepts at the bottom of the tree are all equal to 1. This means that there are concepts in the ontology which have very similar labels, i.e., labels which (often because of our preprocessing steps) get reduced to exactly the same set of tokens. One problem with this phenomenon is that the tree contains long chains of nodes, which are created exclusively for these few siblings. We define an *exclusive chain* as the chain of nodes between an internal node at one level above the bottom of the tree, and another (higher) node which has more than one child. The lengths of these exclusive chains are illustrated in fig. 4a.

We notice that mainly the NCI ontology causes long exclusive chains. The most plausible cause for this is that NCI has a higher average number of representations per concept (2.6) than the other two ontologies (1.7 — FMA and 1.0



(a) Frequency of a given exclusive chain length for nodes at k_m

(b) Frequency of a given number of classes represented in a leaf at level k_m for each ontology. Note the log scale.

Fig. 4: Analysis for the leaf nodes

— SNOMED). To investigate this further, we plot the number of classes which the siblings at the lowest level represent. The result of analyzing the number of classes represented by the leaves in each sibling cluster can be found in fig. 4b

From the figure we notice that, indeed, very often there is a low number of classes represented by the siblings of the final nodes. We also notice that the NCI ontology has the most severe representation clashes.

5.2 Connecting Knowledge Tokens using LSH Forest, i.e. Matching

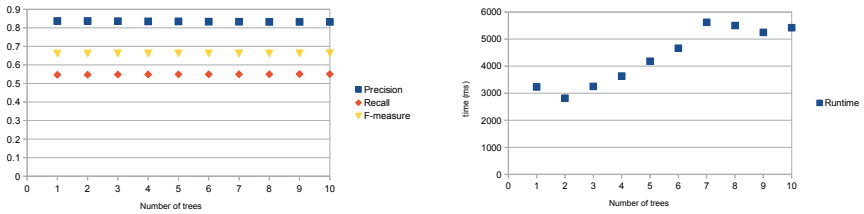
Part 1 When matching the SNOMED and NCI ontologies using a single tree of height 480, we obtain the precision of 0.838, recall of 0.547, and hence F-measure of 0.662. These results are similar to the results of the standard LSH algorithm which attained the precission of 0.842, recall of 0.535, and F-measure of 0.654.

The LSH Forest algorithm, however, uses only 30 % of the amount of hash function evaluations compared to the standard LSH algorithm. Furthermore, the Forest saves around 90 % of the memory used for storing the result of the hash evaluations. This is because the tree saves a lot of resources by only computing and storing the part of the label which is needed. Further, a result is stored only once if the same outcome is obtained from the evaluation of a given hash function for different representations. It should, however, be noted that using LSH Forest also implies a memory overhead for representing the tree structure, while the standard algorithm can place all hash function evaluations in an efficient two dimensional table.

The speed of the two algorithms with the same set-up is very similar. Using the Forest, the alignment is done in 20.6 seconds, while the standard algorithm completes in 21.5 seconds.

Part 2 As can be seen in the distribution of the ontologies over the tree in our previous experiment series (fig. 3) non-similar concepts remain fairly high up in the tree. Hence, when using the improved Rabin hashing technique described above, we can reduce the maximum height of the tree. Based on this information, we now choose the maximum height of the tree to be 30. We also use 10 as the

highest level of interest and ignore all representations which are unable to get a lower positions in the tree. We vary the number of trees used between 1 and 10 and show the impact on the precision, recall and F-measure in fig. 5a and timing in fig. 5b.



(a) Quality measurements minhash

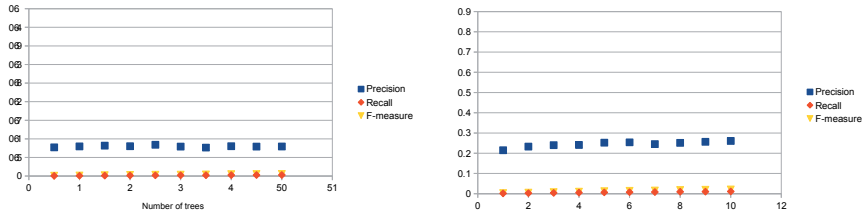
(b) Runtime for minhash

Fig. 5: Quality measurements and runtime behavior for an ontology matching task using different number of trees for minhash.

From the quality measurements, we see that the number of trees has little effect. It is hard to see from the figure, but the precision lowers ever so slightly when more trees are used. Concretely, it goes from 0.836947 when using one tree to 0.831957 with 10 trees. The recall has the opposite behavior growing from 0.546824 to 0.550616. The net effect of these two on the F-measure is a slight increase when more trees are used, namely from 0.661472 to 0.662662. It needs to be noted that also these results are in the same range as the measures in the previous experiment. Hence, we can conclude that constraining the height of a tree does not affect the quality much, if at all. However, as can be seen in the timing chart, the tree works much faster when its height is reduced. When only one tree is used, roughly 3 seconds are needed to obtain results. Increasing the number of trees to 10 only doubles the time, most likely because the system is better able to use multiple threads or the virtual machine might do a better just-in-time compilation. In any case, we note that using the forest and better hashing, we can create a system which is roughly 7 times faster and produces results of similar quality.

Next, we experimented using the RHH and f-RHH hash functions. The quality measurements for these for trees with depth 80 are shown in figs. 6a and 6b. Surprisingly and seemingly contradicting to the findings of [12] the performance of RHH and f-RHH are pretty low when compared to minhash. The reason for this low performance seems to be that in the case of the earlier work [12] the comparison was performed between a large set of complete web pages. The documents which we are working with in these experiments are much smaller, namely tens of words, instead of hundreds or thousands in the case of web pages. Further, we are looking for a high similarity in order to classify something similar, while the earlier work is focused on finding near-duplicate web pages. Finally, when comparing web pages there will often be a large impact from the frequencies

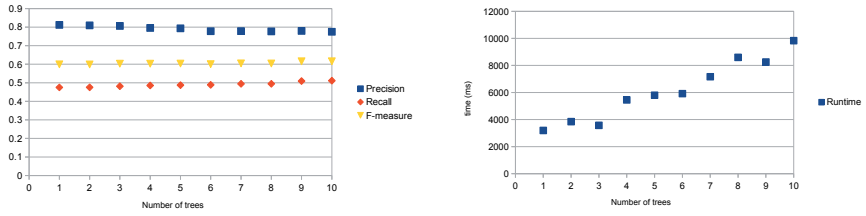
of words. In the current work, however, the frequencies are usually very small numbers. Since these results are not satisfying for the setting we are developing, we will not continue using RHH and f-RHH for further experiments. However, we would still like to highlight the performance difference between RHH and f-RHH. As can be seen from the graphs, f-RHH achieves a much better precision compared to RHH Also the recall and hence F-measure are always higher than what we obtained using RHH. Hence, it would be worth investigating further whether f-RHH works better compared to normal RHH in other use cases.



(a) Quality measurements RHH (b) Quality measurements RHH

Fig. 6: Quality measurements of an ontology matching task using different number of trees using RHH and f-RHH.

Part 3 To try whether we can also use the tree for bigger datasets, we now feed all knowledge tokens created from all three ontologies into the system and present similar measurements in fig. 7.



(a) Quality measurements (b) Runtime

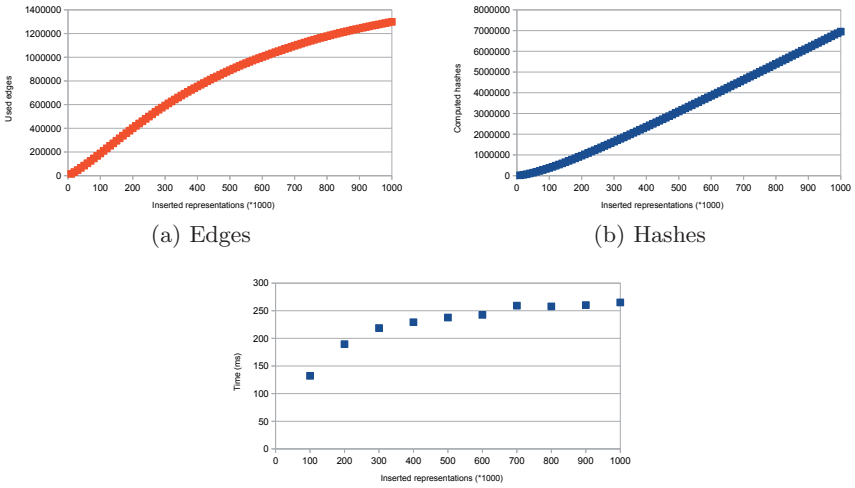
Fig. 7: Quality measurements and runtime behavior for a three way ontology matching task using different number of trees.

Now, we notice the effect on the precision and recall more profoundly. Also the runtime increases faster when the input is larger. We do however see only a three-fold increase when the number of trees is ten-folded. When comparing these results to our earlier work [10] we can see the speed-up of using LSH Forest

and performing multiple alignments at once. In our previous work we used 45.5 seconds for doing three 2-way alignment tasks. Using the LSH Forest we can perform the 3-way alignment in less than 10 seconds. When using a single tree, we measured a time of 3.2 seconds yielding roughly a ten-fold speed-up.

5.3 Adding Dynamics

The results of adding knowledge tokens according to a uniform distribution are in fig. 8. From the figures we note that the number of edges needed grows sub-



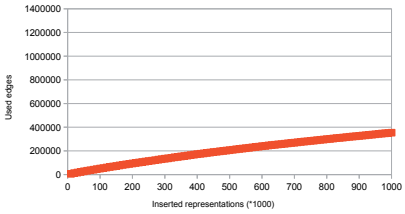
(a) Edges (b) Hashes

(c) Time needed to insert the previous 100000 concepts

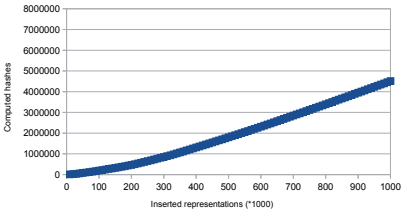
Fig. 8: Cumulative number of edges and hashes; and time needed for uniform adding of knowledge tokens

linear. This is as expected since both the fact that certain knowledge tokens will be selected more than once and the reuse of edges decreases the number of new edges needed. The number of hashes shows an initial ramp-up and then starts growing linear. We also note that the time used for adding is growing, but the growth slows down when more concepts are added. Moreover, if we try to fit a linear curve trough the cumulative runtime measurements, we notice that we can obtain a Pearson product-moment correlation coefficient of 0.9976, indicating that the increase is actually very close to linear.

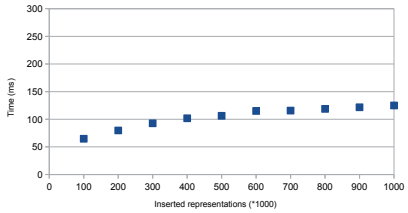
When choosing the representations using a Zipf distribution instead, we obtain the results as depicted in fig. 9. When comparing the charts for insertion using the normal and Zipf distribution, we notice that the later puts much less of a burden upon the system. This is a desirable effect since it means that the



(a) Edges



(b) Hashes



(c) Time needed to insert the previous 100000 concepts

Fig. 9: Cumulative number of edges and hashes; and time needed for adding of knowledge tokens according to a Zipf distribution

system is likely to work well with more organic loads. Also here, we can fit a linear curve trough the cumulative runtime measurements with a high correlation coefficient of 0.9968.

6 Conclusions and Outlook

When trying to understand and follow what is happening around us, we have to be able to connect different pieces of information together. Moreover, the amount of information which we perceive does not allow us to look at each detail, instead we need to focus on specific parts and ignore the rest. When we want to built a system capable of embodying evolution in knowledge, similar challenges have to be tackled. In this paper we investigated one of the first steps needed for this type of system, namely bringing related pieces of knowledge together.

The system we envision is an Evolving Knowledge Ecosystem in which Knowledge Organisms are able to consume Knowledge Tokens, i.e., pieces of knowledge, which have been sown in the environment. In this paper we looked at the application of LSH Forest to dynamically sow knowledge tokens in the environmental contexts.

We found out that LSH Forest is a suitable approach because it is able to balance well between efficiency and effectiveness. This can be observed from the fact that the method scales well, both from a space and runtime perspective; and from the fact that the quality measures are sufficiently high when using minhash. Further, the Forest makes it possible to focus on these parts which need further investigation and it allows for connecting between the knowledge tokens. We also investigated the use of cosine distance using random hyperplane hashing. From our observations we noticed that this approach performs poorly in comparison to minhash. This seems contradictory to earlier findings [12], but is likely because of the fact that the documents which are being compared are very different in nature (short labels vs. complete web pages).

There are still several aspects of using LSH Forest which could be further investigated. First, the problem caused by exclusive chains could be mitigated by measuring the distance between knowledge tokens when they reach a certain depth in the tree. Only when the concepts are different enough, there is a need to continue; this however requires to parametrize the inequality. Another option to reduce at least the amount of used memory and pointer traversals is using PATRICIA trees as proposed by Bawa et al. [2].

Secondly, we noted that the LSH tree allows for removal of concepts and that this operation is efficient. Future research is needed to see how this would work in an evolving knowledge ecosystem. Besides, as described in [1], the knowledge tokens do not disappear at once from an environmental context. Instead, they might dissolve slowly, which could be thought of as a decreasing fuzzy membership in the context. One straightforward method for achieving this would be to use a sliding window which has an exponential decay. Also more complex ideas could be investigated, perhaps even providing a bonus for concepts which are queried often or using hierarchical clustering techniques to remove tokens from areas which are densely populated [13]. This would mean that some tokens remain in the system even when other (less popular or more common) concepts with similar insertion characteristics get removed.

Thirdly, we observed that f-RHH performed better than the traditional RHH. The improvement was still not enough to warrant its use in the context of this

paper, however. As a further direction it would definitely be beneficial to see a large scale comparison between standard RHH, f-RHH, and perhaps multi-proble LSH [14].

Lastly, it would be interesting to see how the Forest would react when the input data becomes that big that it is impossible to keep the tree in the physical memory available. Then, using a distributed setting, ways should be found to minimize the overhead when concepts are added and removed from the tree. One promising idea is the use of consistent hashing for the distribution of knowledge tokens as proposed in [15].

7 Acknowledgments

The authors would like to thank the department of Mathematical Information Technology of the University of Jyväskylä for financially supporting this research. This research is also in part financed by the N4S SHOK organized by Digile Oy and financially supported by TEKES. The authors would further like to thank Steeri Oy for supporting the research and the members of the Industrial Ontologies Group (IOG) of the University of Jyväskylä for their support in the research. Further, it has to be mentioned that the implementation of the software was greatly simplified by the Guava library by Google, the Apache Commons Math^{TM} library, and the Rabin hash library by Bill Dwyer and Ian Brandt.

References

1. Ermolayev, V., Akerkar, R., Terziyan, V., Cochez, M.: Towards Evolving Knowledge Ecosystems for Big Data Understanding. In: Big Data Computing. Taylor & Francis group - Chapman and Hall/CRC (2014) 3–55
2. Bawa, M., Condie, T., Ganesan, P.: LSH forest: self-tuning indexes for similarity search. In: Proceedings of the 14th international conference on World Wide Web, ACM (2005) 651–660
3. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing, ACM (1998) 604–613
4. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1) (January 2008) 117–122
5. Rajaraman, A., Ullman, J.D.: 3. Finding Similar Items. In: Mining of massive datasets. Cambridge University Press (2012) 71–128
6. Cochez, M., Terziyan, V., Ermolayev, V.: Balanced large scale knowledge matching using lsh forest. In Cardoso, J., Guerra, F., Houben, G.J., Pinto, M.A., Velegrakis, Y., eds.: Semantic Keyword-based Search on Structured Data Sources: First COST Action IC1302 International KEYSTONE Conference, IKC 2015, Coimbra, Portugal, September 8-9, 2015. Revised Selected Papers. Volume 9398 of Lecture Notes in Computer Science. Springer International Publishing, Cham (2015) 36–50
7. Broder, A.Z.: On the resemblance and containment of documents. In: Compression and Complexity of Sequences 1997. Proceedings, IEEE (1997) 21–29

8. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing. STOC '02, New York, NY, USA, ACM (2002) 380–388
9. Ermolayev, V., Davidovsky, M.: Agent-based ontology alignment: Basics, applications, theoretical foundations, and demonstration. In: Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics. WIMS'12, New York, NY, USA, ACM (2012) 3:1–3:12
10. Cochez, M.: Locality-sensitive hashing for massive string-based ontology matching. In: Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on. Volume 1., IEEE (2014) 134–140
11. Broder, A.: Some applications of rabin's fingerprinting method. In Capocelli, R., Santis, A., Vaccaro, U., eds.: Sequences II. Springer New York (1993) 143–152
12. Henzinger, M.: Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM (2006) 284–291
13. Cochez, M., Mou, H.: Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM (2015) 505–517
14. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe lsh: efficient indexing for high-dimensional similarity search. In: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment (2007) 950–961
15. Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., Lewin, D.: Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. STOC '97, New York, NY, USA, ACM (1997) 654–663

PVI

TWISTER TRIES: APPROXIMATE HIERARCHICAL AGGLOMERATIVE CLUSTERING FOR AVERAGE DISTANCE IN LINEAR TIME.

by

Michael Cochez and Hao Mou 2015

In T. Sellis, S. Davidson, & Z. Ives (Eds.), SIGMOD '15 : Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, New York: Association for Computing Machinery. This article received the Reproducible label http://db-reproducibility.seas.harvard.edu/papers/

PVII

SCALABLE HIERARCHICAL CLUSTERING : TWISTER TRIES WITH A POSTERIORI TRIE ELIMINATION.

by

Michael Cochez and Ferrante Neri 2015

In SSCI 2015 : Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence. Symposium CIDM 2015 : 6th IEEE Symposium on Computational Intelligence and Data Mining

Reproduced with kind permission of IEEE Computer Society.