HPI Hasso Plattner Institut

Universität Potsdam

# Enriching the Web of Data with Topics and Links

April 2013

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

eingereicht von
Dipl.-Inf. Christoph Böhm

begutachtet von
Prof. Dr. Felix Naumann, Prof. Dr. Gerhard Weikum, Prof. Dr. Sören Auer

verteidigt am
13. September 2013

Published online at the
Institutional Repository of the University of Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2013/6862/
URN urn:nbn:de:kobv:517-opus-68624
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-68624

**Abstract**

This thesis presents novel ideas and research findings for the Web of Data – a global data space spanning many so-called Linked Open Data sources. Linked Open Data adheres to a set of simple principles to allow easy access and reuse for data published on the Web. Linked Open Data is by now an established concept and many (mostly academic) publishers adopted the principles building a powerful web of structured knowledge available to everybody. However, so far, Linked Open Data does not yet play a significant role among common Web technologies that currently facilitate a high-standard Web experience.

In this work, we thoroughly discuss the state-of-the-art for Linked Open Data and highlight several shortcomings – some of them we tackle in the main part of this work.

First, we propose a novel type of data source meta-information, namely the topics of a dataset. This information could be published with dataset descriptions and support a variety of use cases, such as data source exploration and selection. For the topic retrieval, we present an approach coined *Annotated Pattern Percolation (APP)*, which we evaluate with respect to topics extracted from Wikipedia portals.

Second, we contribute to entity linking research by presenting an optimization model for joint entity linking, showing its hardness, and proposing three heuristics implemented in the *LINked Data Alignment (LINDA)* system. Our first solution can exploit multi-core machines, whereas the second and third approach are designed to run in a distributed shared-nothing environment. We discuss and evaluate the properties of our approaches leading to recommendations which algorithm to use in a specific scenario. The distributed algorithms are among the first of their kind, i.e., approaches for joint entity linking in a distributed fashion. Also, we illustrate that we can tackle the entity linking problem on the very large scale with data comprising more than 100 millions of entity representations from very many sources.

Finally, we approach a sub-problem of entity linking, namely the alignment of concepts. We again target a method that looks at the data in its entirety and does not neglect existing relations. Also, this concept alignment method shall execute very fast to serve as a preprocessing for further computations. Our approach, called *Holistic Concept Matching (HCM)*, achieves the required speed through grouping the input by comparing so-called knowledge representations. Within the groups, we perform complex similarity computations, relation conclusions, and detect semantic contradictions. The quality of our result is again evaluated on a large and heterogeneous dataset from the real Web.

In summary, this work contributes a set of techniques for enhancing the current state of the Web of Data. All approaches have been tested on large and heterogeneous real-world input.

**Zusammenfassung**

Die vorliegende Arbeit stellt neue Ideen sowie Forschungsergebnisse für das Web of Data vor. Hierbei handelt es sich um ein globales Netz aus sogenannten Linked Open Data (LOD) Quellen. Diese Datenquellen genügen gewissen Prinzipien, um Nutzern einen leichten Zugriff über das Internet und deren Verwendung zu ermöglichen. LOD ist bereits weit verbreitet und es existiert eine Vielzahl von Daten-Veröffentlichungen entsprechend der LOD Prinzipien. Trotz dessen ist LOD bisher kein fester Baustein des Webs des 21. Jahrhunderts.

Die folgende Arbeit erläutert den aktuellen Stand der Forschung und Technik für Linked Open Data und identifiziert dessen Schwächen. Einigen Schwachstellen von LOD widmen wir uns in dem darauf folgenden Hauptteil.

Zu Beginn stellen wir neuartige Metadaten für Datenquellen vor – die Themen von Datenquellen (engl. Topics). Solche Themen könnten mit Beschreibungen von Datenquellen veröffentlicht werden und eine Reihe von Anwendungsfällen, wie das Auffinden und Explorieren relevanter Daten, unterstützen. Wir diskutieren unseren Ansatz für die Extraktion dieser Metainformationen – die *Annotated Pattern Percolation (APP)*. Experimentelle Ergebnisse werden mit Themen aus Wikipedia Portalen verglichen.

Des Weiteren ergänzen wir den Stand der Forschung für das Auffinden verschiedener Repräsentationen eines Reale-Welt-Objektes (engl. Entity Linking). Für jenes Auffinden werden nicht nur lokale Entscheidungen getroffen, sondern es wird die Gesamtheit der Objektbeziehungen genutzt. Wir diskutieren unser Optimierungsmodel, beweisen dessen Schwere und präsentieren drei Ansätze zur Berechnung einer Lösung. Alle Ansätze wurden im *LINked Data Alignment (LINDA)* System implementiert. Die erste Methode arbeitet auf einer Maschine, kann jedoch Mehrkern-Prozessoren ausnutzen. Die weiteren Ansätze wurden für Rechnercluster ohne gemeinsamen Speicher entwickelt. Wir evaluieren unsere Ergebnisse auf mehr als 100 Millionen Entitäten und erläutern Vor- sowie Nachteile der jeweiligen Ansätze.

Im verbleibenden Teil der Arbeit behandeln wir das Linking von Konzepten – ein Teilproblem des Entity Linking. Unser Ansatz, *Holistic Concept Matching (HCM)*, betrachtet abermals die Gesamtheit der Daten. Wir gruppieren die Eingabe um eine geringe Laufzeit bei der Verarbeitung von mehreren Hunderttausenden Konzepten zu erreichen. Innerhalb der Gruppen berechnen wir komplexe Ähnlichkeiten, und spüren semantische Schlussfolgerungen und Widersprüche auf. Die Qualität des Ergebnisses evaluieren wir ebenfalls auf realen Datenmengen.

Zusammenfassend trägt diese Arbeit zum aktuellen Stand der Forschung für das Web of Data bei. Alle diskutierten Techniken wurden mit realen, heterogenen und großen Datenmengen getestet.

# Contents

# 1 Introduction

Data has become a main pillar of today's IT world. This change is because computational infrastructure, e.g, network and storage capabilities, now facilitate handling massive amounts of data on a daily basis at a fast pace. In the past, large amounts of data have been collected on tapes, which is an established solution for archiving. In contrast, today one can store terabytes of data on spinning or solid-state disks, which allows very fast random access. Furthermore, one can load large data volumes into main memory allowing an even faster access. In addition, fast network connections enable fast moving and sharing of data across long distances, which provides flexible and simple access to data stored in massive data centers.

The sheer availability of such a technical environment leads to novel challenges, ideas, and opportunities. For instance, organizations can not only archive their data in order to meet legal regulations; they can actually reason about future developments of products and trends given observations captured in their historical data. Also, *data itself* has become a valuable asset. That is, business models center around data collected from, for instance, social network users who can then be targeted for Web adds more precisely than in the past. Further, the ability to share large datasets allows research institutions, governments, and public agencies to open up internal data collections in order to realize transparency, repeatability and to allow interoperability.

However, the actual usability of data needs further refinement on many fronts. For instance, a manageable set of access methods (including APIs), data models and respective serializations should be standardized and adopted by a vast majority of data engineers. Also, one requires meaningful meta-information to gain an overview of the wealth of information available. Here, the state-of-the-art lacks common meta-information representations as well as methods for its actual creation. Further, there is a need for highly scalable techniques to create links among entities scattered across the Web. This is important to achieve interoperability for the Web-of-Data. In the following, we introduce Open Data and Linked Open Data in particular leading to a more detailed discussion of these and other shortcomings.

**Open Data.** Data can be considered as Open Data if it is of a general interest, available to the public and their terms of use allow some form of exploration, modification and exploitation, as well as redistribution[1]. Such data available to the public induces a variety of opportunities: Most importantly, everyone can now discover the domain described by the data. This data exploration was formerly reserved for the data owner. Hence, very many people are able to examine the data from different perspectives, which can lead to a variety of diverse findings and inspiration. The interconnection of data from disparate sources allows even further conclusions and a richer picture of the domain under consideration. In case there is a community creating and curating a data source, then, most likely, the information quality improves constantly. *Wikipedia*[2] is the prime-example of an ever-growing community-curated data source that has a very high information quality. This way, it became the de-facto standard encyclopedia for common-world knowledge on the Web.

Currently there is a broad availability of open data. Nevertheless, there are several challenges to overcome before one productively incorporates such data into a solution under consideration.

To start with, although Open Data is available to the public it is not necessarily for free. That is, publishers sell their data, or services based on this data, since their business model bases on data. For instance, *Factual*[3] offers data about public places derived from government sources. The well-known social network *Facebook*[4] is one of Factual's commercial customers. Other data that is not for free can be found at *Socrata*[5], *Microsoft Azure*'s Datamarket[6], or *DataMarket*[7].

However, there are also several providers that offer data for free, sometimes imposing usage restrictions: Some data shall not be used for commercial purposes; in other cases one is required to attribute the original publisher. Many public agencies, such as the *Worldbank*[8] or US departments[9], chose to provide their data under a *Creative Commons*[10] or similar license. Thus, using open data usually starts with exploring respective terms of use in order to make sure that the data at hand can be taken into account for the purpose under consideration.

[1] Further elaboration on how to define Open Data can be found at `opendefinition.org/okd/` and `opendatahandbook.org/en/what-is-open-data/`
[2] `http://www.wikipedia.org`
[3] `http://www.factual.com`
[4] `http://www.facebook.com`
[5] `http://www.socrata.com`
[6] `https://datamarket.azure.com`
[7] `http://www.datamarket.com`
[8] `http://data.worldbank.org`
[9] `https://data.gov`
[10] `http://creativecommons.org`

In addition to the license limitations, one often has to overcome several technical heterogeneities that arise from different data origins: First, there are many ways to access Open Data. It can be downloadable as a whole, e.g., as flat files or database dumps, or accessible via APIs allowing CRUD-style operations (Create, Read, Update and Delete). The *Open Data Protocol*[11], for instance, has evolved into a widely-used data access API. In contrast, for instance, Facebook's *Open Graph Protocol*[12] is a proprietary API to incorporate third-party content into Facebook's social network and vice-versa. Vendor-specific solutions harbor the risk of frequent API changes and discontinuation. Google's *Data Protocol*[13] shows that global players also abandon house-made proprietary solutions.

Then, Open Data comes in a variety of different formats: Some sources allow the download of the entire collection as CSV files or database dumps while others offer pieces of data in JSON, XML, or publisher-specific serializations.

After accessing the data and processing the respective format, the next challenge is to understand the schema of the data and the semantics of the data as a whole and single fields in particular. An `amount` field, for instance, can be parsed as a string, an integer or floating point value. Then, the question remains what the field refers to, which unit is to be used, and what it actually means, e.g., it might comprise a price or a number of pieces. Thus, understanding data requires a detailed study of proper data documentation to avoid misusage.

Evidently, the aforementioned challenges hinder inspiration and innovation leading to novel ideas, findings, and applications.

**Linked Open Data.** Seemingly independent from these real-world challenges arising from data heterogeneity, the idea of Linked Open Data (LOD for short) emerged from the Semantic Web community. Today, Linked Open Data often stands for the core vision of the Semantic Web, i.e., un-siloing and connecting data on the Web[14]. In 2006 Tim Berners-Lee featured so-called Linked Open Data as one of his informal design issues [BL06]. LOD bases on techniques developed in the area of Semantic Web research, also initially proposed by Berners-Lee et al. [BLHL01]. In analogy to the HyperText Web, links shall connect arbitrary real-world entities described with RDF, i.e., the Resource Description Framework [KC04]. RDF is a graph data model that can be seen as a set of triples consisting of subject, predicate, and object. The subject is the resource

[11] `http://www.odata.org`

[12] `http://developers.facebook.com/docs/opengraphprotocol`

[13] `https://developers.google.com/gdata`

[14] `http://www.scilogs.com/web_science/what-is-the-semantic-web-really-all-about`, `http://tomheath.com/blog/2009/03/linked-data-web-of-data-semantic-web-wtf`

under consideration. The predicate describes the relationship among the subject and the object, which is another resource or a literal. Actual data values, e.g., strings, numbers, dates, etc. appear as literals. Then, the graph's nodes are subjects and non-literal objects connected through edges labeled with predicate names. The Linked Data design issues state four rules in order to make LOD most valuable for the Web [BL06, SH13]:

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up an URI, provide useful information, using the standards.
4. Include links to other URIs so that they can discover more things.

The first rule states a common-sense in the Semantic Web field to clearly identify things. The second rule proposes the use of the standardized and wide-spread HTTP naming scheme, which includes a commitment to the established Domain Name System. Rule three asks publishers to return valuable information if such HTTP URI is dereferenced, i.e., an HTTP request is issued. This shall facilitate human-readability. Note that it does not suggest what to return in particular. Thus, one can return the entity with *all* its predicates and objects, or only predicates *with literal values*, or *all neighboring entities* in the RDF graph versus just some text. The fourth rule can lift the data to the next level in the value chain, since it asks publishers to connect their data entries to other data sources. Given such connections, one can easily follow links to discover further interesting information for the entities under consideration. Note that this is a major step forward: Data shall not exist in isolated silos anymore. Instead, data should reside in a context consisting of other entities described in further LOD sources on the Web. This way, one could answer Join-queries spanning multiple sources without additional effort, e.g., entity reconciliation, beforehand.
Beyond being connected, all LOD sources provide the same access methods, namely the linked data interface as well as SPARQL endpoints [BCH07], and return their information using a common data model, RDF. To be complete, there are several RDF serializations, e.g., XML [BM04], N3 [BL05, BLC11], JSON-LD [SLK+12], or the Query Result JSON format [Sea11].
Along with RDF there are two standards for describing the underlying schema of the data, namely *RDFS* [BG04] and *OWL* [W3C09]. The former, RDF Schema, defines a vocabulary for describing RDF vocabularies. RDFS, with its namespace `http://www.w3.org/2000/01/rdf-schema#`, is commonly used in LOD. The latter, the Web Ontology Language, allows a more concise formal specification of vocabularies and respective data. For instance, `sameAs` can be used to express the equivalence of entities described

in individual sources. Schema information is a first-class citizen in LOD sources, since RDFS and OWL use RDF to describe this information. Thus, the schema *can* be inherent in the data at hand. However, observations show that LOD often comes as a loose RDF graph without a fixed set of relationships or a well-defined type of an entity.

Apart from a variety of RDF serializations, LOD with its unified data model, the common access method, the schema captured in the data and the links across sources may sound like a silver bullet for today's data on the Web. Indeed, it is a promising design vision, since it tackles many problems resulting from simply "throwing data on the Web". Further, with SPARQL [PS08, HS12], the RDF query language, it captures tremendous potential: Imagine that one could possibly query the *Web of Data* – formed by interlinked individual LOD sources – in a way we are used to query relational datasources with *SQL*. The LOD community currently examines efficient federated SPARQL query processing [HMZ10, SPFW13]. However, one is aware that the schema-less nature of LOD poses a major problem for query federation [PHHD10]. In general, federated query answering is a challenging task that has been under consideration in the database community for decades [SL90, Kos00, OV11].

Within the past years the so-called LOD cloud, comprising interlinked LOD sources, evolved tremendously. Figures 1.1 and 1.2 depict interlinked data sources that fulfill the basic LOD criteria[15]. From 09/2007 to 09/2011 the number of sources grew from 28 to 295, i.e., 31.6 billion triples with 504 million (out)links[16]. At the time of writing (03/2013), the number of sources approached 330 and the LOD cloud comprises data from the media, geography, life sciences, and governments as well as large amounts of user-generated content – see Table 1.1.

| Domain | # Datasets | # Triples | in % | # (Out-)Links | in % |
|---|---|---|---|---|---|
| Media | 25 | 1,841,852,061 | 5.82 | 50,440,705 | 10.01 |
| Geographic | 31 | 6,145,532,484 | 19.43 | 35,812,328 | 7.11 |
| Government | 49 | 13,315,009,400 | 42.09 | 19,343,519 | 3.84 |
| Publications | 87 | 2,950,720,693 | 9.33 | 139,925,218 | 27.76 |
| Cross-domain | 41 | 4,184,635,715 | 13.23 | 63,183,065 | 12.54 |
| Life sciences | 41 | 3,036,336,004 | 9.60 | 191,844,090 | 38.06 |
| User-generated content | 20 | 134,127,413 | 0.42 | 3,449,143 | 0.68 |
| | 295 | 31,634,213,770 | | 503,998,829 | |

Table 1.1: Number of triples as well as RDF links per domain.
Source: `lod-cloud.net/state/#domains`, Version 0.3 as of Sep. 2011 by Bizer et al.

[15] `http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets/CKANmetainformation` states that data items shall be accessible via dereferencable URIs and that datasets must have at least 50 RDF in- or outgoing links.

[16] `http://lod-cloud.net` as of September 2011

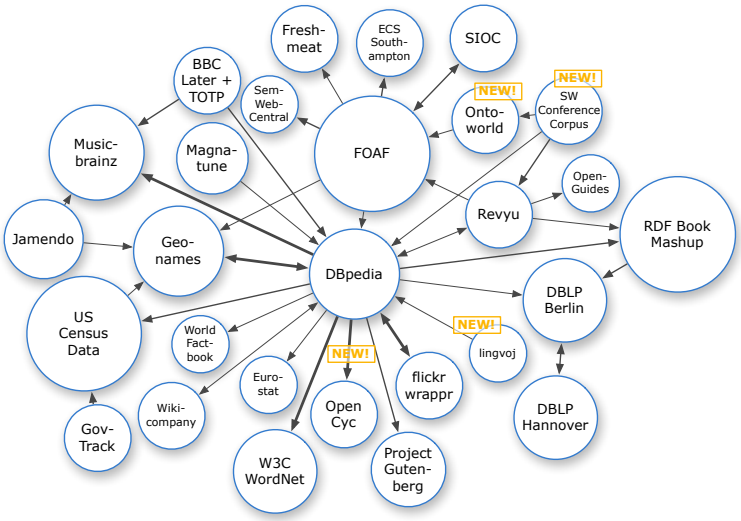


Figure 1.1: The LOD cloud as of 11/2007.
Source: `lod-cloud.net` by Cyganiak and Jentzsch.

Additionally, there exist vast amounts of data in webpages. For this, one uses *Microdata*, *Microformats*, or *RDFa* to embed machine-readable information into HTML pages. Microdata has been around for years and is currently further evolving in the context of the *HTML5* standard [BLN+12, Hic12]. Microformats are a set of purpose specific formats developed independently[17]. RDFa [ABMH12] development has been driven by the Semantic Web community and represents a specification of attributes to express structured data in any markup language. Currently, there is a wealth of information embedded into websites. Table 1.2 shows recent numbers obtained from a *Bing* Web crawl as of January 2012 [MP12]. In this crawl approximately 5% of the pages contain embedded data. The latest development for embedding data into webpages is *schema.org*[18] – an initiative driven by today's global search engine players.

For some reason, however, the LOD design vision has not yet come into play for the everyday Web-experience. That is, it has neither been adopted by commercial data providers nor has it become the underpinning for currently evolving web applications that gather very many users. In the latter case it has simply not yet been accepted by the millions of engineers creating today's Web. Yahoo's Web of Concepts [DKP+09] and Google's Knowledge Graph [gkg12] are notable exceptions since they borrow ideas from LOD and contribute to the field. We observe that Google's Knowledge Graph is powered by large amounts of structured data and, apparently, Web content is constantly aligned with that

[17] `http://microformats.org`
[18] `http://schema.org`

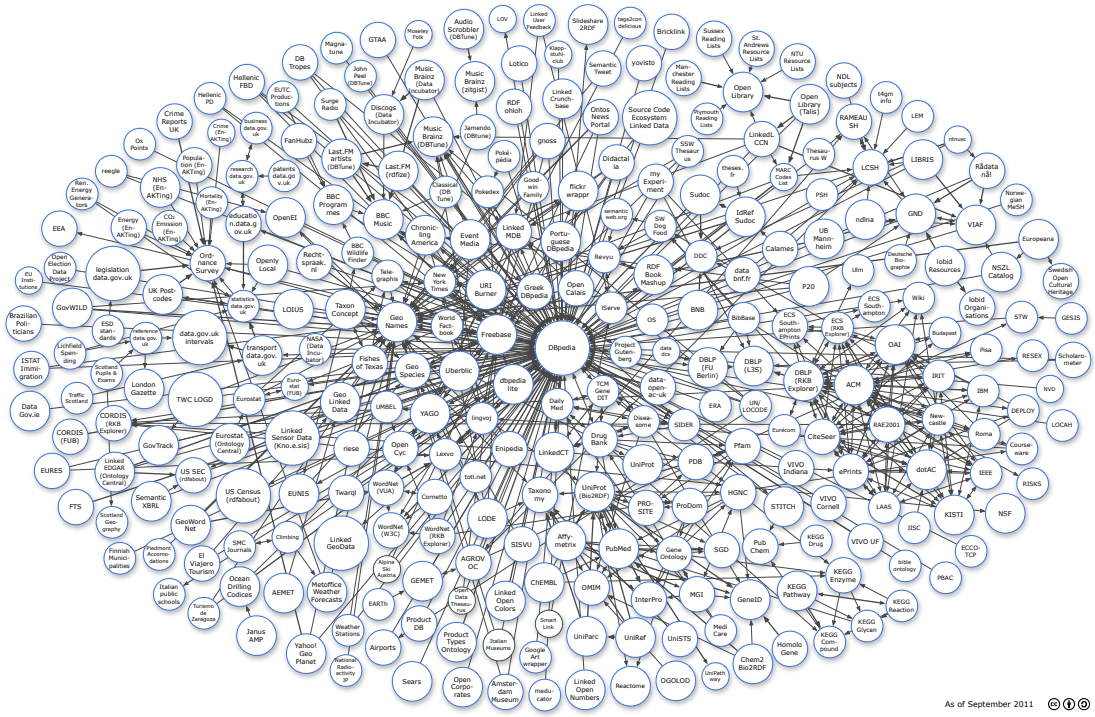


Figure 1.2: The LOD cloud as of 09/2011.
Source: `lod-cloud.net` by Cyganiak and Jentzsch.

data. Technical details remain unpublished so far. The LOD community, however, is aware of the discrepancy between LOD and the real Web. At most large international conferences in this realm this deficit is under discussion, e.g., at the Semantic Web Conference (ISWC) or the World Wide Web Conference (WWW) – lately in the *Linked Data on the Web workshop* (LDOW) at WWW 2012 [Biz12]. Among others, reasons for this discrepancy might include[19]:

**(1) Lack of Timeliness, Data Quality, and Influence.** Before an organization includes a data source into its business model, it requires certain guarantees. For instance, the data must adhere to some well-structured data model; all entities' descriptions should be "equally complete", i.e., they should provide the same set of properties; all property values must have a particular type; etc. Also, being up-to-date at all times is a prerequisite. Further, if such requirements are not met, there are no simple means to contribute

[19]This argumentation is the authors personal opinion resulting from following the LOD development. Nevertheless, Hogan et al. also categorize current problems of the Web of Data [HHP+10]. They identify accessibility issues, syntactical problems, data inconsistencies, as well as non-authoritative contributions.

| Site | Triples |
|---|---|
| facebook.com | 2,737,580,921 |
| yahoo.com | 856,294,566 |
| tabelog.com | 659,670,199 |
| tripadvisor.co.uk | 559,423,276 |
| youtube.com | 480,077,405 |
| tripadvisor.com | 478,847,932 |
| venere.com | 365,571,734 |
| myspace.com | 266,485,235 |
| tripadvisor.it | 184,566,490 |
| yellowpages.com | 164,303,902 |

Table 1.2: Top sites by number of triples. Source: Table 2 in [MP12].

to or influence a data source of interest in order to improve it. In [PHHD10], Polleres et al. discuss several LOD specific data quality issues inhibiting consumer applications from fully exploiting that data. Zaveri et al. acknowledge the wide range of Web data quality and give a comprehensive literature review an data quality metrics that can be applied to LOD [ZRM+12].

**(2) Lack of Reliable Availability.** Since many sources stem from academia or the public sector and thus base on non-stable, non-commercial funding, there is no guarantee that a data source will be up and running for the next years. As for the US government's data site (*Data.gov*), for instance, there was a budget reduction from $37 to $2 million in 2011[20]. Currently, the EU is promoting the Digital Agenda to "turn government data into gold"[21]. For this, it grants €100 million from 2011 to 2013 to improved data-handling technologies. But it is not clear what could happen a few years later. Nevertheless, at the time of writing (03/2013), releasing public-sector information seems to emerge as a major use-case for LOD. Note that MONDECA, a company dedicated to providing technology for the management of knowledge structures, maintains a list of SPARQL endpoint availability[22], which usually ranges around 60%. Of course, this issue also applies to other resources freely available on the Web, including arbitrary APIs, etc.

**(3) Lack of Communication Towards Web Engineers.** The Semantic Web started with the ambitious goal to capture the world's knowledge and facilitate sophisticated techniques, such as logical inference. The Linked Open Data cloud in particular has been built as a gigantic academic experiment. Up to this point, the LOD development has

[20] http://www.guardian.co.uk/news/datablog/2011/apr/05/data-gov-crisis-obama
[21] http://europa.eu/rapid/pressReleasesAction.do?reference=IP/11/1524
[22] http://labs.mondeca.com/sparqlEndpointsStatus/index.html

mainly been driven by a few hundred senior researchers (from academia and industry) without attracting many small (e.g., start-ups) or large players that dominate the Web consumed by billions of people every day. This lack of attraction might be caused by a lack of communication, e.g., at respective developer conferences. As a result, developers consider the Semantic Web technology stack, including RDF and SPARQL, as academic, cumbersome, and bulky. However, some transmissions from academia to industry can be found. For instance, `schema.org` is a project powered by the major search engine providers.

**(4) LOD is still in its Infancy.** While the former reasons stem from an outside LOD point-of-view, there are shortcomings in the actual deployment of the current Web of Data. Hogan et al. evaluate the Web of Data conformance [HUH+12] with respect to rules taken from the de-facto guide book on how to publish linked data on the Web [HB11]. The authors give an impressive survey of the LOD deployment status, analyzing naming issues, interlinkage, resources descriptions, and accessibility. They conclude, for instance, that there is a lack of human readable meta-information and that data providers, often, do not provide locally-known in-links.

Links are indeed a major issue. In [HB11], Heath and Bizer discuss three important types of RDF links, namely relationship links, identity links (mainly `sameAs` links), and vocabulary links. The Web of Data currently does not contain as many links as one would hope for. For instance, `sameAs` links connect two coreferent resources that describe the same real-world entity [HB11]. Hogan et al. state that only 29.8% of the top-level domains link to external domains [HUH+12][23]. *SameAs.org*, a service collecting, managing and providing equivalent URIs [GJM09], has links for approximately 43 million unique URIs. Bizer et al. state that there are (only) approximately 504 million links among 31.5 billion triples in the LOD cloud – see Table 1.1. Obviously, most links stem from life science and bibliographic data sources. Cross-domain-, media-, and user-generated content sources, where links would be most valuable for general purposes, do not provide as many links. Unfortunately, many of the existing links are questionable [DSFM10, HHM+10]. Halpin et al. found only 51% of the `sameAs` links to be correct[24]. The `sameAs` deployment status is further considered in [DSSM10, HZU+12]. In the latter paper, the authors estimate `sameAs` link accuracy to be app. 97% – observing that many of them are trivially correct. This is a poor status, given that already the

[23] The corpus used has been crawled in 2010 and contains 947 million unique triples spanning content from 778 data providers, i.e., top level domains.

[24] In [HHM+10] the authors used copy of the LOD cloud hosted by OpenLink, which amounts to 58,691,520 sameAs triples from 1,202 unique domain names.

name, *Linked* Open Data, features links as a major building block for an interweaved Web of Data.

Another major issue is the lack of meaningful meta-information published with a data source, i.e., meta-information that can be understood by human beings but also interpreted by machines. Some publishers consider this meda-data issue – for instance when creating open government data [FCFOP+11]. Jain et al. state that "currently there is no mechanism to describe that *Jamendo* captures music related information, whereas *Geonames* captures geographical information. This is a serious drawback if we envision applications that could seamlessly harness the vast number of facts present in the cloud." [JHY+10] Further, Frosterus et al. state "metadata available for [public] datasets is often minimal, heterogeneous, and distributed, which makes finding a suitable dataset for a given need problematic." In particular, "finding suitable datasets based on different selection criteria such as subject topic, size, licensing, publisher, language etc. is not supported." [FHL11] To overcome this, it requires novel meta-information that could be published as part of voiD description [ACHZ09, ACHZ11], provenance information [BCC+13], SPARQL Service Descriptions [Wil12] or in data metadata repositories.

This thesis provides novel research findings for the latter two issues. We devise approaches for the discovery of links among entities described in disparate sources [BMNW12]. This field of research coined many names for the problem, e.g., entity matching, entity alignment, or entity assignment, and deduplication. Note that, given the current size of the Web of Data, these techniques must scale to very large and super-heterogeneous datasets. Further, most recent entity matching approaches conduct joint reasoning, i.e., matching decisions are interdependent across types of data and processing steps. In our approach, we perform joint reasoning in a scalable manner. In particular, we present a general optimization model for joint entity matching. For this model, we describe an approximation of the global optimum and elaborate on three different implementations: The first implementation, developed with Gerard de Melo, runs on a single machine but can exploit multi-core processors. The second and third versions run on clusters of machines in a shared-nothing manner still performing joint reasoning. For this, we leverage Map/Reduce as well as pure message-passing. Also, we feature an additional approach, developed with Toni Grütze [Grü11], dedicated to the matching of concepts from the Web of Data [GBN12]. Here, we came up with a novel grouping scheme and incorporate it into an existing approach.

As a second major contribution, we propose novel meta-information for the description of graph-structured data, such as RDF [BKN12]. For this, Johannes Gosda and Eyk Kny contributed to the implementation and experimentation as part of their master

theses [Gos11, Kny12]. We suggest to use sets of concepts for representing the topic of a dataset. We present a scalable approach, coined Annotated Pattern Percolation, to derive this information. This way, we facilitate an abstract description of the semantic content, which can be used in a multitude of scenarios. For instance, it supports human understanding. Also, given a topic under consideration, it can be used to select and process an appropriate partition of a large dataset. To the best of our knowledge, the state-of-the-art does not comprise a comparable approach for the abstract description of graph data.

**Structure of this Work.** In the following chapter we introduce different aspects of contributing to the Web of Data. We distinguish the creation of data (Section 2.1) and meta-information from data (2.2) as well as the creation of links among data entries (Section 2.3). Then, this thesis contributes the following research findings: In Chapter 3, this work discusses our novel approach for meta-information derived from graph-structred data. In Chapter 4, we present our scalable algorithms performing joint inference for aligning entities in the Web of Data. Third, in Chapter 5, this work introduces our scalable and holistic approach for the well-known problem of concept alignment in the Web of Data – a special case of ontology alignment. Eventually, Chapter 6 gives on overall conclusion of this work.

# 2 Contributing to the Web of Data: State-of-the-Art

From a consumer's point-of-view the Web of Data consists of the actual data, meta-information describing that data, and links among data from different providers. In the following, we give an overview of techniques, platforms, and tools that facilitate the population and growth of the Web of Data. Some of these contributions stem from different research communities; others have been developed by Web engineers. Note that we additionally mention techniques that have not been created for the Web of Data in particular but deal with equivalent challenges in different domains. The selection of these additional techniques has been conducted with respect to the technical contributions of this thesis, i.e., the extraction of topics from graph-structured data and the discovery of links among entities in distributed data sources.
We first discuss the creation of the actual data including some prominent Linked Open Data (LOD) examples from various domains (Section 2.1) followed by a discussion of different means for the creation of meta-information (Section 2.2). In the third part, we discuss selected techniques for matching entity representations, i.e., methods that can be used for creating links among LOD (Section 2.3).

## 2.1 Creating Data and Schemata

**Data Publishing Pipelines.** There is a broad discussion on how to publish Linked Data on the Web and the respective life cycle management [ATvN+12, CHM11, Hyl11, HVTH12, Woo11]. In general, it is a process comprising data modeling (including naming and describing things) and data generation or data conversion. Conversion can be conducted fully automatic or in a scripting manner. Mostly, the latter leads to better results since scripts created for a particular input dataset can fully incorporate data modeling decisions. Note that there has not been a consensus on how to convert existing, mostly relational data to graph data. However, the community is currently developing standards and tools to support this process. A nice overview of tools can be found in Chapter 6.2 of [Woo11]; examples include Triplify [ADL+09] and D2RQ [BS04]; W3C recommendations are available in [ABPS12, DSC12].

**The Major Data Publishing Platform.** To reach a wide audience data publishing platforms need to announce the existence of a dataset. *Ckan* is a prominent data publishing platform. In particular, *ckan* is a data catalog system, which contains dataset meta-data including pointers to the actual data. Though this is a powerful resource for exploring and finding data on the Web, it has the problem of dead pointers, e.g., reference entities cannot be shown or the actual dataset access method is outdated. Thus, the *ckan* system apparently requires significant manual effort to maintain dataset information for the publisher on the one hand and to discover up-to-date datasets for the consumer on the other hand. Since recently, all dataset meta-information can also be obtained as RDF data. Beside numerous regional installations, e.g., Berlin Open Data[25], there are currently three major installations of the *ckan* system: `datahub.org`, `data.gov.uk`, and `publicdata.eu`. Likely, the first, `datahub.org`, is the most prominent example as it contains thousands of datasets contributed by various parties. Also it has a group dedicated to LOD[26].

`Data.gov.uk` is the UK government's official *ckan* installation to transparently organize public sector datasets. Though not all datasets are available as LOD, the UK government plays a major role in pioneering LOD in the public sector. Also note that the W3C guidance on publishing open government data [BH09] suggests that data should be published as soon as it is available in its original format so that it is instantly available to the public. Then, over time, one should set up a catalog including documentation and offer human-readable as well as machine-processable transformations.

`PublicData.eu` is developed by the Open Knowledge Foundation, a UK-based non-profit organization. This *ckan* installation has been created as part of the LOD2 project[27], a research project funded by the European Commission dedicated to the development of methods for exposing and managing large amounts of structured information. On the project site one can find an intuitive grouping of the data: `publicdata.eu/group`. On the other hand, presented meta-information, such as maintainer or provenance, is relatively scarce. An example for data published as part of the LOD2 project is statistical data from the world bank[28]. The LATC project[29] is another project in the same realm targeting (among others) infrastructure development, data publishing as well as best practices for publishing Linked Data.

[25] `http://daten.berlin.de`
[26] `http://thedatahub.org/group/lodcloud`
[27] `http://lod2.eu`
[28] `http://lod2.eu/BlogPost/1219-publishing-world-bank-linked-data.html`
[29] `http://latc-project.eu`

**Linked Open Data Examples.** A platform like *ckan* is a means of publishing and organizing pointers to datasets. In the following we briefly discuss selected examples for datasets from different domains published as LOD on the Web. GeoSpecies[30] contains information about various species, such as Fungi or Animalia, organized hierarchically and linked to DBpedia and other LOD. Further, other life-science-related sources, such as the Gene Ontology[31] and HomoloGene[32], can be accessed as LOD – though, unfortunately, not reliably at the time of writing.

From the public sector there is, for instance, the Linked Open Government Data portal [DLE+11] developed at the RPI TWC facilitating simple conversion and publication of raw government data as LOD. Note that the US government collaborates with the RPI TWC for publishing LOD[33].

For public data that has not been published as LOD, we adopted a classic integration pipeline performing data scrubbing and mapping, entity matching, as well as data fusion [BFH+12]. The integration result is then published as LOD at `govwild.org`. In other cases, public agencies, such as the European Environment Agency, publish LOD directly, e.g., emission trading data as well as data about habitat distribution[34].

DBpedia [BLK+09], Freebase [BEP+08, MGM07], and Yago [HSBW12, SKW07] are well-known Wikipedia-based cross-domain datasets. DBpedia originally used infoboxes and a static mapping from Wikipedia infobox templates to classes for the extraction of structured data. Today, the DBpedia team additionally offers community-based mapping creation and an online Wikipedia content extraction. Freebase contains data contributed from the community seeded with information from (among others) Wikipedia and MusicBrainz[35]. Freebase is not necessarily an LOD source but comes in a graph format and comprises many links to other sources on the Web. Yago combines Wikipedia data and WordNet [Fel98]. Initially, Suchanek et al. mapped Wikipedia categories to WordNet synsets to create a large term hierarchy. Recently, Hoffart et al. additionally incorporated temporal and spatial data [HSBW12]. The datasets above mentioned have been (partly) extracted from mostly unstructured Wikipedia data. Note that currently an effort called *Wikidata*[36] evolves – an initiative to expose Wikipedia knowledge as structured information. In particular, the project aims at underpinning Wikipedia in that it provides statements about things including their provenance. With this, Wikipedia data extraction efforts could become obsolete.

[30] `http://lod.geospecies.org`
[31] `http://thedatahub.org/dataset/bio2rdf-goa`
[32] `http://thedatahub.org/dataset/bio2rdf-homologene`
[33] See `data.gov` and `http://logd.tw.rpi.edu`.
[34] `http://rdfdata.eionet.europa.eu/`
[35] `http://wiki.freebase.com/wiki/Data_sources`
[36] `http://www.wikidata.org`

As for Media data, the New York Times (NYT) and the BBC are prime examples for LOD publishers. The NYT uses tags to organize its media content into topic pages. From this information it publishes data about people, organizations, and locations, which was mostly mapped manually to other sources like DBpedia, Freebase or Geonames. Unfortunately, at the time of writing, it seems like the NYT is not going to publish data on a regular basis – latest updates date two years back. In contrast, the BBC constantly releases program-, music-, and other data.
GeoNames[37] is an impressive data source containing over 10 million geographical names and consisting of over 8 million unique features integrated from a multitude of sources from many public agencies, Wikipedia, the World Factbook, etc.

**Ontology Learning.** Before the Semantic Web community approached the direct conversion and generation of data for plugging it into the LOD cloud there was vivid activity in the field of ontology learning from unstructured data. Here, we proposed an approach that combines heterogeneous evidences into a graph and then selects the target ontology structure leveraging an algorithm based on the dominating set problem [BGL09]. We then examined structural properties of the result. A prominent work in a similar vein, with regard to structural properties, has been published in the life science field [AXH+10]. Of course, there are various frameworks and tools covering the entire ontology life-cycle: Prominent examples include the framework by Maedche and Staab [MS01] as well as Protege by Noy et al. [NSD+01]. The most recent review of the field has been published by Wong et al. [WLB12]. Additional surveys of ontology learning approaches can be found in [BCM05, DG08, HEBR11, ZN10]. The broader field of ontology engineering has been reviewed in [CFLGP03, DSW+00, SMB10].

**Leveraging Web Tables.** Apart from these efforts to create integrated and structured LOD sources there is a recent trend to query knowledge in tables on websites: A prominent work in this realm is by Cafarella et al. [CHW+08] who first explored efficient techniques to gather structured data on the Web and examine the potential of large table corpora extracted from the Web. In a later paper they describe *Octopus*, a system that offers operators to combine the extraction of tables from the Web and the enrichment with context from respective source pages plus additional information form other sources [CHK09]. Recently, Crestan et al. examine and quantify the different types of tables that can be found on the Web [CP11]. Further, Dalvi et al. estimate the value of the long tail in the Web and find that it is worth investigating techniques for the extraction since there are no large aggregator sites that capture all entities of a specific

[37] `http://www.geonames.org/ontology/documentation.html`

domain under consideration [DMP12]. For instance, to achieve a coverage of 90% of all restaurants one has to access at least 1000 websites. These observations motivate the consideration of raw Web data for the creation of LOD. Note that this analysis is part of Yahoo's effort to build a web of concepts [DKP+09].

## 2.2 Creating Meta Information

Meta-data, i.e., data about data, is important for any type of information – no matter whether one deals with video files, geographic shape information or the description of arbitrary things in a graph data model. It is important to be aware of simple statistics, such as dataset size and value distributions, the origin of the data, transformation processes it underwent, and a sort of semantic description.

Meta-data is useful in a multitude of scenarios: the most obvious case is when data engineers search for information about a specific topic. How do they know what a dataset at hand is about and how can they quickly discover connections to other open sources that they already work with? A data source should provide this information in a standardized way. A second application is crawling the LOD cloud: Here, raw statistics, e.g., the number of triples, resources, links, etc., are of interest for scheduling tasks and provisioning resources. Also, semantic information, such as considered types or related resources, can facilitate useful segmentation of the data. Query answering for LOD is another scenario where dataset statistics can support decision making and help achieve better results more efficiently. A wide availability of well-defined meta-data expedites data discovery, semantic integration, and usage.

**Vocabularies for the Description of Meta-data.** The Vocabulary of Interlinked Datasets (voiD) addresses the need for meta-data for LOD. VoiD is an RDF-based schema to describe linked datasets [ACHZ09, ACHZ11]. By providing a standardized vocabulary, it aims at facilitating the discovery of linked datasets as well as their usage. VoiD offers two main classes: A `void:Dataset` describes collections of data published and maintained by a single provider. A `void:Linkset` describes entities linking to other sources. Along with datasets, a number of properties describe technical or statistical features. Similar to voiD there is DCAT, a vocabulary for the description of data catalogs targeting Web data catalog integration [MEA12]. The PROV Ontology facilitates the description of provenance information from a variety of systems from various domains [BCC+13].

Given these means for meta-data description the question remains where the data originates from: It can either be manually created or generated leveraging computational

methods. This thesis proposes a novel method for the generation of topics – a sophisticated type of meta-data.

**RDF Meta-data Creation.** The aforementioned vocabularies facilitate the description of relatively simple meta-information, such as statistics, as well as informal descriptions and data quality statements. Tools meant to create such type of meta-information, e.g., the IBM information analyzer, mainly originate from traditional database vendors. Since they are not suitable for graph-structured RDF data, we do not discuss them here. For RDF data we have developed ProLOD [BNA+10] for iterative RDF data profiling. ProLOD aims at determining data quality issues. On the other hand, Guéret et al. examine LOD links in particular [GGSL12]: The authors examine three network measures and two LOD-tailored approaches to spot poor-quality links and determine the dataset link quality as a whole. For this, they use manually evaluated links resulting from Silk linkage rules (see below).
Besides tools there are libraries, such as the NXParser[38], which are capable of creating simple statistics about the data. RDFStats[39] computes statistics and outputs them using the so-called RDFStats statistics vocabulary. Finally, many developers use handcrafted scripts to perform meta-data extraction. On Grimnes' web blog one can find interesting results of such an approach[40]. Others use high-end hardware to perform statistics computation for Web-scale datasets [JAaS+10]. In [BLN11] we report on a set of Map/Reduce algorithms to create voiD descriptions for Web-scale datasets. Further, there is, for instance, Virtuoso's database function RDF_VOID_STORE[41], which creates descriptions for RDF graphs. The authors of the voiD vocabulary maintain a list of tools at `semanticweb.org/wiki/VoiD`.

**Community Detection in Graphs.** In Chapter 3 we propose a novel, abstract and well-defined meta-information that describes the topic of a dataset [BKN12]. We define a topic as a set of types that are related in a specific area of the real world. For the creation of topics we introduce a novel community detection algorithm for annotated entity graphs. An annotated entity graph consists of vertices representing entities and edges capturing relationships among these entities – just like RDF. Additionally, entity vertices comprise annotations, e.g., the type of the entity or arbitrary attributes. In particular, our approach determines communities of vertices with respect to topics, i.e., it groups the entities according to the topic they deal with. Since entities can be part

[38] `http://code.google.com/p/nxparser/`
[39] `http://rdfstats.sourceforge.net`
[40] `http://gromgull.net/blog/?s=btc`
[41] `http://virtuoso.openlinksw.com/`

of multiple topics, an entity can be part of multiple communities. We thus present an approach to detect overlapping communities.

Community detection in the general cases has been under broad consideration for many years. Thus, there is a large body of work that emerged from several fields, such as computer science (e.g., social networks and electronic circuits) or biology. Along these lines, there are several excellent surveys and empirical evaluations [BvH06, For10, LLM10, POM09, Sch07].

Lately, attributed graphs have gained significant interest. An attributed graph is similar to what we call an annotated entity graph, i.e., vertices may have attributes. Given such an attributed graph, Silva et al. determine attribute-structure correlation patterns, i.e., dense subgraphs induced by particular attribute sets [SMZ12]. With these patterns, the authors target questions, such as how particular interests in social networks form communities. That is, the authors seek attribute sets that justify dense subgraphs via correlation. For this, they create so-called *attribute-structure correlation patterns* using significance tests with respect to null graph models. Zhou et al. propose SA-Cluster, a graph clustering approach that aims at deriving vertex clusters with homogeneous vertex attribute occurrences [ZCY09]. SA-Cluster incorporates vertex attributes through the addition of attribute nodes and respective edges. Then, the method uses random-walk vertex distances for a k-medoids-based clustering. Note that this requires the $k$ to be defined beforehand. Observe that variable attribute value importance influences the random-walk distance matrix. A recent paper by Cheng et al. tackles this scalability problem, i.e., many computations of the random-walk vertex distance matrix [CZHY12]. In particular, the authors incrementally update only these portions of the matrix that depend on variable edge probability values. Moser et al. discuss the mining of dense subgraphs such that respective vertices share a significant portion of additional boolean features, i.e., vertex attributes [MCRE09]. Instead, Mougel et al. discover sets of subgraphs, such that in each set subgraph vertices share a large number of attributes [MPR$^+$10].

However, all these approaches use vertex attributes as additional or complementary information when grouping vertices into dense communities and require certain constraints on the attribute sets to hold for the dense groups. Instead, the approach we present in Chapter 3 uses relations among these attributes to discover vertex communities and do not require density. In particular, we exclusively combine patterns of attributes showing significant correlation to form communities.

Further, most methods target distinct communities and have not (yet) shown to scale to large graphs with more than a million vertices per dataset like we face them in the LOD world. Silva at al. use by far the largest dataset for experimentation in [SMZ12],

namely, approximately 108,000 vertices from DBLP, 272,000 vertices from LastFM, or 294,000 vertices from CiteSeer. All other experiments have been conducted on a few thousand vertices representing authors from DBLP.

Further, there is a significantly smaller set of techniques that create overlapping communities [BGMi05, DPV05, DWP+07, Gre08, PSPLPMM10, PDFV05]. The latest survey in this field is by Xie et al. [XKS13]. These methods compare to our approach in that we also derive overlapping partitions of nodes.

Note that graph clusterings in general, including the aforementioned approaches, seek to maximize the ratio of intra- and inter-cluster links. For this objective, Newman and Girvan propose to maximize the modularity value, which is an NP-hard problem and thus often solved with heuristics [BDG+07, GMC10, NG04]. Note that modularity is not invariant to scale, i.e., it cannot produce communities smaller than a specific size [FB07]. Arenas et al. also argue about disadvantages of modularity and introduce an extended version of modularity for network motifs [AFFG08]. This way, the authors seek communities that comprise more motifs [MSOI+02] than a randomized version of the network at hand. We do not rely on cluster modularity, since the interconnection of instances through their classes seems to be a more meaningful criterion for topic discovery in annotated entity graphs. We thus *directly model the intuition of related vertices and respective annotations via patterns* by using selected patterns to expand communities.

Grimnes et al. apply traditional clustering to Semantic Web resources [GEP08]. Since they use graph distance, graph reachability, or ontological resource similarity, this approach also does not target a topical partitioning of the data.

Finally, note that as computational power grew during recent years, the community detection field evolved rapidly. That is, many of the approaches mentioned above emerged simultaneously to our approach.

**Relational Database Summarization.** From a conceptual point-of-view, closely related to graph meta-data creation is work on summarizing relational database schemas. The work most related to ours is by Wu et al. [WRSM08]. The authors describe a system for discovering topical structures in relational databases in order to support semantic browsing and large-scale data integration. Their work, however, categorizes relational tables according to topics – not entities. Wu et al. leverage different evidences, including attribute values, for this categorization. This is costly and would not scale to the size of current LOD sources. Further, we target uncovering topical structures inherent in the graph – not in the data values, i.e., literals.

Yu and Jagadish devise an approach for summarizing hierarchical schemata [YJ06]. For

this they use size, schema coverage, and importance as optimization criteria. Importance is estimated based on the cardinality of an element as well as its interlinkage. The authors present iterative algorithms for the creation of summaries, which consist of abstract elements and links.

In a similar vein, Yang et al. recently presented an approach to discover informative join paths through a set of query tables [YPS11]. Thus, in this work, summary graphs are created for a subset of large relational schemata.

**Latent Dirichlet Allocation.** Another line of related research is topic modeling, which aims to describe topics of text corpora. Here, Latent Dirichlet Allocation (LDA) [BNJ03] has attracted considerable interest. LDA is a generative, probabilistic Baysian model, which derives topics from text corpora. In LDA a topic is a probability distribution over words, i.e., given a probability threshold a topic is a set of words. Comparably to LDA, in Chapter 3 we produce sets of annotations; but we work on annotated entity graphs - not on texts. Note that authors recently tackle scalability issues for LDA: Zhai et al. propose a parallel LDA implementation based on Map/Reduce [ZBGAA12]. Their approach does not use Gibbs Sampling; it rather employs variational inference since it fits better into the Map/Reduce framework. The authors further propose two extensions: a so-called informed prior to bias topic discovery as well as a technique for the discovery of topics in multilingual corpora.

**Miscellaneous Types of Topics.** Finally, there is a number of recent works that deal with different notions of topics: For instance, authors define a topic as a "spatially coherent meaningful theme", i.e., words that are often close in space [YCH+11]. This approach works on geo-tagged data and thus creates topics from geographic regions instead of documents or entities in a graph. The authors of [JHL11] define a topic as "semantically coherent content that is shared by a significant number of documents in the corpus". They discover topics by observing significant changes in a document corpus. Others cluster Web search queries into missions based on topics [ADOM11]. Here, topics are defined as "the sum of what can be perceived, discovered, or learned about any real or abstract entity". A classifier determines the probability that two sets of queries are topically related, which is the input for an agglomerative clustering.

The approaches just mentioned demonstrate the diverse perception of *what is a topic* – no matter whether one deals with graphs, relations, or texts. In Chapter 3 we provide our definition as well as an abstract representation of topics in attributed graph-structured data.

## 2.3 Creating Links

The third building block of the Web of Data, besides data and respective meta-information, are the links that interconnect entities. The entity matching research field evolved over many years. Thus, to gain an overview is a challenging task. Hence, a number of surveys was created to be able to grasp the vast majority of approaches – often dedicated to different domains and use-cases. Table 2.1 at Page 29 lists more than a dozen summarizing works from the last decade and thus illustrates the vivid development and change in the field. Remember that, interestingly, computational capabilities evolved tremendously over the same time and thus the nature of developed approaches changed, i.e., techniques range from early pairwise string similarity-based methods to complex joint reasoning over many sources. Reasoning in a joint manner is to take a global view when making local decisions, e.g., the identity of two entities can depend on other entities connected trough arbitrary relationships. Joint reasoning includes interdependent decisions, i.e., (earlier) decisions influence other (later) decisions. In the literature, the terms "joint reasoning", "interdependent -", or "collective decisions"are used interchangeably.
In the scope of this work, we go into detail for two groups of techniques: Since we target the large-scale entity matching scenario for the Web of Data, we first discuss works that promise processing large datasets and thus facilitate large-scale LOD entity matching. Second, we elaborate on selected approaches specifically dedicated to the Linked Open Data world – because we deal with LOD.
Since it is a challenge to differentiate approaches from the entity matching field, we mark differences to the technique we propose in Chapter 4 using a $\Diamond$. We use the $\Diamond$ mark for the sake of clarity, since, given the very many approaches, discussions have shown that it is often not apparent for the reader.

**Large-scale Entity Matching.** Table 2.2 at Page 30 lists all approaches considered in the subsequent discussion. It further depicts datasets and respective sizes used for experiments the authors report on in their work. In Chapter 4 we discuss results obtained using a dataset comprising roughly 120 million unique entity descriptions.
The work that presents experiments on the largest scale (among those in Table 2.2) has been published recently by Papadakis et al. [PIN+12]. The authors use blocking based on entity URI infixes, URI infixes of neighboring entities, as well as literal tokens. Blocking means to group data according to a blocking key and then conduct pairwise *intra*-block comparisons. This way, one saves the many *inter*-block comparisons. In [PIN+12] the authors investigate the correlation of *Blocking Cardinality* and *Comparison Cardinality* with respect to effectiveness and efficiency, respectively. This work does not consider

precision and recall. ◊ As opposed to our work, the proposed approach does not perform joint decisions where discovered similarities are taken into account for further processing. However, it incorporates shallow mutual dependencies since it leverages neighboring entity URIs.

Earlier work by Papadakis et al. determines attribute-name independent blocks for linking sources with loose schemata and noisy data [PINF11]. For this, they simply ignore attribute names but rely on shared attribute value tokens to build blocks. To deal with the very large number of blocks they elaborate on a block scheduling strategy that processes blocks according to the ratio of gain and cost per block. ◊ This work deals with a pair of sources and does not perform joint processing.

Kolb et al. devise load balancing strategies for running blocking-based matching algorithms with Map/Reduce [KTR12]. Here, mappers compute blocking keys, and reducers perform the actual matching per block. To distribute the load, the authors compute a block distribution matrix used to assign blocks or block fragments to compute nodes according to different schemes, i.e., a schema solely depending on the input or a range-partitioning scheme. If blocks span across multiple compute nodes they run an additional matching step. ◊ Given large compute clusters, Kolb et al. can essentially run blocking-based approaches on arbitrary dataset sizes. However, it remains unclear whether presented observations hold for large block sizes beyond a critical limit where significant block distribution overhead (including additional comparisons) occurs. Also, Kolb's work does not consider joint inference. As for blocking with Map/Reduce, Rong et al. suggest a signature based on an inverted index as blocking key for deduplicating millions of author names and synthetic strings [RLDZ11]. Niu at al. have taken a similar approach with *Zhishi.links* for large LOD datasets [NRZW11]. Here, the reducer consists of two steps: It first computes a name and a geographic similarity and then, if the first value exceeds a given threshold, it determines a similarity based on property-value pairs giving higher weight to functional properties.

Hogan et al.'s work deals with roughly the same dataset size as we do and is particularly inspired by the requirements of a Semantic Web search engine [HHU+12, HZU+12]. In their work, the authors aim to analyze large semantic Web data corpora. For this, they employ a distributed architecture coordinated by a dedicated master and process data in multiple phases – run, gather, flood, run, etc. That is, they run local computations, gather partial results on the master, and push aggregated result back to the slaves. This way, they compute results based on `sameAs` symmetry and transitivity, explicit rules leveraging inverse functional properties as well as max-cardinality restrictions and shared in/out links plus attribute values. Last, they apply OWL rules for a final refinement. ◊ The proposed approach performs in a distributed manner but does not conduct

distributed joint inference. Instead, it employs distributed sorts and scans of the corpus to achieve fairly low runtimes for simple methods on a large corpus.

In earlier work, Hogan et al. gained experience with matching based on quasi inverse functional properties [HPUZ10]. Here, they compute statistics in few passes over the data – basically performing sorts and counts. From these statistics they induce aggregated probabilities for entities being `sameAs` other entities. ◊ Again, this is not a joint matching approach.

SiGMa is an approach closely related to our Multi-Core algorithm (discussed in Section 4.2). Its solution is a binary matrix that is approximated iteratively with a greedy algorithm [LJPD+12]. It performs joint decisions but uses only local neighborhoods. Also, the authors argue that unique mappings per source induce a higher result quality[42]. ◊ The proposed approach matches two sources and achieves fairly low runtime. However, SiGMa assumes aligned relationships and properties. Also, SiGMa propagates matching information through the graph but does neither update priority scores nor reorganize internal data structures accordingly.

Whang and Garcia-Molina do not process *very* many entities (with respect to other publications listed) [WGM12]. However, they present a sound framework for the integration of existing entity resolution algorithms. These algorithms are then scheduled such that they produce a joint result. That is, a logical execution plan determines the order of execution (including re-executions) of individual algorithms to ensure interdependent considerations, e.g., papers should be reconsidered after venues have been merged. Given the logical plan, the framework creates a physical execution plan incorporating resource constraints. ◊ Note that the author's framework needs existing matching techniques tailored to specific entity types and attributes. The framework thus operates on well-structured input for which an engineer needs to specify an influence graph. This graph captures the semantic relationships among entity types.

Rastogi et al. argue that all state-of-the-art collective entity matchers are probabilistic and suffer from a fundamental problem, namely scalability [RDG11]. They propose to partition the input into neighborhoods (they use canopy clustering), which can be considered locally on a single machine. For collective matching decisions they propose a simple as well as a maximal message passing scheme and show that their framework produces provably sound and accurate results (empirically shown) for well-behaved entity matchers. ◊ Throughout the paper they use a Markov-Logic-based entity matcher, which requires manually created domain-specific rules. Rastogi et al. report on high precision

[42] In fact, Lacoste-Julien et al. claim that most entity matching solutions rely on unique mappings per source [LJPD+12]. We support this claim in that we argue that such constraint increases the result quality. However, this discussion is highly controversial, i.e., reviewers often suggest the opposite.

results and low runtimes on highly structured, and clearly typed dataset. Large-scale experiments are run on Hadoop at Yahoo. It remains unclear how to identify suitable partitions for highly heterogeneous LOD entities.

PARIS aligns instances, classes, and relations using (inverse) functional properties [SAS11]. Quasi functionality is computed taking observed property occurrences into account – just like proposed by Hogan et al. [HPUZ10] (see above). PARIS iteratively adjusts probabilities for matches. Hence, match probabilities recursively depend on previous matching decisions. The authors run PARIS until reaching a fixpoint. ◊ The input for PARIS are two well-structured ontologies. Note that the authors also assume unique mappings per source. This way, they achieve high precision on OAEI data[43].

Herschel et al. examine the very general case of detecting intra-source entity matches in graph-structured data [HNST11]. They assume an iterative process examining candidate pairs maintained in a priority queue. The authors organize their data, including dependencies among entities and the candidate queue in relational tables to exploit fast sorting, etc. Note that relational storage is beneficial only if the queue is updated in many places. Sorting a table for only few changes does not make sense. To overcome this shortcoming the authors incorporate buffering.

◊ The general framework (graph data, iterations and a priority queue) is very similar to ours. The queue can be organized according to an arbitrary ranking. In particular, the authors elaborate on a ranking that minimizes entity pair reconsiderations. Instead, we use the priority queue to refine intermediate solutions towards a global optimum following a set of constraints. Here, the relational database used by the authors would constantly have to handle very many updates.

Herschel et al. work with a reference graph encoding entities and dependencies. A domain expert has to define this graph beforehand and specific relational tables have to be created. This works for clearly structured data. Instead, we operate on the Web data graph almost as is. Nevertheless, we can apply arbitrary similarities involving the graph neighborhood recursively without particular preprocessing.

Herschel et al. further discuss a distributed approach and argue that the Map/Reduce-framework is not suitable due to interdependent classifications. We show that smart partitioning of the input data can facilitate distributed collective decisions.

Herschel et al. chose a master-slave setup and maintain a registry that contains information about all candidate pairs to be updated and classification results. This way, they guarantee the equivalence of the distributed and the non-distributed approaches. However, holding the entire reference graph on a single machine leads to a severe trade-off

[43] The (O)ntology (A)lignment (E)valuation (I)nitiative is a yearly competition in matching ontologies organized in various tracks [EFH+09, EFM+10, EFH+11]. Each track focuses on a different challenge.

between efficient access and available compute resources. In our implementation, we do not have a central coordination at the price of violating the unique mapping constraint. Work by Hu et al. deals with pairwise non-joint matching of classes and properties (no instances) [HCZQ11]. The authors use an existing tool (Falcon-AO [HQ08]) to process 4,433 ontologies gathered by the Falcon search engine. The matching has been conducted on six PCs and took nearly one year for the creation of six million mappings. After filtering the result to ensure high quality 3.1 million mappings remained. The authors then study mapping-, ontology-, and top-level domain graph properties, such as power-law-distribution and the small-world phenomenon.

A second paper by Hu et al. reports on a machine-learning-based technique [HCQ11]. The approach creates training data from existing `sameAs` statements and functional properties as well as cardinality constraints to select discriminative property values. Then, property combinations for entity matchings are selected, e.g., latitude and longitude have shown to be effective. ◊ Unfortunately, the authors restrict the evaluation of their non-joint approach to ten predefined URIs.

Finally, a prominent work that dates three years back is by Arasu et al. [ARS09]. They propose a declarative framework with precise semantics for collective intra-source entity matching with constraints. That is, the authors contribute a Datalog-like language to manually specify the matching process and respective algorithms with theoretical underpinning plus physical optimizations. In this work, "collective" stands for the matching of different types of entities simultaneously, e.g., publications, authors, and venues. "Constraints" allow to specify dependencies among these types, e.g., a publication has exactly one venue.

**Entity Matching for LOD.** Some of the above discussed approaches have already been tailored specifically to LOD. Thus, a distinction between the two sets of techniques (large-scale and made-for-LOD) is not clear-cut. However, in the following, we briefly discuss prominent approaches that emerged from the Semantic Web field. Most of these techniques competed in the Ontology Alignment Evaluation Initiative (OAEI) [EFH+09, EFM+10, EFH+11]. ◊ Note that none of the approaches below conducts joint inference and they are mostly designed to operate on two well-structured sources.

The *Silk Link Discovery Framework* is highly prominent [VBGK09]. In 2009, Volz et al. presented the framework including a declarative language for specifying linkage rules. With these rules, one can select entity predicate values, and transform as well as compare them to other values using well-known similarity functions. A year later the authors added an implementation based on Map/Reduce, which is basically undocumented at the time of writing. A look into the public sources reveals that the mapper conducts

the matching and the reducer filters resulting entity pairs. Next, Isele et al. bundle this approach into a software that consumes RDF data streams and matches incoming entities against a set of locally known entities hold in an in-memory cache [IJB10]. In 2011, the authors presented a feature called *MultiBlock*, which essentially indexes entities by all its predicates and hereby allows to form blocks without sacrificing recall [IJB11]. ◊ This approach implements a variant of the well-known blocking technique. Recently, the authors formalized expressive linkage rules and present a supervised rule learning algorithm based on genetic programming [IB12]. This way, they overcome one of the major shortcomings of their framework, i.e., Silk linkage rules were manually created, which remained a tedious task. The authors show the viability of their approach in extensive evaluations on a multitude of datasets. ◊ In general, the framework is not meant to conduct interdependent decisions that propagate through the graph constructed from multiple sources. Also, it is not a suitable approach for noisy and heterogeneous data at Web-scale where many distinct properties exist and textual labels are sparse but interconnections across many sources are available. Also, the latest addition requires training data for the rule learning.

Ngonga Ngomo and Lyko also tackle the problem of learning linkage rules with a system called *EAGLE* [NL12]. Similar to Isele and Bizer [IB12], they employ genetic programming as supervised learning strategy but incorporate active learning to overcome the need for a large training dataset.

Earlier, Ngonga Ngomo and Auer presented an approach coined *LIMES* to limit the number of entity comparisons based on the triangle equation [NA11]. They conclude that given the distances of the dataset's entities to reference points, one can estimate whether the pairwise distance is above a fixed lower bound. Then, pairwise distances have to be computed only for pairs where this estimate is below the fixed value. ◊ This work is complementary to ours, as it describes how the prior computation for our algorithm could be optimized if we used a metric.

Similar to the general Silk approach, Hassanzadeh et al. present a framework to declaratively specify linkage rules – but for relational tables [HKL+09, HXM+09]. *LinQL* is the respective language that facilitates standard data manipulation in combination with link discovery. A main contribution of this work is to provide a translation of LinQL to SQL, which allows to benefit from relational operators. For semantic knowledge in particular, they show how synonym and hyponym relationships can be leveraged when stored in relational tables.

*AgreementMaker* is a publicly available framework to express an alignment process combining existing techniques for concept matching [CAS09]. It allows to specify first- and second-layer matchers that compare concept features, such as labels, comments, and

annotations, as well as structural properties, respectively. Then, third-layer matchers combine output from the previous layers. Matchers can be run in parallel or sequentially. *COMA++* is very prominent among frameworks for combining approaches [ADMR05, DR02]. This system is not exclusively dedicated to ontology integration. COMA++ consists of a repository to store data relevant to the matching process. Additionally, it comprises an execution engine, which performs the actual matching plus the similarity combination.

*RiMOM* tackles the selection of a good combination of strategies for matching ontologies [LTLL09]. Li et al. calculate a label- and a structure-based ontology similarity factor. Given these indicators, RiMOM determines a weighting scheme for linguistic matchers and how to propagate similarity values in the alignment process.

*SERIMI* also performs in multiple phases [AHSdV11]. It first selects predicates with literal values that have an entropy higher than average and then creates match candidates by retrieving entity pairs with similar textual labels for these predicates. Second, SERIMI disambiguates sets of target entities using a Resource Description Similarity, which captures the intra-candidate-set similarity.

*Codi*, publicly available as well, stands for Combinatorial Optimization for Data Integration and is a framework to declaratively model soft and hard constraints [NMS10, NNMS10]. Soft constraints represent similarities among ontology elements, i.e., potential correspondences, and hard constraints capture logical axioms. The authors leverage Markov-logic to determine the maximum probability of all corresponding possible worlds. Note that this work targets a global optimum where decisions influence each other.

Finally, a visionary work by Cudré-Mauroux et al. describes a framework with two goals, namely the retrieval of entity equivalences as well as postdating entities, i.e., representations that "model the same referent, but taken at different times" [CMHJ+09]. The authors' system *idMesh* deals with multiple sources in a decentralized manner and manages uncertainty. Given entities from multiple Web sources, the authors build factor-graphs to infer equivalences and time relationships.

| Title and Authors | Reference |
|---|---|
| Data Matching, Christen | [Chr12b] |
| A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication, Christen | [Chr12a] |
| Towards Large-Scale Schema and Ontology Matching, Rahm | [Rah11] |
| Results of the Ontology Alignment Evaluation Initiative, Euzenat et al. | [EFH$^+$09, EFM$^+$10, EFH$^+$11] |
| Holistic Concept Matching in the Web of Data (Chapter 3), Grütze | [Grü11] |
| Evaluation of entity resolution approaches on real-world match problems, Köpcke et al. | [KTR10] |
| Frameworks for entity matching: A comparison, Köpcke and Rahm | [KR10a] |
| An Introduction to Duplicate Detection, Naumann and Herschel | [NH10] |
| Handbook on Ontologies, Staab and Studer | [Sta09] |
| Ontology Matching, Euzenat and Shvaiko | [ES07] |
| Duplicate Record Detection, Elmagarid et al. | [EIV07] |
| An Empirical Study of Instance-Based Ontology Matching, Isaac et al. | [IvdMSW07] |
| A Survey on Ontology Mapping, Choi et al. | [CSH06] |
| Ontology Mapping: The State of the Art, Kalfoglou and Schorlemmer | [KS05] |
| A Survey of Schema-Based Matching Approaches, Shvaiko and Euzenat | [SE05] |
| A Survey of Approaches to Automatic Schema Matching, Rahm and Bernstein | [RB01] |

Table 2.1: Surveys related to the Entity- and Concept Matching research field (ordered by year of publication).

| Title and Authors<br>Dataset and respective sizes | Reference |
|---|---|
| Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data, Papadakis et al.<br>Billion Triples Challenge 2009 data: 182m entities | [PIN+12] |
| Load Balancing for MapReduce-based Entity Resolution, Kolb et al.<br>Product descriptions: 110k Publications: 1.4m | [KTR12] |
| Scalable and distributed methods for EM, Consol. and Disambig. over linked data corpora, Hogan et al.<br>Crawled data: 947m unique triples from 783 pay-level-domains | [HZU+12] |
| SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases, Lacoste-Julien et al.<br>YAGO: 1.4m entities IMDb: 3.1m, 4.8m entities Freebase: 474k entities | [LJPD+12] |
| Joint Entity Resolution, Whang and Garcia-Molina<br>Synthetic records: 1k Spock data: 1m persons, 800k addresses, 350k schools, and 660k jobs | [WGM12] |
| LINDA: Distributed Web-of-Data-Scale Entity Matching, Böhm et al. (part of this thesis, see Chapter 4)<br>Billion Triples Challenge 2011 data + DBpedia + Yago + Freebase + Geonames: >115m entities | [BMNW12] |
| Large-scale Collective Entity Matching, Rastogi et al.<br>Bibliographic data: 2.3m publications: 4.6m authors | [RDG11] |
| PARIS: probabilistic alignment of relations, instances, and schema, Suchanek et al.<br>Yago: 2.8m instances, 292k classes DBpedia: 2.4m instances, 318 classes IMDB: 4.8m instances, 15 classes | [SAS11] |
| Scalable Iterative Graph Duplicate Detection, Herschel et al.<br>IMDB data: 35k movie names, 800k actors Freedb.org: 10k CDs | [HNST11] |
| Efficient entity resolution for large heterogeneous information spaces, Papadakis et al.<br>DBPedia/IMDB: 23.5k and 23.2k entities DBPedia/DBPedia: $2 \times 1.2$m entities | [PINF11] |
| How Matchable Are Four Thousand Ontologies on the Semantic Web, Hu et al.<br>4,433 ontologies gathered by the Falcon Search Engine: 2m classes and properties | [HCZQ11] |
| A Self-Training Approach for Resolving Object Coreference on the Semantic Web, Hu et al.<br>569.5m triples, i.e., 76.4m URIs, including 8m triples used for training data creation | [HCQ11] |
| Some Entities are more equal than others, Hogan et al.<br>Crawled data: 20m triples | [HPUZ10] |
| Large-Scale Deduplication with Constraints using Dedupalog, Arasu et al.<br>Bibliographic data: 460k entries | [ARS09] |

Table 2.2: State-of-the-art large-scale entity matching approaches (ordered by year of publication) with respective datasets used for experimentation.

# 3 Topic Mining

A main goal of the LOD vision is to allow easy understanding and re-use of formally raw (and sometimes hidden) data in order to facilitate novel applications incorporating data from distributed sources. Given the wealth of data, it is a challenging undertaking for data engineers to choose appropriate sources for a task at hand. Jain et al. state that there is currently no mechanism to capture the conceptual description – *the topics* – of a dataset [JHY+10]. Currently, 63% of the 295 LOD sources (and 73% of the 41 cross-domain sources) do not provide any meta-information at all[44]. This is a major shortcoming of the current LOD implementation.

Consider, for instance, an online warehouse with internal product data to be augmented with open data, e.g., from some open media database or a source describing geographic places. To accomplish this augmentation, one could conduct a Web search and might find the Linked Movie Database, Geonames, or DBpedia. The Linked Movie Database and Geonames do not provide any semantic meta-information. DBpedia offers a concise voiD description but the semantic content, i.e., the *topics* it covers, remains unclear. For a human expert, the covered topics might be easy to guess in cases, such as IMDB or GeoNames; but which topics does DBpedia cover? DBpedia describes diverse entities from the media, politics, arts, science, sports, and many more domains. Other prominent examples for cross-domain data sources are Yago [HSBW12], Freebase [BEP+08], Zhishi.me [NSW+11], or OpenCyc[45]. Recently, Best Buy opened its product database and now offers product specifics, descriptions, and consumer reviews[46].

The *topic* of a data(sub)set can be understood as the dataset's subject, i.e., it states what the interplay of all its entities is about. In contrast, a *concept* or *type* of an entity is an abstract idea derived from specific entities, i.e., it generalizes the notion of a specific group of entities. We use *types* as annotations of nodes in a graph.

Politics, arts, science, and sports are examples for topics we aim to discover. If given at all, they are not exposed in a structured manner. We propose novel structured meta-information, namely *sets of types* that constitute topics. Then, a topic can be expressed by all *distinct types* (not the entities itself) of all entities forming a topic. Note that these

[44] `http://lod-cloud.net/state/#data-set-level-metadata` as of September 2011.
[45] `http://sw.opencyc.org`
[46] `https://bbyopen.com/announcing-bbyopen-metis-alpha-best-buy-product-catalog-semantic-endpoints`

```
@prefix void: <http://rdfs.org/ns/void#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix : <http://example.com/ontology#> .

:ds a void:Dataset ;
  dcterms:title "The Prominent People DB" ;
  dcterms:description "This is data about musicians and more." ;
  void:uriLookupEndpoint <example.com/> ;
  void:uriRegexPattern "^example.com/ontology#.+" ;
  void:exampleResource :entity/Bob_Marley ;
  void:exampleResource :entity/The_Rolling_Stones ;
  void:numberOfSubjects 4000 ;
  void:numberOfPredicates 20 ;
  void:numberOfObjects 3000 ;
  void:numberOfTriples 40000 ;
  dcterms:subject :topic01 ;
  dcterms:subject :topic02 .
:topic01 a :topic ;
  :contains :type/album ;
  :contains :type/band ;
  :contains :type/manager ;
  :contains :type/event ;
  :contains :type/club .
:topic02 a :topic ;
  :contains :type/place ;
  :contains :type/stadium ;
  :contains :type/player ;
  :contains :type/footballclub ;
  :contains :type/brand .
```

Listing 3.1: Example for topic descriptions as part of a `void:Dataset` description.

topics may overlap, e.g., the concept of a geographic place can play a role in different topics, such as sports and transport. We are aware that it is a matter of opinion which entities exactly form a topic and which do not. However, we believe that there is a rough common understanding and further provide a set of parameters for our approach so that the output can range from high-level to fine-grained and thus meets different requirements (see Section 3.5, Page 53).

For a seamless integration into existing meta-data standards, one could incorporate topics into voiD descriptions or expose them in the form of `http://purl.org/dc/terms/subject` property values. Then, users could easily grasp what a dataset is about. Consider Listing 3.1 as an example. Here, we can find the regular voiD information, such as the datasource title, the SPARQL endpoint, and example resources. In addition, the description contains the topics of the data – see `:topic01` and `:topic02` defining sets of types. From these two collections of types inherent in the data one can conclude that the data at hand is about music industry and sports related entities. This conclusion

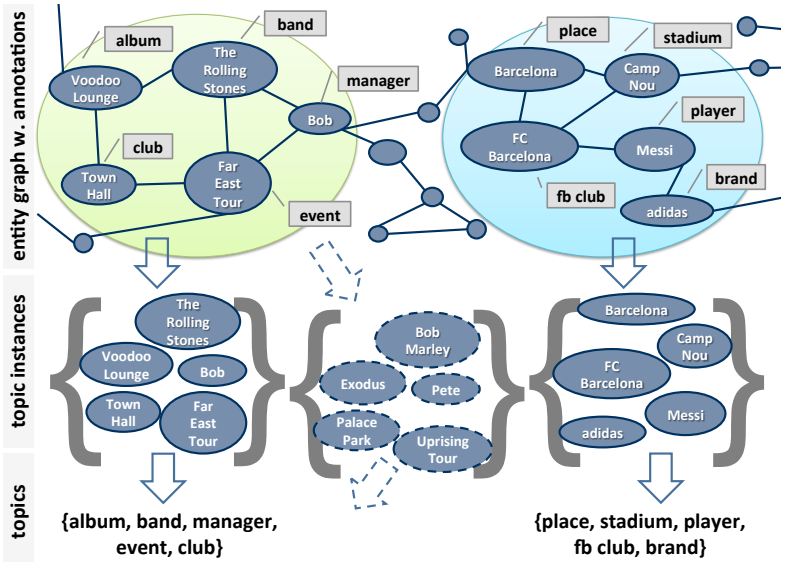


Figure 3.1: Running example: filled ellipses (nodes) indicate entities; edges represent relationships among entities; rectangles are annotations. The large transparent ellipses illustrate different topics.

could neither be drawn from the title nor from the example resource information.

Further, we envision several technical use cases: For instance, topics can facilitate source selection for querying the LOD cloud [HHK+10, TZS10]. Here, one could compute the topics of a query and then determine sources in the cloud whose topic significantly overlaps with those from the query. Topics can also provide further evidence for link discovery across data sources. That is, a link among entities from different sources makes sense only if both entities stem from overlapping topics. Last, RDF indexing approaches can be supported: topic instances represent node sets of a graph that should be co-located on disk and indexed respectively [WPWCM11].

Consider Figure 3.1 as a general example that we use throughout this chapter. The top section shows our input: We work with a graph consisting of nodes representing real-world entities (blue ellipses) and edges representing relationships among these entities (blue lines). Additionally, our input contains annotations for entity nodes, i.e., each entity's abstract type (grey rectangles). This graph in the top section, for instance, contains the band `The Rolling Stones` and their album `Voodoo Lounge` or the football club `FC Barcelona` and the stadium `Camp Nou` as entity nodes. Moreover, it comprises instances of `managers` and `brands`, etc. Given this graph, one can intuitively distinguish different topics, e.g., *music industry* for the left-hand nodes and *sports* for the right-hand nodes[47]. Thus, the example in the top section shows two topic instances that can be

[47]The actual labeling of topics is not part of this work.

identified in the graph (light blue and light green background areas).
The middle section depicts these topic instances, i.e., sets of entities. Note that topic instances instantiate topics, i.e., they are of a specific topic. Thus, there can be multiple *topic instances* that instantiate a single *topic*. In the example, there are two *music industry* topic instances; one contains entities from the rock music business; a second instance comprises reggae music related entities. Also, the middle section contains one topic instance for *sports*.
The bottom section shows the unique abstract representations of the topic instances: We use sets of types to describe a topic. In our example, there are two topics: $music\,industry = \{album, band, manager, event, club\}$ and $sports = \{place, stadium, player, fbclub, brand\}$. If published as part of voiD descriptions this could be represented as in Listing 3.1. Such dataset descriptions clearly help a user understand what is in the dataset under consideration.
In the following we formalize the problem of mining topics from semantically heterogeneous graph-structured data and propose an approach called *(A)nnotated (P)attern (P)ercolation* (APP) for mining overlapping topics and topic instances. Our approach is based solely on the data's graph structure and does not require any literal values. The result of our approach consists of (1) types forming topics, (2) patterns of type relations per topic, and (3) sets of entities from the data instantiating the topics. Note that since we work with a graph capturing entities, our approach is applicable to all sorts of data that can be converted to such a graph. This graph structure is inherent in RDF data but could also be derived from tabular information. In Section 3.5 we evaluate our approach on DBpedia and show that we can reconstruct a portion of the topics that we extracted from Wikipedia.

## 3.1 Annotated Pattern Percolation

**Topics and Instances.** Given a large RDF dataset with heterogeneous content, we want to derive topics and partition the data into respective topic instances. An RDF triple $(s, p, o)$ consists of a subject, a predicate, and an object. The subject is the entity the RDF statement is about. In LOD, subjects and predicates are given as URIs. Objects are either URIs of other entities or literals. In the following, we define topics as well as topic instances.

**Definition 3.1** (topic)
Given a set $R$ of RDF triples including $(s, \texttt{rdf:type}, o)$ statements, a topic $T$ is a set of types $\{t_1, \ldots, t_n\} \subseteq \{o | (s, \texttt{rdf:type}, o) \in R\}$ that are related in a specific area of the real world. □

Note that we deliberately chose the fuzzy term "related" in order to reflect the many different kinds of coherence among types in the real world.

**Definition 3.2** (topic instance)
Given a set $R$ of RDF triples and a topic $T = \{t_1, \ldots, t_n\}$, a topic instance $I = \{e_1, \ldots, e_m\}$ is a set of entities such that the types of these entities constitute the topic $T$, i.e., $T = \bigcup_{e \in I} type(e)$. The function $type(e)$ yields the types of an entity. □

Entities in topic instances shall reflect the type relationships of topics, i.e., entities with strongly related types should be related similarly. A *topic* and an *instance* share the same relationship as a *type* and an *entity*. An entity's type describes its abstract idea – though there are many individual representations. An instance's topic describes what the instance's entities are all about – though there can be different instances (i.e., sets of entities) about the same topic. Observe that the topic- and instance cardinality can differ, since two entities (of an instance) can have equal types or single entities can have multiple types. Further note that Definition 3.1 and 3.2 (as well as Definition 3.3) are customized for RDF data. However, they could easily be restated in order to fit life-science graphs or social network data, etc.

**Data Model.** For our topic mining approach, we cast a set of triples into an *annotated entity graph* $G$, with nodes representing entities, and edges representing relationships with other entities. Specifically, we consider only RDF triples $(s, p, o)$ where $s$, $p$, and $o$ are URIs. Literals and reifications are not taken into account. We thus deal only with structure inherent to the data. Types of entities, i.e., concepts associated via `rdf:type` predicates, constitute entity annotations and are compiled into a set of values associated with an entity. Another set contains all types used in $G$. For the definition of these sets, we use the + sign to denote the recursive application of the `rdfs:subClassOf` predicate. Thus, given a set of RDF triples $R$, $(t1, \texttt{rdfs:subClassOf}^+, t2)$ defines $t2$ to be any super-type of $t1$ present in $R$ – not necessarily a direct one.

**Definition 3.3** (annotated entity graph)
An annotated entity graph is a vertex-colored graph $G = (V, E)$ with a set of nodes $V$ representing entities and edges $E$ capturing relations among entities. Entity node annotations $type()$ are the vertex colors. Given a set of RDF triples $R$, the set of all types $C$ as well as $V$, $E$ and the function $type()$ are defined as follows:

$$
\begin{aligned}
C &:= \{o \mid \exists s : (s, \texttt{rdf:type}, o) \in R\} \cup \\
&\quad \{t \mid \exists s : (s, \texttt{rdf:type}, o), (o, \texttt{rdfs:subClassOf}^+, t) \in R\} \\
V &:= \{s, o \mid (s, p, o) \in R\} \setminus C \\
E &:= \{(s, o) \mid (s, p, o) \in R \wedge s, o \in V\} \\
\mathit{type}(s) &:= \{o \mid (s, \texttt{rdf:type}, o) \in R\} \cup \\
&\quad \{t \mid (s, \texttt{rdf:type}, o), (o, \texttt{rdfs:subClassOf}^+, t) \in R\}
\end{aligned}
$$

□

Intuitively, an annotated entity graph comprises RDF nodes that occur as subject or object in RDF triples. It does not contain nodes that represent entity type information. An edge among two nodes exists if there is a triple containing both nodes. Entity node annotations *type*() are derived from entity types. $C$ comprises all possible annotations – that is, it contains all types from the RDF graph.
Given an annotated entity graph derived from an RDF dataset, the problem we tackle is to derive topics as defined in Definition 3.1. Further, we want to partition the entity graph in a way such that each partition is a topic instance as in Definition 3.2.

**Patterns.** The *annotated entity graph* is the structure we work with throughout our approach. To mine topics as well as topic instances, we compute a number of different subgraphs. More specifically, we find patterns that match in an annotated entity graph. These patterns shall inherently model the notion of *relations among types in the real word* from Definition 3.1 and may thus vary depending on the use case. In Section 3.2, we elaborate on the patterns for topic mining. In the following, we define the general notion of a pattern and a pattern match.

**Definition 3.4** (annotated pattern)
Given a set $N$ of node variables and the set $C = \bigcup_{v \in V} \mathit{type}(v)$ of types from an annotated entity graph (from Definition 3.3), an annotated pattern of size $s$ is a triple $p = (N_p, E_p, \mathit{ann}())$ where $(N_p, E_p)$ is a connected graph and $\mathit{ann} : N_p \to C$ is a function. The set $N_p \subseteq N$ constitutes the nodes, $E_p \subseteq N_p \times N_p$ represents edges, and $|N_p| = s$. The function $\mathit{ann}()$ returns the annotation of the node $q \in N_p$. □

Intuitively, an *annotated pattern* of size $s$ is a connected structure formed by $s$ arbitrary nodes but with specific type annotations. Note that in such patterns the nodes are anonymous, i.e., they stem from node variables. The structure itself is important, since it represents specific relationships. For instance, the pattern $p = (\{(a, b, c)\}, \{(a, b), (b, c), (c, a)\},$

$ann()$) with $ann(a) = t_1$, $ann(b) = t_2$, $ann(c) = t_3$ comprises the node variables $a$, $b$, and $c$ as well as edges $(a, b)$, $(b, c)$, $(c, a)$. Importantly, $p$ forms a triangle of nodes annotated with $t_1$, $t_2$, and $t_3$. In contrast, the edges $\{(a, b), (b, c)\}$ would represent a sequence of annotated nodes – another pattern. In the following, for ease of reading, we denote annotated patterns without node variables and the function $ann()$. Thus, we denote the aforementioned pattern as $p = (\{t_1, t_2, t_3\}, \{(t_1, t_2), (t_2, t_3), (t_3, t_1)\})$. Further, we use the terms *pattern* and *annotated pattern* synonymously. With the definition of an *annotated pattern*, we can now define how we locate such patterns in an entity graph by means of an *annotated pattern match*.

**Definition 3.5** (annotated pattern match)
A match of an annotated pattern $p = (N_p, E_p, ann())$ in an entity graph $G = (V, E)$ is a subgraph $G' = (V', E')$ of $G$ such that

- $|N_p| = |V'|$,
- $\bigcup_{v \in N_p} ann(v) \subseteq \bigcup_{v \in V'} type(v)$,
- $\forall (v, w) \in E_p, \exists (v', w') \in E'$ with $ann(v) \in type(v') \wedge ann(w) \in type(w')$.

□

Intuitively, an annotated pattern matches in an entity graph if the graph contains an isomorphic subgraph for the pattern. That is, the subgraph comprises nodes annotated with respective types and these annotated nodes must (at least) be connected in the same way as the node variables in the pattern. Consider Figure 3.2 as an example of two annotated pattern matches:

- $p_1$ =({*club*, *album*, *band*}, {(*club*, *album*), (*album*, *band*)}) (given in red)
  matches in the graph from Figure 3.1 at $G'_1$ = ({1,2,3},{(1,2), (2,3)}).
- $p_2$=({*band*, *manager*, *event*},{(*band*, *manager*), (*manager*, *event*), (*event*, *band*)})
  (given in light blue)
  matches in the graph from Figure 3.1 at $G'_2$ = ({3,4,5},{(3,4), (4,5), (5,3)}).

## 3.2 Annotated Patterns

In the following we elaborate on determining annotated patterns that represent relationships among types in the real world (as required in Definition 3.1). Specifically, we present two alternative approaches that create patterns with different properties. For one, we extend the notion of network motifs [MSOI+02] with annotations (Section 3.2.1).

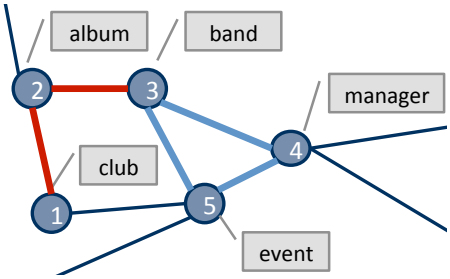


Figure 3.2: Two annotated pattern matches in entity graph from Figure 3.1.

Further, we derive patterns from a weighted graph capturing relationships among types (Section 3.2.2). These patterns are building blocks for topics, since they capture fine-grained relationships among small groups of types. In order to build topics of a larger size, we combine these patterns by using a neighborhood criterion for patterns. Given such neighborhoods, we compute the transitive closure of neighboring patterns to create topics and respective instances (Section 3.3).

### 3.2.1 Annotated Motif Patterns

We now present our first approach for the creation of annotated patterns, such that these patterns represent a common-sense perception of relations in the real world and thus work well for our approach. For this, we extend the notion of network motifs [MSOI+02]: Milo et al. define motifs as "recurring, significant patterns of interconnections". That is, a pattern is considered to occur "significantly often" in a given graph if its occurrence frequency is significantly higher in this graph than in random graphs with equal node properties. Milo et al. use a cutoff probability ($P = 0.01$) for motifs to occur in random graphs. Such random graphs shall have the same number of nodes as well as a similar node degree distribution as the original graph. In our case, we further ensure that nodes are equally annotated, i.e., the random graphs have a similar type distribution. To incorporate node annotations, we use an extended version of network motifs, namely annotated motifs.

**Definition 3.6** (annotated motif)
Given an annotated entity graph $G = (V, E)$ with $type()$ and the set of all annotations $C$, an annotated motif $m = (C_m, E_m)$ of size $s$, is an annotated pattern of size $s$ that matches significantly more often in $G$ than in random graphs of equal size with similar node properties. □

Consider again Figure 3.1. Here, one could, for instance, find three annotated patterns as depicted in Figure 3.3 (assuming statistical significance for these patterns):

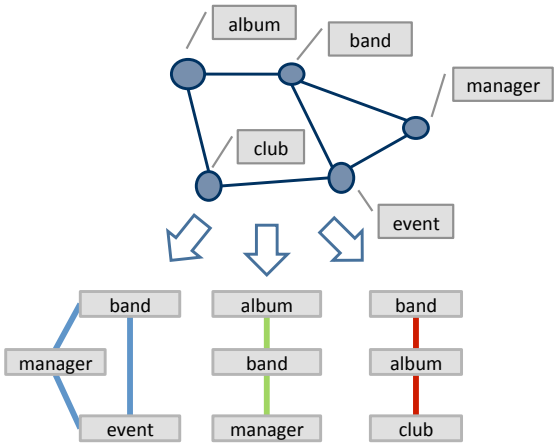


Figure 3.3: Patterns extracted from annotated entity graph in Figure 3.1.

- $p_1$=($\{club, album, band\}$,$\{(club, album), (album, band)\}$), and
- $p_2$=($\{album, band, manager\}$,$\{(album, band), (band, manager)\}$), and
- $p_3$=($\{band, manager, event\}$,$\{(band, manager), (manager, event), (event, band)\}$).

In order to consider patterns to occur significantly more often than in random graphs, we generate graphs that roughly preserve node degrees. That is, given the annotated nodes from an entity graph, each node has a distinct counterpart in the random graph with the same annotation and edge degree. The random graph generation essentially shuffles around edges of the original graph. Then, we use a one-sided t-Test with $\alpha = 0.01$ in 30 random graphs[48] to test whether the occurrence frequency of a pattern at hand is higher in the original than in the generated graphs.
In the following, if we use annotated motifs as patterns of relations among types, we denote this with *AM patterns*.

### 3.2.2 Mutual Information Patterns

Annotated motifs as relation patterns are an elegant choice, since they directly rely on the entity graph. That is, one can determine only patterns that actually exist in this specific case. Unfortunately, it is a computationally expensive process, because one needs to determine all annotated patterns in a (preferably large) number of random graphs (see Section 3.4). In order to retain the statistical argument for relation patterns, but provide a pattern discovery process that is more efficient, we present an approach that

[48]The number of random graphs was determined by available compute resources.

works with an abstract model of relations among types. Interestingly, resulting annotated patterns for relations appear to be more intuitive, though they cannot necessarily be found in the entity graph.

**Definition 3.7**
Given an annotated entity graph $G = (V, E)$ with $type()$ and the set of all annotations $C$:

- The annotation count $ac(c) = |\{v \in V | c \in type(v)\}|$ is the number of nodes in $G$ annotated with $c$.
- The annotation edge count $ec(c_1, c_2) = |\{(v_1, v_2) \in E | c_1 \in type(v_1) \wedge c_2 \in type(v_2)\}|$ is the number of edges among nodes annotated with respective types.
- The constant $\alpha = \sum_{c_1, c_2 \in C} ec(c_1, c_2)$ is the total number of edges among annotations via annotated entities.
- The constant $\beta = \sum_{v \in V} |type(v)|$ denotes the total number of annotations in the entity graph.

□

With these notations at hand, we can define an abstract model of relations among types.

**Definition 3.8** (annotation graph)
Given an annotated entity graph $G = (V, E)$ with $type()$ and the set of all annotations $C$, an annotation graph is a weighted graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ is the set of unique annotations and $\bar{E}$ is constructed from the entity graph. Edge weights $w(c_1, c_2)$ are an estimate of the mutual information among annotations in the entity graph:

$$\begin{aligned} \bar{V} &= C \\ \bar{E} &= \{(c_1, c_2) \mid c_1, c_2 \in C \wedge \\ &\quad \exists (e_1, e_2) \in E \ : \ c_1 \in type(e_1) \wedge c_2 \in type(e_2)\} \\ w(c_1, c_2) &= \log \frac{ec(c_1, c_2)/\alpha}{ac(c_1)/\beta * ac(c_2)/\beta} \end{aligned}$$

□

Thus, the annotation graph comprises distinct annotations from an entity graph and captures relationships among types. These relationships are implicitly given in the entity graph, namely through connected entities annotated with respective types. Further, the annotation graph models the strength of these relationships with an estimate of the

mutual information (MI), i.e., the weighted probability of two types occurring together in the entity graph. Note that an annotation graph contains all relations among all types from an annotated entity graph. In a next step, we solve the following problem for the extraction of relation patterns among types (which are nodes in the annotation graph).

**Definition 3.9** (top $k$ maximum edge connected subgraph problem)
Given an edge-weighted graph $\bar{G} = (\bar{V}, \bar{E})$ with $w()$ as well as constants $k$ and $s$, the top $k$ maximum edge connected subgraph problem is to find $k$ connected subgraphs $G_i = (V_i, E_i)$ $(i = 1, ..., k \wedge |V_i| = s)$, such that there is no subgraph $G_j = (V_j, E_j)$ $(j \neq 1, ..., k \wedge |V_j| = s)$ with $\sum_{e \in E_i} w(e) < \sum_{e \in E_j} w(e)$. □

Thus, the problem seeks exactly $k$ connected subgraphs with the highest sum of edge weights. Since the input is our annotation graph, the output is a set of $k$ highly connected annotation node structures, i.e., coherent type structures.

Note that the problem from Definition 3.9 is a variant of the NP-hard maximum edge subgraph problem. However, given the size of the annotation graph (hundreds of nodes), a size $s \leq 5$ (which is sufficient for annotated patterns) as well as a long tail edge weight distribution, it is feasible to approximate $k$ maximum subgraphs with the bottom-up branch-and-bound strategy depicted in Algorithm 1.

The input for the heuristic shown in Algorithm 1 is the weighted annotation graph from Definition 3.8, the requested number of subgraphs $k$ and the size threshold $s$. In this heuristic, subgraphs of size $i$ are built from subgraphs of size $i - 1$ (starting from the edges in $\bar{G}$, see Line 3). A new subgraph of size $i$ is considered only, if the maximum weight it can achieve during expansion is among the top $k * 100$ achievable weights (Line 2). Given $k * 10$ subgraphs of size $s$, we consider each adjacent node (Line 8) and compute the weight gain as well as the upper bound for the new subgraph (Line 11). If the upper bound ranges among the top $k * 100$ upper bounds (Line 12), the subgraph is further considered and included in a set that is 10 times the size of $k$ (Line 14). The reason for this is that after further extensions of the pattern, it could exceed the weight of the $k^{th}$ subgraph. Here, 10 and 100 are empirically chosen factors to ensure that we do not miss the top-$k$ subgraphs. The removal of low entries from $B$ not among the top $k * 100$ upper bounds is implicit in Line 13. The output comprises the top $k$ subgraphs from each sorted set of subgraphs (descending by weight, Line 15).

In the following, if we use maximum edge connected subgraphs from the annotation graph as patterns of relations among types, we denote this with *MI patterns*.

**Algorithm 1** Top $k$ max-weight connected subgraph heuristic

Input: $\bar{G} = (\bar{V}, \bar{E})$ with $w()$, and $k, s$
$B \leftarrow$ empty sorted (asc) set with fixed size k*100
$P_2 \leftarrow$ sorted (desc) set of node sets per edge $\in \bar{E}$
$a \leftarrow$ average edge weight
**for** $i = 3...s$ **do**
    $P_i \leftarrow$ empty sorted (desc) set with fixed size k*10
    **for** each subgraph $p \in P_{i-1}$ **do**
        **for** each candidate node $c$ adjacent to $p$ **do**
            $weight = \sum_{u,v \in p,(u,v) \in \bar{E}} w(u, v)$
            $gain = \sum_{u \in p,(u,c) \in \bar{E}} w(u, c)$
            $ub = weight + gain + \sum_{i \leq j < s} j * a$
            **if** $|P_i| < k * 10 \quad \vee \quad B.first < ub$ **then**
                add $ub$ to $B$
                add $p \cup \{c\}$ to $P_i$
Return: $P_3[1..k]...P_s[1..k]$

## 3.3 Pattern Percolation

The preceding approaches yield annotated patterns that capture relations among types. These patterns are the building blocks for larger topics. We now discuss how to combine these patterns in order to build topics and topic instances. More precisely, we combine *pattern matches* since a combination of *patterns* could result in topics that do not have instances in the given data. We call this process *Annotated Pattern Percolation (APP)*, since patterns percolate via respective matches through the entity graph. Intuitively, patterns percolate through the graph, if the graph comprises overlapping matches for similar patterns. This process is inspired by Clique Percolation [DPV05, PDFV05]. Consider Figure 3.4 as an example for pattern percolation. Let $P = \{p_1, p_2, p_3\}$ be a set of patterns (the red, green and blue one, respectively). In the example, the graph comprises three pattern matches $M = \{G'_1, G'_2, G'_3\}$:

- $p_1$ =({*club*, *album*, *band*}, {(*club*, *album*), (*album*, *band*)}) (given in red)
  matches at $G'_1$ = ({1,2,3}, {(1,2), (2,3)}).
- $p_2$=({*album*, *band*, *manager*},{(*album*, *band*), (*band*, *manager*)}) (given in green)
  matches at $G'_2$ = ({2,3,4}, {(2,3), (3,4)}).
- $p_3$=({*band*, *manager*, *event*},{(*band*, *manager*), (*manager*, *event*), (*event*, *band*)}) (given in blue)
  matches at $G'_3$ = ({3,4,5}, {(3,4), (4,5), (5,3)}).

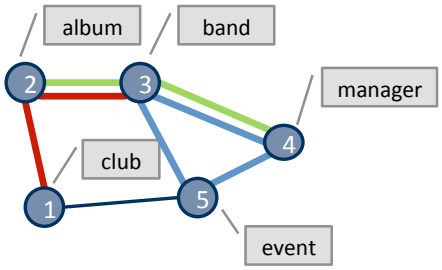


Figure 3.4: Overlapping pattern matches in entity graph.

Note that these matches overlap significantly. That is, $G'_1$ (red) shares two of its three nodes with $G'_2$ (green) and $G'_2$ shares two out of three nodes with $G'_3$ (blue). This way, there is a percolation of $p_1$ at $G'_1$ via $p_2$ at $G'_2$ to $p_3$ at $G'_3$. Then, the entity nodes from the chain of pattern matches form a topic instance: here $I = \{1, 2, 3, 4, 5\}$. The *actual* topic for this instance is the set $T = \bigcup_{e \in I} type(e)$ of annotations from entities in $I$: here $T = \{club, album, band, manager, event\}$.
In the following, we formalize this intuition and discuss how we implement the percolation process. Given a set $P$ of patterns from an annotated entity graph $G$, the set $M$ represents all matches of patterns from $P$ in $G$. To combine matches from $M$, we work with a so-called match graph $G^*$. This graph allows a simple algorithmic approach for the pattern percolation, namely the computation of the transitive closure in $G^*$. First, we define the notion of annotated pattern neighborhood. Then, we build the graph $G^*$ that captures such neighborhood information.

**Definition 3.10** (annotated pattern neighborhood)
Given two annotated pattern matches $G'_1 = (V'_1, E'_1)$ and $G'_2 = (V'_2, E'_2)$ as well as a degree of overlap $d$, $G'_1$ and $G'_2$ are neighboring iff the following holds:

$$\begin{aligned} \exists \hat{V} = V'_1 \cap V'_2 \quad \text{such that} \quad & |\hat{V}| \geq d \\ \wedge \quad & \forall (v, w) \in E'_1 : v, w \in \hat{V} \rightarrow \exists (v, w) \in E'_2 \\ \wedge \quad & \forall (v, w) \in E'_2 : v, w \in \hat{V} \rightarrow \exists (v, w) \in E'_1 \end{aligned}$$

□

Definition 3.10 requires two matches to overlap in $d$ of their nodes. We have chosen $d = \min(|V_1|, |V_2|) - 1$ in order to derive particularly cohesive topics. Requiring all edges among overlapping nodes of the first pattern to be present in the second pattern (and vice versa) further enforces cohesive topics. Of course, one could loosen the criteria by setting $d$ to $\min(|V_1|, |V_2|) - 2$ or removing the edge condition. However, we use a strict setting to ensure pairs of matches that are highly similar and not only overlap in

some minor fraction of their nodes. Remember Figure 3.4 as an example. Here, matches $G'_1$ = ({1,2,3}, {(1,2), (2,3)}) and $G'_2$ = ({2,3,4}, {(2,3), (3,4)}) are neighboring, since they share two nodes (2, 3) and respective edge. Further, $G'_2$ and $G'_3$ are neighbors. $G'_1$ and $G'_3$ share only one node and thus are not neighboring.

Given the notion of annotated pattern match neighborhood, we combine neighboring matches to iteratively build topic instances. Intuitively, to build an instance, we start with an arbitrary match and attach neighboring matches until no more neighbors are available. The result is a chain of neighboring matches. This chain then comprises all nodes of a topic instance. This process is repeated until all matches have been tested for neighboring matches.

In order to implement this approach, we build a match graph $G^*$ from the annotated entity graph and respective pattern matches.

**Definition 3.11** (match graph)
Given the set $M = \{G'_1, \ldots, G'_n\}$ of all annotated pattern matches in an annotated entity graph $G$, the match graph $G^* = (V^*, E^*)$ is constructed as follows:

$$\begin{aligned} V^* &= M \\ E^* &= \{(G'_x, G'_y) | G'_x, G'_y \in M \wedge G'_x, G'_y \text{ are neighboring}\} \end{aligned}$$

□

Thus, $G^*$ contains a node for each match from $M$ in $G$ and edges among nodes iff the matches are neighboring in $G$. Consider Figure 3.5 as an example: The upper part depicts the entity graph with its (neighboring) pattern matches from Figure 3.4. Each of these matches can be found as a node in $G^*$, shown in the lower part. Edges exist among the red and the green match as well as among the green and the blue match.

For the percolation of patterns via matches in $G$, we now compute the transitive closure for all nodes in $G^*$. Then, the closure of a node in $G^*$ also represents all nodes from the annotated entity graph $G$ that can be combined via neighboring matches. In Figure 3.5, the closure of the node for the red match contains the green as well as the blue match. These matches represent nodes $1, \ldots, 5$ from $G$ – a topic instance.

Each closure in $G^*$ thus represents a topic instance $I$, i.e., a subset of entity nodes from $G$. Given all topic instances $I_1, \ldots, I_n$ in $G$, the topics of the data in $G$ are $\{T_i | T_i = \bigcup_{e \in I_i} type(e)\}$ with $i = 1, \ldots, n$.

With this formal process we can implement a system for mining topics from graph structures data, i.e., RDF data in particular.

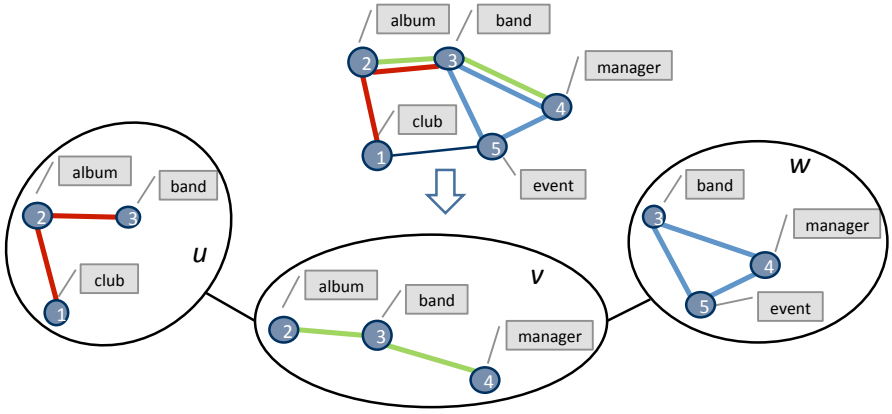


Figure 3.5: Match graph $G^*$ created from annotated entity graph $G$ and pattern matches.

## 3.4 The APP System

In the following, we outline the workflow of the Annotated Pattern Percolation system. Figure 3.6 shows all involved components. Since we deal with large graphs, we have implemented most of the system using the Map/Reduce paradigm [DG04]. With Map/Reduce on a distributed compute platform one can distribute independent map tasks and consume intermediate results in reducers. In Figure 3.6, we marked these components with M/R. Besides pre- and postprocessing, the APP system comprises four major subcomponents:

1. Map/Reduce-based Annotated Pattern Finding (MRAPF),
2. AM Pattern Finding,
3. MI Pattern Finding,
4. Pattern Percolation.

**Preprocessing.** To lower the complexity of our system, the preprocessing performs an annotation selection for entity graph vertices with more than a single annotation. Note that this step is not necessarily required but it lowers the pattern finding runtime in particular. This is because, the finding process enumerates all annotated patterns, where the number of annotations is an important factor (in terms of time and space). This is particularly important when working with RDF data, since in RDF entities often have multiple types. For instances, `Volkswagen` is a `Motor Vehicle Company`, a `Company of Germany`, a `Company established in 1937`, etc. With Kny [Kny12] we developed three strategies for the annotation selection tailored for RDF – each with a

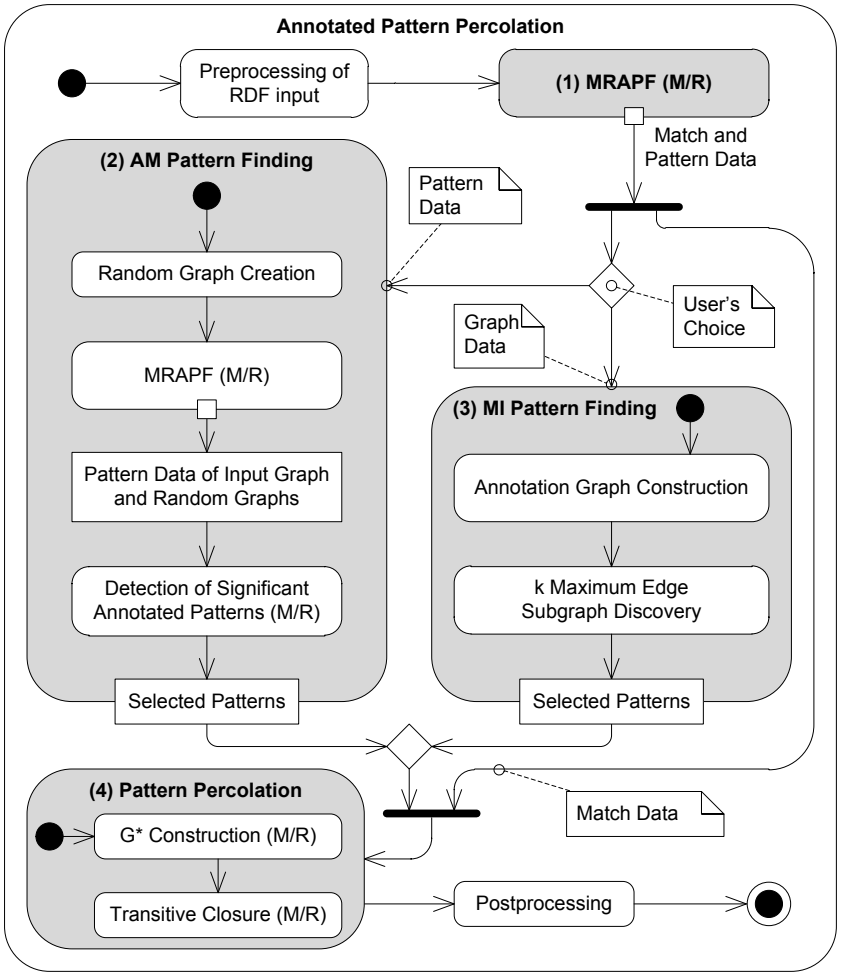


Figure 3.6: Overview of the components forming the Annotated Pattern Percolation system. The notation is BPMN [GDW09]; Filled circles indicate start and end of a (sub)process; rhombs are choices; grey areas show subprocesses.

different underlying intuition. In general, we distinguish the *most specific types* of an entity and their *super-types*. Here, given a type hierarchy, i.e., an ontology, the first ones are those types of an entity such that there are no known sub-types for that entity. The super-types are all ancestors of the specific types in the hierarchy.

We now summarize our annotation selection approaches: The first two approaches select a particular type for an entity by considering the role of the entity it mostly has with respect to other entities. For this, we take relationships among entities into account. The third approach does not consider relationships. Instead, it aims to select a type that semantically covers all known types of an entity.

The first approach, **Most Probable Domain (MPD)**, considers the empirical probability $P(t|p)$ of an entity to be of a specific type $t$ given its relationship $p$ to other entities. Then, given an entity $e$, it selects the type $t$ among the most specific types with the highest average probability $\bar{P}_e(t) := avg(P(t|p))$ over all relationships $p$ of $e$.
The second approach, **Most Significant Domain (MSD)**, determines the empirical probability $P(t)$ for each type over all entities in an RDF dataset. For a particular entity $e$, it selects the type $t$ among all its types (most specific and super-types), where the average probability $\bar{P}_e(t)$ over all the entity's relationships differs the most positively from the expected probability $P(t)$, i.e., $(1 + \bar{P}_e(t) - P(t))/2$.
Finally, the **Most Semantic Coverage (MSC)** approach computes the Semantic Similarity [JC97] among all most specific and respective super-types. It then chooses a type from the super-types that maximizes the average semantic similarity to all most specific types of the entity[49].
In an evaluation, we tested the considered approaches on DBpedia data with DBpedia and Yago types. Note that DBpedia contains only hundreds of types and comprises few hierarchy levels whereas Yago comprises thousands of types and is a deep and manifold hierarchy. Here we found for DBpedia types that MPD and MSC select mostly the same types (81%), which are rather specific. In contrast, MSD chooses more general types. For Yago types we observed that MSD and MSC select similar types. This is because Yago entities have rich annotations and MSD as well as MSC select more general types than MPD.
In a manual evaluation we showed DBpedia entities and Yago types to users and used a majority vote to determine the type *most intuitive* for humans. To transfer the manual selection to DBpedia types, we mapped them to Yago types via cooccurrence frequencies. In the comparison with the automatic selection, we observed for DBpedia that MPD selects the same type for 47% of considered entities. For Yago types, MSC achieves the best performance, i.e., it selects the same types for 79% of the considered entities.

**Pattern Finding.** After the random graph creating we perform MRAPF, which is an extended version of Map/Reduce-based Pattern Finding (MRPF) [LJC+09]. It performs the enumeration of all matches and respective annotated patterns in an entity graph. Given annotated patterns (starting from size 2, i.e., edges), the mapper extends these by locating incident nodes of matches and constructs all possible combinations. The reducer then removes duplicate patterns. Algorithm 2 details the MRAPF match and pattern extension Map/Reduce tasks. The input for this job is a match and its

[49] If there is only one type annotation available, then this one maximizes the semantic similarity. Thus, the selected type is not necessarily a super-type.

**Algorithm 2** Map/Reduce-based Annotated Pattern Finding (MRAPF)

**Map** ( key = match = $(V', E')$ , value = $setOfPatterns$ ):
**Require:** $G = (V, E)$
1: **for all** $p \in setOfPatterns$ **do**
2: **for all** adjacent nodes $w$ of match **do**
3: **for all** combinations $E''$ of edges among $w$ and match **do**
4: $match' \leftarrow (V' \cup \{w\}, E' \cup E'')$
5: $patterns \leftarrow \emptyset$
6: **for all** combinations of annotations of $w$ and pattern $p$ **do**
7: add new pattern to $patterns$
8: emit(key = $match'$, value = $patterns$)

**Reduce** ( key = $match'$, values = $list\ of\ setOfPatterns$ ):
9: emit(key = $match'$, value = values.first())

list of respective patterns. A match can comprise several patterns, since, in the general case, a node in an entity graph has more than one annotation. For instance, the match $(\{x, y, z\}, \{(x, y), (y, z)\})$ with annotations $type(x) = \{c_1, c_2\}$, $type(y) = \{c_3\}$, and $type(z) = \{c_4\}$ generates pattern $p_1 = (\{c_1, c_3, c_4\}, \{(c_1, c_3), (c_3, c_4)\})$ as well as $p_1 = (\{c_2, c_3, c_4\}, \{(c_2, c_3), (c_3, c_4)\})$. In practice, our preprocessing reduces the number of annotations per entity to one. Thus, in line 1 there is only one pattern $p$. Note that the job requires the node adjacency in memory, which can be handled using effective data structures. With that, we examine all incident nodes (line 2) and determine possible matches (lines 3 and 4). We then compute all annotated patterns (lines 5-7). Again, in practice, there is exactly one pattern per match. In line 8, the mapper emits the current match and all its annotated patterns. Since an extended match can be found more than once, e.g., both vertex sets $\{x, y\}$ and $\{y, z\}$ can be extended to $\{x, y, z\}$, the reducer eliminates duplicates by emitting only one set of patterns.

In practice, we split this phase such that we first extend matches and eliminate duplicates and then determine annotated patterns per distinct match. This way, we save tremendous network traffic by not sending unnecessary annotated pattern matches from mappers to reducers. Note that though MRAPF is a computationally expensive process, it can be accomplished on small-scale Hadoop clusters due to highly tuned pattern- and match canonization [McK81], serializations, and comparison.

Observe that MRAPF is also part of the AM Pattern Finding. Here, MRAPF runs on a number of random graphs (created using smart shuffling of edges in the original graph). Given the total counts of patterns for the input graph as well as the random graphs, the system computes and selects significant annotated patterns, i.e., annotated motifs, that

can be used for the pattern percolation.
For finding MI patterns, the system requires only the original input graph for the annotation graph construction and the maximum edges connected subgraph finding. For the graphs we considered (DBpedia and Yago), both computations can be accomplished on a single machine. The graph construction involves edge grouping by types and count computations which can be done efficiently. For the connected subgraph finding we leverage our branch-and-bound strategy, shown in Algorithm 1.

**Pattern Percolation.** Eventually, a Map/Reduce program takes the annotated patterns of choice as well as all annotated pattern matches as input, computes $G^*$, and determines the transitive closure for all nodes in $G^*$. For the closure computation, we employ a scalable approach based on *Disjoint Sets* [MP10] implemented as part of Kny's Master's Thesis [Kny12]. *Disjoint Sets* basically allow constant-time set operations, such as *find* as well as *union*, and can be used to build local closures in mappers that can then be merged by a single reducer. Eventually, a post-processing, the final step in Figure 3.6, forms topic instances (sets of entities) and derives topics (sets of annotations).

**Complexity of APP.** The runtime behavior of APP varies for the different phases. The enumeration of all annotated patterns (MRAPF in Figure 3.6) has a high worst-case runtime, because, in theory, there can be $\binom{n}{m} * a^m * c$ patterns in an annotated entity graph. Here, $n$ is the number of entities in the graph; $m$ is the pattern size; $a$ is the number of distinct annotations; and $c$ is the number of connected graphs with $n$ nodes[50]. However, this is the case only if entity nodes are fully connected and each entity is annotated with all available annotations. In practice, however, edge degrees often follow a power-law distribution and our preprocessing selects a single annotation per entity.
When creating AM patterns ((2) in Figure 3.6), we run MRAPF on a number of random graphs with equal node properties. The creation of a single random graph is linear in the number of edges since we essentially remove or shuffle edges. Checking the significance of annotated patterns, i.e., annotated motifs, is a straight-forward statistical computation that can be done in constant time.
The retrieval of MI Patterns ((3) in Figure 3.6) consists of the construction of the annotation graph and the detection of $k$ maximum weight connected subgraphs. The annotation graph construction requires a single traversal of the graph to collect all annotations and their relationships and is thus linear in the number of edges and the maximum number of annotations per entity graph node. The top-$k$ maximum weight connected subgraph

[50] `http://oeis.org/A001349`

problem (Definition 3.9) is NP-hard, since the *dense k-subgraph problem* [FKP99] is already NP-hard. To efficiently deal with this complexity issue, we employ the bottom-up branch-and-bound heuristic shown in Algorithm 1, which works well in light of the typical power-law distribution of edge degrees and for small subgraph sizes (we use $s \leq 5$). The complexity of Algorithm 1 is in $O(s*k*o)$ where $s$ is the size of required subgraphs, $k$ is the number of desired patterns, and $o$ is the maximum out degree of a node in the annotation graph.
In theory, the construction of the match graph $G^*$ ((4) in Figure 3.6) again depends on the theoretic number of patterns in an annotated entity graph (see above). In practice, there are less patterns due to sparse entity graphs and a single annotation per entity node. The final closure computation can then be done in linear time.
In conclusion, these complexity estimations imply feasible runtimes. Nevertheless, it is a challenging task to deal with large graphs and it requires highly-tuned data structures for serialization and comparisons. Also, the Map/Reduce framework helps to deal with the load since resource consumption can be distributed across compute nodes.

## 3.5 Experiments

In the following we discuss results of our approach for mining topics from graph-structured data. Note that the entire design of APP shall model the intuition from Definitions 3.1 and 3.2. Again, we are aware of that the underlying "relatedness" of types is fuzzy and a matter of opinion. However, it is a step forward taking into account that there are many works dealing with topics; but none of them defines the notion of topics under consideration – as discussed in Section 2.2. Since there is no common topic definition, there is no common standard to compare results to and it is thus inherently challenging to judge the quality of our results. In the scope of this thesis, we compare results from different settings and point to differences to the original clique percolation (CLP) [DPV05, PDFV05], which relies on entity cliques, whereas we use annotation connectivity. Further, we present a comparison with topics extracted from Wikipedia portals[51]. This is a first attempt to compare topics against a common manually created standard.

**Input.** For the evaluation we used DBpedia 3.7 (September 2011), which can be downloaded from `wiki.dbpedia.org/Downloads37`. With the data, Bizer et al. provide an ontology[52], which we additionally used for the annotated entity graph creation. Fur-

[51] `http://en.wikipedia.org/wiki/Wikipedia:Portal`
[52] `http://wiki.dbpedia.org/Downloads37#ontologyinfoboxtypes`

ther, one can download links to Yago2, currently maintained by Hoffart[53]. We used these Yago2 links to create a second version of the annotated DBpedia entity graph. That is, in both cases the graph consists of DBpedia entities. However, the two versions are *annotated with different sets of types*, i.e., DBpedia and Yago2 types.

**Reference Topics.** Wikipedia articles correspond to real-world entities and can thus be used to construct a Wikipedia entity graph. In conclusion, Wikipedia portals, containing articles, can be viewed as sets of entities, i.e., topic instances. These *portals* are similar to the high-level classifications of the Yahoo! Directory[54] or dmoz[55]. At the time of writing, Wikipedia featured about 1,100 portals. Portals are intended to present a given topic and introduce "the reader to key articles, images, and categories that further describe the subject." They are organized as a tree with twelve top-level portals (culture, geography, history, etc.). A typical portal is divided into common information, selected articles, selected facts, current news, and the like. Many of the links to other articles change regularly, and many seem randomly chosen rather than providing a stable and principle reference page. Nevertheless, since it is a manually created reference set, we extract topics from these portals (from all levels) and compare the results with our approach. In the following, we focus our evaluation on *topics*, i.e., *sets of types*.
We extracted topics from Wikipedia portals in the following manner: We were given sets of Wikipedia articles – each corresponding to a portal. For each article we determined the respective DBpedia entity by converting Wikipedia URLs to DBpedia URIs and removing those not present in DBpedia. For all entities in a set derived from a portal we then determined associated DBpedia and Yago types, which results in sets of types, i.e., reference topics.

**Topic Comparison.** After this procedure we have a set of reference topics for comparison. For the comparison of our topic set with reference topics, we need an assignment of our topics to *appropriate* reference topics (our topics are not labeled), and a measure to evaluate the assignment.
For the assignment, we aim at an automatic process since we deal with hundreds to thousands of topics. There are many different solutions to the bipartite assignment problem, which include algorithms to achieve a stable marriage property, or to guarantee a maximum weighted matching. For simplicity, we have chosen the following straight-forward assignment procedure: We first compute Jaccard set similarity values $sim(T_x, T_y)$ for each pair of topics (one from our set and one from the reference set). Then, we assign

[53] `http://wiki.dbpedia.org/Downloads37#linkstoyago2`
[54] `http://dir.yahoo.com`
[55] `http://www.dmoz.org`

a topic $T_x$ from our set to a reference topic $T_y$, iff $sim(T_x, T_y)$ is the maximum for both topics. That is, there are no topics $T'_x$ or $T'_y$ such that $sim(T_x, T_y) < sim(T_x, T'_y)$ or $sim(T_x, T_y) < sim(T'_x, T_y)$. We are aware that requiring the match to be the best fit for either one of the topics is a very strict criterion. However, this models our intuition that not all topics from our approach have a corresponding reference topic, nor can all reference topics be found in our topics. Also, the afore-mentioned more sophisticated matching approaches would optimize the matching towards a global maximum and would thus not respect individual results retrieved.

After the assignment, we have pairs of topics: one from our APP result and another one from the reference set. Now, we calculate precision, recall, and F-measure values to determine how *well* two topics match. Precision is the fraction of correctly retrieved types; recall describes the fraction of relevant types retrieved; F-measure is the (weighted) harmonic mean of precision and recall.

To examine the *overlap* of resulting topics we determine Jaccard set similarity values *for each pair of topics in the result* and show average values and standard deviation. The Jaccard set similarity is the ratio of the intersection and the union size of two sets and thus yields values between 0 and 1. Table 3.1 shows pairwise topic overlap for the Wikipedia portal reference sets.

| | | | **#annotations** | | **Jaccard index** | |
|---|---|---|---|---|---|---|
| | | **#topics** | **avg.** | **std.** | **avg.** | **std.** |
| *DBpedia* | MPD | 1,061 | 9.2 | 9.3 | 0.068 | 0.096 |
| | MSD | 1,052 | 9.7 | 8.7 | 0.100 | 0.109 |
| | MSC | 1,061 | 9.2 | 9.3 | 0.068 | 0.096 |
| *Yago* | MPD | 1,176 | 22.2 | 64.3 | 0.013 | 0.040 |
| | MSD | 1,025 | 10.1 | 12.7 | 0.060 | 0.088 |
| | MSC | 1,151 | 15.0 | 29.6 | 0.019 | 0.054 |

Table 3.1: Pairwise inter-topic overlap (all pairs) of reference topics extracted from **Wikipedia portals** for DBpedia entity graph with *DBpedia* and *Yago* annotations. MPD, MSD, and MSC refer to the different graph preprocessings discussed in Section 3.4.

**Annotated Entity Graphs.** Given the DBpedia RDF input as well as `rdf:type` relationships to DBpedia and Yago types, we created entity graphs. Table 3.2 illustrates the annotated entity graphs after applying the different preprocessings described in Section 3.4, namely MPD, MSD, and MSC. We used these graphs for creating the results discussed later in this section. In particular, the table shows the dimension of the entity graphs. Remarkably, the graph with DBpedia annotations has more vertices then

the other one. This is because not all DBpedia entities have respective Yago types. On the other hand, the Yago-annotated entity graph comprises thousands of annotations whereas the DBpedia versions have 230-250 annotations. For the Yago-annotated graphs with different preprocessings there is a tremendous difference of how annotations are distributed across a specific graph. Here the Most Probable Domain approach (MPD) selects very many different annotations such that on average a single annotation is used only 18 times. Instead, the Most Semantic Coverage annotation selection (MSC) uses less than a tenth of the annotations, which results in a higher annotation usage frequency. In general, Yago-annotations occur fewer times than DBpedia-annotations.

| | | **#vertices** | **#edges** | **#annot.** | **avg. #annot. freq.** |
|---|---|---|---|---|---|
| *DBpedia* | MPD | 1,404,378 | 5,415,394 | 251 | 5595 |
| | MSD | ” | ” | 235 | 5976 |
| | MSC | ” | ” | 252 | 5573 |
| *Yago* | MPD | 890,737 | 3,409,886 | 49537 | 18 |
| | MSD | ” | ” | 5177 | 172 |
| | MSC | ” | ” | 4079 | 218 |

Table 3.2: Size of DBpedia entity graph with *DBpedia* or *Yago* annotations. We show the number of vertices, edges, and annotations as well as the average frequency of an annotation for each graph.

**Parameters.** We now recap the parameters that influence the Annotated Pattern Percolation. For the experiments discussed in the following, we chose settings such that we cover a reasonable range of possible results.

- The annotation selection during the preprocessing (see Section 3.4) is important, because it determines the type of annotations that resulting topics can contain.

- The type of patterns used for percolation is crucial since they model the interconnection of entities in the real world (see Definition 3.1 and 3.2). We propose AM and MI patterns (see Section 3.2.1 and 3.2.2). AM patterns produce reliable matches in an an entity graph since they directly stem from this graph. Manual inspections, however, indicate that MI patterns yield more intuitive results.

- The pattern size $s \in \mathbb{N}^+$ influences the coverage of topics, i.e., the higher $s$, the narrower the topics.

- The number of patterns used for percolation is crucial for the number of topics created. The fewer patterns percolate, the fewer topics can be retrieved. In the following, we use $k$ to indicate the number of percolating patterns.

- The pattern overlap $d \in [1, \ ..., \ \min(|V_1|, |V_2|) - 1]$ (see Definition 3.10) further influences the size of created topics. The higher $d$, the smaller and narrower are resulting topics.

- There is a set of additional parameters that can further influence the annotated pattern percolation result. For instance, the pattern match overlap definition could take edge directions and labels into account (so far, we disregard directions and labels). Finally, there are several pattern and topic filter approaches. We neglect these further parameter settings to narrow the space for experiments.

**Exemplary Results.** Table 3.3 shows exemplary results for topics extracted from the MPD DBpedia entity graphs with different annotations: Topics 1 through 6 comprise DBpedia annotations; Topics 7 through 11 have Yago types. Topic 1 deals with administrative and military-related issues and comprises additional related types, such as geographic objects. The second topic is similar but does not cover the geographic dimension. Topic 3 deals with School or University related Sports entities whereas topic 4, 5, and 6 are about the media business. Here, Topic 4 is the broadest since it contains relatively general types. Topic 5 adds the television-dimension and Topic 6 additionally states what the media could be about, i.e., it includes Company and Military Conflict.

Obviously, given a large input entity graph it is challenging to grasp and differentiate the many resulting topics. To this end, we examined strategies to organize topics in a hierarchy based on topic containment [Kny12]. A topic contains another topic if it comprises all its annotations or covers respective semantics, i.e., it is semantically more general. However, remember that we additionally envision technical use cases, which do not require "human readability". Nevertheless, the more manifold annotations are, the more helpful is a structured view on the result.

Consider Topics 7 through 11 as more complex examples, i.e., results from a Yago-annotated graph. Here, the topics often come with a specific time, space, or other additional dimension since the Yago type system covers these. In conclusion, there are more variations of related topics. Topic 7 deals with english football; the second is about the british television domain. Topics 9-11 cover the media: Topic 9 is about albums released by japanese labels in 2001, whereas the 10th topic deals with Australian music, and the 11th is about groups from the folk music genre.

| | **DBpedia** annotations |
|---|---|
| 1 | Administrative Region, Company, Military Conflict, Office Holder, Person, Person Function, Radio Station, River, Settlement, Village |
| 2 | Member Of Parliament, Military Person, Person, Person Function, School Settlement |
| 3 | Road, School, Settlement, Soccer Player, University |
| 4 | Film, Musical Artist, Person, Single |
| 5 | Album, Musical Artist, Settlement, Single, Television Show |
| 6 | Album, Band, Company, Country, Film, Military Conflict, Musical, Artist, Single, Town |

| | **Yago** annotations |
|---|---|
| 7 | Association Football Forwards, English Football Clubs, Geoclass Populated Place, Living People, Person, University, Yago Geo Entity |
| 8 | BBC Television Dramas, BBC Television Programmes, British Films, English-language Films, Independent Films, ITV Television Programmes, Living People |
| 9 | 2001 Albums, Japanese Record Labels, Living People |
| 10 | Album, American Record Labels, Debut Albums, Living People, Victoria (Australia) Musical Groups |
| 11 | American Folk Rock Groups, American Record Labels, Living People, Musical Groups Established In 2006 |

Table 3.3: Example topics extracted from DBpedia entity graph with different annotations.

**Clique Percolation.** In the following, we consider clique percolation results, which serve as a baseline. With the baseline we can compare precision, recall and F-measure values, but also size and variance as well as topical overlap of APP results. The clique percolation method is similar to APP, but percolates entity cliques, not annotated patterns. At the beginning of this project there was no scalable implementation of the clique percolation method. However, our implementations implicitly implements CLP through requiring patterns to be cliques. Table 3.4 shows pairwise overlap of resulting sets of annotations; Table 3.5 depicts respective precision-, recall-, and F-measure values. For ease of comparison with further measurements we focus in the discussion on the values for percolating cliques of size 4. In general, clique percolation retrieves reasonable results with respect to Wikipedia portals for all entity graph annotation selection approaches (F-measure values up to 0.8). However, note that only few topics created can be mapped to Wikipedia portal topics. This is particularly the case for the entity graph with Yago annotations (Table 3.5) where only about a tenth of the topics created can be mapped to the reference topic set. This is important since precision and recall refers to the comparison of *mapped* topics, i.e., the two sets of types. Respective values in Table 3.5 (and the following) are averaged over all pairs of mapped topics.

| | | | #annotations | | Jaccard index | |
|---|---|---|---|---|---|---|
| | | clique size | avg. | std. | avg. | std. |
| *DBpedia* | MPD | 3 | 4.5 | 2.3 | 0.068 | 0.106 |
| | | 4 | 3.4 | 1.2 | 0.059 | 0.128 |
| | MSD | 3 | 4.4 | 2.2 | 0.070 | 0.105 |
| | | 4 | 4.9 | 1.9 | 0.166 | 0.159 |
| | MSC | 3 | 4.5 | 2.2 | 0.068 | 0.105 |
| | | 4 | 3.4 | 1.3 | 0.059 | 0.128 |
| *Yago* | MPD | 3 | 5.0 | 6.2 | 0.027 | 0.060 |
| | | 4 | 4.9 | 2.2 | 0.058 | 0.084 |
| | MSD | 3 | 4.5 | 2.5 | 0.055 | 0.103 |
| | | 4 | 4.0 | 1.4 | 0.090 | 0.140 |
| | MSC | 3 | 4.6 | 3.6 | 0.023 | 0.068 |
| | | 4 | 4.3 | 1.7 | 0.064 | 0.116 |

Table 3.4: Pairwise inter-topic overlap (all pairs) of topics created with **clique percolation** for the DBpedia entity graph with *DBpedia* and *Yago* annotations.

| | | clique size | #topics | #mappings | prec. | rec. | F-m. |
|---|---|---|---|---|---|---|---|
| *DBpedia* | MPD | 3 | 6,061 | 592 | 0.84 | 0.77 | 0.78 |
| | | 4 | 863 | 251 | 0.85 | 0.83 | 0.81 |
| | MSD | 3 | 8,823 | 629 | 0.86 | 0.80 | 0.81 |
| | | 4 | 2,697 | 311 | 0.85 | 0.82 | 0.81 |
| | MSC | 3 | 6,106 | 577 | 0.84 | 0.77 | 0.78 |
| | | 4 | 864 | 246 | 0.87 | 0.82 | 0.81 |
| *Yago* | MPD | 3 | 27,314 | 894 | 0.50 | 0.38 | 0.35 |
| | | 4 | 5,324 | 359 | 0.47 | 0.40 | 0.34 |
| | MSD | 3 | 8,222 | 584 | 0.81 | 0.76 | 0.75 |
| | | 4 | 1,602 | 242 | 0.81 | 0.81 | 0.79 |
| | MSC | 3 | 15,122 | 750 | 0.62 | 0.60 | 0.56 |
| | | 4 | 3,192 | 325 | 0.61 | 0.64 | 0.56 |

Table 3.5: Precision, Recall, and F-Measure (with respect to Wikipedia portals) of topics created with **clique percolation** for the DBpedia entity graph with *DBpedia* and *Yago* annotations.

**AM Pattern Percolation.** Table 3.6 and 3.7 depict respective results for the AM pattern percolation. The small number of mappings to portal topics roughly holds for topics created with 50,000 AM patterns (Table 3.7). However, the size of the topics, i.e., the number of annotations per topic, is larger than for CLP and there is a higher (in some cases much higher) variance among topic sizes (Table 3.6) – these values are comparable the those for the portals.
The inter-topic overlap is slightly higher (Jaccard values of 0.1-0.18) in some cases. Note that larger and more manifold topics are a desirable feature. Nevertheless, these more complex topics make it more challenging to retrieve similar topics like in a reference set. Thus, precision and recall values drop in some cases. Interestingly, AM pattern percolation performs significantly better in the case where CLP fails, namely on MPD selected Yago annotations (Table 3.7, line 4).

| | | **#annotations** | | **Jaccard index** | |
|---|---|---|---|---|---|
| | | **avg.** | **std.** | **avg.** | **std.** |
| *DBpedia* | MPD | 5.3 | 3.7 | 0.106 | 0.107 |
| | MSD | 5.1 | 2.9 | 0.131 | 0.112 |
| | MSC | 5.3 | 3.7 | 0.104 | 0.106 |
| *Yago* | MPD | 5.2 | 76.7 | 0.181 | 0.085 |
| | MSD | 5.3 | 4.1 | 0.107 | 0.108 |
| | MSC | 7.1 | 11.0 | 0.065 | 0.089 |

Table 3.6: Pairwise inter-topic overlap (all pairs) of topics created with **AM pattern percolation** for DBpedia entity graph with *DBpedia* and *Yago* annotations (k=50,000).

| | | ***#topics*** | **#mappings** | **prec.** | **rec.** | **F-measure** |
|---|---|---|---|---|---|---|
| *DBpedia* | MPD | 7,629 | 569 | 0.71 | 0.75 | 0.70 |
| | MSD | 8,711 | 633 | 0.78 | 0.78 | 0.75 |
| | MSC | 7,772 | 574 | 0.71 | 0.75 | 0.70 |
| *Yago* | MPD | 5,025 | 159 | 0.51 | 0.60 | 0.51 |
| | MSD | 7,049 | 477 | 0.73 | 0.75 | 0.71 |
| | MSC | 6,790 | 426 | 0.50 | 0.59 | 0.50 |

Table 3.7: Precision, Recall, and F-Measure (with respect to Wikipedia portals) of topics created with **AM pattern percolation** for DBpedia entity graph with *DBpedia* and *Yago* annotations (k=50,000).

**MI Pattern Percolation.** Table 3.8 and 3.9 depict results created with MI patterns on DBpedia-annotated entity graphs. Experiments on *Yago*-annotated graphs did not succeed since the top one million MI Yago-patterns could not be matched. Thus, we observe that MI patterns do not directly stem from the entity graph – they cover latent relationships instead. By MI pattern definition, these relationships occur in the graph. Nonetheless, types connected in patterns are not necessarily directly connected (via entities) in the graph. To overcome this, we propose a variation of the pattern match neighborhood in Definition 3.10 that does not require equality of annotations. It rather takes their semantic similarity into account. However, due to a lack of respective experiments, the semantic similarity pattern match neighborhood is not within the scope of this thesis and thus left for future work. Instead of MI pattern experiments with Yago-annotated graphs, we show measurements considering a varying number $k$ of patterns used for percolation in DBpedia-annotated graphs. The selection of $k$ has been done such that the number of topics created roughly approaches the number of topics created with AM patterns.

Considering Tables 3.8 and 3.9 we find that discovered topics are mostly larger than those created with AM patterns. The topic size variance is also larger than respective values for AM results, which negatively influences the precision and recall measurement with respect to Wikipedia portals. Further, though MI patterns create fewer topics (which is a matter of $k$), we observe up to 10% more mappings to DBpedia portal topics. These are the highest values (percentages with respect to created mappings) in this evaluation. Though this indicates structural similarity of MI pattern topics with the reference topics, it influences precision and recall values negatively.

| | | **#annotations** | | **Jaccard index** | |
|---|---|---|---|---|---|
| | | avg. | std. | avg. | std. |
| k=100,000 | MPD | 9.58 | 8.11 | 0.06 | 0.09 |
| | MSD | 4.9 | 2.33 | 0.07 | 0.12 |
| | MSC | 7.13 | 5.86 | 0.05 | 0.1 |
| k=300,000 | MPD | 8.94 | 7.69 | 0.05 | 0.08 |
| | MSD | 7.41 | 4.74 | 0.1 | 0.1 |
| | MSC | 7.54 | 6.1 | 0.05 | 0.08 |
| k=600,000 | MPD | 1.83 | 1.21 | 0.0 | 0.01 |
| | MSD | 1.62 | 1.0 | 0.0 | 0.0 |
| | MSC | 1.9 | 1.3 | 0.0 | 0.01 |

Table 3.8: Pairwise inter-topic overlap (all pairs) of topics created with **MI pattern percolation** for DBpedia entity graph with *DBpedia* annotations (for a different number of percolating patterns $k$).

| | | *#topics* | #mappings | prec. | rec. | F-measure |
|---|---|---|---|---|---|---|
| k=100,000 | MPD | 2,495 | 472 | 0.51 | 0.63 | 0.53 |
| | MSD | 404 | 150 | 0.54 | 0.72 | 0.58 |
| | MSC | 1,210 | 300 | 0.50 | 0.66 | 0.53 |
| k=300,000 | MPD | 4,776 | 663 | 0.53 | 0.66 | 0.56 |
| | MSD | 2,739 | 465 | 0.56 | 0.69 | 0.59 |
| | MSC | 2,789 | 520 | 0.52 | 0.65 | 0.54 |
| k=600,000 | MPD | 5,372 | 696 | 0.53 | 0.66 | 0.56 |
| | MSD | 5,780 | 631 | 0.62 | 0.72 | 0.65 |
| | MSC | 3,120 | 543 | 0.52 | 0.65 | 0.55 |

Table 3.9: Precision, Recall, and F-Measure (with respect to Wikipedia portals) of topics created with **MI pattern percolation** for DBpedia entity graph with *DBpedia* annotations (for a different number of percolating patterns $k$).

## 3.6 Discussion

In this chapter we explored the problem of mining topics and respective instances from heterogeneous graph-structured data. To this end, we presented a novel approach called Annotated Pattern Percolation, APP for short, which exploits the structural information inherent to annotated graphs. Note that we avoided textual information from labels, descriptions, etc. While APP is applicable to data from many fields, we exemplary applied it to DBpedia – a prominent cross-domain linked dataset. An evaluation against topics extracted from existing Wikipedia meta-information, namely Wikipedia portals, shows that we can reasonably reconstruct a portion of this information.

We discussed differences among the original clique percolation method as well as our APP approach and found that our results are structured differently – in particular they are more manifold, i.e., they contain larger topics with a wider range of their size. The broader coverage of topics created by APP makes a comparison to the community-curated reference topic set even more challenging. Table 3.10 summarizes our findings. The original method CLP yields better F-measure values due to relatively few mappings from the result to the reference data. However, AM pattern topics are larger than CLP topics but they can be mapped similarly to the reference topic set. MI pattern topics are similar in size and respective variance like Wikipedia portal topics and additionally map better to the reference data. Finally, note that our APP method achieved higher precision values for the earlier version 3.6 of DBpedia, which we published in [BKN12].

In conclusion, we believe that extracted topics enable users to gain conceptual insights into cross-domain datasets. Further, discussed parameters enable a selection of a proper

APP setting when applied for technical use-case. Also, the plug-and-play nature of APP, i.e., allowing the use of different pattern types, enables easy adoption of other notions of topics. In the future, for instance, one could incorporate frequent subgraphs or Encyclopedic Knowledge Patterns (EKPs) [NGPC11]. Intuitively, EKPs are small ontologies that contain a concept $c$ and its relations to the most relevant concepts $c_i ... c_j$ that can be used to describe $c$.

The automatic labeling of topics is also part of future work in order to improve the user experience. As for usability, we conducted first experiments with organizing topics hierarchically and a web application that allows to browse the resulting topic hierarchy and respective instances.

Finally, there are further options to incorporate more structural knowledge into our method. For instance, one can regard edge directions and labels as well as their semantic similarity.

Unfortunately, since topics in graph-structured data are not an established field of research, any evaluation is inherently challenging and subject to different points of view.

| | **CLP** | **APP with AM** | **APP with MI** |
|---|---|---|---|
| Topic size | small<br>low variance | larger than CPL<br>comparable variance | size and variance<br>most similar |
| #Mappings | 250–300 | 500 | double % |
| F-measure | $0.6 - 0.8$ | $0.5 - 0.7$ | 0.55 |
| Overlap | similar | similar | similar |

Table 3.10: Summary of **comparison** of baseline and APP **to Wikipedia portal reference** topic set.

# 4 Entity Alignment

Linked Open Data (LOD) is a way of interconnecting structured data sources on the Internet and creating a *Web of Data*. As the name suggests, the key of LOD is that it can provide extensive cross-linkage between sources at the level of entities. Links are typically represented in the form of additional triples with the `sameAs` predicate. For example, two `sameAs` links suffice to connect data about the director David Lynch in DBpedia, Freebase, and the BBC. This way, one can answer queries that join biographic data from DBpedia with data about Lynch's music compositions from the BBC. Obviously, this form of entity linkage at Web-scale has enormous potential.

Unfortunately, this cross-linkage between LOD sources is not nearly as extensive as one would hope. Exact numbers are not known. An estimate for the number of `sameAs` links is in the order of 400 million[56]. For example, the New York Times archive provides `sameAs` links for only around 5,000 people, 3,000 organizations, and 1,000 locations[57]. A huge number of links exist between the major knowledge bases DBpedia and Freebase, etc. However, these links are trivial as they build upon Wikipedia and can use article titles as a common denominator. In this thesis, we present automatic methods that discover highly accurate previously unknown links, while scaling to the enormous proportions of the Web of Data.

Unlike existing entity matching systems for LOD, such as [HKL+09, VBGK09], which require dataset-specific matching rules, we aim at a fully automated, domain-independent system.

At first glance, this problem seems to be identical to the classical *record linkage* task, also known as *entity resolution*. Given several sets of records, e.g., different databases for the same domain, identify all equivalence classes of records. The problem has received much attention in the literature, driven by applications, such as de-duplication and other forms of data cleansing [Coh00, KR10b, NH10, EIV07]. A close look, however, reveals that there are fundamental differences between the typical record-linkage setting and our problem of entity mapping in LOD: First, the fine-grained and loose-schema nature of RDF triples makes it much harder to identify appropriate inputs for similarity functions. Second, traditional record-linkage methods work best in specific domains where similarity

[56] `http://lod-cloud.net` as of September 2009

[57] `http://data.nytimes.com`

measures, thresholds, and other components can be customized. In contrast, the vast heterogeneity of LOD sources, spanning many different domains, poses a major challenge. Third, we are facing a much larger number of sources, as opposed to the typical data-warehouse scenario. This is an additional difficulty, but also an opportunity, because we can potentially exploit `sameAs` transitivity over many sources. Finally, many billions of triples is a daunting scale that poses very high performance and scalability demands, which standard record-linkage methods do not meet.

These challenges have not been fully addressed by any recent work geared towards LOD, where the main aim is to assist data publishers in creating links between two well-structured data sources. They often rely on existing `sameAs` triples for training and on comparable schemas across sources. None of these methods were designed for and tested against Web-scale data. Since we implement domain-independent similarity functions and design our approach to work in a distributed fashion, we can cope with the size and diversity of the Web of Data as it continues to grow. We anticipate new LOD sources with previously unseen properties and highly dynamic data including HTML-embedded semantic information.

Our approach uses an optimization model that captures the joint evidence for entities in one source corresponding to entities in other sources. The evidence is based on a similarity among the neighborhood of an entity and that of another entity to which a `sameAs` link could possibly exist. Consider the example of the entity `David Lynch` in Freebase and a candidate entity on the BBC site. For inferring that they are truly the same, we consider both sides' attributes such as `types`, `birthplace`, `awards` and also connections to movies, compositions, etc. This typical situation motivates our joint reasoning approach: The evidence that one refers to the same `David Lynch` on both sides increases with the evidence that a certain movie, connected to `David Lynch` in Freebase, is the same as a movie referenced by the `David Lynch` candidate entity in the BBC site. Note that this reasoning proceeds recursively, i.e., a decision about identity of neighboring entities may in turn depend on further neighboring entities. Our approach further takes connected actors, musicians, and places into account (which may in turn depend on yet more connected entities).

In contrast to other joint entity resolution approaches [HNST11, KSRC09, SD06, WCRM09], we have encoded our joint evidence task into a weighted graph, and impose additional constraints on the output. Our LINDA (LINked Data Alignment) approach iteratively processes the graph by increasing prior `sameAs` edge scores depending on the joint evidence of the incident nodes' local neighborhoods. This framework can be instantiated on a shared memory multi-core system, as well as using a distributed Map/Reduce- or message-passing-based approach. This way, LINDA can efficiently process graphs

with $> 100$ millions of nodes and it is feasible to partition the data and parallelize the processing on cloud platforms.
In the following, we first discuss the underlying optimization model for our joint entity matching (Section 4.1) and then describe three algorithms for solving our optimization problem, i.e., finding good mappings (Sections 4.2 through 4.4). In Section 4.5 we discuss the similarity functions we plug into our framework and conclude with experiments in Section 4.6.

## 4.1 Optimization Model

Given Linked Data subject-predicate-object triples, we would like to match entity identifiers that come from different sources and represent the same real-world entity.

**Input.** As specified by the RDF standard, we start with a set of $(s, p, o)$ triples as input. Similar to Definition 3.3, we cast the set of triples into an *entity graph* $G$, with nodes corresponding to entity URIs, and labeled edges representing predicates. To build the graph, we consider only RDF triples where $s$, $p$, and $o$ are URIs. Literals are compiled into a set $L(s)$ of values associated with an entity. For each URI $e$ we denote the data source from which it originates as $S(e)$.

**Definition 4.1** (entity graph)
Given a set of RDF triples $R$, an entity graph is a directed, labeled multigraph $G = (V, E)$ with a set of nodes $V$ representing entities and edges $E$ capturing relations among entities.

$$\begin{aligned} V &:= \{s \mid (s,p,o) \in R\} \cup \{o \mid (s,p,o) \in R\} \\ E &:= \{(s,p,o) \mid (s,p,o) \in R \wedge s, o \in V\} \\ L(s) &:= \{o \mid (s,p,o) \in R \wedge o \text{ is literal}\} \\ S(s) &:= \text{the data source that } s \text{ originates from} \end{aligned}$$

□

**Desired Output.** Our output is a squared binary matrix $X$, telling us whether any two URIs represent the same entity or not, i.e., whether a `sameAs` link (as specified in the OWL standard) should exist between them. The output is subject to certain constraints. We require the `sameAs` relationship to be symmetric and transitive, which corresponds to establishing equivalence classes of entities.
We do not aim to discover `sameAs` links within a single data source. Usually, single sources are much better maintained than the links across sources. High-quality datasets,

such as DBpedia or Freebase, are well-curated and hardly contain duplicate entities. Moreover, approaches for creating `sameAs` links within a single source could use source-specific properties and thus differ from methods for links across sources. Therefore, intra-source links are orthogonal to this work.
Our entity matching algorithm makes joint decisions for multiple URI pairs when producing `sameAs` links. It is initialized with a *prior similarity* between entities, based on the immediate neighborhoods of the URIs given by their respective data sources. The algorithm iteratively considers similarities between entities in the neighborhoods in order to reinforce or invalidate the matching between two URIs. Output matchings are gradually built into an *assignment matrix* $X$. Dynamically re-computed similarity values are tracked in a *similarity matrix* $Y$ of the same shape as $X$, for which the entries are maintained in a priority queue.

**Definition 4.2** (assignment matrix)
Given an entity graph $G = (V, E)$, an assignment matrix $X$ is a symmetric $n \times n$ matrix with $n = |V|$ and $x_{a,b} \in \{0, 1\}$. An entry $x_{a,b}$ states whether our algorithm outputs "$a$ `sameAs` $b$". □

The assignment matrix $X$ represents our solution for the entity matching problem. Feasible assignments are constrained as follows:

**Definition 4.3** (consistency)
An assignment matrix $X$ is consistent if it satisfies the following constraints:

1. **Reflexivity** $\forall a \in V : x_{a,a} = 1$
2. **Symmetry** $\forall a, b \in V : x_{a,b} = x_{b,a}$
3. **Transitivity** $\forall a, b, c \in V : x_{a,b} \cdot x_{b,c} \leq x_{a,c}$
4. **Unique mapping per data source (*optional*)**
$$\forall a, s \neq S(a) : \sum_{b:S(b)=s} x_{a,b} \leq 1$$

□

The optional last constraint states that an entity $a$ cannot simultaneously match two entities $b$, $b'$ that are both from the same second source, since this would imply that $b$ and $b'$ are duplicates of each other. This constraint allows us to focus on cross-source links rather than mappings within data sources. In practice, however, we also found that the strategy of picking only the best match within each dataset also avoids many false positives.

### 4.1.1 Objective Function

To quantify the quality of the output $X$, we define the following objective function: Let $\text{sim}(a, b, G, X)$ be a similarity function between two entities $a$ and $b$ that may depend on

the entity graph $G$ as well as the current assignment matrix $X$. For constant $G$ and $X$, sim should be a semimetric, i.e., it does not necessarily satisfy the triangle inequality. This similarity function should return scores in $[-\infty, +\infty]$, positive scores in the case of likely entity matches and negative scores in the case of likely non-matches. Then, the *maximum consistent assignment problem* is to find an assignment of $X$ that comprises maximum similarity information from sim but also satisfies our constraints.

**Definition 4.4** (maximum consistent assignment)
Given an entity graph $G = (V, E)$ and a similarity function $\text{sim}(a, b, G, X)$, the maximum consistent assignment problem (MCA) is to find

- a consistent assignment matrix $X$ with values $x_{a,b} \in \{0, 1\}$
- that maximizes

$$\sum_{a,b \in V : S(a) \neq S(b)} x_{a,b} \ \text{sim}(a, b, G, X).$$

□

The sim function deliberately depends on $X$, the desired output of the assignment algorithm: For instance, in our previous example (at the beginning of this chapter), the similarity of the two `David Lynch` entities depends on whether the movies `Twin Peaks Fire Walk With Me` and `Twin Peaks (Prequel)` represent the same real-world entity. Thus the similarity function can reward joint assignments that are globally consistent.
The seemingly recursive structure of the objective function, with $X$ being an argument to sim, is similar to optimizing non-linear functions. Once we choose a specific sim function, we obtain an optimization problem with the $X$ entries as variables.
One possible instance of the sim function is to consider the neighbor sets $N(a)$, $N(b)$ of entities $a$ and $b$ and use an aggregated Jaccard coefficient for the names in $N(a)$ and $N(b)$ as a similarity function. This could lead to the following simple objective:
$\max \sum_{a,b \in V : S(a) \neq S(b)} \big(x_{a,b} \cdot avg_{a' \in N(a), b' \in N(b)} x_{a',b'} \cdot Jaccard(a', b')\big)$.
While this objective may appear intuitive, we found that this specific objective function does not achieve a reasonable precision for Linked Data. Instead, we propose a more sophisticated function, as discussed in Section 4.5. Our assignment algorithm iteratively computes entries in $X$, based on evaluating sim on the previous iteration's $X$ values.

### 4.1.2 Complexity

We now show that the maximum consistent assignment problem is NP-hard, no matter whether we include or disregard the optional unique mapping constraint. We first reduce *CLIQUE* to MCA. Then, we give a reduction of the *minimum multiway cut problem* to

show NP-hardness when the optional unique mapping constraint from Definition 4.3 must hold. Note that for this we rely on the decision version of our maximization problem: Given such a graph and a similarity function, is there a consistent assignment matrix such that the sum from Definition 4.4 has at least a specific value?

**Theorem 4.1**
The *maximum consistent assignment problem* without the unique mapping constraint is NP-hard.

*Proof.* We reduce the CLIQUE problem [Kar72] to MCA.

- The CLIQUE problem is to determine whether $G = (V, E)$ contains a clique of size at least $k$.
- Given $G$, we define an instance of MCA as follows:
  - Let $G' = (V, \emptyset)$ be an entity graph.
  - Define $\text{sim}(a, b, X, G') = 1$ if $(a, b) \in E$ and $\text{sim}(a, b, X, G') = -\infty$ otherwise.
- Then, a solution $X$ for MCA induces cliques in $G$, since the consistency requires symmetry and transitivity, which essentially is a complete subgraph.
- These cliques are maximal, since otherwise more edges could be added to $X$ increasing its score.
- To determine the maximum clique size, one determines the maximum number of positive entries per row in $X$, which shall be at least $k - 1$.
- Conversely, any clique of size $\geq k$ implies an $X$ comprising entries for respective clique edges. This $X$ is consistent, because cliques are fully connected and thus symmetry and transitivity holds.
- Thus, solving MCA in polynomial time means to determine the maximal clique size in polynomial time.

□

**Theorem 4.2**
The *maximum consistent assignment problem* is NP-hard if the unique mapping constraint must hold.

*Proof.* We reduce the NP-hard minimum multiway cut problem [DJP+94] to our MCA problem.

- Given an undirected graph $G = (V, E)$ with edge weights $w(e) \geq 0$ for $e \in E$ and a set of terminals $T \subseteq V$, the objective is to find a set $C \subseteq E$ of edges with minimal cost $w(C) = \sum_{e \in C} w(e)$ such that every $t \in T$ is disconnected from all $t' \in T$ (with $t' \neq t$) in $G' = (V, E \backslash C)$, i.e., each $t$ resides in a distinct connected component.

- We define an instance of MCA as follows:
  - Define $\text{sim}(a, b, G, X) = w(a, b)$ if $(a, b) \in E$ and $-\infty$ otherwise.
  - Define $S(v) = S_0$ if $v \in T$ and $S_v$ otherwise.

- Then, an assignment $X$ provides a multiway cut $C = \{(u, v) \in E \mid x_{u,v} = 0\}$, because the unique mapping constraint from Definition 4.3 enforces that terminals are not reachable from one another.

- Intuitively, since $C$ consists of all edges not selected to maximize $X$'s score, it is minimal.

- The cost of $C$ is

$$\begin{aligned} w(C) &= \sum_{e=(u,v) \in E : x_{u,v} = 0} w(e) \\ &= \left( \sum_{e \in E} w(e) \right) - \left( \sum_{e=(a,b) \in E : x_{a,b} = 1} w(e) \right) \\ &= \left( \sum_{e \in E} w(e) \right) - \left( \sum_{a,b \in V : S(a) \neq S(b)} x_{a,b} \ \text{sim}(a, b, G, X) \right) \\ &= \left( \sum_{e \in E} w(e) \right) - k. \end{aligned}$$

- The cost $w(C)$ of $C$ is minimal, since $k := \sum_{a,b \in V : S(a) \neq S(b)} x_{a,b} \ \text{sim}(a, b, G, X)$, i.e., the score of $X$, is maximal by definition.

- Conversely, any multiway cut $C'$ with cost $k'$ implies an assignment $X'$ with $x'_{u,v} = 1$ if $(u, v)$ is connected in $(V, E \backslash C')$, and 0 otherwise.

- $X'$ is consistent, because reachability in an undirected graph is reflexive, symmetric, and transitive.

- The unique mappings constraint is fulfilled, because each node can be connected only to a single $t \in T$ and hence only to one node from $S_0$. All other nodes have distinct sources $S_v$.

- The objective score for $X'$ is

$$\begin{aligned}
&\sum_{a,b \in V : S(a) \neq S(b)} x'_{a,b} \, \mathrm{sim}(a, b, G, X') \\
&= \sum_{e=(a,b) \in E \setminus C'} w(e) \\
&= \left( \sum_{e \in E} w(e) \right) - \left( \sum_{e \in C'} w(e) \right) \\
&= \left( \sum_{e \in E} w(e) \right) - k'
\end{aligned}$$

- Hence, a minimum multiway cut can be found by finding an optimal assignment.

□

In the following, we present three different approaches for the NP hard MCA problem we previously defined. Each of these approaches shall yield a (ideally near-optimal) solution. However, due to their different processing environments, i.e., one runs on multi-core machines and the other two perform in a distributed fashion, they exhibit varying properties, which we discuss throughout the following sections. Eventually, we instantiate these algorithms with specific similarity functions, which together forms the LINDA system used in the experiments.

## 4.2 Assignment Algorithm for Multi-core Machines

We now present an algorithm that computes a consistent assignment matrix $X$ with a high value of the objective function. This first algorithm runs on a single machine and is able to benefit from multiple cores during operation. The method additionally benefits from the availability of additional machines during the initialization phase. Our algorithm iteratively adjusts the values in $X$ with carefully chosen, greedy improvements in the objective function. $X$ is initialized as a diagonal matrix, i.e., with 1 on the diagonal and 0 elsewhere. In addition, we use a *similarity matrix* $Y$ with pairwise, real-valued similarities: positive values support matches between entities; negative values indicate non-matches. $Y$ is initialized with values that reflect the pair-wise similarity of entities, as further described in Section 4.5. Given the previous iteration's values of $X$, we then compute new similarity values for $Y$, which are maintained in a priority queue, i.e., sorted by similarity values, and used to adjust elements of $X$ in priority order. Note that both $X$ and $Y$ are sparse symmetric matrices that do not need to reside entirely in memory.

**Algorithm 3** Multi-Core Assignment Algorithm

```
1:  procedure LINDA(G)
2:      X ← I_|V|                                          ▷ identity matrix
3:      Q ← initial similarities                           ▷ with Map/Reduce
4:      while Q non-empty do
5:          dequeue (y_{a,b}, {a, b}) with highest y_{a,b} from Q
6:          if y_{a,b} < 0 then break
7:          X_0 ← X
8:          for all a' ∈ E(a, X_0), b' ∈ E(b, X_0) do      ▷ assignment
9:              x_{a',b'} ← 1                              ▷ update X
10:             x_{b',a'} ← 1
11:         S ← {(a', b') | sim(a', b', X, G) ≠ sim(a', b', X_0, G)}
12:         for all (a', b') ∈ S : x_{a',b'} = 0 do in parallel   ▷ update Y in parallel
13:             fetch (y_{a',b'}, {a', b'}) from Q
14:             y* ← sim(a', b', G, X)
15:             if y* ≠ y_{a',b'} then                     ▷ similarity changed
16:                 remove (y_{a',b'}, {a', b'}) from Q
17:                 if y* > 0 then
18:                     enqueue (y*, {a', b'}) in Q
19:     return X
```

Moreover, the entries of $Y$ need to be materialized only on demand as needed for the priority queue. Negative scores do not need to be retained at all, so the $Y$ matrix as a whole is a conceptual construct only. Nevertheless, in our implementation, both data structures must be highly efficient to cope with the many value retrievals and updates. In general, we consider this not to be a bottleneck since main memory capacities increase tremendously. If main memory is not sufficient, there are highly-efficient disk-backed data structures.

Algorithm 3 describes the method formally. In line 3, the updatable priority queue $Q$ is initialized with initial similarity scores $y_{a,b} = \text{sim}(a, b, G, X)$, in case they are greater than zero. In practice, sim is defined to have a negative score for most entity pairs. Using Map/Reduce, we can efficiently determine which entity pairs can possibly have positive scores without having to compute a large quadratic number of similarities (details in Sec. 4.5). The computation of initial similarities can be done in less than 12 hours for all datasets, ranging from 44 to 120 million entities, discussed in Section 4.6.

The algorithm then repeatedly dequeues the entity pair $a, b$ with the highest similarity score from the queue and sets $x_{a,b}$ to 1 – see the for-loop starting in line 8. To enforce transitivity (see Definition 4.3), it considers not only $a$ and $b$ but also all *equivalents* $E(a, X) = \{a' | x_{a,a'} = 1\}$ of $a$ and $E(b, X) = \{b' | x_{b,b'} = 1\}$ of $b$ already identified with them. The for-loop (line 8) essentially merges the two equivalence classes for $a$ and $b$. The loop also ensures that the symmetry constraint from Definition 4.3 is satisfied.

Since sim may depend on $X$, modifying $X$ means that similarity scores need to be recomputed. In line 11, the algorithm determines which pairs of entities may need updating. Note that this selection varies for different similarity measure. This is particularly important for the distributed implementations we present in Sections 4.3 and 4.4, because it may need additional graph traversals to reach required fractions in the graph. For the similarity function we define later in Section 4.5, updates are required for

- the candidate pairs involving entities in $E(a, X) \cup E(b, X)$, since their local neighborhood has changed, as well as

- pairs $(a', b')$ where $a' \in \{s \mid (s, p, o) \in E, o \in E(a, X)\} \cup \{o \mid (s, p, o) \in E, s \in E(a, X)\}$ and $b' \in \{s \mid (s, p, o) \in E, o \in E(b, X)\} \cup \{o \mid (s, p, o) \in E, s \in E(b, X)\}$, because the similarity of entities connected to entities in $E(a, X) \cup E(b, X)$ might have changed.

Note that we update only if entities are potential match candidates, i.e., if there is an initial similarity above zero. In our implementation discussed in Section 4.6, we require an ngram overlap among entities under consideration, i.e., $\text{ngram}(a') \cap \text{ngram}(b') \neq \emptyset$. On multi-core machines, the algorithm adjusts the scores of these pairs in $Y$ in parallel. The new similarity scores $y_{a',b'}$ computed in line 14 reflect the fact that we now have $E(a, X) = E(b, X)$, highlighting the joint mapping strategy of our algorithm.

Our algorithm produces a consistent (Definition 4.3) matrix $X$ if $\text{sim}(a, b, G, X) = -\infty$ whenever $S(a) = S(b)$ (i.e., if $a$ and $b$ are from the same source). This is because, reflexivity is fulfilled due to the initialization of $X$ in line 2. Symmetry and transitivity are fulfilled, because all pairwise equivalences between equivalence classes are set simultaneously in lines 8 and 9 rather than just the original mappings. Whenever an entity would need to be connected to two entities from the same source, $y^*$ in line 14 is $-\infty$, so the algorithm ends up not making the assignment.

Observe that our algorithm may converge to a local maximum of the objective function. However, our framework allows easy modifications in the priority order of the queue, and thus the algorithm's steps. In particular, randomization can avoid such issues. For example, one can pick $(a, b)$ pairs from the priority queue at random biased by the pair's $y_{a,b}$ value. Then, with low but non-zero probability we skip the highest-priority pairs and proceed with alternatives.

## 4.3 Assignment Algorithm with Map/Reduce

The previously described Algorithm 3 computes a consistent assignment matrix for the objective from Definition 4.4. It exploits multi-core systems in that it performs $y_{a,b}$ (re)computations in parallel (line 12). In Section 4.6, we will show that it is feasible to apply this approach to large fractions of the Web of Data. Nevertheless, it is bound by the number of CPU cores available for parallelism as well as (in our implementation) the amount of memory available to store the resulting assignment matrix $X$ and (parts of) the priority queue $Q$.

To overcome these bounds, we introduce a Map/Reduce-based version of our assignment algorithm that allows scaling our solution to immense amounts of data, considering that today's large cluster sizes range from hundreds to thousands of compute nodes. With Map/Reduce, a problem is divided into many independent map tasks that consume and emit key-value-pairs as well as a number of reduce tasks that aggregate intermediate output to resulting key-value-pairs [DG04]. Figure 4.1 illustrates the workflow of our Map/Reduce-based approach (MR-LINDA for short).

Before diving into the figures' details, remember that the acceptance of a `sameAs` edge between vertices $a$ and $b$ induces a series of similarity (re)computations: These computations may depend on an arbitrary fraction of the input graph. Now consider an input entity graph (see left of Figure 4.1) that is distributed across a cluster and a mechanism to follow edges in the graph. In our Map/Reduce-based approach, reducers handle specific graph partitions. Hence, if $t$ is the number of edge hops from $a$ or $b$ to related nodes with edges whose sim values may change, then the algorithm will use $t$ subsequent reducer calls to trigger the respective similarity recomputations. For the sim function we define in Section 4.5, a new `sameAs` edge induces two chunks of recomputations. First, all queue entries in which $a$ or $b$ are involved must be reconsidered. Second, the contexts of equivalents in $E(a, X)$ and $E(b, X)$, as discussed in Section 4.2, must be taken into account. Thus, similarity recomputation require two edge traversals: one to reach the equivalents and a second to reach the contexts. These traversals can also be seen in Figure 4.1 (cf. *notify* and *update* messages).

In our Map/Reduce-based approach, each node holds a portion $Q_i$ of the queue $Q$ and a respective partition $G_i$ of the entity graph $G$. The entity graph partition comprises all queue entry vertices' contexts, i.e., their local neighborhoods. Given a pair of entities $(a, b)$ from the queue, the node to hold the Q entry and respective vertex contexts is determined by a modulo operation of the first component ($a$). The queue and entity graph partitions are stored as sorted lists on the compute nodes. This allows fast merge-join-like access of required entries on the compute nodes directly, instead of shuffling the

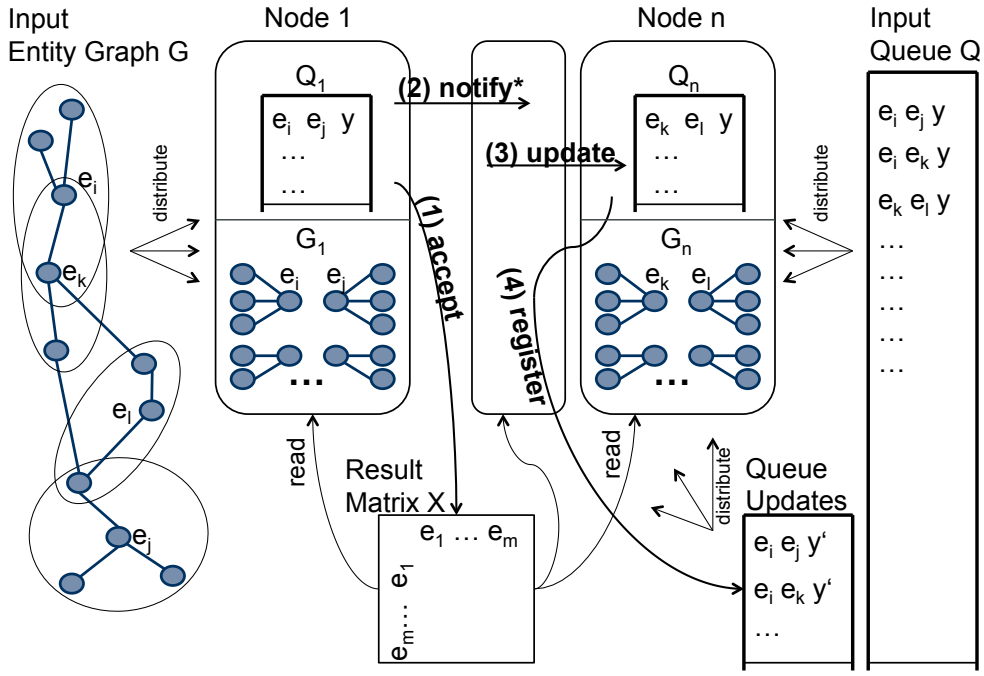


Figure 4.1: Map/Reduce Workflow for the LINDA algorithm.

graph and the queue across the cluster [LS10]. The only information sent from mappers to reducers are messages about which pairs of vertices to reconsider.

Algorithm 4 shows the map and reduce phases for our first distributed approach. Additionally, Figure 4.1 illustrates the flow of the messages. Note that the message flow is tailored to our sim function as defined later in Section 4.5. In the general case, however, other sim functions might require further notification steps (indicated by the $*$ in the figure) to reach the required graph partition.

The input to the mappers in Algorithm 4 are the nodes' specific queue partitions $Q_i$ as well as the messages sent in previous phases. Each mapper reads $Q_i$ and stores the respective pair of entities in a buffer $B$ if its $y$ value is among the top $K$ entries (line 5). We refer to $K$ as the *acceptance rate*. That is, mappers *bulk-accept* sets of mappings (of size $K$) from the queue. Further, mappers forward messages from previous phases (line 7). After mapper completion, a procedure *map-close()* accepts all pairs of equivalents for buffered queue entries for the resulting $X$ (line 12 and step (1) in Figure 4.1), i.e., it writes to a file in the distributed file system. Note that this is where our Map/Reduce-approach merges respective equivalence classes. Additionally, *map-close()* emits two messages: `notification` indicates the new entry in $X$ and triggers the update for equivalents' contexts as well as respective queue entries for $a'$ (line 13 and step (2) in Figure 4.1). `updateTargets` triggers the update of queue entries for $b'$.

**Algorithm 4** Map/Reduce Assignment Algorithm

```
1: Input: queue partition Q_i, messages from previous iterations
2: B ← empty sorted set (asc) of fixed size K
3: procedure MAP(Key k = (a, b), Value v)          ▷ v is similarity y_{a,b} or a message
4:     if v is similarity ∧ v > B.first.v then
5:         remove first from B, add (k, v) to B                          ▷ buffer entity pair
6:     else if v is notification or update
7:         emit (k, v)                                  ▷ forward message from previous phase
8: procedure MAP-CLOSE( )
9:     for all ((a, b), y_{a,b}) ∈ B do
10:        for all a' ∈ E(a, X_0), b' ∈ E(b, X_0) do                    ▷ merge equivalence classes
11:            k' = (a', b'), k'' = (b', a')
12:            accept k'
13:            emit (k', notification), (k'', updateTargets)
14: procedure COMBINE(Key k = (a, b), Values V)                        ▷ V is a set of messages
15:    n, t, u = false
16:    for all v ∈ V do                                                ▷ aggregate messages
17:        if v is notification then n = true
18:        if v is updateTargets then t = true
19:        if v is update then u = true
20:    if n == true then emit (k, notification)
21:    if t == true then emit (k, updateTargets)
22:    if u == true then emit (k, update)
23: procedure REDUCE(Key k = (a, b), Value v)                          ▷ v is a message
24:    if v is notification then                          ▷ react to new equivalence mapping
25:        determine contexts C(a) and C(b) from G_i
26:        for all (a', b') ∈ C(a) × C(b) do
27:            emit ((a', b'), update)
28:        for all (a, b*) ∈ Q_i do
29:            emit ((a, b*), update)
30:    if v is updateTargets then                    ▷ trigger update for queue entries
31:        for all (a, b*) ∈ Q_i do
32:            emit ((a, b*), update)
33:    if v is update then                                             ▷ perform update
34:        if umc holds for a' ∈ E(k.a, X), b' ∈ E(k.b, X) then
35:            register y' = sim(a, b, X, G_i) for Q_i
36:        else
37:            for all a' ∈ E(k.a, X), b' ∈ E(k.b, X) do
38:                register y' = −∞ for Q_i
```

These messages are distributed across the compute cluster in the same manner as the queue and entity graph partitions. For this distribution, we implemented a custom partitioner that is used in different places of the algorithm to ensure proper distribution according to entity pairs. Thus, a message arrives at the compute node where the

respective queue and graph data resides. The partitioned data resides in a distributed file system. However, due to the principle of locality, reducers are launched where the data resides locally. To reduce message traffic, we implemented a simple combiner (line 14-22) that passes only one message per message type and target. Message duplicates can occur, for instance, when updates were triggered by different sources.

Then, a reducer reacts according to the message it receives (the value in line 23). Given a `notification`, it uses its graph partition $G_i$ to determine the contexts of $a$ and $b$ and emits `update` messages (line 27 and step (3) in Figure 4.1). Further, given its partition $Q_i$ of the queue, it triggers the update of respective entries (line 29). It also triggers updates when receiving an `updateTargets` message (line 32), which ensures the update of the queue entries for the second component of a previously accepted pair. The actual update of queue entries is triggered when receiving an `update` message. Then, the reducer checks whether the unique mapping constraint still holds (*umc* in line 34) and computes as well as registers the new $y'$ value (step (4) in Figure 4.1). If the constraint is violated, it registers negative values for all pairs of equivalents (line 38). Technically, updated $y$-values are written to the distributed file system and then merged during the following read of the queue partitions in the mapper.

This *map()*, *map-close()*, *combine()*, and *reduce()* sequence proceeds until a predefined number of iterations has been reached, a certain number of entries in $X$ has been computed or a specific fraction of the queue has been processed. Note that the round-trip of the effect of a new entry in $X$ takes two iterations (since we do two edge traversals). That is, the `notification` from *mapper 1*, the `update` from *reducer 1*, the forward of the `update` in *mapper 2* and the actual $y$ value computation in *reducer 2*. Therefore, unique mapping constraint violations can occur when accepting `sameAs` edges, say in *mapper 2*, while queue updates for decisions from previous mappers (*mapper 1*) are still pending. We quantify this effect in Section 4.6. In certain cases, MR-Linda can also violate the transitivity constraint. We discuss this issue in the following Section 4.4.

An implementation detail worth mentioning is that we use 64 bit hash values for the identification of a URI, i.e., a representation of a real-world object. To check whether our constraints from Definition 4.3 hold, we need dataset information during the assignment process. In our multi-core-implementation of LINDA we can retrieve such information from a database. However, this is not a desired feature for a distributed implementation. Therefore, we encoded additional information, such a dataset id, into the 64 bit URI identifier. This allows for a very fast check of certain constraints.

## 4.4 Assignment Algorithm with Message-passing

The previously described Map/Reduce-based implementation of LINDA allows performing joint entity matching on a very large scale. This is because, since the publication of the Google Map/Reduce paper [DG04] (about a decade ago), the Map/Reduce computing paradigm has been adopted by a wide range of global industrial and academic players and thus runs on clusters with hundreds to thousands of compute nodes.

Map/Reduce was initially used to process large amounts of textual data. Though, it has shown to be able to solve particular problems for graph data [Coh09, KTF09, LS10]. We previously discussed how it can be used to perform joint entity matching in entity graphs. Nevertheless, the Map/Reduce-based approach has shortcomings: In general, Map/Reduce is meant to process data once and distill aggregates that can be used in further steps, which are not necessarily based on Map/Reduce. It is not meant to run in iterations, since iterations require processing the same data multiple times – this is the case in our Map/Reduce-based LINDA implementation where we iterate over the queue many times. Recently, a set of approaches dedicated to efficient iterations in a distributed fashion over large datasets emerged [BHBE10, ELZ+10, ETKM12]. A second set of approaches particularly focuses on graph data processing [LGK+12, MAB+10]. Later in this section, we show how we adopt the processing model from [MAB+10] for LINDA.

**Constraints in MR-LINDA.** Due to the independent bulk processing of chunks from the queue, our Map/Reduce approach can produce mappings that might violate certain constraints. For one, the optional unique mapping constraint may be disregarded when accepting mappings while negative queue updates from previous iterations are still pending. Further, it might happen that two mappers merge overlapping equivalence classes. Consider Figure 4.2 as an example. On the left, there are three equivalence classes, namely $E(a, X), E(b, X) = E(c, X)$, and $E(d, X)$. Further, there are two similarity values, shown as red edges, that are currently under consideration, i.e., they are acceptance candidates. When accepting the matchings ($(a, b)$ and $(c, d)$) simultaneously, our Map/Reduce-based LINDA takes care of the proper merge of $E(a, X)$ and $E(b, X)$ as well as $E(c, X)$ and $E(d, X)$. However, in this new situation, shown in the middle section of the figure, $E(a, X)$ is not equal to $E(d, X)$. This is because the two merges take place independently and thus cannot regard vertices currently added to the existing equivalence class $E(b, X) = E(c, X)$. That is, in certain cases the transitivity constraint can be violated.

To overcome this constraint violation based on the simultaneous extension of an existing equivalence class, we need a setup in which processes extending classes can exchange

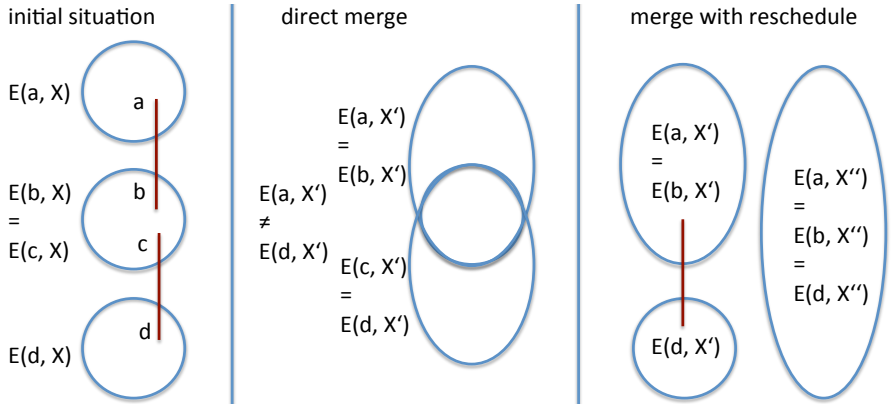


Figure 4.2: Different equivalence class merges in LINDA. Ellipses illustrate equivalence classes.

information to "block" the class under consideration. Then, we could first merge $E(a, X)$ with $E(b, X)$ and reschedule the merge of $E(c, X)$ with $E(d, X)$ for a later iteration (see right part of Figure 4.2). Therefore, we chose to implement a third version of our LINDA algorithm. For this approach, we aimed at a close coordination among the distributed processes dealing with different fractions of the graph.

**BSP Processing Model.** In particular, we adopted an idea recently presented in the Pregel paper [MAB+10]. Here, the authors employ the Bulk Synchronous Parallel processing model (BSP) [Val90] for graphs. In its original form, BSP consists of three steps:

1. Concurrent asynchronous processing of independent tasks on a set of processors. This processing may change the state of each task's environment.

2. Asynchronous communication among the independent tasks.

3. When independent tasks hit a so-called synchronization barrier, they wait until all tasks completed processing and communication.

In the Pregel model, a graph vertex can be in two different states: it is either `active` or `inactive`. The framework proceeds in supersteps and graph vertices may pass messages among each other. In each superstep, all active vertices take action. That is, they receive messages sent in the previous superstep, perform local computation (which can change their state or mutate the graph topology), and send messages to other vertices. A vertex can halt only itself, i.e., set itself to `inactive`. Then, such an inactive vertex does not take action in the next superstep, unless it receives a message from some other vertex in the graph. The framework stops when all vertices are inactive. Technically, a

cluster compute node is responsible for a set of graph vertices. Specifically, it triggers the local computation on all vertices; it buffers, sends, and fetches messages in batches; and it takes care of the global superstep synchronization. Since graph vertices may exchange information among each other, this processing model is also called message-passing-scheme.

**MP-LINDA's Data Structure.** For the implementation of LINDA based on message-passing (MP-LINDA for short), we extend our entity graph from Definition 4.1.

**Definition 4.5** (extended entity graph)
Given an entity graph $G = (V, E)$, an initial queue $Q$ with similarity values, and a consistent assignment matrix $X$, an extended entity graph $G' = (V, E, E_0, E_Y, E_X)$ comprises additional undirected labeled edges $E_0$, $E_Y$, and $E_X$.

$$
\begin{aligned}
E_0 &:= \{(a,b) \mid a,b \in V \wedge \text{there is a positive prior for } a \text{ and } b\} \\
E_Y &:= \{(a,b) \mid a,b \in V \wedge \exists (y_{a,b}, \{a,b\}) \in Q\} \\
E_X &:= \{(a,b) \mid a,b \in V \wedge x_{a,b} = 1\} \\
y((a,b)) &:= y_{a,b}, \text{ if } \exists (y_{a,b}, \{a,b\}) \in Q, \quad -\infty \text{ otherwise} \\
p((a,b)) &:= z_{a,b}, \text{ if there is a positive prior } z_{a,b} \text{ for } a \text{ and } b
\end{aligned}
$$

□

Intuitively, we encode the queue $Q$ (see line 3 in Algorithm 3) populated with initial similarities as well as the resulting $X$ (see line 2 in Algorithm 3) into the entity graph. Of course, an initial $X$ contains only ones in the diagonal. Additionally, the extended entity graph comprises edges among entities, if there is a *positive prior*. Such prior exists, if entities are matching candidates. The value itself can be used as smoothing in the similarity computation, which we discuss in the following Section 4.5. Thus, the graph can now capture the current state of the algorithm by adding or removing edges in $E_Y$, setting values $y((a,b))$ for edges in $E_Y$, or augmenting the graph with edges in $E_X$. Observe that $y((a,b))$ denotes a variable label for an edge $(a,b)$. Later, we use the notion $y((a,b)) := s$ to denote that the variable is being updated with $s$. Further, note that in theory all edges in $E_0$, $E_Y$, and $E_X$ are undirected. However, in our implementation discussed below, undirected edges exist twice, since vertices hold local copies of adjacent edges. In the following, for ease of reading, we use *entity graph* (or *graph* for short) to refer to the extended entity graph as defined in Definition 4.5. Also, we use the function $C(a)$ to denote the set of vertices in the context of $a$, i.e., all vertices adjacent to $a$ via edges in $E$. Further, we use functions to access certain vertex

properties. In our implementation, we leverage these properties to guarantee mutual exclusion of vertices during processing to avoid overlapping structural updates in the graph, i.e., avoid constraint violations as depicted in Figure 4.2.

- Given a vertex, the function $id : V \to \mathbb{N}$ returns the vertex id.
- Given a set of vertices $V' \subseteq V$, $max_{id}(V') := argmax_{v \in V'}(id(v))$ returns the vertex with the largest id in $V'$.
- Given a vertex, the function $E : V \to \mathcal{P}(V)$ yields the set of all equivalent vertices $\{w|(v,w) \in E_X\} \cup \{v\}$ of $v$, i.e., it abbreviates $E(v, X)$ for a current state of $X$.
- Similarly, given a vertex, $E' : V \to \mathcal{P}(V)$ yields the set of equivalent vertices $\{w|(v,w) \in E_X\}$ without $v$.

Pregel additionally defines so-called *aggregators* to manage global coordination. In our message-passing-based version of LINDA, we use such an aggregator to manage a global sorted set $A$, which holds all *distinct current similarity values* in descending order that are larger than a predefined threshold. Note that $A$ represents the *order* of similarity values $y(e)$ currently existing for edges $e$ in $E_Y$. Thus, $E_Y$ and $A$ together form the priority queue. On the set $A$ we use operators, such as $\cup$ and $\setminus$ as well as the function $\max(A)$, as follows:

- The operator $\cup$ adds an element to the set $A$ (which remains sorted implicitly).
- $\setminus$ removes a value from the set $A$.
- $\max(A)$ returns the *largest* value from $A$ (without removing it).

**MP-LINDA Algorithm.** Given these definitions, we can formally specify our message-passing-based assignment algorithm. We denote messages as `[message,o]` in brackets. Messages usually comprise their origin, which is the vertex $o$ in the example `[message,o]`.
MP-LINDA proceeds in two phases shown in Tables 4.1 and 4.2 (Pages 82 and 83), respectively. The first phase performs the selection of new mappings as well as, most importantly, the coordination of active vertices. Coordination is necessary, because otherwise non-consistent equivalence classes might emerge when classes are extended twice simultaneously. To avoid such simultaneous modifications of classes, MP-LINDA takes two actions: For one, if a vertex in a class has more than one suitable candidate for a new mapping, it selects a single edge for the actual mapping creation. The second (more general) case is that at least two vertices from the same class have suitable candidates

for a new mapping – remember the class $E(b, X)$ in Figure 4.2. In this case, MP-LINDA selects one of the vertices in the class and leaves the remaining mapping candidates for further iterations as shown on the right in Figure 4.2, i.e., it reschedules them. Here, we first accept the mapping among $a$ and $b$, which creates the consistent $X'$, and then consider $(c, d)$, which creates $X''$. For the selection of a vertex among a set of vertices, we use the vertex id, i.e., we select the vertex with the largest id. Of course, this selection can also be driven by any other deterministic function.

MP-LINDA's second phase performs similarity recomputations for edges in $E_Y$ that might have changed due to newly created mappings. Since we use a joint mapping strategy that incorporates intermediate states of $X$, i.e., edges in $E_X$, we need to request and transfer the context (connected via edges in $E$) of respective vertices to other fractions in the graph. Below we provide the steps taken for the similarity function that we define later on in Section 4.5.2. This function makes use of the neighbors of a given entity. For other similarity functions, appropriate adaptations need to be made. For ease of presentation, in what follows we omit the similarity update of queue entries involving equivalents of currently aligned entities. Similarly, we do not include the comparison of context aspects as defined in Section 4.5.2.

We now elaborate on the details of the algorithm depicted in Tables 4.1 and 4.2: The algorithm consists of 12 supersteps (step for short). The first four steps make up the creation of new alignments (Table 4.1). Step 5 trough 12 perform similarity updates (Table 4.2). We call a full run from Step 1 trough 12 an *iteration*. The algorithm iterates until $A$ is empty, i.e., there are no more similarity values to process, or a predefined threshold for values in $A$ has been reached. It can also stop processing after a given number of iterations.

Before the first iteration, MP-LINDA populates the global sorted set $A$ by iterating over all edges in $E_Y$ (Superstep 0). Then, in Step 1, it selects the maximum current similarity value from the global set. If this value is zero, the queue has been fully processed and MP-LINDA can stop processing.

In Step 2, each active vertex retrieves all vertices that are adjacent via edges in $E_Y$ labeled with the current maximum similarity value. Intuitively, it dequeues all pairs with the current maximum similarity. Also, it removes the maximum value from the global set $A$ (line 7). It then sends three types of messages: `[E'(a),a]` informs a selected node about the equivalence class to merge with. Note that the message contains the equivalence class $E'(a)$ (without a) and the origin of the message $a$ (which in combination is the complete equivalence class). All non-selected vertices $b_2$ receive a `[reschedule,a]` message, which informs them to not add the mapping $(b_2, a)$ under consideration, as another mapping for this vertex ($a$) has been selected. Further, active nodes inform all

other nodes in their equivalence class that they have taken action.

In Step 3, vertices react according to the message they receive. If they received equivalence classes to merge with, they check whether they have the maximum id among among all active vertices in the class (line 13). If this holds, they forward the class to merge with to all vertices in their own equivalence class. Otherwise, they add the current maximum similarity back to the global set $A$, which causes a reconsideration in the next iteration.

In Step 4, vertices stop processing if their respective mapping candidate counterpart was not among the selected vertices in the previous phase, i.e., they received a reschedule message (lines 20-22). Otherwise, a vertex does the actual setting of new mappings, i.e., it adds edges to $E_X$ and removes respective counterparts from $E_Y$ (lines 23-25).

Then, the second phase of the algorithm starts within the same superstep (Step 4 cont. in Table 4.2). At this point, equivalence classes have been exchanged among matching vertices. To determine which direction to send the context for the similarity reconsideration, we again use the ids (lines 27, 28) and forward the equivalence class to the selected context.

This equivalence class is the target in the following Step 5. In Steps 5 through 7, equivalents of the context of the selected matching candidate are sent to equivalents of the context of the other matching candidate.

In Step 8, MP-LINDA determines whether contexts of equivalents of matching candidates $a$ and $b$ are connected via an edge in $E_Y$, i.e., whether a similarity value to reconsider exists. If this is the case, we send the context of the active vertex to the other context.

Steps 9 through 11 collect information for the reconsideration of similarity values. Step 9 first sends the larger set of neighbors to the smaller set of neighbors of the counterpart. Note that we chose to deliver the larger context to send fewer (but larger) messages, since there are fewer message targets in the smaller context. Also note that the current active vertex $c_b$ is an equivalent of a context vertex of a newly matched vertex, i.e., it is a vertex that must reconsider its similarity edge values.

In Step 10, vertices receive contexts, compute the intersection with their own context, and send it back. Though we receive only the context (not the equivalents of each context vertex), we implicitly compare equivalence classes (of contexts), since, due to transitivity of equivalence, one vertex $v$ (here from a context) from a class $E(v)$ is sufficient to determine an overlap of $E(v)$ with another class $E(w)$.

In Step 11, the context equivalents of newly identified vertices receive the context as well as respective context matches from vertices to reconsider the similarity with. With the superset of all context matches (line 45), MP-LINDA can now recompute similarities. The function $\text{sim}'$ refers to the similarity sim we define in the following Section 4.5.

However, its parameters are defined differently, because we explicitly precompute and pass neighbor matches $M$ rather than computing them within the function. The resulting value $s$ is then set as edge label and send to the respective counterpart. In Step 12, the counterpart sets the newly computed $y$ value $s$ and adds it to the global set $A$, which concludes an iteration.

Table 4.1: The equivalence class merge phase of the Message-Passing-based alignment algorithm.

| | Super-step | Active Vertex | Action / Message Content | Target | Comment |
|---|---|---|---|---|---|
| 1 | 0 | $\forall v \in V$ | $\forall e = (v, w) \in E_Y$ with $y(e) > t$ | | Initialize the global sorted set $A$. |
| 2 | | | $A \leftarrow A \cup \{y(e)\}$ | | |
| 3 | 1 | $\forall v \in V$ | $\bar{y} := \max(A \cup \{0\})$ | | Determine current maximum similarity. |
| 4 | | | if $\bar{y} = 0$ then HALT | | Terminate if the queue is empty. |
| 5 | 2 | $\forall a \in V$ | $Y = \{w \mid (a, w) \in E_Y \wedge y((a, w)) = \bar{y}\}$ | | Determine vertices with maximum similarity. |
| 6 | | | if $(Y \neq \emptyset)$ | | If the current vertex has y-edges with value $\bar{y}$. |
| 7 | | | $A \leftarrow A \setminus \{\bar{y}\}$ | | Remove maximum similarity from set $A$. |
| 8 | | | send `[E'(a),a]` | $b_1 = max_{id}(Y)$ | Send equivalence class to selected vertex. |
| 9 | | | send `[reschedule,a]` | $b_2 \in Y \setminus \{b_1\}$ | Notify non-selected vertices to stop processing. |
| 10 | | | send `[active,a]` | $b_3 \in E'(a)$ | Notify vertices in equivalence class. |
| 11 | 3 | $b$ | if received `[E'(a),a]` | | If $b$ was selected by a vertex $a$. |
| 12 | | | $m := max_{id}(\{w \mid$ received `[active,w]`$\}$ $)$ | | If $b$'s id is the largest among |
| 13 | | | if $(id(b) > id(m))$ | | active vertices $w$ in equivalence class. |
| 14 | | | send `[E'(a),a,b]` | $b' \in E(b)$ | Send equivalence class to be merged. |
| 15 | | | else | | |
| 16 | | | $A \leftarrow A \cup \{\bar{y}\}$ | | Else reschedule this similarity |
| 17 | | | send `[reschedule,b]` | $a' \in E(a)$ | and inform equivalence classes. |
| 18 | | | if received `[reschedule,*]` | | If $b$ was not selected by another vertex $a$. |
| 19 | | | $A \leftarrow A \cup \{\bar{y}\}$ | | Reschedule this similarity. |
| 20 | 4 | $b' \in E(b)$ | $\forall v \in$ `[E'(a),a]` | | Stop processing if received |
| 21 | | | if received `[reschedule,v]` | | a reschedule message for any vertex |
| 22 | | | return from $b'$ | | in received equivalence class. |
| 23 | | | $\forall v \in$ `[E'(a),a]` | | |
| 24 | | | $E_X \leftarrow E_X \cup \{(b', v)\}$ | | Add sameAs edges. |
| 25 | | | $E_Y \leftarrow E_Y \setminus \{(b', v)\}$ | | Remove edge that represents queue entry. |

Table 4.2: The context transfer phase of the Message-Passing-based alignment algorithm.

| | Super-step | Active Vertex | Action / Message Content | Target | Comment |
|---|---|---|---|---|---|
| 26 | 4 cont. | $b' \in E(b)$ | $\forall$ `[E'(a),a,b]` received | | If this is a selected vertex. |
| 27 | | | if $id(b) > id(a)$ | | Choose direction for sending context. |
| 28 | | | send `[E'(a),a]` | $c \in C(b')$ | Pass equivalence class to b's context. |
| 29 | 5 | $c \in C(b')$ | send `[E(c)]` | $a' \in E(a)$ | Send context vertices' equivalents to equivalents of $a$. |
| 30 | 6 | $a' \in E(a)$ | send `[E(c)]` | $c' \in C(a')$ | Forward to the context of $a$'s equivalents. |
| 31 | 7 | $c' \in C(a')$ | send `[E(c)]` | $c_a \in E(c')$ | Forward to context equivalents of $a$'s equivalents. |
| 32 | 8 | $c_a \in E(c')$ | $\forall c_b \in$ `[E(c)]` | | |
| 33 | | | if $(c_a, c_b) \in E_Y$ | | If $a$'s context shares $E_Y$-edges with $b$'s context, |
| 34 | | | send `[`$C(c_a), c_a$`]` | $c_b \in E(c)$ | send $a$'s context $C(c_a)$ to $b$'s context equivalents. |
| 35 | 9 | $c_b \in E(c)$ | $\forall$ `[`$C(c_a), c_a$`]` received | | Vertices receive contexts for $E_Y$-edges partners. |
| 36 | | | if $(\lvert C(c_b)\rvert < \lvert C(c_a)\rvert)$ | | Determine the smaller context. |
| 37 | | | send `[`$C(c_a), c_a, c_b$`]` | $c \in C(c_b)$ | Send larger context to the smaller context. |
| 38 | | | else | | (besides origin $c_b$, attach $E_Y$-edge target id $c_a$) |
| 39 | | | send `[`$C(c_b), c_a, c_b$`]` | $c \in C(c_a)$ | |
| 40 | | | send `[`$C(c_a), c_a$`]` | $c_b$ | Resend context to self. |
| 41 | 10 | $c$ | $\forall C_l \in$ `[`$C_l, c_a, c_b$`]` received | | All smaller contexts receive larger contexts $C_l$. |
| 42 | | | $C_m := C_l \cap E(c)$ | | Determine context overlap (entity matches). |
| 43 | | | send `[`$C_m, c_a$`]` | $c_b$ | Send matches back to the sender. |
| 44 | 11 | $c_b$ | $\forall c_a$ : `[`$C_m, c_a$`]` and `[`$C(c_a), c_a$`]` received | | Receive overlap and context for $c_a$. |
| 45 | | | $M := \bigcup C_m$ | | Create set of all context entity matches. |
| 46 | | | $s := \mathrm{sim}'(c_a, c_b, p(c_a, c_b), M, C(c_a))$ | | Compute new similarity. |
| 47 | | | $y((c_b, c_a)) := s$ | | Set new similarity. |
| 48 | | | send `[`$c_b, s$`]` | $c_a$ | Inform $E_Y$-edge partner about update. |
| 49 | 12 | $c_a$ | $\forall$ `[`$c_b, s$`]` received | | If $E_Y$-edge values were reconsidered. |
| 50 | | | $y((c_a, c_b)) := s$ | | Update respective $y$ value. |
| 51 | | | $A \leftarrow A \cup \{s\}$ | | Add value to global set. |
| 52 | | $\forall v \in V$ | GOTO step 1 | | Start next iteration with step 1. |

## 4.5 The LINDA System

The LINDA (LINked Data Alignment) system implements the different assignment algorithms of Sections 4.2 - 4.4 in combination with judiciously designed similarity functions. Note that due to the many domains Linked Data stems from it is highly challenging to build similarity functions that balance the precision among all domains under consideration. It is not within the scope of this thesis to explore the space of possible similarities, since this is an orthogonal project and subject to many discussions. Also, we consider precision to be highly subjective and assessable only with large and thorough user studies. In this thesis, we mainly focus on how to perform *distributed* and *joint* entity mapping. Nevertheless, in the course of this work, we found that the careful design and selection of similarity ingredients is a central part of an entity matching system and also determines the influence of contextual information. In the following, we propose one possible instantiation for our LINDA algorithms that we found to perform well for the diverse datasets we considered (described in Section 4.6).
In our implementation, LINDA computes two kinds of similarities between entities $a$ and $b$, namely a *prior similarity* $\text{sim}_0$ based on literals and constraints as well as a *contextual similarity* $\text{sim}_C$, which considers the current state of the assignment matrix $X$:

1. The *prior similarity* $\text{sim}_0(a, b)$ is a token-based similarity between literal sets $L(a)$ and $L(b)$. These literals typically include labels and descriptions, etc. This similarity is computed once beforehand and used to initialize the similarity matrix $Y$ and later as a smoothing prior (see Section 4.5.1). In theory, one computes prior similarities for all entity pairs. In practice, however, we use a token frequency cut-off, which (as a side-effect) reduces comparisons and employ Map/Reduce to distribute remaining computations. Intuitively, the prior similarity must produce a high-quality set of matching candidates that can be refined with further information.

2. The *contextual similarity* $\text{sim}_C(C(a), C(b), G, X)$ determines the similarity of the contexts of $a$ and $b$. Specifically, we consider all graph neighbors of $a$ and $b$ and also the equivalence classes among these neighbors, based on the current state of $X$. $\text{sim}_C$ is thus recomputed in each iteration (see Section 4.5.2). Intuitively, the contextual similarity degrades or reinforces matching candidate pairs discovered with the prior similarity.

The two similarity measures are combined to an overall *similarity score* $\text{sim}(a, b, G, X)$ as follows.

**Definition 4.6** (similarity score)
Given an entity graph $G = (V, E)$, an assignment matrix $X$, and two parameters $\alpha$, $\theta$, the similarity score for entities $a, b \in V$ is:

$$\text{sim}(a, b, G, X) = \text{sim}_0(a, b) + \alpha \, \text{sim}_C(C(a), C(b), G, X) - \theta.$$

□

$C(a)$ and $C(b)$ denote the contexts of $a$, $b$ and are defined later. The parameter $\alpha$ controls the influence of contextual similarities. $\theta$ is used to ensure that the overall scores are renormalized to values around 0, since positive scores shall reflect likely mappings and negative scores imply dissimilarities – as required by the objective function in Definition 4.4. Without such renormalization, i.e., with $\theta = 0$, the objective would encourage selecting as many mappings as possible, as long as the respective items have at least some very small probability of matching. Setting $\theta > 0$ ensures that there is a penalty for matching items that are too dissimilar. We empirically select the parameter values in Section 4.6.2.

### 4.5.1 Prior Similarities

Prior similarities $\text{sim}_0(a, b)$ reflect the direct evidence of two entities matching. Within the scope of this thesis we chose an intuitive function similar to the well-known Jaccard Coefficient based on literals. For the prior similarity definition we use the following functions:

- Given an entity $a$, $T(a)$ is 1 if $a$ is likely to be a class and 0 otherwise. We leverage $T(a)$ to avoid matching classes with non-classes.
- Given an entity $a$, $I(a)$ is `true` if a URI appears to refer to an information resource, e.g., a webpage, rather than a real-world entity. We use this to not consider information resources at all.
- Given an entity $a$, $N(L(a))$ provides the bag of word $n$-grams from all string literals $L(a)$ of $a$. Literals longer than 8 tokens are truncated to length 8. If an individual literal is shorter than $n$, we use the entire literal. Before forming n-grams, we apply word stemming, character normalization, and stopword removal. N-grams that occur more than $1,200$ times are removed, as they are unlikely to be discriminative (value has been chosen empirically). In case we deal with a label literal, we additionally add the entire label a fixed number of times to the bag $N(L(a))$ to give more weight to such literals. We determine labels through respective predicates, such as `rdfs:label`, `skos:prefLabel`, `foaf:name`, etc. For

all experiments presented in Section 4.6 we use 3-grams and add label literals 3 times.

- Given an entity $a$, the source $S(a)$ of $a$ is defined as part of the entity graph in Definition 4.1.

**Definition 4.7** (prior similarity)
Given two entities $a$ and $b$ and their bags of literal n-grams $N_a = N(L(a))$, $N_b = N(L(b))$, the prior similarity is defined as

$$\text{sim}_0(a,b) = \begin{cases} -\infty & \text{if } S(a) = S(b) \text{ or } T(a) \neq T(b) \\ & \text{or } I(a) = \texttt{true} \text{ or } I(b) = \texttt{true} \\ & \text{or } N_a \cap N_b = \emptyset \\ \dfrac{|N_a \cap N_b|}{\min(|N_a|, |N_b|) + \ln(|(|N_a| - |N_b|)| + 1)} & \text{otherwise.} \end{cases}$$

□

Note that because $\text{sim}_0(a,b)$ yields $-\infty$ for cases where $S(a) = S(b)$, $T(a) \neq T(b)$, $I(a)$ or $I(b)$, or $N_a \cap N_b = \emptyset$, we enforce to not match entities within a data sources, classes with non-classes, information resources, and entities that do not share a single n-gram, respectively. Unlike the well-known Jaccard set similarity measure, our prior similarity normalization is biased towards the size of the smaller bag and hence better accounts for data heterogeneity. This is important, because a frequent case for two matching entities is that one source contains only few literals, whereas another includes long textual descriptions.

### 4.5.2 Contextual Similarities

The contextual similarity $\text{sim}_C(C(a), C(b), G, X)$ of two entities $a$ and $b$, which depends on the current values stored in the assignment matrix $X$, is defined as follows.

**Definition 4.8** (entity context)
Given an entity graph $G = (V, E)$, the context $C(a)$ of an entity $a$ is a set of *context tuples* $(r, n, w)$, where $r$ is a relation, i.e., an edge label in the entity graph (and a predicate in the original RDF data), $n$ is a neighboring vertex of $a$ in the entity graph, and $w$ is a numeric weight. The context of an entity $a$ includes:

- $\{(r, n, w) \mid (a, r, n) \in E\}$, i.e., objects $n$ of triples with $a$ as subject,
- $\{(\texttt{inv:}r, n, w) \mid (n, r, a) \in E\}$, i.e., subjects $n$ of triples with $a$ as object. □

Here, `inv:`$r$ is used to represent the inverse relation of $r$. The weights $w$ of context tuples are computed as $\frac{1}{\log(\text{freq}(r,n))}$, where $\text{freq}(r,n)$ is the total number of occurrences of relation $r$ with entity $n$. Hence, we assign higher weights to less frequent context tuples, since, for instance, $(\texttt{bornIn}, \texttt{ParisFrance}, w_1)$ is less discriminative than $(\texttt{bornIn}, \texttt{ParisTexas}, w_2)$, as Paris, France is the birthplace of more people than Paris, Texas. With the definition of the context $C(a)$ of a vertex $a$, we can now define the contextual similarity.

**Definition 4.9** (contextual similarity)
Given an entity graph $G$, an assignment matrix $X$, and two entities $a$, $b$ with contexts $C_a = C(a)$, $C_b = C(b)$, the contextual similarity is

$$\text{sim}_{\text{C}}(C_a, C_b, G, X) = \begin{cases} \sum\limits_{\substack{(r_a,n_a,w_a)\\ \in C_a}} \max\limits_{\substack{(r_b,n_b,w_b)\\ \in C_b}} x_{n_a,n_b} \cdot w_a \cdot \text{sim}(r_a, r_b) & \textit{if } |C_a| \leq |C_b| \\ \sum\limits_{\substack{(r_b,n_b,w_b)\\ \in C_b}} \max\limits_{\substack{(r_a,n_a,w_a)\\ \in C_a}} x_{n_a,n_b} \cdot w_b \cdot \text{sim}(r_a, r_b) & \text{otherwise.} \end{cases}$$

□

Intuitively, this function finds matching pairs of context tuples and sums up their similarity values. Specifically, it first determines the smaller context set $C_s$ and then, for each tuple $\in C_s$, it sums up weighted similarities for the best matching context tuples from the other set. We choose to start with the context tuple matching with the smaller set, because the sum shall yield values $\leq |C_a|$ (given that tuple similarities are in $[0..1]$). The similarity of individual context tuples $(r_a, n_a, w_a)$, $(r_b, n_b, w_b)$ depends on the entity matching as well as the predicate similarity.
The entity matching considers the assignment matrix $X$ at position $x_{n_a,n_b}$, which expresses whether the respective entities $n_a$, $n_b$ are currently considered equivalent.
For the weighted predicate similarity $w_a * \text{sim}(r_a, r_b)$, we split the predicate names of $r_a$ and $r_b$ into words, remove stopwords, and perform stemming. This procedure yields two token sequences $s_a$, $s_b$ of length $l_a$, $l_b$, respectively. Then, we compute the distance $d(s_a, s_b)$ of $s_a$ and $s_b$ as the sum of the smallest edit distances for each word of the smaller sequence to words of the other sequence. The similarity $\text{sim}(r_a, r_b)$ is then computed as $1 - \frac{d(s_a,s_b)}{\max(l_a,l_b)}$.

## 4.6 Experiments

In the following, we present measurements for our LINDA system. Table 4.3 summarizes presented approaches and points to the respective evaluation section. We discuss algorithmic choices, i.e., the influence of the contextual similarity, as well as precision and

recall for our multi-core approach (referred to as LINDA1). To reason about accuracy we rely on manual assessments of result samples. However, we have also published accuracy results for the simple but prominent OAEI 2010 benchmark in [BMNW12] (Table 1). We do not consider precision and recall values for the distributed approaches (MR-LINDA and MP-LINDA), because accuracy is not the focus of this work. Instead, as part of this thesis, we demonstrate the scalability for the Map/Reduce- and the message-passing-based approach. In the following, we describe the creation of a sample from the Web of Data, called *LODsample*. Then, we quantitatively describe all datasets used in this section.

Table 4.3: Presented versions of the LINDA algorithm.

| Name (and shorthand) | Presentation section | Evaluation section |
|---|---|---|
| Multi-core LINDA (LINDA1) | 4.2 | 4.6.2 |
| Map/Reduce-based LINDA (MR-LINDA) | 4.3 | 4.6.3 |
| Message-passing-based LINDA (MP-LINDA) | 4.4 | 4.6.4 |

### 4.6.1 Data

To evaluate the algorithmic choices of our system on a real-world sample of the Web of Data, we created a dataset that, unlike existing instance matching datasets, consists of a variety of different URIs from heterogeneous domains. In particular, we created a list of seed URIs from diverse domains shown in Table 4.4, which we used as starting points for a depth-2 LOD crawl. Note that we deliberately chose seeds such that matches occur but made sure that the matching task is challenging by adding entities with similar labels, e.g., the city of Berlin, the poet Irving Berlin, and Mount Berlin.

With this LOD sample, we created two sets of manually labeled matching candidate pairs, named CAND1 and CAND2. The labeling was conducted by two annotators plus a third person for resolving conflicts. A pair is labeled as *true* if both entity URIs represent exactly the same real-world object. In contrast, *sameas.org* (a public aggregator for `sameAs` links) identifies, e.g., the movie The Godfather with the Godfather trilogy – a pair we would reject. The Fleiss' kappa scores [Fle71] for the annotations were 0.838 (CAND1) and 0.822 (CAND2). Entities from *CAND1* pairs stem from different datasets, share at least one bi-gram, and one of the two entities is from our list of seed URIs. CAND1 comprises 11,830 entity URI pairs out of which 30 are positive candidates (note that 11,279 include a *lod.geospecies.org* URI). This demonstrates that simple string similarity measures cannot achieve satisfying results. Thus, the detection of these few positive candidates is the actual challenge for a matching algorithm. *CAND2* pairs ful-

Table 4.4: Seed URIs used for the creation of a LOD sample.

| | |
|---|---|
| | The nobel prize winner Albert Einstein |
| 1 | http://dbpedia.org/resource/Albert_Einstein |
| 2 | http://data.nytimes.com/einstein_albert_per |
| 3 | http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000000417c |
| | The musician Anne Jackson |
| 4 | http://data.linkedmdb.org/resource/actor/47299 |
| 5 | http://dbpedia.org/resource/Anne_Jackson |
| | The band Boards of Canada |
| 6 | http://dbpedia.org/resource/Boards_of_Canada |
| 7 | http://dbtune.org/musicbrainz/resource/artist/69158f97-4c07-4c4e-baf8-4e4ab1ed666e |
| | The poet James Joyce |
| 8 | http://dbpedia.org/resource/James_Joyce |
| 9 | http://www4.wiwiss.fu-berlin.de/gutendata/resource/people/Joyce_James_1882-1941 |
| | the US state Georgia |
| 10 | http://dbpedia.org/resource/Georgia_%28U.S._state%29 |
| 11 | http://sws.geonames.org/4197000/ |
| | the country Georgia |
| 12 | http://dbpedia.org/resource/Georgia_%28country%29 |
| 13 | http://sws.geonames.org/614540/ |
| | The gangster movie The Godfather 2 |
| 14 | http://dbpedia.org/resource/The_Godfather_Part_II |
| 15 | http://rdf.freebase.com/ns/guid.9202a8c04000641f80000000000861a6 |
| | The city of Berlin |
| 16 | http://dbpedia.org/resource/Berlin |
| 17 | http://rdf.freebase.com/ns/guid.9202a8c04000641f80000000000094d6 |
| 18 | http://sws.geonames.org/2950159/about.rdf |
| | Irving Berlin, the musician |
| 19 | http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000005f5f6 |
| 20 | http://www.bbc.co.uk/music/artists/5e645519-a175-4fe0-9a9b-eb9dc9f506b5#artist |
| | A mountain called Berlin |
| 21 | http://dbpedia.org/resource/Mount_Berlin |
| 22 | http://rdf.freebase.com/ns/guid.9202a8c04000641f80000000004df425 |
| | The CDU politician Angela Merkel |
| 23 | http://dbpedia.org/resource/Angela_Merkel |
| 24 | http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000008480e |
| | The species Guinea Pig |
| 25 | http://dbpedia.org/resource/Guinea_pig |
| 26 | http://www.uniprot.org/taxonomy/10141 |
| 27 | http://lod.geospecies.org/ses/3KbUP (not accessible anymore) |

fill the same similarity criteria as CAND1 but must involve a DBpedia URI and cannot contain entities from *lod.geospecies.org*. These criteria ensure that we have a reasonable percentage of true positives. CAND2 contains 423 candidate entity pairs, 51 of which are positive examples. This dataset is more diverse than CAND1, because we no longer require one of the two entities to be from our seed set.

To judge the scalability of our approach we instead use much larger datasets. In particular, we augmented the Billion Triple Challenge Data (BTC) as of 2011 [Bil] with

major Web of Data hubs, namely DBpedia (version 3.6), Yago2, Freebase (as of January 2011), as well as GeoNames (as of mid 2011). The BTC collection is highly heterogeneous, including data generated by content management systems and data embedded into webpages. Each dataset used for the augmentation reasonably enlarges the entire set on the one hand but also ensures a high coverage of prominent entities in the LOD network.
Given the raw RDF triples, we first preprocess the data with Map/Reduce. In particular, we remove identical triples, blank nodes as well as reification statements and compute prior similarities. We then load the resulting entity graphs into MongoDB instances (version 2.2.2) [PHM10], which may physically reside on multiple machines and thus scale horizontally. Additionally, we determine context weights and predicate similarities – eventually also stored in MongoDB. For the largest dataset we considered (BTC+++ with 636 million unique triples, see below) the entire preprocessing took roughly 15 hours[58]. With this setup, we can run LINDA on a single machine or produce the input for the distributed versions. In detail, we created the following datasets:

- LODsample is the depth-2 crawl as described above.
  It comprises 1,210,596 triples and describes 443,863 entities.
  Its compressed size in the HDFS[59] is 188MB. The database size is 2GB.

- BTC+ is the union of the BTC data and DBpedia.
  It comprises 157,503,668 triples and describes 43,853,192 entities.
  Its compressed size in the HDFS is 27GB. The database size is 44GB.

- BTC++ is the union of the BTC data, DBpedia, Yago, and GeoNames.
  It comprises 288,332,370 triples and describes 70,082,459 entities.
  Its compressed size in the HDFS is 43GB. The database size is 90GB.

- BTC+++ is the union of the BTC data, DBpedia, Yago, GeoNames, and Freebase.
  It comprises 636,048,989 triples and describes 120,986,900 entities.
  Its compressed size in the HDFS is 92GB. The database size is 182GB.

Each of these datasets serves a different purpose: We use LODsample to evaluate algorithmic choices. We managed to run our message-passing-based LINDA on BTC+. We process BTC++ and BTC+++ with the multi-coreversion of our approach to judge the output accuracy. Also, the Map/Reduce version of LINDA easily processes BTC++ and BTC+++.

[58]The runtime refers to the Map/Reduce cluster described later in Section 4.6.3.
[59]HDFS is a shorthand for the Hadoop distributed file system.

Table 4.5: Most frequent preprocessed literal words in BTC+++.

| token | frequency | token | frequency | token | frequency |
|---|---|---|---|---|---|
| de | 2,001,030 | bibsonomi | 390,112 | section | 325,936 |
| song | 800,732 | river | 389,139 | born | 320,191 |
| school | 739,138 | district | 387,133 | forc | 315,885 |
| la | 727,343 | love | 383,565 | text/plain | 312,885 |
| descript | 710,045 | public | 383,101 | season | 311,704 |
| lyric | 622,029 | unit | 368,450 | application/rss+xml | 309,600 |
| metrolyr | 614,331 | nation | 368,232 | world | 306,906 |
| databas | 600,224 | inform | 362,726 | station | 302,938 |
| world | 584,034 | wikipedia | 356,366 | william | 298,127 |
| freebas | 580,106 | american | 353,227 | artist | 297,101 |
| source: | 576,005 | ¡a | 349,679 | creek | 291,348 |
| rdf | 530,333 | le | 344,312 | citi | 291,133 |
| state | 511,126 | transport | 339,434 | id | 290,401 |
| livejournal.com | 478,302 | el | 336,804 | full | 289,892 |
| user | 478,046 | type | 336,454 | ru | 289,425 |
| john | 477,706 | lake | 330,493 | link | 286,562 |
| act | 468,061 | counti | 329,049 | en | 276,642 |
| book | 418,718 | church | 328,747 | 1985 | 275,600 |
| includ | 413,886 | ca | 328,444 | episod | 273,206 |
| attribut | 412,446 | list | 326,193 | interest | 272,531 |

Table 4.5 depicts the most frequent literal tokens in the BTC+++ data after we applied our preprocessing described in Section 4.5.1. Note that the table shows *tokens* and respective frequencies. However, we work with *3-grams* with a maximum frequency of 1,200. Nevertheless, the table illustrates what kind of literal data our prior operates on.

Figure 4.3 illustrates the network structure of each dataset. In particular, it shows the number of entities with a given number of entities in their local neighborhood, i.e., the vertex out-degree. For LODsample, the maximum out-degree is $26,927$[60]. The second to the maximum is in the order of $2,900$. The mean and standard deviation values are 4.2 and 117, respectively. For the BTC+++ data, the maximum, mean and deviation values are 831770, 6.6, and 388, respectively. The latter numbers are slightly smaller but roughly hold for BTC+ and BTC++. In general, our LODsample shows comparable properties, e.g., very many low degree entities, etc. An exception is the range $[10-20[$, which has fewer entities than the range $[20-30[$. Apparently, we cannot cover all degrees of interconnectivity with the given size of our sample.
Figure 4.4 illustrates the number of matching candidates per entity. Here, we group entities by the number of candidate entities they have a positive prior $\geq 0.1$ with. For

[60]Not shown for readability.

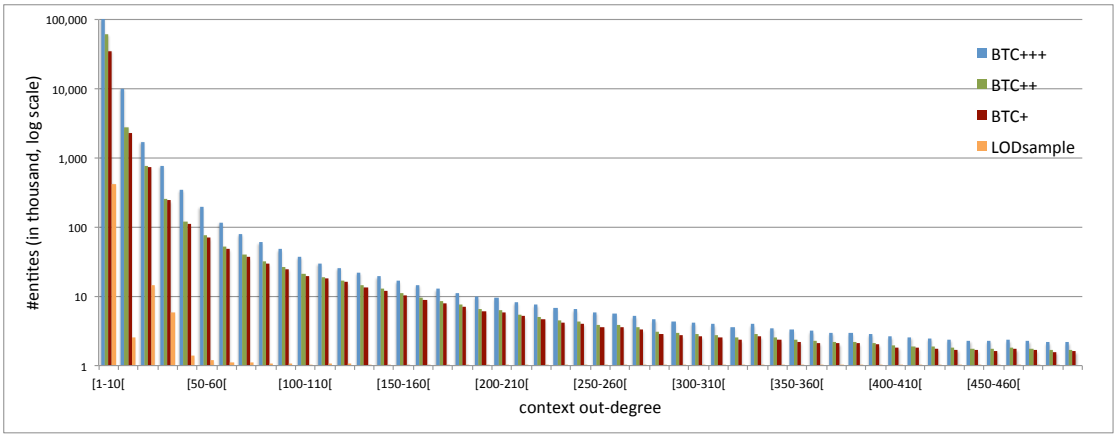


Figure 4.3: Number of entities (y axis) with respective context out-degree (x axis).

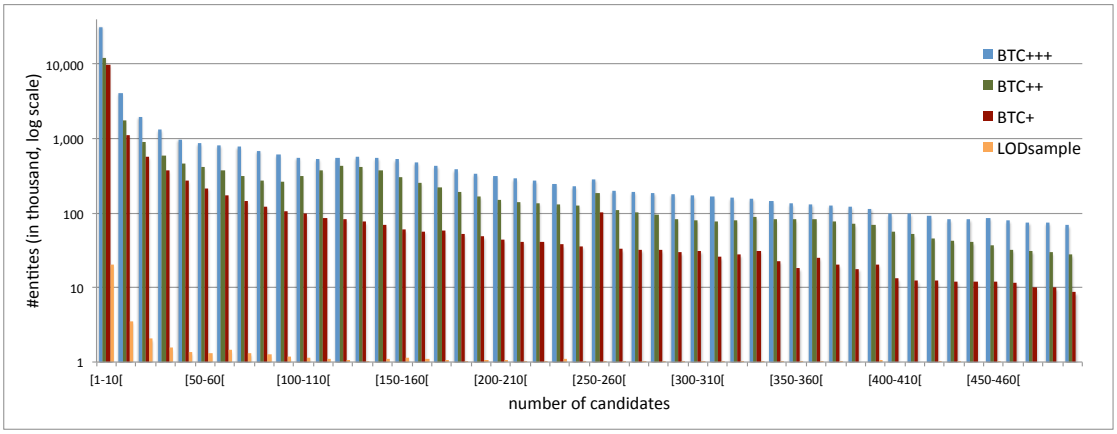


Figure 4.4: Number of entities (y axis) with respective number of matching candidates (x axis), i.e., entities such that there is a prior $\geq 0.1$.

BTC+++, the maximum number of matching candidates is in the range $[49,950 - 49,960[^{60}$. There is a remarkable number of entities (hundreds) with a number of candidates around $1,200^{60}$. Then, the number of entities slowly increases to millions for smaller numbers of candidates. Observe that there is a notable difference in the distribution for the various BTC datasets we use. That is, for instance, there are $336,933$ entities with 200 candidates in the BTC+++ data, whereas there are only $166,399$ and $47,442$ of such entities in the BTC++ and BTC+ data, respectively. In conclusion, since these candidates are mostly reflected in the queue used for the iterative processing of LINDA, it requires varying effort to update and reorganize the queue in the case of newly discovered entity alignments. We observed this effect, for instance, when processing the BTC++ or BTC+++ data with our first approach. Then, LINDA progresses slowly in

the beginning for both datasets due to initial matches for highly connected entities with very many candidates to be reconsidered. The processing accelerates after a certain set of entity pairs has been passed. However, the speed-up takes significantly longer for BTC+++ than for BTC++ data. Most likely, the higher mean number of candidates is one of the reasons.

### 4.6.2 Multi-core LINDA

**Algorithmic Choices.** We now evaluate varying settings for $\alpha$ and $\theta$ to suggest a setting that works well for Linked Open Data. For this, we ran LINDA1 on the LODsample data and determined precision and recall. Consider Table 4.6. The first result-row of the table depicts the setting we propose as default ($\theta = 0.7$, $\alpha = 2$) and shows that our system delivers high precision results at reasonable levels of recall. Below, we show the influence of individual parameters, i.e., results obtained by varying $\alpha$ and $\theta$ values. Overall, the algorithm is remarkably robust with respect to changing parameter values. As expected, higher $\theta$ leads to lower recall. The precision increases until $\theta = 0.7$ and remains stable until $\theta = 0.9$. However, we chose to suggest $\theta = 0.7$ as default to not sacrifice recall (though it remains stable for this particular dataset). Using $\theta$ fixed at 0.7 and varying $\alpha$, we find that $\alpha = 2$ performs best on CAND2. On CAND1, $\alpha = 1..3$ achieved the best results.

Running our algorithm using only the contextual similarities, we obtain high precision with very low recall with respect to CAND1 and no results for CAND2. Without contextual similarities, both precision and recall drop on CAND2, which highlights our joint assignment approach as opposed to conventional pairwise similarities. Note that the contextual influence, of course, highly depends on the strength of the prior similarity. That is, the better the prior similarity performs, the lower is the contextual influence. For instance, we found that other prior similarities, e.g., a *set* instead of a *bag* in Definition 4.7, cause a higher variance for the recall. Using conventional Jaccard scores with our system as a baseline, we find that we outperform the recall in any case and achieve a much higher precision for CAND2.

We also tested the influence of our context tuple weighting and found that uniform weights $w = 1$ instead of frequency-based weights cause a drop in precision and recall.

Finally, we compared our results with a snapshot of *sameas.org* (as of May 2011), which is a large aggregator of known collections of `sameAs` links gathered from numerous sources. It had a perfect precision on our dataset – although in general it contains numerous links that are not, strictly speaking, correct. However, its very low recall highlights the need for large-scale interlinking across the Web of Data.

Note that the results shown in Table 4.6 illustrate the performance for varying contextual influence ($\alpha$) and different renormalizations ($\theta$). Obviously, respective values depend on the interplay of both parameters and, of course, the choice of prior and contextual similarity functions. The expected precision/recall trade-off can be observed for the changing $\theta$ or by varying the number of accepted mappings from the queue (which essentially is varying $\theta$). In practice, however, there usually is a precision plateau from where higher thresholds not necessarily cause higher result confidence – in our case at $\theta = 0.7 \ldots 0.9$. For the results in Table 4.6 we processed the entire queue, i.e., all matching candidates with an initial similarity $> 0$. For experiments with much larger datasets, presented later in this chapter, we had to stop processing after a given number of iterations or after a specific amount of time.

**Accuracy.** Next, we show results for LINDA in its multi-core version on large datasets. The entire system is implemented in Java and uses a thread pool to parallelize the computation of updated scores $y^*$ (see the loop starting in line 12 of Algorithm 3). All LINDA1 experiments with large datasets were conducted on a single server with 80 virtual cores (Intel Xeon E7-4870 @ 2.40GHz) and 512GB RAM.

We ran the algorithm on BTC++ for six days and created roughly 716,000 mappings[61]. The BTC++ data queue had a length of 248,264,357 mapping candidate pairs. Remember that LINDA1 starts up slow due to the queue setup and since early matches cause many queue updates and reorganizations. These slow iterations take roughly 2 days. Then, LINDA produces more then 100,000 mappings per day (not requiring the total CPU power of the machine we had available).

The BTC+++ data queue had a length of 712,176,782. Here, setup and slow iterations take roughly 5 days. Remember Figure 4.4, which illustrates the tremendous number of candidates per entity. We experienced much faster start-up times when dealing with fewer candidate pairs per entity. For BTC+++ we created approximately 1,009,000 mappings in 19 days[61].

Table 4.7 illustrates the presence of sources among the resulting mappings[62]. Remember that the difference among BTC++ and BTC+++ is that the latter dataset additionally includes Freebase data (roughly 350 million triples / 50 million URIs more). For the BTC++ result, 60% of the mappings contain a DBpedia URI. Thus, DBpedia is most prominent in the result. The fraction of mappings with Yago is 53%, which is the second

[61] Note that the total runtime and thus the number of mappings created was determined by the time the machine was available. Therefore, we created this somewhat odd number of mappings. Also note that we used $\alpha = 1$ to equally weigh prior and context similarity for this experiment.

[62] Numbers refer to the first 500k mappings created.

Table 4.6: Algorithmic Settings.

| Variation | CAND1 | | CAND2 | |
|---|---|---|---|---|
| | Prec. | Rec. | Prec. | Rec. |
| LINDA (default: $\theta = 0.7$, $\alpha = 2$) | 0.71 | 0.46 | 0.93 | 0.49 |
| LINDA ($\theta = 0.15$, $\alpha = 2$) | 0.57 | 0.46 | 0.80 | 0.55 |
| LINDA ($\theta = 0.2$, $\alpha = 2$) | 0.60 | 0.46 | 0.82 | 0.55 |
| LINDA ($\theta = 0.3$, $\alpha = 2$) | 0.67 | 0.46 | 0.84 | 0.53 |
| LINDA ($\theta = 0.4$, $\alpha = 2$) | 0.67 | 0.46 | 0.84 | 0.51 |
| LINDA ($\theta = 0.5$, $\alpha = 2$) | 0.71 | 0.46 | 0.84 | 0.51 |
| LINDA ($\theta = 0.6$, $\alpha = 2$) | 0.71 | 0.46 | 0.86 | 0.49 |
| LINDA ($\theta = 0.7$, $\alpha = 2$) | 0.71 | 0.46 | 0.93 | 0.49 |
| LINDA ($\theta = 0.8$, $\alpha = 2$) | 0.71 | 0.46 | 0.93 | 0.49 |
| LINDA ($\theta = 0.9$, $\alpha = 2$) | 0.71 | 0.46 | 0.93 | 0.49 |
| LINDA ($\theta = 1.0$, $\alpha = 2$) | 0.62 | 0.31 | 0.93 | 0.49 |
| LINDA ($\theta = 1.5$, $\alpha = 2$) | 0.67 | 0.23 | 0.90 | 0.37 |
| LINDA ($\theta = 2.0$, $\alpha = 2$) | 0.67 | 0.23 | 0.93 | 0.27 |
| LINDA ($\theta = 3.0$, $\alpha = 2$) | 0.25 | 0.04 | 0.88 | 0.14 |
| LINDA ($\theta = 4.0$, $\alpha = 2$) | 0.25 | 0.04 | 0.50 | 0.02 |
| LINDA ($\theta = 0.7$, $\alpha = 1$) | 0.71 | 0.46 | 0.89 | 0.47 |
| LINDA ($\theta = 0.7$, $\alpha = 2$) | 0.71 | 0.46 | 0.93 | 0.49 |
| LINDA ($\theta = 0.7$, $\alpha = 3$) | 0.71 | 0.46 | 0.83 | 0.49 |
| LINDA ($\theta = 0.7$, $\alpha = 4$) | 0.63 | 0.38 | 0.73 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 5$) | 0.63 | 0.38 | 0.73 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 6$) | 0.59 | 0.38 | 0.69 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 7$) | 0.59 | 0.38 | 0.67 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 8$) | 0.63 | 0.38 | 0.67 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 9$) | 0.63 | 0.38 | 0.67 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 10$) | 0.59 | 0.38 | 0.67 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 15$) | 0.63 | 0.38 | 0.66 | 0.42 |
| LINDA ($\theta = 0.7$, $\alpha = 20$) | 0.63 | 0.38 | 0.63 | 0.39 |
| LINDA ($\theta = 0.7$, $\alpha = 25$) | 0.67 | 0.38 | 0.71 | 0.43 |
| LINDA ($\theta = 0.7$, $\alpha = 30$) | 0.53 | 0.31 | 0.59 | 0.31 |
| LINDA ($\theta = 0.7$, $\alpha = 40$) | 0.53 | 0.31 | 0.58 | 0.29 |
| LINDA ($\theta = 0.7$, $\alpha = 50$) | 0.53 | 0.31 | 0.58 | 0.29 |
| LINDA ($\theta = 0.7$, $\alpha = 100$) | 0.53 | 0.31 | 0.58 | 0.29 |
| LINDA ($\theta = 0.7$, $\alpha = 200$) | 0.53 | 0.31 | 0.58 | 0.29 |
| LINDA ($\theta = 0.7$, $\alpha = 1000$) | 0.53 | 0.31 | 0.58 | 0.29 |
| No Contextual Similarity | 0.75 | 0.46 | 0.91 | 0.41 |
| Only Contextual Similarity | 1.00 | 0.04 | 0.00 | 0.00 |
| Uniform context weights | 0.71 | 0.38 | 0.91 | 0.41 |
| Jaccard prior | 1.00 | 0.08 | 0.33 | 0.02 |
| *sameas.org* | 1.00 | 0.19 | 1.00 | 0.09 |

Table 4.7: Fraction of mappings from respective sources created with LINDA (in %). Underlined fractions refer to sets we used for the manual evaluation (see Table 4.8.)

| Source | BTC++ | BTC+++ |
|---|---|---|
| DBpedia to Any | 60.10 | 34.50 |
| DBpedia to Yago | 37.12 | 10.62 |
| DBpedia to Any but Yago | 22.97 | 23.88 |
| Freebase to Any | 4.95 | 73.50 |
| Freebase to DBpedia | 0.99 | 17.41 |
| Freebase to Any but DBpedia | 3.96 | 55.71 |

most prominent[63]. 37% of the mappings are among DBpedia and Yago. For BTC+++, most of the mappings include a Freebase URI (73.5%) and less than 20% of the mappings are among Freebase and DBpedia.

For the manual evaluation we chose to sample from the major fractions but also evaluate mappings specifically not among the most prominent sources in the results. Table 4.8 depicts the outcome of the manual evaluation of random samples from BTC++ and BTC+++ results[64]. For BTC++, we find a reasonable accuracy of 0.73 for randomly selected links. The percentage of true positives is 0.92 for links sampled from the BTC+++ result. To be critical, the latter result is partly caused by many links among the Wikipedia-based knowledge bases, namely Freebase, DBpedia, and Yago. Entities from these sources often have a large textual overlap and thus have high similarity scores. These high scores cause early positions of such pairs in the queue LINDA processes. Therefore, the none-source specific sample from the BTC+++ result does de-facto include source specific links.

The samples for DBpedia to Yago (BTC++) and DBpedia to Freebase (BTC+++) yield an accuracy of 0.96 and 0.92, respectively. For BTC++, DBpedia to Any achieves an accuracy of 0.63 – observe that this is the hardest matching task evaluated. Freebase to Any (BTC+++) achieves a higher accuracy of 0.96. Again, this is partly because it still contains mappings among Wikipedia-based sources. An evaluation for a complementary set of mappings, i.e., mappings among Freebase and any source that is not Wikipedia-based, yields an accuracy of $0.84 \pm 0.075$ (BTC+++).

With these results, LINDA clearly performs in the range of state-the-art matching approaches but without source-specific similarity measures, or highly-tuned domain-specific rules, etc. Instead, we chose a general similarity and exploit joint evidence for matches. Obviously, this choice gracefully balances accuracy among available sources without the introduction of custom dataset-specific fine-tuning and works well on a large scale.

[63] Not shown since the table refers to most prominent datasets.

[64] The manual judgement was done by a single person – who was not the author of this work.

Table 4.8: Accuracy for results on BTC++ and BTC+++. The size per sample is 120.

| Sample | Accuracy |
|---|---|
| BTC++ Any to Any Random Sample | 0.73 ± 0.09 |
| BTC++ DBpedia to YAGO Sample | 0.96 ± 0.03 |
| BTC++ DBPedia to Any but Yago Sample | 0.63 ± 0.10 |
| Total (sample size is 360) | 0.80 ± 0.05 |
| BTC+++ Any to Any Random Sample | 0.92 ± 0.06 |
| BTC+++ Freebase to DBpedia Sample | 0.92 ± 0.05 |
| BTC+++ Freebase to Any but DBpedia Sample | 0.96 ± 0.03 |
| Total (sample size is 360) | 0.94 ± 0.03 |

### 4.6.3 MR-LINDA

**Setup.** The distributed versions of the LINDA algorithm operate entirely on Hadoop. At this point the open source community deserves a special mention, since Hadoop, a wide-spread implementation of Map/Reduce, has been developed and pushed forward under the Apache umbrella, see `hadoop.apache.org`. We use Cloudera's CDH3u5, which is Hadoop version 0.20.2[65]. Our cluster consists of ten (one master, nine slaves) Dell PowerEdge R720 machines equipped with two Intel Xeon E5-2640 CPUs (12 virtual cores each), 64GB RAM, and eight internal 2TB SAS 7200 RPM HDDs. All cluster nodes run Debian Linux (Kernel 3.2.21.1.amd64-smp) and are connected via 10GBit ethernet. Slave nodes are configured to run up to ten map tasks and ten reduce tasks each in parallel. For MR-LINDA, the memory usage is limited to 4GB per task. MP-LINDA requires more memory (we allow up to 15GB) since each compute node loads a complete fraction of the graph into main memory. In either case, MR-LINDA and MP-LINDA, the map and reduce tasks represent compute nodes. That is, when we state to run LINDA on 20 nodes, it is executed as 20 map or reduce tasks. However, in any case, the input data is partitioned respectively and resides in the Hadoop distributed file system (HDFS). Then, due to locality, the data is mostly loaded from a local disk in a machine where a task is launched. The Map input split size is set such that a mapper consumes a complete graph partition. For instance, when processing the BTC+++ data on 60 nodes the input split size is set to 4GB whereas it must be 12GB when working with 20 nodes.

**Data Distribution.** Figure 4.5 illustrates the distribution of the queue for the BTC+++ data across compute nodes. It shows the average number of entity pairs on a single node

[65] `http://hadoop.apache.org`, `http://www.cloudera.com`

per similarity range for different numbers of nodes. Obviously, the more nodes we use, the more balanced is the distribution of queue similarity values. For instance, on 20 nodes there are 5.5 million pairs with values among $0.3 - 0.4$ and roughly 2 million pairs with values among $1 - 1.1$. Instead, on 60 nodes there are 1.8 and 0.7 million pairs for the respective ranges, i.e., the number of pairs in different ranges varies less – though, of course, the factor remains the same.

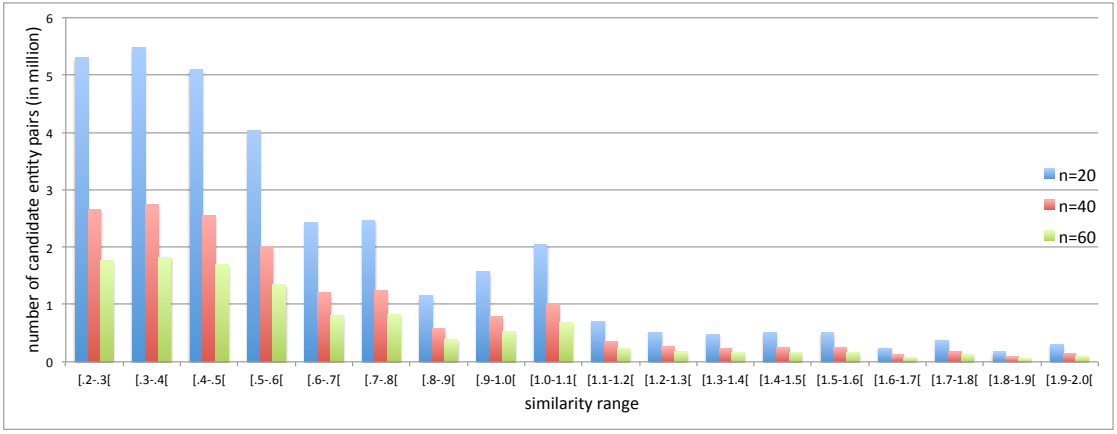


Figure 4.5: Average number of candidate entity pairs with respective initial similarity value in BTC+++ per queue partition for our Map/Reduce-based LINDA.

**Scalability.** Figure 4.6 and 4.7 show the number of created mappings vs. runtime for a varying number of compute nodes $n$ (20, 40, and 60) and different acceptance rates $K$ (50 and 100). In the first figure, we depict values for $K = 50$ over hundred iterations. Here, we observe that the cumulative runtime for $n = 20$ is significantly higher than for $n = 40$ or 60, which is because respective queue partitions scanned on each node in each iteration are significantly larger (see Figure 4.5). This is also the case for the graph partitions handled on each compute node. Additionally, there are fewer nodes processing updates and message traffic, which, in Map/Reduce, implies sorting large amounts of key/value pairs. Interestingly, the runtime for $n = 40$ and 60 behaves similarly (among the first 100 iterations). A possible explanation for this similar runtime is a trade-off between computation time split among compute nodes and message traffic causing physical network load increasing with more compute nodes involved.

As for the number of created mappings, we observe a slightly increasing growth for all values of $n$. This increase is caused by a lower bound for the cumulative number of created mappings in iteration $i$, i.e., $i * n * K$. The increase of the growth happens because when accepting mappings LINDA merges equivalence classes – and thus creates

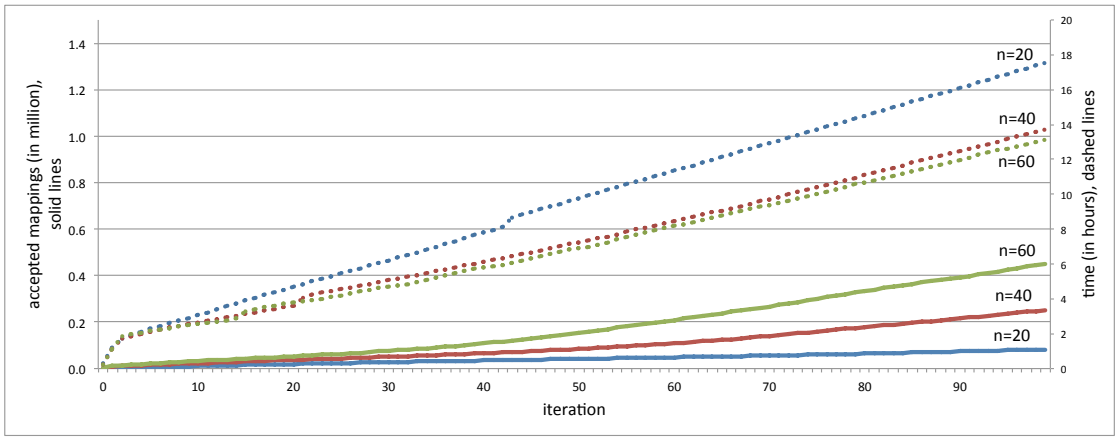


Figure 4.6: Total number of mappings created and time elapsed for our Map/Reduce-based LINDA on BTC+++ data. Acceptance rate $K = 50$.

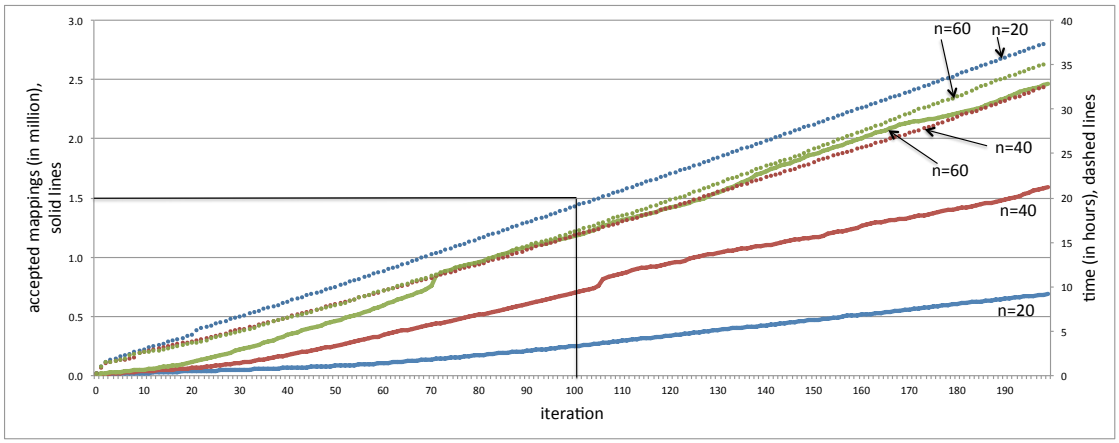


Figure 4.7: Total number of mappings created and time elapsed for our Map/Reduce-based LINDA on BTC+++ data. Acceptance rate $K = 100$. The marked rectangle is the range (number of mappings and time) shown in Figure 4.6.

more mappings. For instance, with 60 nodes in iteration $i = 100$ with an acceptance rate $K = 50$ MR-LINDA created $453k$ mappings in $13 : 10h$.

Figure 4.7 shows respective numbers for $K = 100$ over 200 iterations and $n = 20$, 40, and 60. After 200 iterations with acceptance rate $K = 100$, MR-LINDA created $688k$, $1,592k$, or $2,463k$ mappings in $37h$, $32h$, and $35h$, respectively. Note that $n = 40$ runs faster from iteration $i \approx 100$ (considering the absolute time per iteration, i.e., disregarding the number of created mappings). This observation supports the hypothesis discussed above and indicates that there is an ideal $n$ with respect to the time per iteration, which, of course, highly depends on the computing infrastructure. In our case,

all compute nodes run on 9 physical machines. We expect the most beneficial value for $n$ to be higher in the case of broader physical distribution. Note that our bulk acceptance strategy, i.e., we accept $K$ pairs from the queue per node and iteration, causes a large number of messages, i.e., key/value pairs, sent across the network. In case of broader distribution, fewer processes would send through a single network interface.
On the other hand, obviously, $n = 60$ produces more mappings – effectively leading to more mappings per time unit. Also, in other experiments running for approximately 500 iterations, we observed that the time per iteration eventually decreases (likely due to less message traffic) and thus the number of created mappings per time unit can further increase.

**Message Traffic.** Figures 4.8 through 4.10[66] discuss the update message traffic and respective queue updates actually performed. The first figure illustrates the number of messages sent to trigger similarity reconsiderations for entity pairs. Instead, Figure 4.9 and 4.10 depict the number of similarity recomputations that lead to changed similarity values in the queue. We show these values for $n = 20$ and 60 with $K = 100$. Figure 4.8 illustrates that MP-LINDA sends a large number of update messages across the network. In iteration two, the setting $n = 20$ and 60 cause the maximum of 7.2 and 7.6 billion messages, respectively. This message traffic also causes the longest runtime per iteration observed. However, message traffic slows down in the following iterations. For n=60 it remains fluctuating among 5 and 35 million messages per iteration. For n=20 the traffic levels out to below 5 million with occasional peaks.

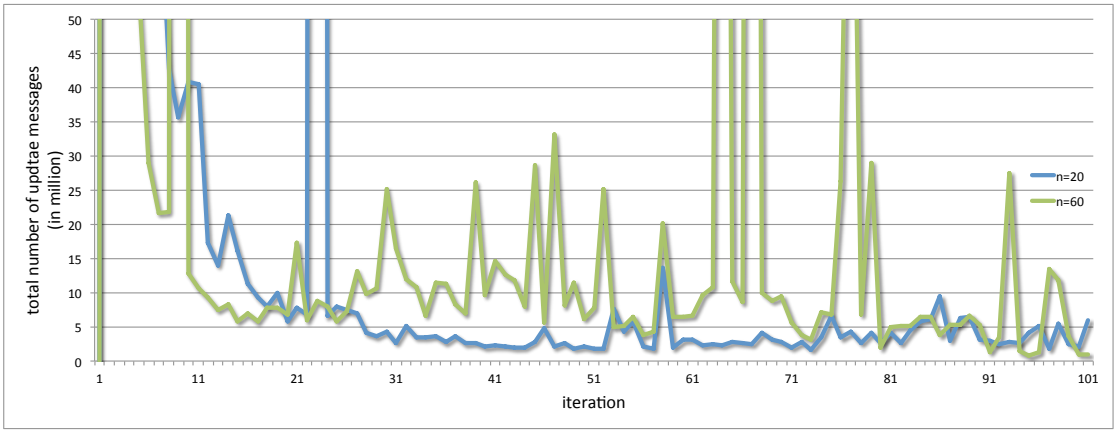


Figure 4.8: *Total* number of *update messages* sent during MR-LINDA processing on BTC+++ data. Acceptance rate $K = 100$.

[66] Y-axis in Figure 4.8 through 4.10 are cut-off for readability reason.

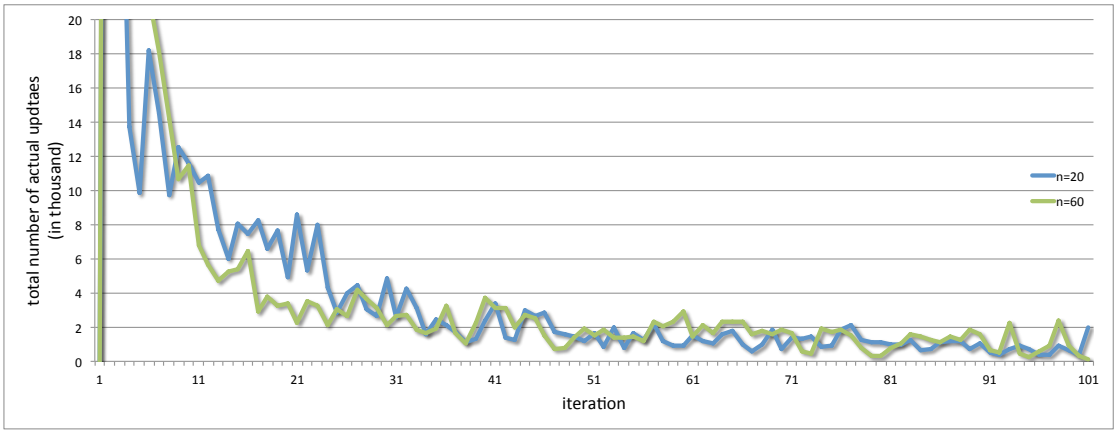


Figure 4.9: *Total* number of *actual updates* during MR-LINDA processing on BTC+++ data. Acceptance rate $K = 100$.

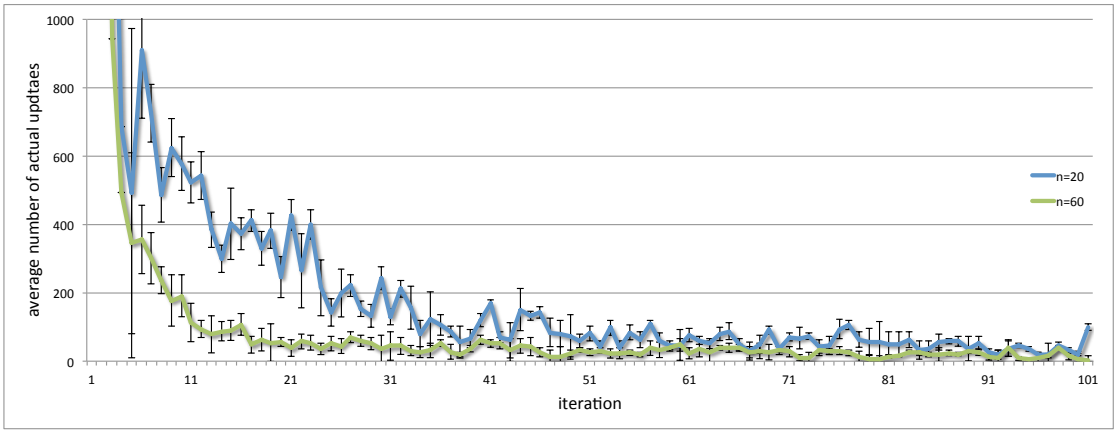


Figure 4.10: *Average* number of *actual updates* and standard deviation per compute node during MR-LINDA processing on BTC+++ data. Acceptance rate $K = 100$.

Note that these graphs should be compared with care, since in a given iteration both settings process different portions of the queue and thus handle other fractions of the entity graph. Nevertheless, as expected, we observe that $n$ not only increases the traffic but also shortens time between peaks.

In Figure 4.9 we consider *actual* updates performed and observe that the number of changed similarity values in the queue behaves similarly for both settings. After $n = 60$ starts out with 156,325 updates in iteration two, it levels out quickly. The setting $n = 20$ causes 136,373 updates during the second iteration and levels out more slowly, because it processes respective queue entries later than the other setting.

Figure 4.10 shows these numbers for a single node in the system (average and standard

deviation over 20 and 60 nodes). Clearly, on average a single node among 60 computes less actual updates than a node in a system of 20. Considering the standard deviation, however, we find that some nodes in either setting perform as many updates as nodes in the other setting.

**Unique Mapping Constraint.** Finally, we quantify the percentage of mappings violating the unique mapping constraint. Remember, these constraint violations occur since queue entry reconsiderations (including negative updates) require two iterations after a change of the current $X$. Figure 4.11 shows the percentage of mappings violating the uniqueness constraint for $n = 20$, 40, and 60 with $K = 100$. For the latter two settings, roughly 40% violate the constraint. With $n = 20$ this fraction is approximately 5 percentage points smaller. The dashed lines additionally show the size of the violating fraction, if we allowed three mappings per source, i.e., an entity could be mapped to three entities from a single other source. In this case, for iterations around 100, the values are roughly 15 percentage points smaller (even more for early iterations).
We see several options to avoid this undesirable effect. For one, we could add an efficient global coordination that registers $X$ updates and informs compute nodes on-demand. Second, smarter partitioning of the data could reduce the violations. Then, nodes were responsible for certain (not necessarily distinct) pairs of sources and it should not happen that distinct nodes map a single entity to entities from the same other source at the same time.
Another option is to perform additional post-processing that removes mappings until the constraint holds. For such post-processing, we are given a set $E$ of equivalent entities as well as respective acceptance scores $score(e, e')$ from the MR-LINDA algorithm, and we seek a partitioning $P_1...P_n$, such that the unique mapping constraint from Definition 4.3 holds in each partition $P_i$, $n$ is minimal, and $\sum_{P_i, 1 \leq i \leq n} \sum_{e_1, e_2 \in P_i} score(e_1, e_2)$ is maximal.
Given the source $S(e)$ for each entity $e$, there is a partitioning $S_1...S_m$ of $E$ by source. Let $|S_1| \geq ... \geq |S_m|$. Thus, $m$ is the maximal size of a partition $P_i$ since we have entities from $m$ sources in $E$. Also, the number of required partitions $n$ is $|S_1|$, since all entities in $S_1$ need to reside in a different partition. Then a simple greedy heuristic to build a partitioning is the following:

- Create $n = |S_1|$ new partitions $P_1...P_n$ with a single entity from $S_1$ each;
- For the remaining partitions $S_j = S_2...S_m$ and for $e \in S_j$,
- assign $e$ to $P_i$ where the gain $\sum_{e' \in P_i} score(e, e')$ is maximal and remove $P_i$ as option for further $e'' \in S_j$.

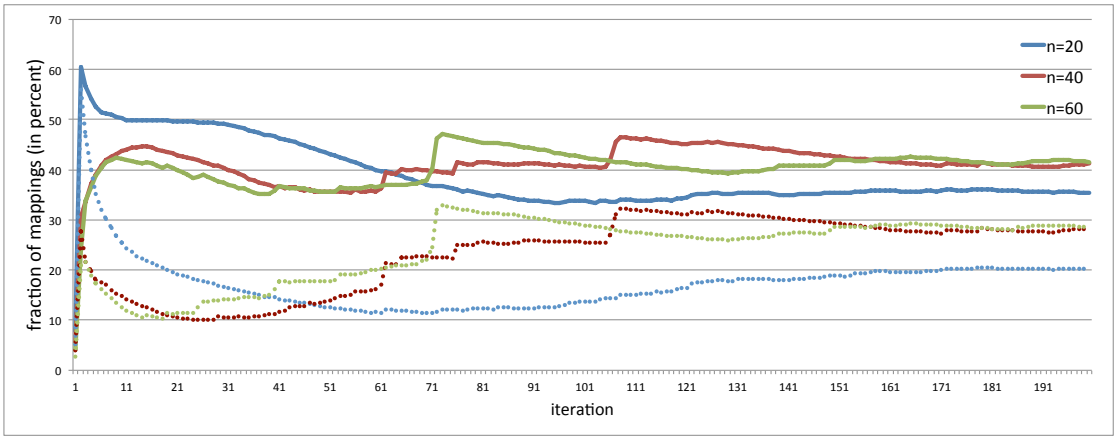


Figure 4.11: Percentage of mappings violating the optional unique mapping constraint from Definition 4.3 produced by our Map/Reduce-based LINDA on BTC+++ data. Solid lines refer to the constraint as it is. Dashed lines refer to the assumption that three mappings per source are acceptable. Acceptance rate $K = 100$.

A fourth option is to aim at a distributed implementation of LINDA that closely coordinates compute nodes' actions. We chose this option and implemented MP-LINDA, which we evaluate in the following.

### 4.6.4 MP-LINDA

**Setup.** Our third version of LINDA, MP-LINDA, has been implemented on top of *Giraph*[67] – a project donated by Yahoo! to the Apache Software Foundation. Giraph implements the graph processing model discussed in the Pregel paper [MAB+10] and aims to add fault tolerance by building on Hadoop[68] and using ZooKeeper[69] as a central coordination process. For our implementation we used a Giraph version as of early 2012. At this time, the project was in a very early state and lots of work on the internal memory management, the RPC framework and its usability was ongoing. Nevertheless, Giraph was the only open option to realize a message-passing-based implementation in Java – not least due to the nascent and very supportive community.

For our experiments we used the cluster described in the previous Section 4.6.3 and created a JSON dump of the BTC+ entity graph. During this process we limited the vertex out-degree to $1,000$ (sampled at random) since MP-LINDA passes vertex neighborhoods to matching candidates. We conduct this out-degree reduction to avoid serialization and deserialization of very large messages and heavy network load. We expect this reduc-

[67] `http://incubator.apache.org/giraph/`
[68] `http://hadoop.apache.org`
[69] `http://zookeeper.apache.org`

tion to not seriously affect the result for two reasons: (1) Hub vertices (with out-degree $\gg 1,000$) in the entity graph still comprise many relationships. (2) Vertices formerly connected to hubs only loose none-discriminative relations. Nevertheless, we are aware that the edge removal lowers the effect of our joint mapping approach. However, we expect more mature processing platforms and a highly-tuned implementation of MP-LINDA to be able to deal with larger out-degrees.

**Scalability.** Figure 4.12 illustrates the number of mappings created as well as reschedule messages sent over 150 iterations. In iteration 1 and 2 MP-LINDA creates 60 and 32 mappings, respectively[70], and it sends 28 and 11 reschedule messages. In the following, it creates $2-6$ mappings per iteration and rarely sends reschedule messages. We acknowledge the low number of created mappings and discuss this effect below. Note that this behavior is exactly the same for any number $n$ of compute nodes involved. This is because, compute nodes execute a specific function on each vertex. This execution is distributed but does not influence the outcome of the vertices, i.e., mapping decisions and messages sent.

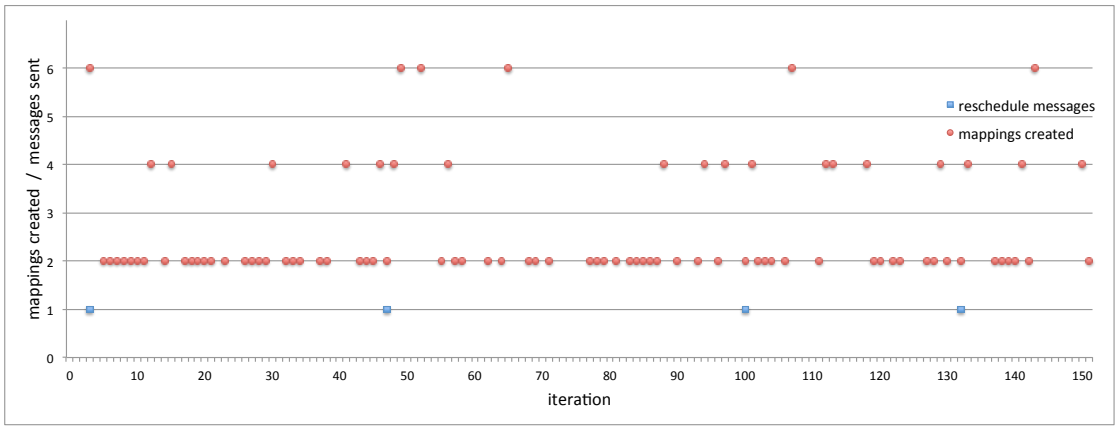


Figure 4.12: Number of mappings created and reschedule messages sent by MP-LINDA on $n = 20$, 40, and 60 nodes over iterations (values for different $n$ are identical by algorithm design).

Figure 4.13, however, shows the time per iteration required to execute *all vertices'* compute functions. As expected, this time differs largely for a varying number of compute nodes, e.g., it is roughly $4min$ on 60 nodes whereas it is $9-12min$ on 20 nodes. We are aware that these times are long for only very few mappings created. However, remember that we deal with the BTC+ data and thus handle almost 44 million vertices on 20, 40, or 60 compute nodes physically distributed on 9 machines. Though the machines

[70]Y-axis in Figure 4.12 is limited to 6 for readability reason.


Figure 4.13: Time per iteration required by MP-LINDA on $n = 20$, 40, and 60. Solid graphs represent measurements; dashed lines show estimated numbers.

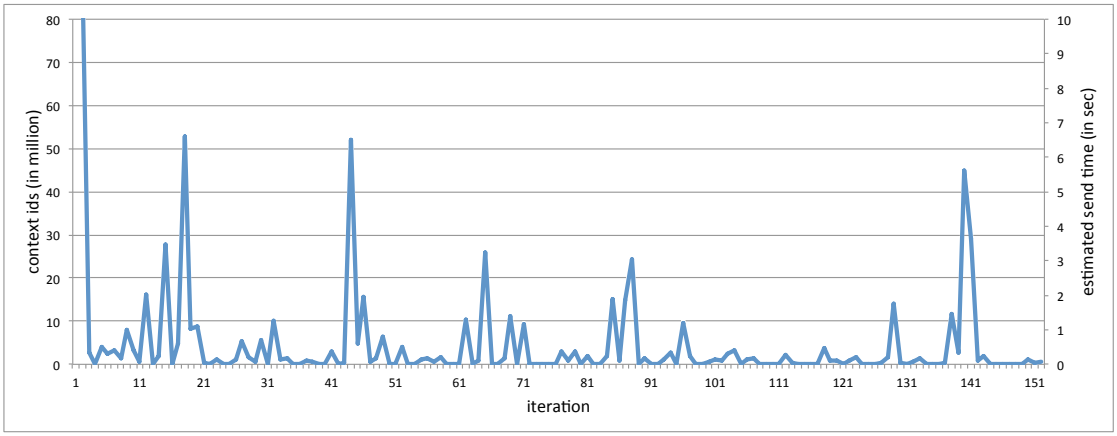


Figure 4.14: Number of context entity ids sent by MP-LINDA per iteration and estimated network transfer time over iterations. The primary y-axis (left) shows the number of ids; the secondary axis (right) is for the network time required.

are powerful and can easily cope with the respective number of processes in parallel, 44 million calls to the compute function add up to a sizable amount of time. Figure 4.13 additionally shows the *estimated* runtime for $n = 20$ and 60, given the respective other measurement. Specifically, given the number for $n = 20$ (solid blue graph), we estimated that iterations for $n = 60$ (dashed green line) would require roughly as long as they actually took (solid green line). In conclusion, given values for $n = 60$, we follow that the computation for all vertices with $n = 180$ would take approximately 1-1:15min (dashed orange line) plus some network communication overhead.

To quantify the network overhead, Figure 4.14 depicts the number of contextual neigh-

borhood entity ids sent across the network in each iteration. Again, the number differs largely – depending on the number of mappings created. In iteration one, there are 114 million ids passed to other vertices. Later, it ranges among $0.5 - 5$ million per iteration. On the secondary y-axis (on the right) we depict the estimated time to ship these ids through the network given 64 bit entity ids and assuming 50% actual throughput on gigabit ethernet, i.e., $1024^3/2$ bit/second. With this, we follow that the network traffic is acceptable (below 7 seconds) for all considered iterations. However, in our experiments, we observed longer runtimes besides the actual computation times shown in Figure 4.13. We attribute this to message serialization and deserialization as well as RPC communication overhead, etc. We expect these additional times to vanish, given a more elaborate system than Giraph in early 2012.

To further clarify the runtimes, Figure 4.15 shows elapsed times per node and step during the first 4 iterations. Green areas indicate small values up to 2 seconds; yellow fields indicate mid-range values; red spots emphasize the highest values. In general, the first two iterations take longer, which is because they generate more mappings then the following iterations. Also, the second step takes orders of magnitude more time than others. This is where MP-LINDA determines $y$-value edges, maximum ids, and sends messages to respective candidates as well as equivalents. Step 7 takes also longer then others due to the transfer of context equivalents to other vertices' context equivalents, i.e., the number of generated messages multiplies from Steps 4 through 7. However the latter effect disappears when fewer mappings are created in later iterations. For completeness, we refer the reader to the appendix where we additionally show times without message flushing overhead (Figure 7.1), i.e., spent in the compute function of the vertices only, as well as time deviations from average values (Figure 7.2 and 7.3). In Figure 7.1, we can observe active and none-active nodes and that the message creation in Step 7 is indeed the most time consuming procedure. We are aware that MP-LINDA messages should be implemented more efficiently.

## 4.7 Discussion

In this chapter we presented LINDA – the first fully automatic system for entity matching in the Web of Data that is able to scale far beyond the size of the Billion Triples Challenge dataset. We contribute an optimization model that formally captures the problem, show its hardness, and offer three implementations of a greedy strategy to iteratively build an optimal solution. Unlike previous approaches for Linked Data, LINDA is designed to operate on all LOD sources in their entirety, without source-specific customization. Our experiments demonstrate that the LINDA system successfully harnesses joint evidence

| Iteration | step | node 1 | node 2 | node 3 | node 4 | node 5 | node 6 | node 7 | node 8 | node 9 | node 10 | node 11 | node 12 | avg | max |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 3,680 | 4,101 | 3,739 | 2,935 | 12,294 | 3,289 | 2,852 | 2,363 | 2,939 | 3,229 | 3,355 | 3,160 | 3,690 | 12,294 |
| | 2 | 541,202 | 556,741 | 528,050 | 529,322 | 525,557 | 509,777 | 531,772 | 508,985 | 515,508 | 514,064 | 509,766 | 513,441 | 529,590 | 571,677 |
| | 3 | 1,699 | 1,745 | 1,767 | 1,623 | 1,708 | 1,680 | 1,818 | 1,990 | 1,643 | 1,807 | 1,821 | 1,711 | 1,744 | 1,990 |
| | 4 | 2,106 | 1,835 | 3,999 | 2,207 | 1,870 | 3,675 | 1,968 | 2,006 | 3,200 | 4,451 | 2,342 | 1,858 | 2,643 | 6,417 |
| | 5 | 2,479 | 2,627 | 2,454 | 2,770 | 2,713 | 2,479 | 2,584 | 2,545 | 2,670 | 2,763 | 3,027 | 2,858 | 2,613 | 3,027 |
| | 6 | 5,752 | 1,873 | 6,230 | 2,659 | 1,798 | 3,067 | 2,057 | 5,118 | 4,646 | 6,651 | 5,946 | 2,727 | 3,718 | 6,651 |
| | 7 | 258,814 | 261,385 | 262,589 | 263,458 | 253,253 | 258,522 | 269,247 | 247,437 | 258,700 | 259,522 | 293,099 | 275,705 | 266,911 | 308,904 |
| | 8 | 3,830 | 4,524 | 3,314 | 3,191 | 4,712 | 2,984 | 4,542 | 2,898 | 3,293 | 3,108 | 3,762 | 2,952 | 3,357 | 4,712 |
| | 9 | 11,300 | 44,838 | 11,360 | 5,471 | 11,229 | 13,105 | 53,670 | 53,813 | 13,009 | 13,079 | 6,063 | 5,452 | 16,322 | 53,813 |
| | 10 | 32,343 | 33,205 | 31,101 | 30,938 | 23,859 | 27,840 | 29,355 | 29,836 | 31,015 | 27,354 | 27,453 | 28,992 | 29,397 | 39,241 |
| | 11 | 5,482 | 5,885 | 5,616 | 6,620 | 6,618 | 5,852 | 6,818 | 6,881 | 6,879 | 6,351 | 7,323 | 6,350 | 6,371 | 7,323 |
| | 12 | 3,776 | 2,845 | 2,954 | 3,179 | 3,489 | 2,556 | 3,619 | 4,140 | 2,681 | 3,710 | 3,428 | 2,792 | 3,172 | 4,140 |
| 2 | 1 | 2,380 | 2,422 | 2,446 | 2,518 | 2,568 | 2,503 | 2,378 | 2,420 | 2,514 | 2,416 | 2,848 | 2,441 | 2,530 | 2,850 |
| | 2 | 546,984 | 560,157 | 523,338 | 515,278 | 520,669 | 506,198 | 518,383 | 499,717 | 510,699 | 511,787 | 502,519 | 502,365 | 520,016 | 568,538 |
| | 3 | 1,667 | 1,677 | 1,617 | 1,642 | 1,622 | 1,847 | 1,701 | 1,593 | 1,604 | 1,645 | 1,833 | 1,726 | 1,680 | 1,847 |
| | 4 | 3,114 | 3,133 | 1,816 | 1,957 | 1,788 | 1,628 | 1,775 | 1,791 | 1,831 | 1,789 | 2,123 | 1,866 | 1,978 | 3,133 |
| | 5 | 1,767 | 1,850 | 1,801 | 1,844 | 1,987 | 1,900 | 1,852 | 1,974 | 1,886 | 1,973 | 1,949 | 1,819 | 1,926 | 2,248 |
| | 6 | 1,904 | 1,904 | 1,801 | 3,999 | 1,783 | 3,684 | 3,555 | 1,818 | 7,986 | 5,721 | 2,724 | 2,247 | 2,990 | 7,986 |
| | 7 | 87,308 | 89,392 | 93,835 | 94,969 | 87,247 | 88,951 | 96,819 | 95,674 | 92,921 | 90,210 | 89,224 | 97,575 | 91,825 | 101,778 |
| | 8 | 2,093 | 2,791 | 2,173 | 2,242 | 3,766 | 5,407 | 2,133 | 2,285 | 2,442 | 4,624 | 2,833 | 2,274 | 3,020 | 5,407 |
| | 9 | 6,458 | 18,465 | 8,000 | 6,462 | 8,659 | 6,448 | 6,394 | 6,462 | 8,066 | 6,424 | 6,442 | 9,603 | 8,549 | 21,212 |
| | 10 | 15,757 | 22,194 | 19,498 | 20,150 | 16,412 | 19,436 | 17,793 | 20,384 | 19,546 | 17,363 | 20,758 | 20,520 | 19,209 | 23,824 |
| | 11 | 2,894 | 3,534 | 2,446 | 2,572 | 3,114 | 2,883 | 2,553 | 2,883 | 2,705 | 2,666 | 2,666 | 3,390 | 2,880 | 3,921 |
| | 12 | 2,067 | 2,133 | 2,123 | 1,924 | 2,094 | 2,153 | 2,028 | 2,073 | 2,110 | 2,150 | 2,190 | 1,937 | 2,106 | 2,266 |
| 3 | 1 | 2,639 | 2,759 | 2,564 | 2,402 | 2,617 | 2,396 | 2,624 | 2,689 | 2,533 | 2,539 | 3,097 | 2,635 | 2,612 | 3,097 |
| | 2 | 542,889 | 558,289 | 521,509 | 514,539 | 514,423 | 507,224 | 517,396 | 498,261 | 510,887 | 507,501 | 504,829 | 505,745 | 520,060 | 570,928 |
| | 3 | 1,664 | 1,729 | 1,672 | 1,762 | 1,717 | 1,644 | 1,787 | 1,698 | 1,598 | 1,581 | 1,888 | 1,838 | 1,760 | 1,978 |
| | 4 | 2,037 | 1,885 | 1,675 | 1,724 | 1,901 | 1,810 | 1,703 | 1,665 | 1,757 | 1,683 | 2,047 | 1,752 | 1,769 | 2,047 |
| | 5 | 1,604 | 1,620 | 1,581 | 1,617 | 1,683 | 1,618 | 1,989 | 1,963 | 1,808 | 1,875 | 1,854 | 1,695 | 1,722 | 1,989 |
| | 6 | 1,786 | 3,383 | 1,725 | 1,611 | 1,742 | 1,621 | 1,804 | 1,901 | 1,757 | 1,808 | 2,014 | 1,699 | 1,882 | 3,383 |
| | 7 | 22,828 | 19,461 | 28,569 | 28,362 | 20,395 | 15,407 | 23,210 | 21,452 | 27,319 | 18,188 | 20,358 | 22,328 | 22,579 | 28,569 |
| | 8 | 1,977 | 1,847 | 2,252 | 1,947 | 1,764 | 1,784 | 1,859 | 2,021 | 1,944 | 1,890 | 2,137 | 1,835 | 1,912 | 2,252 |
| | 9 | 2,579 | 2,577 | 2,576 | 2,600 | 2,584 | 2,590 | 2,579 | 2,579 | 2,611 | 2,624 | 2,589 | 2,587 | 2,632 | 3,429 |
| | 10 | 3,107 | 1,959 | 2,088 | 2,006 | 1,906 | 2,139 | 2,433 | 2,133 | 2,137 | 2,166 | 2,342 | 2,127 | 2,385 | 5,943 |
| | 11 | 1,805 | 1,719 | 1,805 | 1,886 | 1,873 | 1,849 | 1,910 | 1,723 | 2,002 | 2,046 | 2,074 | 1,862 | 1,892 | 2,139 |
| | 12 | 2,006 | 2,155 | 2,096 | 1,935 | 1,858 | 1,908 | 2,242 | 1,985 | 1,856 | 1,883 | 2,159 | 1,893 | 1,997 | 2,242 |
| 4 | 1 | 2,576 | 2,641 | 2,507 | 2,702 | 2,507 | 2,382 | 2,652 | 2,546 | 2,593 | 2,603 | 2,887 | 2,488 | 2,594 | 2,887 |
| | 2 | 541,170 | 557,535 | 518,373 | 518,876 | 518,702 | 509,320 | 516,202 | 499,454 | 512,668 | 510,041 | 503,297 | 503,807 | 519,788 | 570,014 |
| | 3 | 1,594 | 1,626 | 1,619 | 1,613 | 1,717 | 1,716 | 1,697 | 1,651 | 1,885 | 1,851 | 1,803 | 1,685 | 1,717 | 1,885 |
| | 4 | 1,691 | 1,973 | 1,884 | 1,641 | 2,015 | 1,750 | 1,804 | 2,927 | 1,772 | 1,701 | 1,981 | 1,920 | 1,875 | 2,927 |
| | 5 | 1,693 | 1,663 | 1,705 | 1,595 | 1,638 | 1,707 | 1,837 | 1,818 | 1,602 | 1,624 | 1,792 | 1,664 | 1,703 | 1,903 |
| | 6 | 1,953 | 1,751 | 1,642 | 1,855 | 1,759 | 1,920 | 1,876 | 1,791 | 1,830 | 1,753 | 2,091 | 1,749 | 1,797 | 2,091 |
| | 7 | 1,645 | 1,766 | 1,620 | 1,733 | 1,746 | 1,665 | 1,831 | 2,025 | 1,624 | 1,722 | 1,943 | 1,831 | 1,765 | 2,025 |
| | 8 | 1,628 | 1,819 | 1,629 | 1,746 | 2,334 | 1,705 | 1,915 | 1,838 | 1,757 | 1,866 | 1,826 | 2,014 | 1,896 | 3,361 |
| | 9 | 1,644 | 1,722 | 1,624 | 1,713 | 1,720 | 1,765 | 1,777 | 1,821 | 1,945 | 1,903 | 1,857 | 1,741 | 1,766 | 1,945 |
| | 10 | 1,709 | 1,835 | 1,882 | 1,736 | 1,949 | 1,724 | 1,903 | 1,825 | 1,792 | 1,741 | 1,787 | 2,001 | 1,817 | 2,001 |
| | 11 | 1,770 | 1,706 | 1,735 | 1,618 | 1,650 | 1,687 | 2,072 | 1,862 | 1,641 | 1,642 | 1,821 | 1,706 | 1,742 | 2,072 |
| | 12 | 1,921 | 2,005 | 1,964 | 2,041 | 1,938 | 2,015 | 1,997 | 2,105 | 2,034 | 2,082 | 2,213 | 1,819 | 2,006 | 2,213 |

Figure 4.15: Time per compute node and step in MP-LINDA (12 out of 20 nodes, in milliseconds).

and constraints, but also scales by making use of distributed architectures.

Each version of our system comes with varying properties, which we summarize in Table 4.9. Our first version, i.e., the multi-core approach, is bound by hardware resources and performs relatively slow, since it manages evolving similarities as well as the result on a single machine. However, it achieves good accuracy and delivers a consistent result. Therefore, it is the approach of choice if the amount of mapping candidates, which determine queue and result size as well as similarity reconsiderations per iteration, does not exceed critical hardware limits.

Our Map/Reduce-based approach is highly scalable at the price of violating constraints. Due to its scalability, it is the method of choice for very large datasets. Note that we managed to run LINDA1 and MR-LINDA on a dataset with roughly 120 million entity representations ranging from loosely to highly connected vertices with out-degrees in the thousands.

Finally, we successfully ran our message-passing implementation on a dataset with roughly 44 million entities. MP-LINDA delivers a consistent result through the addition

Table 4.9: Comparison of presented algorithms.

| | multi-core LINDA | MR-LINDA | MP-LINDA |
|---|---|---|---|
| runtime | slow | fast | medium |
| scalability | low | high | high |
| result quality | consistent | semi-consistent | consistent |
| potential | domain-independent use with few candidate pairs | very large datasets w. unique mapping negligible | large datasets on large clusters |

of close coordination among compute nodes and omitting the bulk accept strategy of MR-LINDA. However, to allow this coordination, vertices communicate in each superstep which requires a large cluster and a highly efficient message-passing implementation to physically distribute the many compute function calls (one per vertex and superstep). Note that this chapter presents previously unseen experiments on very large datasets that sometimes ran for days. In all experiments, we did not utilize pre-existing `sameAs` links as additional evidence for entity mappings. However, it is fairly straightforward to seed LINDA with such information during the $X$ matrix initialization and possibly improve precision and recall of our system. However, for the results presented in Section 4.6.2 we aimed at a stress test and avoided potentially false links and to prepare for cases where links are unavailable. We anticipate such cases as the Web of Data keeps growing and will include more dynamic datasets beyond the current dominance of relatively static reference collections.

The most promising, but also challenging, direction to enhance this work is to examine the entity graph distribution on the compute cluster. So far we distribute the graph using a fixed hashing but do not incorporate any smart strategy taking the entity connectivity or sources into account. However, smart data distribution could potentially avoid constraint violations in MR-LINDA but also reduce message traffic in either distributed implementation.

Also, for future work we envision to automatically generate RDFa for entities detected in news, blogs, and community forums and then to determine links among the newly discovered and existing entities. In such a scenario, LINDA needs to deal with data continuously generated in an online fashion.

To tune LINDA's accuracy we could incorporate any type of rules by simply adding them to the prior similarities or by manually assigning specific predicates a higher weight. In this manner, our algorithms can be made competitive with systems using fine-tuned dataset-specific similarity measures.

# 5 Concept Alignment

In the previous chapter we explored the problem of creating equivalence classes of entities from the Web of Data. These entities can be either instances or classes, i.e., we link different types of entities simultaneously and in a joint manner. In the following, we specifically target schema-level links across sources from the Web of Data. Schema-level links are links among concepts (also called classes, types, or vocabulary terms). The detection of these schema level correspondences is a well-established topic in the database community [BLN86]. In this chapter, we deal with the respective problem for Linked Open Data but restrict ourselves to correspondences among classes and do not match properties. Such links among these abstract entities are inherently important when querying across multiple LOD sources to allow for a connection of entities of different types (from different sources). Also, they could, for instance, be used to seed our LINDA algorithm and thus further improve LINDA's performance. Finally, a vision is to use such links among classes in combination with data source topics as discussed in Chapter 3, e.g., to detect overlapping topics across data sources, which requires schema-level links to allow for a comparison of topics, i.e., sets of types.

As of September 2011, the LOD cloud comprised 295 sources, which fulfill the basic LOD principles[71], i.e., dereferencable HTTP URIs that provide useful information. The majority of these 295 data sources (190) use proprietary vocabulary terms, i.e., source-specific class definitions. Out of these 190 sources, only 15 offer mappings to other widely deployed vocabularies. Nevertheless, 159 provide dereferencable URIs for their proprietary terms, i.e., descriptive information for these "new terms" are available.

This observation lead to the goal of utilizing such descriptive information and to develop a very fast concept alignment approach that can handle thousands to millions of concepts as a preprocessing for other methods. However, to ensure high-quality output, the alignment should still consider the data in its entirety, i.e., not simply perform pairwise local similarity computations. Also, we target an approach that can process content from the Web, which is heterogeneous in structure, granularity and size as well as completeness and quality.

In the following we present an approach, coined Holistic Concept Matching (HCM) [GBN12],

[71] `http://lod-cloud.net/state/`

that processes concept definitions only, i.e., it neglects instance data due to its immense size. Property definitions are ignored, since we found that their definitions and actual use differ largely across sources. HCM disregards ontological structure for the actual alignment creation (if available at all). This is for scalability reasons and since the structure across diverse ontologies varies largely and is thus not necessarily beneficial. However, HCM exploits structural information, such as subclass relationships, to verify derived alignments and find semantic contradictions. HCM's underlying concept alignment strategy is applicable to diverse domains.

## 5.1 Holistic Concept Matching

For HCM we adopted the well-known grouping strategy in order to achieve results very quickly. In particular, we group the input data by topic and thus create small groups of concepts that can be aligned individually. Through the grouping, we can process Web-scale input data in a few hours on a single machine. However, at the same time we target a holistic view on the data to leverage the information at hand as a whole. As we group by topics, we can still infer relationships holistically, i.e., draw conclusions not only based on a pairwise similarity but additionally rely on alignments and dependencies among other topically related members from other ontologies but in the same group.
Consider, for instance, five sources $(A, B, C, D, E)$ and let $\{a_1, \ldots, a_m, b_1, \ldots, b_n, c_1, \ldots, c_o, d_1, \ldots, d_p, e_1, \ldots, e_q\}$ be concepts from these sources. Running a traditional approach for pairwise ontology alignment would result in many isolated runs, e.g., $A-B$, $A-C$, $A-D$, etc. This is computationally expensive, because these alignment approaches often base on complex lexical and structural properties [CAS09, LTLL09, JMSK09]. Further, these approaches are not designed to process incomplete and highly heterogeneous data from the Web.
We employ *additional knowledge* to group respective entities. For instance, let the three sources $A, B, C$ store media related entities and $D, E$ provide information from the life sciences. We build groups of concepts, e.g., $\{a_1, \ldots, a_m, b_1, \ldots, b_n, c_1, \ldots, c_o\}$ and $\{d_1, \ldots, d_p, e_1, \ldots, e_q\}$ – essentially leading to fewer alignment candidates, e.g., $A-D$ concept pairs will not be considered. Note that these groups must not directly relate to the input ontologies. For instance, if $d_1 = human$, then it could also reside in the first group. However, within these groups, we can run computationally more complex approaches to take an holistic view based on multiple ontologies.
Our general approach, depicted in Figure 5.1, proceeds as follows: The input to HCM are concept information from the Web of Data, i.e, concept URIs, labels, descriptions, and structure (if available). For each concept, we extract a *knowledge representation*

capturing the meaning of a concept (upper part of Figure 5.1). This representation can be a simple descriptive string, a feature vector, or a more complex data structure. We then apply topical grouping based on this knowledge representation in order to create smaller sets of concepts (lower right of Figure 5.1). Within these sets, we then create alignments by identifying highly similar knowledge representations and by reasoning among respective concepts using additional structural information from the input as well as other candidates from the same group (lower left of Figure 5.1).
Note that many techniques can be plugged into this approach. Depending on the knowledge representation, one can choose the specific representation of a concept's meaning, topical similarities as well as concept similarities in order to find alignments. In the remainder of this work, we discuss the adoption of Wikipedia category forest [JHS+10] for HCM (Section 5.2). The topical grouping is done using a set similarity index [XWLY08] (Section 5.3). For the alignment generation we combine the Wikipedia category forest similarity [JHS+10] and a rule-based verification approach [JMSK09] (Section 5.4).

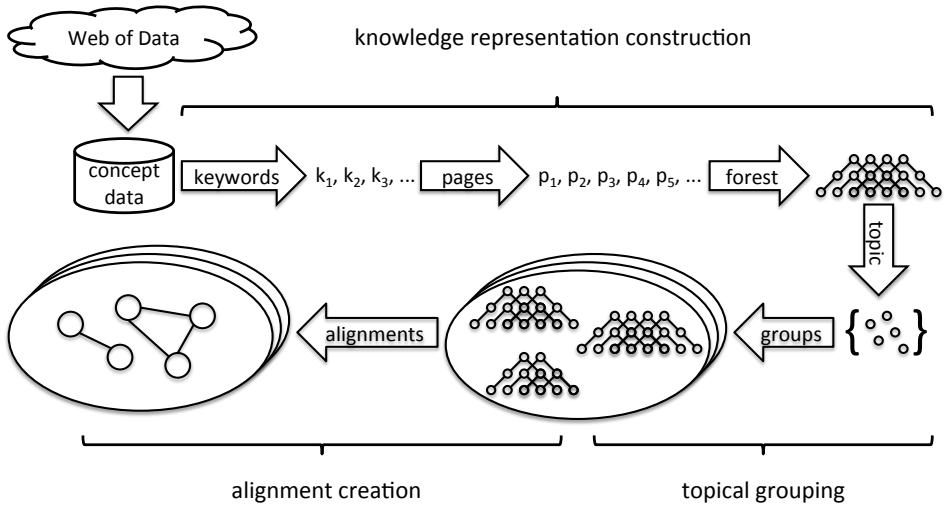


Figure 5.1: Holistic Concept Matching Workflow.

## 5.2 Knowledge Representation

For the comparison of different concepts we use an abstract representation of its semantic content, i.e., a so-called knowledge representation. Given a set $C$ of concepts from many different ontologies gathered from the Web of Data, we build such knowledge representation for each concept $c \in C$. To this end, we chose *Wikipedia category forests* (WCFs) as proposed for the BLOOMS algorithm in [JHS+10, JYV+10]. BLOOMS is a state-of-the-art alignment algorithm for matching heterogeneous ontologies from many

domains. However, BLOOMS is not meant to scale to the amount of data we target to process and it does not provide semantic verification and conclusions incorporated in the alignment creation process. A WCF is a set of Wikipedia category trees created as follows:

1. Given a single concept $c \in C$, create a list of keywords $kw(c) = \{k_1, \ldots, k_n\}$.
2. Given $kw(c) = \{k_1, \ldots, k_n\}$, a Wikipedia search for all keywords returns a ranked list of Wikipedia pages $R = \{p_1, \ldots, p_m\}$.
3. For each of the $d$ top-ranked pages $\{p_1, \ldots, p_d\}$ in $R$ and a height parameter $h$, we construct a Wikipedia category tree (discussed below in this section).

This process is also illustrated in the upper part of Figure 5.1. In summary, given a concept, we extract keywords from its URI and its description; with these keywords we run a Wikipedia search resulting in pages; for the top-ranked pages, we build trees, which form so-called Wikipedia category forests.

**Keyword Query Generation.** To determine keywords $kw(c)$ for a concept $c$, we use two orthogonal methods: The first uses tf.idf scored descriptions and the second leverages the class id, i.e., the concept URI.
The TFIDF$_n$-extractor consumes concept description, i.e., comments and labels. Specifically, we merge description texts, tokenize them, and neglect stop words. We then determine tf.idf scores for each token with respect to the overall corpus of concept descriptions. Finally, we select the top $n$ tf.idf ranked tokens as keywords $kw(c)$.
The ID-extractor, on the other hand, processes the concept's URI. This URI is the concept's unique identifier in its ontology definition. We use a straight-forward approach to determine a concept id suffix: We truncate the prefix of the URI until the last occurrence of either #, :, or /. The suffix must not contain any other character than letters or underscores. To extract tokens, we split the suffix at camel-case characters and underscores. After removing stop words, this results in the set of keywords $kw(c)$. For instance, the concept id for `http://umbel.org/umbel/rc/SpacePlatform_Manned` would be `SpacePlatform_Manned`. From this we create $kw(c) = \{space, platform, manned\}$.
The combination of both extractors, coined I/T$_n$-extractor, first applies the ID-extractor and then, if no WCF can be constructed (see next steps), it runs the TFIDF$_n$-extractor and returns tf.idf scored keywords from concept descriptions instead. Observe that this combination, the I/T$_n$-extractor, maximizes the amount of concepts that can be represented by a WCF.

**Wikipedia Keyword Search.** Next, we conduct a Wikipedia full-text search using a query with all keywords from $kw(c)$ delimited by spaces. From the resulting Wikipedia pages we choose the top $d$ pages as tree roots for the creation of a WCF. The parameter $d$, we call it the forest depth, influences a forest's coverage of conceptual meanings. That is, in order to catch all potential meanings of a concept, many Wikipedia articles should be considered. On the other hand, a deep forest might contain trees not related to the actual concept since the respective Wikipedia page is ranked too low. Thus, the selection of the forest depth parameter is a trade-off between semantic coverage and irrelevance.

**Wikipedia Category Forests.** For the forests construction, we employ the Wikipedia category hierarchy to build individual trees. Given the result $R$ from the Wikipedia keyword query, we build trees recursively from the $d$ top-ranked root pages $p \in R$ to the maximal tree height $h$ in the following recursive manner:

1. In recursion $1 \leq h$, for all $p_1, \ldots, p_d$, determine categories $\{\alpha_1, \ldots, \alpha_q\}$.
2. In recursion $2 \leq h$, for all $\alpha_1, \ldots, \alpha_q$, determine super-categories $\{\beta_1, \ldots, \beta_r\}$.
3. In recursion $3 \leq h$, for all $\beta_1, \ldots, \beta_r$, determine super-categories $\{\gamma_1, \ldots, \gamma_s\}$.
4. etc.

The height $h$ influences the level of abstraction of a WCF. The higher the trees, the more category hierarchy layers are considered, which leads to more abstract categories in the WCF. Instead, the lower the trees, the less probable is a topical overlap among related concepts. We illustrate different parameter configurations in the experiments in Section 5.5.
Consider the resource `umbel:MannedSpacecraft` as an example. The ID-extractor yields {`manned, spacecraft`}, which results in the following Wikipedia search result for $d = 3$: {`Human_spaceflight, Spacecraft, Orion_(spacecraft)`}. Figure 5.2 depicts the tree for the root article `Spacecraft` ($h = 2$). Obviously, the higher the tree layer, the more abstract Wikipedia categories the layer contains. Further, note that some tree nodes occur multiple times, e.g., `spaceflight` or `aerospace engineering`. These nodes play a significant role in the next phase described in Section 5.3.

## 5.3 Match Candidate Groups

Previously, we created our knowledge representation of choice, i.e., Wikipedia category forests – one for each concept. Given these knowledge representations, we now determine groups of topically related concepts. Similar to our definition in Section 3.1, a topic is

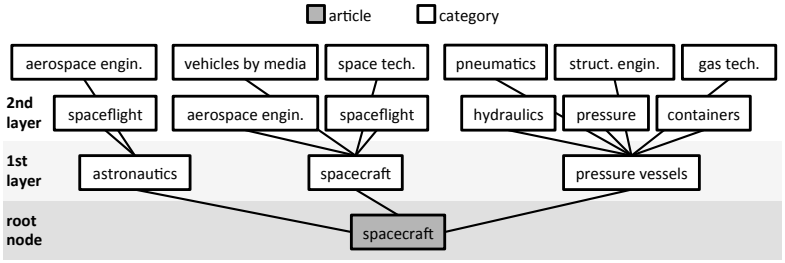


Figure 5.2: Wikipedia category tree for the Wikipedia article "Spacecraft" with $h = 2$. The dark gray rectangle represents the Wikipedia root page $p$. The 1st layer is highlighted in light gray. The white background indicates the 2nd layer categories.

a field of the real world wherein different concepts interplay. To determine concepts' interplay we examine respective knowledge representations. For the following, the underlying assumption is that a search for alignments among concepts within (and not across) topics does not cause a major loss of recall. On the other hand, the grouping by topic reduces the runtime of the alignment generation step by splitting the problem space into smaller independent alignment tasks. This procedure is often referred to as blocking or partitioning [Chr07].
Given a set $F$ of WCFs, we aim to identify disjoint groups $G_1, \ldots, G_n \subset F$ of topically related forests to allow for an independent and parallel processing. That is, we want groups, such that $G_i \cap G_j = \emptyset \ \forall \ i \neq j, 1 \leq i, j \leq n \ \wedge \ \bigcup_{i=1\ldots n} G_i = F$. In short, we implemented the following procedure:

1. For each WCF $f \in F$, extract the $topic(f) = \{t_1, \ldots, t_m\}$. We define a topic of a WCF as a set of selected WCF tree nodes.

2. Given a topic $topic(f_1)$, identify all forests $f_i \in F$ with a high topical overlap.

3. Given pairs $(f_1, f_i)$ with a high topical overlap, form groups of forests by associating topically overlapping forests transitively.

**Topic Extraction.** The topic $topic(f)$ of a WCF $f$ is a subset of its tree nodes, i.e., selected elements from the different tree levels $\{p, \alpha_1, \ldots, \alpha_q, \beta_1, \ldots, \beta_r, \ldots\}$. We aim at a selection of tree nodes that disregards highly specific categories like `Soviet manned space program` and very generic categories like `Humans` or Wikipedia article lists, because these would represent a very generic or too detailed view of the concept's topic. We employ a simple yet promising approach for the topic extraction, i.e., we again utilize the tf.idf measure. Nodes common in a specific tree are scored high due to the high term frequency value, whereas nodes popular in many trees are scored low due to the inverse document frequency. Observe that with this scoring scheme, highly specific nodes in

trees do not necessarily receive a high score, unless they occur multiple times in a forest. We rank a WCF's tree nodes with respect to tf.idf and select the top $m$ nodes for representing the topic. The choice of $m$ influences the quality of the tree node set representing the concepts' topics. Experiments indicated $m = 10$ to be a reasonable setting. Higher values lead to very specific topics, whereas smaller $m$ lead to low representativity.

**Topic Comparison and Group Creation.** Next, we compare the topic sets, i.e., sets of WCF tree nodes, to identify topically related forests. We use the Jaccard coefficient $J$ and a corresponding threshold $\theta$ to determine the similarity of topic sets: $J(topic(f_1), topic(f_2)) \geq \theta$. However, the pairwise comparison of all forest topic sets leads to a quadratic runtime with respect to the number of forests $|F|$ (in our experiments discussed in Section 5.5, $|F|$ is roughly $300k$). To reduce the runtime, we apply a set-based similarity self-join technique. In particular, we selected *ppjoin* [XWLY08], which delivered the most promising results (see Section 5.5).
Ppjoin performs a set similarity self-join. Internally, it reduces the amount of pairs to be considered with an inverted index and two filtering techniques on the candidate sets, i.e., prefix and positional filtering[72].
With the topic comparison, we can now determine concept groups. Remember that each concept has a knowledge representation, i.e., a Wikipedia category forest. In the following we use the terms concept and WCF (forest for short) interchangeably.
To determine groups of topically related WCFs, we form the transitive closure of topically related forests. In particular, given a set $F$ of Wikipedia category forests, we assign the forests $f_1, f_2$ to the same group if $J(topic(f_1), topic(f_2)) \geq \theta$ or there is a group with $topic(f_\alpha), topic(f_\beta)$ and $J(topic(f_1), topic(f_\alpha)) \geq \theta \wedge J(topic(f_\beta), topic(f_2)) \geq \theta$. Then, each transitive closure of forests refers to a group of concepts. We reason about $\theta$ later in Section 5.5.

## 5.4 Concept Alignment

At this point we operate on groups of topically related Wikipedia category forests, where each forest refers to a single concept. Note that we can process these groups individually and can thus easily run the concept alignment in parallel. Also, given the relatively small groups, we can incorporate additional ontological knowledge from the sources. We now describe the concept alignment within each group – a domain-independent iterative process including alignment acceptance and verification. In short, we proceed as follows:

[72] For the ppjoin, we adapted the open source code available at `https://code.google.com/p/similarity-join-tools/`.

1. We extend a group $G$ of topically related concepts with other related WCFs. These additional relations stem from the original ontology definitions.

2. We then compare all pairs of forests $f_1, f_2 \in G$ leveraging a forest overlap scoring function $O(f_1, f_2)$. Given a threshold $\rho$, we select all forest pairs with a respective overlap above the threshold, i.e., $O(f_1, f_2) \geq \rho$, and add these pairs to an alignment candidate set $M$.

3. Finally, we create a semantically consistent alignment graph $D$ by iteratively adding alignment candidates from $M$. With each addition of an alignment, we test a set of conditions to avoid contradicting relations. On the other hand, we additionally add relations concluded from the newly accepted alignment.

**Topical Group Extension.** To incorporate immutable axioms from the underlying source ontologies, we first extend each topical group $G$ with *further related WCFs*. A WCFs $f_2$ is related to $f_1 \in G$ if the underlying concept of $f_2$ has an ontology-level relation to the concept of $f_1$. These relations include `owl:equivalentClass` and `owl:disjointWith` as well as `rdfs:subClassOf` relationships. We restricted our selection to these relationships since we consider them to be most relevant to reveal alignment conflicts while keeping the group size low (which is a desirable property). In the following, the group $G$ refers to the *extended* topical group.

**Wikipedia Category Forest Comparison.** For the comparison of forests within groups, we employ the tree similarity introduced by Jain et al. [JYV+10]. Given arbitrary trees $t_1, t_2$ from different WCFs, the tree overlap determines common nodes and is defined as follows:

$$Overlap(t_1, t_2) = \frac{\log \sum_{v \in (t_1 \cap t_2)} \left(1 + e^{d(v,t_1)^{-1}-1}\right)}{\log 2|t_1|}$$

Here, $d(v, t)$ is the distance of a node $v$ to the root in a tree $t$. Thus, the tree overlap depends on the distance $d$ of common nodes $v \in (t_1 \cap t_2)$ to the tree root. The higher the depth of a shared node, i.e., the more nodes are between the node and the root page, the smaller the influence of the shared node. The depth of the root page $p$ is set to $d(p, t) = 1$.
Observe that tree overlap is not symmetric: $Overlap(t_1, t_2) \neq Overlap(t_2, t_1)$. The authors of BLOOMS+ use this asymmetry to identify parent-child relationships among concepts. They induce concept equivalence if $Overlap(t_1, t_2) = Overlap(t_2, t_1)$.
We extend the tree overlap to a similarity among category forests. For this, we compare all pairs of trees and select the best matching pair. Given two WCFs $f_1$ and $f_2$ from a

group $G$ with $f_1 \neq f_2$, we define the forest similarity as the maximal harmonic mean of the overlaps among all tree pairs $(t_1, t_2)$ with $t_1 \in f_1, t_2 \in f_2$:

$$O\left(f_1, f_2\right) = \arg \max_{\substack{t_1 \in f_1, \\ t_2 \in f_2}} \left( \frac{2 \cdot Overlap(t_1, t_2) \cdot Overlap(t_2, t_1)}{Overlap(t_1, t_2) + Overlap(t_2, t_1)} \right)$$

For the selection of alignment candidates, i.e., the set $M$, we use a minimum overlap threshold $\rho$. That is, $M$ contains all pairs $(f_1, f_2)$ with $O(f_1, f_2) \geq \rho$. A reasonable threshold is essential for the alignment quality, because otherwise irrelevant candidates lower the precision. Also, the following semantic reasoning might not be able to eliminate erroneous candidates. Additionally note that the total runtime of the semantic verification grows with the number of candidates to be considered. Of course, too few candidates leads to a low recall. In Section 5.5 we evaluate different thresholds and their influence on the alignment quality.

**Alignment Graph Creation.** From the previous step, we have a set of scored match candidates $M$. In the following, we create the alignment graph $D$, which captures the final result. We gradually build this graph by adding existing relations inherent in the source ontologies and augmenting it with highly-scored, none-contradicting relations from $M$.

**Definition 5.1** (alignment graph)
Given a group $G$ of WCFs, an alignment graph $D = (V, E)$ is a directed edge-labeled graph where the forests in $G$ are the nodes, i.e., $V = G$, and $E$ includes relations among forests representing concepts. The functions *cert*, *type*, and *origin* capture edge properties.

| | | |
|---|---|---|
| $cert:$ | $G \times G \rightarrow [0, 1]$ | denotes the alignment **certainty**. |
| $type:$ | $G \times G \rightarrow \{equi, disj, parent, child, onto\}$ | captures the **type** of a relation. |
| $origin:$ | $G \times G \rightarrow \{def, detect, infer\}$ | provides the **origin** of an edge. |

□

Intuitively, the higher the *certainty* of a relation, i.e., the *cert* value, the more reliable it is. The *type* label differentiates edges, i.e., an edge $e$ with $type(e) = equi$ connects two concepts identified to be equal. Instead, $disj$ marks edges connecting dissimilar concepts. Edges marked with *parent* or *child* stem from `rdfs:subClassOf` relationships. *Onto*-labeled edges indicate concepts to originate from the same source ontology. The *origin* of a relation captures what produced the edge. A $def$ edge is an axiom found in the source ontology. An edge $e$ with $origin(e) = detect$ denotes a verified alignment of two concepts with $O(f_1, f_2) \geq \rho$. An edge labeled with $infer$ indicates an edge transitively concluded from other alignments.

To populate the alignment graph $D$ with edges, we perform the following steps:

1. Initialize the alignment graph $D$ with $E = \emptyset$.

2. Add relations $e$ among WCFs for concepts originating from the same ontology with $type(e) = onto, cert(e) = 1.0$, and $origin(e) = def$.

3. Populate $D$ with ontology relation edges $e$ labeled $cert(e) = 1.0$ and $origin(e) = def$ derived from `subClassOf`, `equivalentClass`, and `disjointWith` triples from the underlying RDF data. Mark these edges with respective *type* labels, i.e., *child/parent*, *equi*, or *disjoint*.

   - For each added ontology relation $e = (f_1, f_2)$, add the inverse $e^{-1} = (f_2, f_1)$ with equal $origin(e^{-1})$ and $cert(e^{-1})$. The $type(e^{-1})$ equals $type(e)$, unless $type(e) = child$ or $type(e) = parent$. Then, add $type(e^{-1}) = parent$ or $type(e^{-1}) = child$, respectively.
   - For added relations $e, e^{-1}$ and existing edges $e' \in E$, include further relations $a$ derived from $E \cup \{e, e^{-1}\}$. Figure 5.3 illustrates additional conclusions we draw. Subfigure (a) shows an equivalence closure for $type(e) = type(e') = equi$; (b) shows the closure for an existing disjoint edge, i.e., $type(e') = disj$, and an added equivalence $e$, which leads to $type(a) = disj$; similarly, (c) depicts a disjoint closure for added *disjoint* relations; (d) and (e) show *parent* and *child* closures for $type(e), type(e') = parent$ or *child*, leading to $type(a) =$ *parent* or *child*, respectively.

4. For each candidate alignment $e = (f_1, f_2)$ in $M$ sorted (desc.) by forest similarity values $O(f_1, f_2)$, examine semantic conditions. If there is no semantic contradiction, add $e$ to $E$ with $cert(e) = O(f_1, f_2)$, $type(e) = equi$, and $origin(e) = infer$. The detection of semantic contradictions uses the current state of $D$ and determines whether the alignment $e = (f_1, f_2)$ contradicts an alignment $e' = (f_1, f_2)$ in $D$. That is, $type(e) \neq type(e')$, $origin(e') \neq infer$, or $cert(e') \geq cert(e)$. Additionally, we test for *multiple-entity correspondences*, *crisscross correspondences*, and *disjointness-subsumption* contradictions originally introduced as part of the ASMOV ontology alignment approach [JMSK09].

   - For each added alignment $e = (f_1, f_2)$, further add the inverse $e^{-1} = (f_2, f_1)$ with same $type()$, $origin()$, and $cert()$ values to $E$.
   - For added $e, e^{-1}$ and existing edges $e' \in E$, include further relations $a$ derived from $E \cup \{e, e^{-1}\}$ (as discussed above and illustrated in Figure 5.3).

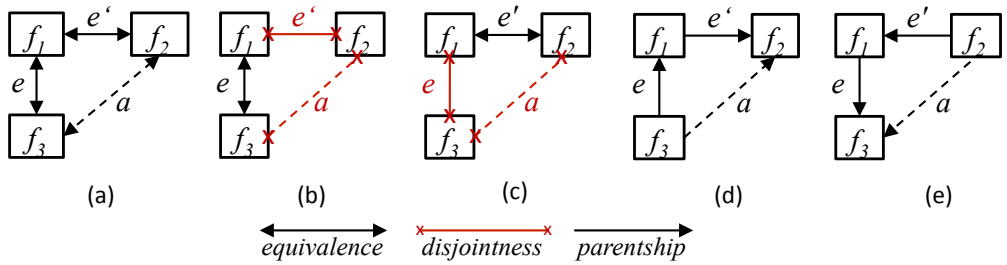


Figure 5.3: Five types of alignment conclusions, which infer a closure $a$ from an additional relation $e$ and an existing edge $e'$ in the alignment graph.

In summary, the previous process first populates the alignment graph with ontological constraints and then adds alignment candidates in a greedy manner. During the process, we check the consistency of the graph and additionally add inverse relations as well as relations concluded from the new state of $D$.
In the resulting alignment graph $D$, edges $e$ labeled $origin(e) = detect$ and $type(e) = equi$ form the result of our approach.

## 5.5 Experiments

**Setup.** All HCM experiments ran on a Windows 2008 R2 Enterprise Server with two quad-core processors (Intel Xeon X5355 @ 2.66GHz) and 32GB of RAM. To evaluate the performance of our approach in a Web-scale scenario, we used the Billion Triples Challenge data as of 2011 [Bil] (BTC). Our HCM approach is implemented in Java.
The extraction of conceptual information from the data can be done in a single pass over the data. We selected concepts, relations, and textual descriptions by considering resources occurring in triples with the following predicates: `rdfs:type`, `rdfs:subClassOf`, `rdfs:domain`, `rdfs:range`, `owl:equivalentClass`, `owl:disjointWith`, `rdfs:label`, and `rdfs:comment`. With this selection, we retrieved approximately one million concepts from the BTC data.
In contrast to BLOOMS [JHS+10], we cannot use the Wikipedia service, because we issue very many searches as we process thousands of concepts. The many online searches would thus lead to high network overhead. In contrast, we created a full-text index for Wikipedia[73] (articles and category hierarchy). The index was created with *Apache Lucene*[74] (v2.3.0) and a modified version of the *LuceneSearch*[75] MediaWiki-plugin implementation (v2.1.3).

[73] We used the English Wikipedia as of August 3rd, 2011 with roughly 11 million pages.
[74] `http://lucene.apache.org/java/docs/index.html`
[75] `http://www.mediawiki.org/wiki/Extension:Lucene-search`

Additionally, we used MongoDB [PHM10] (v1.6.5) running on the same machine to store intermediate results created throughout HCM.
The aim of this evaluation is to show the scalability of our approach while achieving reasonable result quality. Remember that HCM aligns concept definitions only and shall be usable as preprocessing for more sophisticated approaches such as LINDA. We therefore target a quick but reliable process.

**Knowledge Representation Runtimes.** We first consider runtimes for the knowledge representation construction, i.e., the creation of WCFs, including keyword query generation, Wikipedia search, and Wikipedia category forest creation (see Section 5.2).
Given the one million concepts form the BTC data, the overall *keyword extraction* time is 7 minutes. The average number of keywords per concept is 2.77. As expected, the *index query* time to retrieve Wikipedia pages is linear to the number of extracted keyword sets, i.e., concepts under consideration. For instance, given 293k keyword sets from the I/$T_3$-extractor, the query phase took 64 minutes.
Note that the input to the keyword set extraction comprises one million concepts. However, the ID-extractor cannot yield a result for all input concepts, since we do not deal with blank nodes (roughly 50% of all input concept URIs are blank nodes) and malformed URIs. Further, not all keyword sets created yield a Wikipedia query result, leading to 238k keyword sets that can be used to build category forests. However, the combination of the concept URI and description-based approach, i.e., the I/$T_3$-extractor, leads to another 55k usable keyword sets. That is, for the following we deal with 293k concepts and respective knowledge representations.
The *forest construction* runtime mainly depends on the forest depth $d$ (the number of trees in the forest) and the tree height $h$ (the number of levels in a tree). Table 5.1 shows runtimes for the forest construction with different parameters[76]. The runtime depends linearly on the forest depth $d$. An increasing tree height $h$ leads to an exponential runtime increase. Similar to Jain et al., we set $h = 4$ and $d = 10$ for the following experiments [JHS+10]. With this configuration, the WCFs consist of 9.53 trees on average. In total, given fixed values for $h$ and $d$, the knowledge representation construction amounts to a linear runtime with respect to the number of input concepts dominated by the WCF construction.

**Group Size Measurements.** Given the knowledge representations for the BTC data, we now examine the grouping output. Figure 5.4 depicts the WCF group distribution for the BTC data over varying topic similarity thresholds $\theta$. For one, we illustrate the

[76]The creation of the Wikipedia page and category index took 5:30$h$ utilizing 4 cores.

| $h \setminus d$ | **5** | **10** | **15** | **20** |
|---|---|---|---|---|
| **1** | 0:06h | 0:09h | 0:11h | 0:14h |
| **2** | 0:15h | 0:23h | 0:28h | 0:38h |
| **3** | 0:31h | 0:52h | 1:18h | 1:34h |
| **4** | 1:13h | **3:03h** | 3:55h | 5:45h |
| **5** | 4:14h | 8:28h | 14:01h | 15:40h |

Table 5.1: Forest construction runtime for varying forest $h$ and $d$ parameters as well as the I/T$_3$-extractor (in hours).

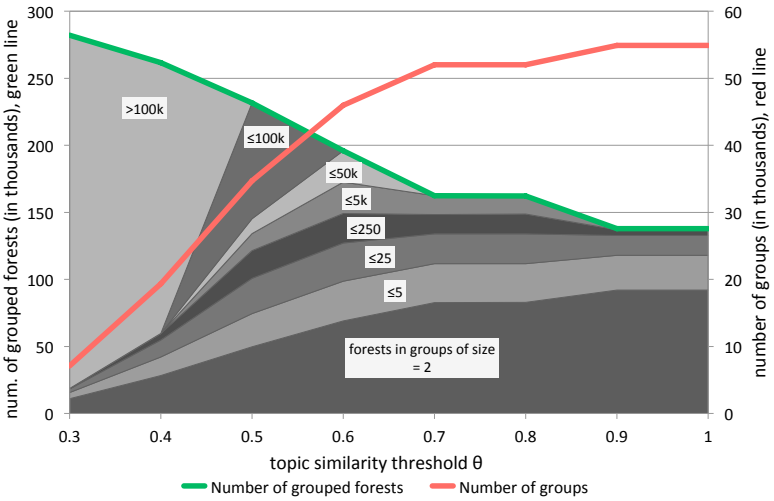


Figure 5.4: Forest group distribution for the I/T$_3$ keyword extractor, $h = 4$, $d = 10$, topics with 10 tree nodes over varying topic similarity $\theta$. The green graph shows the number of forests that can be grouped (left axis). The shaded areas below indicate the number of forest in groups of a specific size, i.e., the group size distribution. The red line shows the total number of groups (right axis).

number of forests that can be grouped for a given $\theta$ (green graph, left axis). Shaded areas below the green graph depict the number of forests in groups of a given size, e.g., groups of $size = 2$, $size \leq 5$, $size \leq 25$, etc. Also, we show the total number of created groups over $\theta$ (red graph, right axis).

For the fraction of WCFs that can be grouped, a higher $\theta$ leads to a decrease, because fewer WCF pairs have a sufficient topical overlap. The shaded areas further show that the higher $\theta$, the more WCFs fall in small groups, which is because a stricter overlap criterion induces more individual groups. Vice versa, a lower $\theta$ results in larger groups since smaller groups are merged.

The total number of groups clearly increases for higher $\theta$ due to the fact that there are more smaller groups for a higher threshold. The figure shows that $\theta = 1.0$ leads to 138k (out of 293k) forests that have at least one correspondence with an equal topic. The

majority (92k) appears in groups of size two.
For the following experiments we set $\theta = 0.7$ as default for three reasons: (1) This setting avoids large WCF groups, which would raise the runtime of the alignment generation phase. With $\theta = 0.7$, the maximal group size of 4k leads to a reasonable (and feasible) runtime on a single machine. (2) $\theta = 0.7$ provides a significant topical overlap among forests and minimizes arbitrary correspondences. (3) This threshold enables the identification of groups for more than 55% of all available WCFs. That is, we group 162k WCFs. Remaining WCFs do not have a sufficient topic overlap and can thus not be considered for the alignment generation.

**Group Creation Runtimes.** In general, the runtime of the candidate group creation, i.e., the blocking of WCFs, depends on the number of input WCFs, i.e, their respective topics and the self-join technique of choice. Figure 5.5 compares runtimes for three set similarity join techniques over a varying number of topics (topics include the top $m = 10$ nodes from the WCFs and $\theta$ is set to 0.7). The red graph shows the quadratic runtime for the naïve approach that performs all pairs' comparisons. The yellow line indicates values for *mpjoin* introduced by Ribeiro and Härder [RH09]. Mpjoin tries to minimize candidate set creation effort.
The green line shows the ppjoin runtime. The latter approach aims to minimize the candidate set sizes [XWLY08]. Obviously, ppjoin performs best showing a near-linear runtime. Here, the similarity join for 295k forest topics takes 6 minutes. Remember that these numbers refer to a large and heterogeneous sample from the current Web of Data, i.e., the BTC data. Wang et al. introduce a Map/Reduce implementation of ppjoin [WWL$^+$10], which leads to the conclusion that our approach can also deal with larger future datasets.

**Accuracy on BTC data.** To determine the accuracy of our holistic concept matching approach on BTC data, we examine the alignment generation algorithm's result (Section 5.4) using different similarity ranges. That is, we test different threshold settings for $\rho$, namely $O(f_1, f_2) = 1$, $1 > O(f_1, f_2) \geq 0.95$, $0.95 > O(f_1, f_2) \geq 0.9$, ..., $0.75 > O(f_1, f_2) \geq 0.7$. We did not consider alignments with $O(f_1, f_2) < 0.7$, which is along the lines with findings by Jain et al. where the authors mention an optimal threshold of 0.85 [JYV$^+$10]. Given the result from our approach, we created a random sample of 50 alignments for each similarity range. These samples were then manually evaluated by two independent annotators. In case of conflicts, a third annotator took the decision. Raters could use one out of three judgements: **Equivalence** indicates a correct alignment of two equivalent concepts. **Similar** indicates a fuzzy match. The two

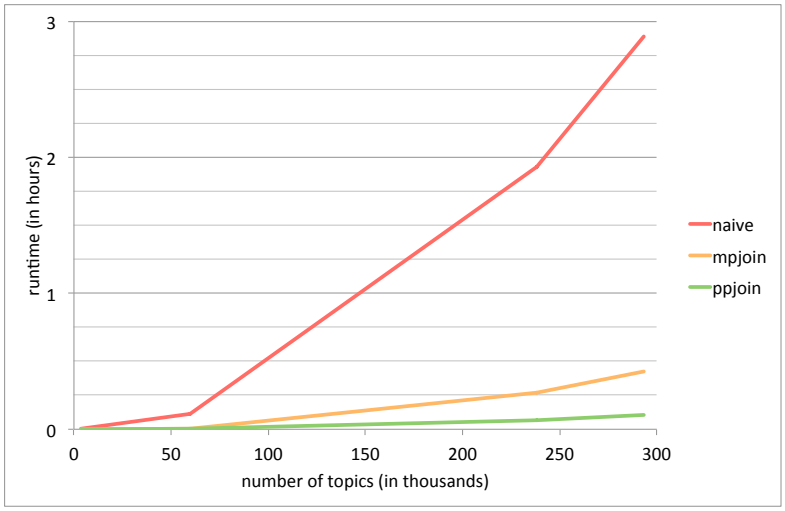


Figure 5.5: Runtime for the forest grouping with different set similarity join algorithms. $h = 4$, $d = 10$, $\theta = 0.7$, and $m = 10$.

concepts are related and the sets of instances of the concepts have a significant intersection. **Disjoint** indicates an incorrect alignment of two concepts whose sets of concept instances are disjoint.

Considering only exact equivalence ratings, the inter-annotator agreement, Cohen's kappa, is $\kappa = 0.909$. Treating similar-rated alignments as correct, the inter-annotator agreement decreases to $\kappa = 0.817$. This decrease is because of a varying perception of relatedness. Clearly, similarity is more subjective than strict equivalence. For instance, the concepts *daml:Ammeters* and *umbel:Voltmeter* are related due to their common purpose. On the other hand, respective instance sets can be disjoint.

Table 5.2 shows the result of our manual evaluation, i.e., accuracy and respective confidence intervals. In the range of $1 > O(f_1, f_2) \geq 0.95$, 57.5% of the alignments were labeled as equivalences and additionally 15% as similar. Of course, the number of exact matches decreases for smaller forest similarities. Consider the following examples showing that our approach retrieves non-trivial and interesting relations among concepts from ontologies with different scopes.

- `yago:PsychoactiveFungi` and `umbel:HallucinogenicMushroom`
- `dbpedia:BirdsOfAustralia` and `opencyc:Parrot`
- `umbel:AirConditioner` and `dbpedia:HeatExchangers`

| | | | equivalences | equ. or similar |
|---|---|---|---|---|
| | $O(f_1, f_2)$ | $= 1.0$ | 0.964 ±0.036 | 0.964 ±0.036 |
| $1.0 >$ | $O(f_1, f_2)$ | $\geq 0.95$ | 0.575 ±0.143 | 0.725 ±0.129 |
| $0.95 >$ | $O(f_1, f_2)$ | $\geq 0.9$ | 0.481 ±0.145 | 0.706 ±0.131 |
| $0.9 >$ | $O(f_1, f_2)$ | $\geq 0.85$ | 0.071 ±0.066 | 0.294 ±0.131 |
| $0.85 >$ | $O(f_1, f_2)$ | $\geq 0.8$ | 0.053 ±0.053 | 0.200 ±0.114 |
| $0.8 >$ | $O(f_1, f_2)$ | $\geq 0.75$ | 0.053 ±0.053 | 0.126 ±0.092 |
| $0.75 >$ | $O(f_1, f_2)$ | $\geq 0.7$ | 0.071 ±0.066 | 0.331 ±0.136 |

Table 5.2: Accuracy and confidence intervals for random samples created from HCM alignments. The left column shows varying WCF similarity ($O(f_1, f_2)$) ranges. The middle column depicts accuracy values for considering only *equivalence* judgements as correct alignments. The rightmost column shows values for considering both, *equivalent* and *similar* pairs, as correct.

Obviously, since the concepts from these pairs stem from heterogeneous ontologies, alignments cannot be determined easily by humans. However, our holistic concept matching approach can deal with this kind of data, because it exploits the broad knowledge for many domains present in Wikipedia and does not neglect existing information from the sources.

In [GBN12] we additionally show results for the benchmark track of the well-known Ontology Alignment Evaluation Initiative Campaign [EFM+10]: here, $\rho = 0.7$ lead to a precision of 85% and a recall of 55%.

**Runtimes on BTC data.** Figure 5.6 illustrates runtimes of the alignment generation (Section 5.4) for the groups created with $\theta = 0.7$ as depicted in Figure 5.4. Since the actual alignment performs a pairwise comparison of Wikipedia category trees (whose number and size is bound by $d$ and $s$) for all pairs of forests in a group, the runtime is quadratic in the number of group members. Thus, the overall maximum runtime (31h) depends on the largest group, which comprises roughly 4k Wikipedia concept forests. The second largest group requires less than half of the time (14h) for the alignment. The vast majority of groups can each be aligned in less than 2h, since these groups do not contain many concepts (see Figure 5.4). Thus, in practice, all groups could be processed in parallel within the runtime of the largest group.

The alignment graph creation is linear in the number of ontological and candidate relations. The polynomial alignment verification per added relation additionally influences the overall runtime. In Figure 5.6, we further show the polynomial regression, given the measurements for the actual groups.

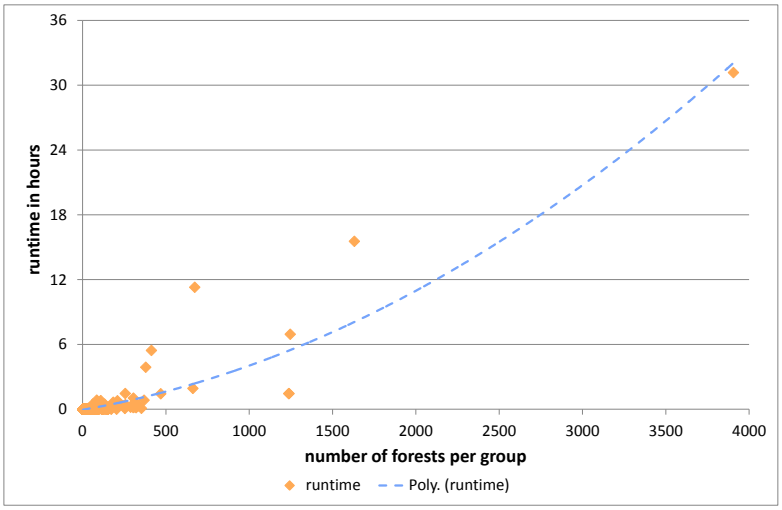


Figure 5.6: Alignment generation runtime different WCF group sizes. The dashed line indicates the quadratic regression for the total alignment generation time per group. The experiment was conducted with the $I/T_3$ keyword extraction, forest $h = 4$ and $d = 10$, topics with $m = 10$ tree nodes and $\theta = 0.7$ as well as a forest overlap $\rho = 0.7$.

## 5.6 Discussion

In this chapter we tackled the problem of aligning concepts from many ontologies available on the Web of Data – simultaneously and in a holistic manner. To this end, we proposed an abstract workflow consisting of knowledge extraction, grouping, and alignment generation to specifically enable scalability while examining many ontologies in its entirety. We plug state-of-the-art techniques and novel ideas into this workflow and report on promising results with respect to scalability and alignment quality in a Web-scale alignment scenario. For representing knowledge, we chose Wikipedia category forests. For grouping the input, we leverage topical information represented as sets of highly scored Wikipedia category tree nodes as well as a state-of-the-art set similarity index. Last, the alignment generation leverages Wikipedia category forest overlaps and performs a semantic verification. Thus, our approach is specifically tailored towards a quick solution but still draws conclusions from a large subset of the ontological relations in the original sources.

Table 5.3 summarizes the runtimes of the entire HCM approach. Disregarding the preprocessing, i.e., concept data extraction and Wikipedia index construction, most steps can be done in a short amount of time. The forest construction requires the longest time ($3h$) before the actual matching. The alignment phase runtime varies depending on the group size, i.e., it is roughly $2h$ for the majority of groups while the largest group requires $31h$ for the matching. In total, the process required $44{:}30h$ dominated by the alignment

creation for the largest concept group. Disregarding the Wikipedia index creation time this value is below $40h$. Given these short runtimes and the modularity of our approach, we clearly implemented an alignment workflow that is ready for future large concept sets. Note that all phases, except the grouping, can be distributed easily. The grouping itself is fast when using a state-of-the-art set similarity self-join technique.
Future directions might include the exploration of other knowledge representations and matching techniques that can be plugged into the workflow. The challenge is to identify approaches that can capture different semantic notions of an input concept but allow for a grouping that minimizes pairwise considerations. Another direction is to examine other relationship types, where concept instance data could be helpful for the creation of knowledge representations. Finally, more elaborate grouping methods could for instance support incremental updates and thus facilitate online ontology alignments for the growing Web of Data.

| | time in hours |
|---|---|
| BTC data preprocessing | 03:30 |
| Keyword query generation | 00:07 |
| Wikipedia index creation | 05:30 |
| Wikipedia search | 01:04 |
| Forest construction | 03:03 |
| Forest grouping | 00:06 |
| Concept alignment (max. time) | 31:10 |
| Sum | 44:30 |

Table 5.3: Summary of runtimes throughout the different phases of our approach.

# 6 Conclusion

**Summary of the Presented Methods.** In this thesis we presented methods for enhancing the current state of the Web of Data. First, in Chapter 3, we coined the notion of topics for graph-structured data and discussed our approach called Annotated Pattern Percolation. This APP method detects overlapping communities of topically related entities in annotated entity-graphs. For this community detection, we employ only the structure inherent to the graph and are independent of textual information, such as labels or descriptions. In an evaluation against manually created reference data from Wikipedia, we compared properties of varying results and found that we can automatically reconstruct a portion of our reference data at reasonable precision. Note that, since topics in graph-structured data are not an established field of research, this evaluation is, to the best of our knowledge, the first of its kind.
The second main part of this work, Chapter 4, discussed how to create links among entities from the Web of Data in a joint and scalable manner. For this entity alignment, heterogeneity and size of respective Web datasets pose major challenges. We capture the problem in an optimization model, i.e., the Maximum Consistent Assignment Problem, and propose three algorithms to compute good solutions – all implemented in our LINDA system. Our first approach produces consistent alignments but lacks scalability. The second approach scales well but violates the result consistency in certain cases. The third method yields a consistent outcome and scales on large compute clusters. We evaluated results on very large heterogeneous real-world datasets and manually determined accuracy values around 0.9 and above (Section 4.6). Note that we are aware of only one work [PIN+12] that reports on entity alignment results obtained with datasets larger than our evaluation data. This work, however, does not perform joint entity linking.
Finally, in Chapter 5, we discussed concept alignment, which is a subproblem of entity alignment. In this case, we presented an approach that groups the input and then runs complex alignment and reasoning for concepts within groups. With this strategy, we can quickly retrieve high-quality output for large real-word data (Section 5.5).

**Combined Use-Case.** The Web of Data is about interconnected entities described with RDF and published on the Web. In the future, one shall be able to query this gigantic knowledge base as if it were a traditional database containing very many tables and

references among these tables. Figure 6.1 illustrates this vision and the role of our contributions, i.e., methods consuming and contributing linked data. To facilitate queries combining a variety of entities of different types from many publishers, the query processing requires information about which chunks of data to process, i.e., to perform query source selection. To perform this selection one could either explicitly provide required topics with the query or compute them from the query. Then, if topics were published with the data, the processing could determine sources where respective entities interplay and execute the query on the selected chunk of data.

For the retrieval of information about entities from different sources, one leverages `sameAs` links (among others) that must be present in the data. HCM and LINDA contribute such `sameAs` links. LINDA has shown to yield accurate output. However, LINDA could additionally benefit from existing links, e.g., achieve higher recall (which we did not aim to quantify yet). HCM, for instance, rapidly produces links among concepts, which could be used to seed LINDA. Then, concept-level links can contribute evidence for two differently typed entities for being truly the same. Topics, on the other hand, could be used to develop a smart partitioning of the data when employing the distributed versions of LINDA.

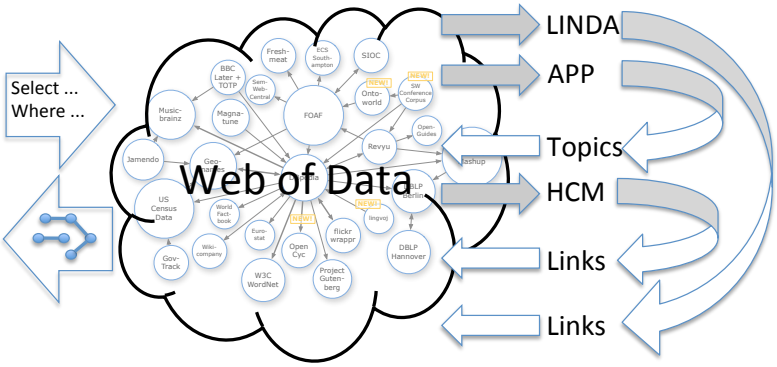


Figure 6.1: Overall Use-Case for APP, HCM, and LINDA.

Finally, observe that, though developed for the Web of Data, our methods also apply to other fields. Our topic mining approach essentially contributes to the general field of community detection in attributed graphs, which is also of interest for social network or life science data. LINDA works on entity graphs, which is an abstract representation of things and respective relationships. The construction of such graphs is not limited to RDF as input. HCM features an entity grouping and intra-group alignment strategy. Many building blocks for this strategy, such as the knowledge representation, can be adapted to other entity alignment scenarios, e.g., when merging multiple complex and heterogeneous relational datasets.

# Acknowledgements

First and foremost, I would like to thank Felix Naumann for the opportunity to do my PhD in his group, his supervision, and for giving me broad flexibility to develop my work and myself during the past years. Further, I am thankful for the opportunity to visit the Max Planck Databases and Information Systems group in 2011, where I got to know Gerhard Weikum and Gerard de Melo, who became de facto co-supervisers. Thanks for the many critical, well-grounded, and inspiring discussions. Also, many thanks to Klaus Berberich for the excellent Hadoop know-how exchange. Additionally, I would like to mention the talented and enthusiastic students who helped to develop and maintain the thousands of lines of code written for the different projects discussed in this thesis: Johannes Gosda, Eyk Kny, Toni Grütze, and Nils Rethmeier. Further, I would like to acknowledge the co-authors and proof-readers of "my" publications. A big thanks to IBM, among others Albert Maier in Böblingen, for financing a significant part of my work through an IBM CAS grant (received by Felix Naumann) and awarding me with two IBM PhD Fellowships. Many thanks to Samira Jaeger for proof-reading this thesis. Thanks to my lovely family for tolerating me whenever I was not so happy with what I was doing. Last but definitely not least, I thank the operations folks at HPI in Potsdam and MPII in Saarbrücken for keeping the computing infrastructure up and running and for fast response times during the Christmas holidays.

# Appendix

| iteration | step | node 1 | node 2 | node 3 | node 4 | node 5 | node 6 | node 7 | node 8 | node 9 | node 10 | node 11 | node 12 | mean | max |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2,647.62 | 3,146.20 | 2,784.80 | 1,945.49 | 11,244.79 | 2,297.10 | 1,899.18 | 1,449.71 | 1,851.14 | 2,167.38 | 2,210.78 | 1,665.30 | 2,639.31 | 11,244.79 |
| | 2 | 7.90 | 28.00 | 28.84 | 44.00 | 7.54 | 26.43 | 34.77 | 51.32 | 32.97 | 66.24 | 18.69 | 18.61 | 28.88 | 66.24 |
| | 3 | 9.33 | 15.04 | 39.93 | 46.48 | 8.68 | 33.76 | 47.42 | 68.16 | 34.16 | 65.21 | 17.21 | 18.83 | 31.06 | 68.16 |
| | 4 | 141.19 | 0.00 | 80.16 | 146.80 | 46.04 | 82.14 | 101.19 | 168.24 | 66.94 | 291.94 | 387.05 | 42.67 | 101.85 | 387.05 |
| | 5 | 477.02 | 596.45 | 528.14 | 489.49 | 513.28 | 475.48 | 469.91 | 429.17 | 511.50 | 497.67 | 661.82 | 495.98 | 515.71 | 661.82 |
| 1 | 6 | 2,841.90 | 0.00 | 4,285.77 | 782.92 | 0.00 | 1,076.02 | 0.00 | 2,985.52 | 2,540.16 | 4,508.47 | 3,300.06 | 779.22 | 1,648.71 | 4,508.47 |
| | 7 | 257,083.63 | 259,666.65 | 260,930.22 | 261,727.53 | 251,448.46 | 256,853.31 | 267,235.45 | 245,480.19 | 257,049.04 | 256,845.03 | 291,204.93 | 273,888.61 | 265,068.56 | 307,059.72 |
| | 8 | 1,040.93 | 1,511.08 | 996.75 | 1,033.23 | 1,316.21 | 1,214.56 | 1,223.61 | 1,103.73 | 1,177.55 | 1,204.79 | 1,622.95 | 1,063.68 | 1,179.72 | 1,622.95 |
| | 9 | 4,945.96 | 14,308.58 | 4,928.57 | 3,215.17 | 3,765.97 | 5,047.35 | 44,210.59 | 45,573.94 | 6,123.51 | 5,081.94 | 3,480.27 | 2,724.41 | 9,111.66 | 45,573.94 |
| | 10 | 30,137.94 | 30,451.76 | 28,292.28 | 29,217.60 | 22,216.10 | 26,212.87 | 27,670.51 | 28,088.06 | 29,309.93 | 25,617.66 | 24,983.34 | 27,220.42 | 27,490.93 | 37,444.34 |
| | 11 | 3,063.96 | 3,921.29 | 3,387.59 | 3,566.93 | 3,710.27 | 3,874.48 | 3,842.74 | 3,961.25 | 4,005.92 | 3,535.11 | 4,130.52 | 3,508.00 | 3,798.77 | 4,434.38 |
| | 12 | 61.33 | 161.98 | 71.68 | 60.01 | 96.76 | 83.75 | 87.07 | 113.70 | 64.13 | 64.83 | 75.84 | 68.17 | 83.05 | 161.98 |
| | 1 | 1,487.68 | 1,538.43 | 1,553.56 | 1,659.89 | 1,570.81 | 1,532.96 | 1,488.24 | 1,501.14 | 1,622.43 | 1,540.39 | 1,898.64 | 1,493.04 | 1,591.86 | 1,898.64 |
| | 2 | 0.00 | 28.03 | 3.04 | 27.28 | 0.00 | 5.42 | 27.30 | 0.00 | 12.13 | 38.59 | 10.11 | 12.04 | 11.18 | 38.59 |
| | 3 | 1.61 | 31.16 | 3.66 | 20.79 | 0.00 | 6.64 | 19.90 | 0.00 | 12.40 | 30.08 | 9.68 | 6.11 | 9.98 | 31.16 |
| | 4 | 9.64 | 56.33 | 10.13 | 44.95 | 0.00 | 7.49 | 59.04 | 0.00 | 19.00 | 59.82 | 17.88 | 19.55 | 21.39 | 59.82 |
| | 5 | 155.89 | 173.24 | 166.62 | 188.27 | 176.10 | 176.91 | 150.67 | 145.78 | 180.76 | 150.52 | 141.44 | 167.49 | 173.71 | 314.48 |
| 2 | 6 | 0.00 | 0.00 | 0.00 | 2,248.34 | 0.00 | 957.00 | 1,286.71 | 0.00 | 6,005.85 | 1,961.32 | 566.80 | 445.96 | 920.50 | 6,005.85 |
| | 7 | 85,652.89 | 87,760.78 | 92,229.67 | 93,365.57 | 85,590.33 | 87,214.08 | 95,095.03 | 94,043.47 | 91,286.02 | 88,571.21 | 87,451.99 | 95,903.82 | 90,154.49 | 100,035.08 |
| | 8 | 351.24 | 507.46 | 372.23 | 372.76 | 441.61 | 461.93 | 488.31 | 560.77 | 459.75 | 349.17 | 347.87 | 479.01 | 419.53 | 560.77 |
| | 9 | 2,373.84 | 14,850.48 | 2,509.92 | 1,882.41 | 4,905.43 | 1,875.79 | 1,383.03 | 1,595.57 | 1,686.01 | 1,561.54 | 1,623.75 | 7,207.18 | 3,901.29 | 18,182.19 |
| | 10 | 14,123.84 | 20,485.96 | 17,818.05 | 18,529.06 | 14,747.24 | 17,733.12 | 16,006.40 | 18,705.22 | 17,852.47 | 15,653.87 | 18,872.17 | 18,829.90 | 17,506.12 | 22,084.19 |
| | 11 | 1,146.36 | 1,501.78 | 634.88 | 800.77 | 1,155.93 | 790.13 | 617.27 | 711.67 | 897.99 | 728.97 | 670.15 | 1,481.86 | 962.04 | 2,049.06 |
| | 12 | 42.36 | 54.84 | 34.96 | 29.04 | 43.33 | 37.70 | 45.38 | 51.21 | 36.95 | 29.91 | 35.35 | 45.00 | 40.51 | 66.98 |
| | 1 | 1,711.83 | 1,797.81 | 1,645.56 | 1,515.82 | 1,668.70 | 1,502.69 | 1,545.39 | 1,636.37 | 1,605.84 | 1,641.19 | 2,090.72 | 1,687.43 | 1,658.63 | 2,090.72 |
| | 2 | 0.00 | 2.49 | 0.00 | 0.00 | 0.00 | 3.92 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.54 | 3.92 |
| | 3 | 0.00 | 2.64 | 0.00 | 0.00 | 0.00 | 8.41 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.74 | 8.41 |
| | 4 | 0.00 | 5.19 | 0.00 | 0.00 | 0.00 | 11.07 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.87 | 11.07 |
| | 5 | 29.00 | 26.26 | 24.65 | 30.27 | 29.34 | 17.42 | 21.03 | 26.84 | 36.98 | 25.50 | 30.52 | 31.59 | 27.50 | 36.98 |
| 3 | 6 | 0.00 | 1,487.77 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 78.30 | 1,487.77 |
| | 7 | 21,169.62 | 17,873.32 | 27,021.40 | 26,739.36 | 18,792.62 | 13,824.72 | 21,525.93 | 19,782.73 | 25,693.06 | 16,527.13 | 18,586.87 | 20,644.76 | 20,924.45 | 27,021.40 |
| | 8 | 57.14 | 32.84 | 59.91 | 64.24 | 45.45 | 27.66 | 40.69 | 43.78 | 126.00 | 45.85 | 42.93 | 40.30 | 51.26 | 126.00 |
| | 9 | 72.29 | 46.64 | 66.47 | 48.47 | 71.63 | 16.21 | 52.03 | 50.75 | 47.25 | 172.33 | 55.22 | 61.55 | 60.83 | 172.33 |
| | 10 | 266.58 | 297.21 | 299.97 | 313.88 | 234.85 | 407.07 | 381.47 | 258.59 | 421.43 | 301.42 | 337.96 | 312.26 | 332.88 | 512.77 |
| | 11 | 64.74 | 49.43 | 60.90 | 43.45 | 67.69 | 19.96 | 44.49 | 45.82 | 74.69 | 62.46 | 39.45 | 71.34 | 53.38 | 84.83 |
| | 12 | 3.86 | 3.10 | 4.57 | 3.34 | 4.29 | 1.74 | 3.08 | 3.32 | 6.42 | 3.01 | 2.84 | 3.86 | 3.64 | 6.42 |
| | 1 | 2,727.94 | 2,834.26 | 2,776.79 | 2,456.55 | 2,625.14 | 2,422.65 | 2,406.65 | 2,389.49 | 2,454.77 | 2,420.83 | 2,904.48 | 2,695.91 | 2,587.77 | 2,904.48 |
| | 2 | 0.00 | 17.62 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.38 | 17.62 |
| | 3 | 0.00 | 17.36 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.38 | 17.36 |
| | 4 | 0.00 | 23.30 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.50 | 24.22 |
| | 5 | 16.90 | 13.55 | 12.99 | 18.65 | 14.54 | 14.86 | 16.08 | 17.17 | 17.50 | 16.06 | 6.75 | 19.93 | 14.84 | 19.93 |
| 4 | 6 | 0.00 | 872.68 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 45.93 | 872.68 |
| | 7 | 10,250.50 | 8,895.00 | 7,392.31 | 11,957.77 | 8,759.69 | 10,927.06 | 8,281.41 | 8,222.17 | 9,490.23 | 9,306.90 | 8,398.11 | 8,570.62 | 9,465.24 | 13,492.59 |
| | 8 | 39.45 | 28.25 | 33.25 | 39.08 | 37.56 | 38.53 | 36.77 | 31.25 | 32.26 | 37.63 | 26.66 | 33.10 | 42.59 | 146.87 |
| | 9 | 205.38 | 142.96 | 139.35 | 279.19 | 167.11 | 183.27 | 160.11 | 164.91 | 1,324.34 | 322.10 | 47.77 | 274.74 | 260.00 | 1,324.34 |
| | 10 | 1,046.35 | 1,478.12 | 1,190.94 | 1,581.14 | 940.07 | 1,907.16 | 949.73 | 1,204.12 | 1,037.67 | 1,292.38 | 1,455.62 | 1,254.97 | 1,422.13 | 2,805.30 |
| | 11 | 202.92 | 92.63 | 229.46 | 249.64 | 88.95 | 82.04 | 69.09 | 89.63 | 166.01 | 96.69 | 26.99 | 75.63 | 107.93 | 249.64 |
| | 12 | 6.91 | 3.71 | 3.52 | 5.03 | 3.38 | 3.78 | 3.19 | 3.36 | 4.99 | 4.82 | 2.49 | 3.60 | 4.40 | 11.43 |

Figure 7.1: Time in vertex compute function of MP-LINDA per compute node and step (in milliseconds).

| iteration | step | node 1 | node 2 | node 3 | node 4 | node 5 | node 6 | node 7 | node 8 | node 9 | node 10 | node 11 | node 12 | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 8.30 | 506.89 | 145.49 | -693.82 | 8,605.48 | -342.22 | -740.13 | -1,189.61 | -788.18 | -471.94 | -428.54 | -974.01 | 2,639.31 |
| | 2 | -20.99 | -0.88 | -0.04 | 15.11 | -21.34 | -2.46 | 5.89 | 22.43 | 4.09 | 37.36 | -10.19 | -10.27 | 28.88 |
| | 3 | -21.73 | -16.02 | 8.87 | 15.42 | -22.38 | 2.70 | 16.36 | 37.10 | 3.10 | 34.15 | -13.85 | -12.23 | 31.06 |
| | 4 | 39.34 | -101.85 | -21.69 | 44.95 | -55.80 | -19.70 | -0.65 | 66.39 | -34.91 | 190.09 | 285.20 | -59.18 | 101.85 |
| | 5 | -38.69 | 80.74 | 12.43 | -26.22 | -2.43 | -40.23 | -45.80 | -86.54 | -4.21 | -18.04 | 146.11 | -19.73 | 515.71 |
| **1** | 6 | 1,193.19 | -1,648.71 | 2,637.06 | -865.79 | -1,648.71 | -572.69 | -1,648.71 | 1,336.81 | 891.45 | 2,859.76 | 1,651.35 | -869.49 | 1,648.71 |
| | 7 | -7,984.93 | -5,401.91 | -4,138.34 | -3,341.03 | -13,620.10 | -8,215.25 | 2,166.89 | -19,588.37 | -8,019.52 | -8,223.54 | 26,136.36 | 8,820.04 | 265,068.56 |
| | 8 | -138.80 | 331.36 | -182.97 | -146.50 | 136.49 | 34.84 | 43.88 | -76.00 | -2.18 | 25.07 | 443.23 | -116.05 | 1,179.72 |
| | 9 | -4,165.70 | 5,196.93 | -4,183.09 | -5,896.48 | -5,345.69 | -4,064.31 | 35,098.94 | 36,462.28 | -2,988.14 | -4,029.71 | -5,631.39 | -6,387.25 | 9,111.66 |
| | 10 | 2,647.01 | 2,960.83 | 801.36 | 1,726.67 | -5,274.83 | -1,278.05 | 179.59 | 597.13 | 1,819.00 | -1,873.27 | -2,507.58 | -270.50 | 27,490.93 |
| | 11 | -734.81 | 122.52 | -411.18 | -231.84 | -88.49 | 75.71 | 43.97 | 162.49 | 207.15 | -263.66 | 331.75 | -290.77 | 3,798.77 |
| | 12 | -21.72 | 78.93 | -11.37 | -23.04 | 13.72 | 0.71 | 4.02 | 30.65 | -18.91 | -18.22 | -7.20 | -14.88 | 83.05 |
| | 1 | -104.18 | -53.43 | -38.30 | 68.03 | -21.05 | -58.90 | -103.62 | -90.72 | 30.57 | -51.47 | 306.79 | -98.82 | 1,591.86 |
| | 2 | -11.18 | 16.85 | -8.14 | 16.10 | -11.18 | -5.76 | 16.13 | -11.18 | 0.96 | 27.41 | -1.06 | 0.86 | 11.18 |
| | 3 | -8.36 | 21.19 | -6.31 | 10.81 | -9.98 | -3.33 | 9.92 | -9.98 | 2.42 | 20.10 | -0.30 | -3.87 | 9.98 |
| | 4 | -11.74 | 34.94 | -11.25 | 23.56 | -21.39 | -13.90 | 37.65 | -21.39 | -2.38 | 38.44 | -3.50 | -1.83 | 21.39 |
| | 5 | -17.82 | -0.47 | -7.09 | 14.56 | 2.39 | 3.20 | -23.04 | -27.93 | 7.05 | -23.19 | -32.27 | -6.22 | 173.71 |
| **2** | 6 | -920.50 | -920.50 | -920.50 | 1,327.84 | -920.50 | 36.50 | 366.21 | -920.50 | 5,085.35 | 1,040.82 | -353.70 | -474.55 | 920.50 |
| | 7 | -4,501.60 | -2,393.70 | 2,075.18 | 3,211.08 | -4,564.16 | -2,940.41 | 4,940.54 | 3,888.98 | 1,131.53 | -1,583.28 | -2,702.50 | 5,749.33 | 90,154.49 |
| | 8 | -68.29 | 87.92 | -47.30 | -46.78 | 22.08 | 42.39 | 68.77 | 141.24 | 40.22 | -70.37 | -71.67 | 59.48 | 419.53 |
| | 9 | -1,527.45 | 10,949.19 | -1,391.37 | -2,018.88 | 1,004.14 | -2,025.50 | -2,518.25 | -2,305.72 | -2,215.28 | -2,339.75 | -2,277.54 | 3,305.89 | 3,901.29 |
| | 10 | -3,382.28 | 2,979.85 | 311.93 | 1,022.94 | -2,758.87 | 227.00 | -1,499.72 | 1,199.11 | 346.36 | -1,852.24 | 1,366.06 | 1,323.78 | 17,506.12 |
| | 11 | 184.31 | 539.74 | -327.16 | -161.27 | 193.89 | -171.91 | -344.77 | -250.37 | -64.05 | -233.07 | -291.89 | 519.82 | 962.04 |
| | 12 | 1.84 | 14.33 | -5.55 | -11.47 | 2.82 | -2.81 | 4.87 | 10.70 | -3.56 | -10.60 | -5.16 | 4.48 | 40.51 |
| | 1 | 53.20 | 139.18 | -13.07 | -142.81 | 10.07 | -155.93 | -113.24 | -22.26 | -52.79 | -17.44 | 432.09 | 28.80 | 1,658.63 |
| | 2 | -0.54 | 1.95 | -0.54 | -0.54 | -0.54 | 3.38 | -0.54 | -0.54 | -0.54 | -0.54 | -0.54 | -0.54 | 0.54 |
| | 3 | -0.74 | 1.90 | -0.74 | -0.74 | -0.74 | 7.67 | -0.74 | -0.74 | -0.74 | -0.74 | -0.74 | -0.74 | 0.74 |
| | 4 | -1.87 | 3.32 | -1.87 | -1.87 | -1.87 | 9.21 | -1.87 | -1.87 | -1.87 | -1.87 | -1.87 | -1.87 | 1.87 |
| | 5 | 1.50 | -1.24 | -2.85 | 2.77 | 1.84 | -10.08 | -6.47 | -0.66 | 9.48 | -2.00 | 3.02 | 4.09 | 27.50 |
| | 6 | -78.30 | 1,409.47 | -78.30 | -78.30 | -78.30 | -78.30 | -78.30 | -78.30 | -78.30 | -78.30 | -78.30 | -78.30 | 78.30 |
| **3** | 7 | 245.17 | -3,051.13 | 6,096.95 | 5,814.91 | -2,131.82 | -7,099.73 | 601.48 | -1,141.71 | 4,768.61 | -4,397.32 | -2,337.58 | -279.68 | 20,924.45 |
| | 8 | 5.89 | -18.42 | 8.65 | 12.98 | -5.80 | -23.60 | -10.56 | -7.48 | 74.74 | -5.40 | -8.33 | -10.96 | 51.26 |
| | 9 | 11.46 | -14.19 | 5.64 | -12.36 | 10.80 | -44.62 | -8.80 | -10.08 | -13.58 | 111.50 | -5.61 | 0.72 | 60.83 |
| | 10 | -66.30 | -35.67 | -32.91 | -19.00 | -98.03 | 74.19 | 48.59 | -74.29 | 88.55 | -31.46 | 5.08 | -20.62 | 332.88 |
| | 11 | 11.36 | -3.95 | 7.52 | -9.94 | 14.30 | -33.42 | -8.89 | -7.56 | 21.31 | 9.08 | -13.93 | 17.95 | 53.38 |
| | 12 | 0.21 | -0.54 | 0.93 | -0.31 | 0.64 | -1.91 | -0.56 | -0.33 | 2.77 | -0.63 | -0.80 | 0.22 | 3.64 |
| | 1 | 140.17 | 246.48 | 189.01 | -131.22 | 37.37 | -165.12 | -181.13 | -198.28 | -133.00 | -166.94 | 316.71 | 108.14 | 2,587.77 |
| | 2 | -1.38 | 16.24 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | 1.38 |
| | 3 | -1.38 | 15.98 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | -1.38 | 1.38 |
| | 4 | -2.50 | 20.80 | -2.50 | -2.50 | -2.50 | -2.50 | -2.50 | -2.50 | -2.50 | -2.50 | -2.50 | -2.50 | 2.50 |
| | 5 | 2.05 | -1.30 | -1.85 | 3.81 | -0.30 | 0.02 | 1.23 | 2.32 | 2.66 | 1.21 | -8.09 | 5.09 | 14.84 |
| | 6 | -45.93 | 826.75 | -45.93 | -45.93 | -45.93 | -45.93 | -45.93 | -45.93 | -45.93 | -45.93 | -45.93 | -45.93 | 45.93 |
| **4** | 7 | 785.26 | -570.24 | -2,072.93 | 2,492.53 | -705.55 | 1,461.81 | -1,183.84 | -1,243.07 | 24.98 | -158.34 | -1,067.14 | -894.62 | 9,465.24 |
| | 8 | -3.14 | -14.33 | -9.34 | -3.51 | -5.03 | -4.06 | -5.82 | -11.34 | -10.33 | -4.96 | -15.93 | -9.49 | 42.59 |
| | 9 | -54.62 | -117.04 | -120.65 | 19.19 | -92.89 | -76.73 | -99.89 | -95.09 | 1,064.34 | 62.09 | -212.24 | 14.73 | 260.00 |
| | 10 | -375.77 | 56.00 | -231.19 | 159.01 | -482.06 | 485.03 | -472.40 | -218.01 | -384.45 | -129.75 | 33.50 | -167.16 | 1,422.13 |
| | 11 | 94.99 | -15.30 | 121.53 | 141.71 | -18.98 | -25.89 | -38.84 | -18.30 | 58.07 | -11.25 | -80.94 | -32.31 | 107.93 |
| | 12 | 2.51 | -0.69 | -0.88 | 0.63 | -1.02 | -0.62 | -1.21 | -1.04 | 0.59 | 0.42 | -1.91 | -0.80 | 4.40 |

Figure 7.2: Difference of individual to average time in vertex compute function of MP-LINDA per compute node and step (in milliseconds).

| Iteration | step | node 1 | node 2 | node 3 | node 4 | node 5 | node 6 | node 7 | node 8 | node 9 | node 10 | node 11 | node 12 | avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | -10 | 411 | 49 | -755 | 8,604 | -401 | -838 | -1,327 | -751 | -461 | -335 | -530 | 3,690 |
| | 2 | 11,612 | 27,151 | -1,540 | -268 | -4,033 | -19,813 | 2,182 | -20,605 | -14,082 | -15,526 | -19,824 | -16,149 | 529,590 |
| | 3 | -45 | 1 | 23 | -121 | -36 | -64 | 74 | 246 | -101 | 63 | 77 | -33 | 1,744 |
| | 4 | -537 | -808 | 1,356 | -436 | -773 | 1,032 | -675 | -637 | 557 | 1,808 | -301 | -785 | 2,643 |
| | 5 | -134 | 14 | -159 | 157 | 100 | -134 | -29 | -68 | 57 | 150 | 414 | 245 | 2,613 |
| 1 | 6 | 2,034 | -1,845 | 2,512 | -1,059 | -1,920 | -651 | -1,661 | 1,400 | 928 | 2,933 | 2,228 | -991 | 3,718 |
| | 7 | -8,097 | -5,526 | -4,322 | -3,453 | -13,658 | -8,389 | 2,336 | -19,474 | -8,211 | -7,389 | 26,188 | 8,794 | 266,911 |
| | 8 | 473 | 1,167 | -43 | -166 | 1,355 | -373 | 1,185 | -459 | -64 | -249 | 405 | -405 | 3,357 |
| | 9 | -5,022 | 28,516 | -4,962 | -10,851 | -5,093 | -3,217 | 37,348 | 37,491 | -3,313 | -3,243 | -10,259 | -10,870 | 16,322 |
| | 10 | 2,946 | 3,808 | 1,704 | 1,541 | -5,538 | -1,557 | -42 | 439 | 1,618 | -2,043 | -1,944 | -405 | 29,397 |
| | 11 | -889 | -486 | -755 | 249 | 247 | -519 | 447 | 510 | 508 | -20 | 952 | -21 | 6,371 |
| | 12 | 604 | -327 | -218 | 7 | 317 | -616 | 447 | 968 | -491 | 538 | 256 | -380 | 3,172 |
| | 1 | -150 | -108 | -84 | -12 | 38 | -27 | -152 | -110 | -16 | -114 | 318 | -89 | 2,530 |
| | 2 | 26,968 | 40,141 | 3,322 | -4,738 | 653 | -13,818 | -1,633 | -20,299 | -9,317 | -8,229 | -17,497 | -17,651 | 520,016 |
| | 3 | -13 | -3 | -63 | -38 | -58 | 167 | 21 | -87 | -76 | -35 | 153 | 46 | 1,680 |
| | 4 | 1,136 | 1,155 | -162 | -21 | -190 | -350 | -203 | -187 | -147 | -189 | 145 | -112 | 1,978 |
| | 5 | -159 | -76 | -125 | -82 | 61 | -26 | -74 | 48 | -40 | 47 | 23 | -107 | 1,926 |
| | 6 | -1,086 | -1,086 | -1,189 | 1,009 | -1,207 | 694 | 565 | -1,172 | 4,996 | 2,731 | -266 | -743 | 2,990 |
| 2 | 7 | -4,517 | -2,433 | 2,010 | 3,144 | -4,578 | -2,874 | 4,994 | 3,849 | 1,096 | -1,615 | -2,601 | 5,750 | 91,825 |
| | 8 | -927 | -229 | -847 | -778 | 746 | 2,387 | -887 | -735 | -578 | 1,604 | -187 | -746 | 3,020 |
| | 9 | -2,091 | 9,916 | -549 | -2,087 | 110 | -2,101 | -2,155 | -2,087 | -483 | -2,125 | -2,107 | 1,054 | 8,549 |
| | 10 | -3,452 | 2,985 | 289 | 941 | -2,797 | 227 | -1,416 | 1,175 | 337 | -1,846 | 1,549 | 1,311 | 19,209 |
| | 11 | 14 | 654 | -434 | -308 | 234 | 3 | -327 | 3 | -175 | -214 | -214 | 510 | 2,880 |
| | 12 | -39 | 27 | 17 | -182 | -12 | 47 | -78 | -33 | 4 | 44 | 84 | -169 | 2,106 |
| | 1 | 27 | 147 | -48 | -210 | 5 | -216 | 12 | 77 | -79 | -73 | 485 | 23 | 2,612 |
| | 2 | 22,829 | 38,229 | 1,449 | -5,521 | -5,637 | -12,836 | -2,664 | -21,799 | -9,173 | -12,559 | -15,231 | -14,315 | 520,060 |
| | 3 | -96 | -31 | -88 | 2 | -43 | -116 | 27 | -62 | -162 | -179 | 128 | 78 | 1,760 |
| | 4 | 268 | 116 | -94 | -45 | 132 | 41 | -66 | -104 | -12 | -86 | 278 | -17 | 1,769 |
| | 5 | -118 | -102 | -141 | -105 | -39 | -104 | 267 | 241 | 86 | 153 | 132 | -27 | 1,722 |
| 3 | 6 | -96 | 1,501 | -157 | -271 | -140 | -261 | -78 | 19 | -125 | -74 | 132 | -183 | 1,882 |
| | 7 | 249 | -3,118 | 5,990 | 5,783 | -2,184 | -7,172 | 631 | -1,127 | 4,740 | -4,391 | -2,221 | -251 | 22,579 |
| | 8 | 65 | -65 | 340 | 35 | -148 | -128 | -53 | 109 | 32 | -22 | 225 | -77 | 1,912 |
| | 9 | -53 | -55 | -56 | -32 | -48 | -42 | -53 | -53 | -21 | -8 | -43 | -45 | 2,632 |
| | 10 | 722 | -426 | -297 | -379 | -479 | -246 | 48 | -252 | -248 | -219 | -43 | -258 | 2,385 |
| | 11 | -87 | -173 | -87 | -6 | -19 | -43 | 18 | -169 | 110 | 154 | 182 | -30 | 1,892 |
| | 12 | 9 | 158 | 99 | -62 | -139 | -89 | 245 | -12 | -141 | -114 | 162 | -104 | 1,997 |
| | 1 | -18 | 47 | -87 | 108 | -87 | -212 | 58 | -48 | -1 | 9 | 293 | -106 | 2,594 |
| | 2 | 21,382 | 37,747 | -1,415 | -912 | -1,086 | -10,468 | -3,586 | -20,334 | -7,120 | -9,747 | -16,491 | -15,981 | 519,788 |
| | 3 | -123 | -91 | -98 | -104 | 0 | -1 | -20 | -66 | 168 | 134 | 86 | -32 | 1,717 |
| | 4 | -184 | 98 | 9 | -234 | 140 | -125 | -71 | 1,052 | -103 | -174 | 106 | 45 | 1,875 |
| | 5 | -10 | -40 | 2 | -108 | -65 | 4 | 134 | 115 | -101 | -79 | 89 | -39 | 1,703 |
| 4 | 6 | 156 | -46 | -155 | 58 | -38 | 123 | 79 | -6 | 33 | -44 | 294 | -48 | 1,797 |
| | 7 | -120 | 1 | -145 | -32 | -19 | -100 | 66 | 260 | -141 | -43 | 178 | 66 | 1,765 |
| | 8 | -268 | -77 | -267 | -150 | 438 | -191 | 19 | -58 | -139 | -30 | -70 | 118 | 1,896 |
| | 9 | -122 | -44 | -142 | -53 | -46 | -1 | 11 | 55 | 179 | 137 | 91 | -25 | 1,766 |
| | 10 | -108 | 18 | 65 | -81 | 132 | -93 | 86 | 8 | -25 | -76 | -30 | 184 | 1,817 |
| | 11 | 28 | -36 | -7 | -124 | -92 | -55 | 330 | 120 | -101 | -100 | 79 | -36 | 1,742 |
| | 12 | -85 | -1 | -42 | 35 | -68 | 9 | -9 | 99 | 28 | 76 | 207 | -187 | 2,006 |

Figure 7.3: Difference of individual to average time of MP-LINDA per compute node and step (in milliseconds).

# Bibliography


[ABMH12] Ben Adida, Mark Birbeck, Shane McCarron, and Ivan Herman. Rdfa core 1.1. `http://www.w3.org/TR/rdfa-core/`, June 2012. W3C Recommendation.

[ABPS12] Marcelo Arenas, Alexandre Bertails, Eric Prud'Hommeaux, and Juan Sequeda. Recommendation: A Direct Mapping of Relational Data to RDF. `http://www.w3.org/TR/rdb-direct-mapping/`, September 2012. W3C Recommendation.

[ACHZ09] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets: On the Design and Usage of voiD, the Vocabulary Of Interlinked Datasets. In *Proceedings of the Linked Data on the Web workshop at the WWW Conference (LDOW)*, 2009.

[ACHZ11] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets with the VoID Vocabulary. `http://www.w3.org/TR/void`, March 2011. W3C Interest Group Note.

[ADL+09] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: Light-weight Linked Data Publication from Relational Databases. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 621–630, 2009.

[ADMR05] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and Ontology Matching with COMA++. In *Proceedings of the ACM SIG Management Of Data Conference (SIGMOD)*, pages 906–908, 2005.

[ADOM11] Luca Maria Aiello, Debora Donato, Umut Ozertem, and Filippo Menczer. Behavior-driven Clustering of Queries into Topics. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, pages 1373–1382, 2011.

[AFFG08] Alex Arenas, Alberto Fernandez, Santo Fortunato, and Sergio Gomez. Motif-based Communities in Complex Networks. *Journal Phys. A: Math. Theor.*, 41:224001, 2008.

[AHSdV11] Samur Araujo, Jan Hidders, Daniel Schwabe, and Arjen P. de Vries. SERIMI - Resource Description Similarity, RDF Instance Matching and Interlinking. *Computing Research Repository*, abs/1107.1104, 2011.

[ARS09] Arvind Arasu, Christopher Re, and Dan Suciu. Large-scale Deduplication with Constraints using Dedupalog. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 952–963, 2009.

[ATvN+12] Sören Auer, Sebastian Tramp, Bert van Nuffelen, Robert Isele, Jens Lehmann, Lorenz Bühmann, Christian Dirschl ü Pablo N. Mendes, Hugh Williams, Orri Erling, and Michael Hausenblas. Managing the Life-Cycle of Linked Data with the LOD2 Stack. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2012.

[AXH+10] Gil Alterovitz, Michael Xiang, David P. Hill, Jane Lomax, Jonathan Liu, Michael Cherkassky, Jonathan Dreyfuss, Chris Mungall, Midori A. Harris, Mary E. Dolan, Judith A. Blake, and Marco F. Ramoni and. Ontology Engineering. *nature biotechnology*, 28(2):128–130, 2010.

[BCC+13] Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. PROV-O: The PROV Ontology. `http://www.w3.org/TR/prov-o/`, March 2013. W3C Proposed Recommendation.

[BCH07] Chris Bizer, Richard Cyganiak, and Tom Heath. How to Publish Linked Data on the Web. `http://linkeddata.org/docs/how-to-publish`, 2007. superseeded by the book Heath2011.

[BCM05] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2005.

[BDG+07] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On Finding Graph Clusterings with Maximum Modularity. In *Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 121–132, 2007.

[BEP+08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the ACM SIG Management Of Data Conference (SIGMOD)*, pages 1247–1250, 2008.

[BFH+12] Christoph Böhm, Markus Freitag, Arvid Heise, Claudia Lehmann, Andrina Mascher, Felix Naumann, Vuk Ercegovac, Mauricio Hernandez, Peter Haase, and Michael Schmidt. GovWILD: Integrating Open Government Data for Transparency. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 321–324, 2012.

[BG04] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. `http://www.w3.org/TR/rdf-schema/`, February 2004. W3C Recommendation.

[BGL09] Christoph Böhm, Philip Groth, and Ulf Leser. Graph-Based Ontology Construction from Heterogenous Evidences. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 81–96, 2009.

[BGMi05] Jeffrey Baumes, Mark Goldberg, and Malik Magdon-ismail. Efficient Identification of Overlapping Communities. In *Proceedings of the ISI*, 2005.

[BH09] Daniel Bennett and Adam Harvey. Publishing Open Government Data. `http://www.w3.org/TR/gov-data`, September 2009. W3C Working Draft.

[BHBE10] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. HaLoop: Efficient Iterative Data Processing on Large Clusters. *Proceedings of the VLDB Endowment (PVLDB)*, 3(1):285–296, 2010.

[Bil] Billion Triple Challenge 2011 Dataset. `http://km.aifb.kit.edu/projects/btc-2011`.

[Biz12] Christian Bizer. Microdata, RDFa, Web APIs, Linked Data: Competing or Complementary? Panel Discussion introductory slides in Proceedings of the Linked Data on the Web workshop at the WWW Conference (LDOW), April 2012.

[BKN12] Christoph Böhm, Gjergji Kasneci, and Felix Naumann. Latent Topics in Graph-Structured Data. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 2012.

[BL05] Tim Berners-Lee. An RDF language for the Semantic Web - Notation3. `http://www.w3.org/DesignIssues/Notation3.html`, August 2005. Design Issues.

[BL06] Tim Berners-Lee. Informal Design Issues: Linked Data. `http://www.w3.org/DesignIssues/LinkedData.html`, June 2006. Design Issues, last change 06/2009.

[BLC11] Tim Berners-Lee and Dan Connolly. Notation3 (N3): A readable RDF syntax. `http://www.w3.org/TeamSubmission/n3/`, March 2011. W3C Team Submission.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284:34–43, 2001.

[BLK+09] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18:323–364, 1986.

[BLN11] Christoph Böhm, Johannes Lorey, and Felix Naumann. Creating voiD Descriptions for Web-scale Data. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 9(3):339–345, 2011.

[BLN+12] Robin Berjon, Travis Leithead, Erika Doyle Navara, Edward O'Connor, and Silvia Pfeiffer. Html5. `http://www.w3.org/TR/html5/`, December 2012. W3C Candidate Recommendation.

[BM04] Dave Beckett and Brian McBride. RDF/XML Syntax Specification. `http://www.w3.org/TR/REC-rdf-syntax/`, February 2004. W3C Recommendation.

[BMNW12] Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. LINDA: Distributed Web-of-Data-Scale Entity Matching. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 2012.

[BNA+10] Christoph Böhm, Felix Naumann, Ziawasch Abedjan, Dandy Fenz, Toni Grütze, Daniel Hefenbrock, Matthias Pohl, and David Sonnabend. Profiling Linked Open Data with ProLOD. In *Proceedings of the International Workshop on New Trends in Information Integration (NTII)*, 2010.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of ML Research*, 3:993–1022, 2003.

[BS04] Christian Bizer and Andy Seaborne. D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2004. `http://d2rq.org`.

[BvH06] Sylvain Brohee and Jacques van Helden. Evaluation of Clustering Algorithms for Protein-protein Interaction Networks. *BMC Bioinformatics*, 7:488–507, 2006.

[CAS09] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. AgreementMaker: Efficient Matching for Large Real-world Schemas and Ontologies. *Proceedings of the VLDB Endowment (PVLDB)*, 2:1586–1589, 2009.

[CFLGP03] Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez. Methodologies, Tools and Languages for Building Ontologies: Where is their meeting point? *Data and Knowledge Engineering (DKE)*, 46(1):41–64, 2003.

[CHK09] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. Data Integration for the Relational Web. *The VLDB Journal*, 2:1090–1101, 2009.

[CHM11] Richard Cyganiak, Michael Hausenblas, and Eoin McCuirc. *Linking Government Data*, chapter Official Statistics and the Practice of Data Fidelity. Springer, 2011.

[Chr07] Peter Christen. Towards Parameter-free Blocking for Scalable Record Linkage. Technical Report TR-CS-07-03, The Australian National University, 2007.

[Chr12a] Peter Christen. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24:1537–1555, 2012.

[Chr12b] Peter Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, 2012.

[CHW+08] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. WebTables: Exploring the Power of Tables on the Web. In *Proceedings of the VLDB Endowment (PVLDB)*, 2008.

[CMHJ+09] Philippe Cudré-Mauroux, Parisa Haghani, Michael Jost, Karl Aberer, and Hermann De Meer. idMesh: Graph-based Disambiguation of Linked Data. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 591–600, 2009.

[Coh00] William W. Cohen. Data Integration using Similarity Joins and a Word-based Information Representation Language. *ACM Transactions on Information Systems*, 18:288–321, 2000.

[Coh09] Jonathan Cohen. Graph Twiddling in a MapReduce World. *Computing in Science and Engineering*, 11(4):29–41, 2009.

[CP11] Eric Crestan and Patrick Pantel. Web-scale Table Census and Classification. pages 545–554, 2011.

[CSH06] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A Survey on Ontology Mapping. *ACM SIGMOD Record*, 35(3):34–41, 2006.

[CZHY12] Hong Cheng, Yang Zhou, Xin Huang, and Jeffrey Xu Yu. Clustering Large Attributed Information Networks: An Efficient Incremental Computing Approach. *Data Mining and Knowledge Discovery*, 25(3):450–477, 2012.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the International Symposium on Operating System Design and Implementation*, 2004.

[DG08] Lucas Drumond and Rosario Girardi. A Survey of Ontology Learning Procedures. In *Workshop on Ontologies and their Applications*. http://ceur-ws.org, 2008.

[DJP+94] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[DKP+09] Nilesh Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, Sathiya Keerthi, and Srujana Merugu. A Web of Concepts. In *Proceedings of the SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, 2009.

[DLE+11] Li Ding, Timothy Lebo, John S Erickson, Dominic DiFranzo, Alvaro Graves, Gregory T Williams, Xian Li, James Michaelis, Jin Zheng, Zhenning Shangguan, Johanna Flores, Deborah L McGuinness; J, and James A Hendler. TWC LOGD: A Portal for Linked Open Government Data Ecosystems. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 9(3), 2011.

[DMP12] Nilesh Dalvi, Ashwin Machanavajjhala, and Bo Pang. An Analysis of Structured Data on the Web. *Proceedings of the VLDB Endowment (PVLDB)*, 5(7):680–691, 2012.

[DPV05] Imre Derényi, G Palla, and Tamás Vicsek. Clique Percolation in Random Networks. *APS Physical Review Letters*, 94, 2005.

[DR02] Hong Hai Do and Erhard Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proceedings of the Internatial Conference on Very Large Databases*, pages 610–621, 2002.

[DSC12] Souripriya Das, Oracle Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF Mapping Language. `http://www.w3.org/TR/r2rml/`, September 2012. W3C Recommendation.

[DSFM10] Li Ding, Joshua Shinavier, Tim Finin, and Deborah L. McGuinness. owl:sameAs and Linked Data: An Empirical Study. In *Proceedings of the Web Science Conference: Extending the Frontiers of Society On-Line (WebSci)*, 2010.

[DSSM10] Li Ding, Joshua Shinavier, Zhenning Shangguan, and Deborah McGuinness. SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in Linked Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, November 2010.

[DSW+00] A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa, and V. R. Benjamins. WonderTools?: A Comparative Study of Ontological Engineering Tools. *International Journal of Human-Computer*, 52(6):1111–1133, 2000.

[DWP+07] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. Community Detection in Large-scale Social Networks. In *WebKDD and SNA-KDD workshop*, 2007.

[EFH+09] Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George A. Vouros, and Shenghui Wang. Results of the Ontology Alignment Evaluation Initiative. In *Workshop on Ontology Matching*, 2009.

[EFH+11] Jérôme Euzenat, Alfio Ferrara, Willem Robert van Hage, Laura Hollink, Christian Meilicke, Andriy Nikolov, Dominique Ritze, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Cássia Trojahn dos Santos. Results of the Ontology Alignment Evaluation Initiative. In *Workshop on Ontology Matching*, 2011.

[EFM+10] Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn dos Santos. Results of the Ontology Alignment Evaluation Initiative. In *Workshop on Ontology Matching*, 2010.

[EIV07] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19:1–16, 2007.

[ELZ+10] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: A Runtime for Iterative MapReduce. In *Proceedings of the ACM International Symposium on High Performance Distributed Computing*, pages 810–818, 2010.

[ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag, 2007.

[ETKM12] Stephan Ewen, Kostas Tzoumas, Moritz Kaufmann, and Volker Markl. Spinning Fast Iterative Data Flows. *Proceedings of the VLDB Endowment (PVLDB)*, 5(11):1268–1279, 2012.

[FB07] S. Fortunato and M. Barthélemy. Resolution Limit in Community Detection. *Proceedings of the National Academy of Sciences (PNAS)*, 104(1):36, 2007.

[FCFOP+11] Kelli de Faria Cordeiro, Fabricio Firmino de Faria, Bianca de Oliveira Pereira, André Freitas, Jo ao Vitor Villas Boas Freitas, Ana Christina Bringuente, Lucas de Oliveira Arantes, and Rodrigo Calhau. An Approach for Managing and Semantically Enriching the Publication of Linked Open Governmental Data. In *In Proceedings of the Workshop in Applied Computing for Electronic Government (WCGE), SBBD.*, 9 2011.

[Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[FHL11] Matias Frosterus, Eero Hyvönen, and Joonas Laitio. *Linking Government Data*, chapter Creating and Publishing Semantic Metadata about Linked and Open Datasets, pages 95–112. Springer, 2011.

[FKP99] Uriel Feige, Guy Kortsarz, and David Peleg. The Dense k-Subgraph Problem. *Algorithmica*, 29:2001, 1999.

[Fle71] Joseph L. Fleiss. Measuring Nominal Scale Agreement Among many Raters. *Psychological Bulletin*, 76:378–382, 1971.

[For10] Santo Fortunato. Community Detection in Graphs. *APS Physical Review*, 486:75–174, 2010.

[GBN12] Toni Grütze, Christoph Böhm, and Felix Naumann. Holistic and Scalable Ontology Alignment for Linked Open Data. In *Proceedings of the Linked Data on the Web workshop at the WWW Conference (LDOW)*, 2012.

[GDW09] Alexander Grosskopf, Gero Decker, and Mathias Weske. *The Process: Business Process Modeling using BPMN*. Meghan Kiffer Press, 2009.

[GEP08] Gunnar A. Grimnes, Peter Edwards, and Alun Preece. Instance-based Clustering of Semantic Web Wesources. In *Proceedings of the Extended Semantic Web Conference (ESWC))*, 2008.

[GGSL12] Christophe Guéret, Paul Groth, Claus Stadler, and Jens Lehmann. Assessing Linked Data Mappings using Network Measures. In *Proceedings of the Extended Semantic Web Conference (ESWC))*, pages 87–102, 2012.

[GJM09] Hugh Glaser, Afraz Jaffri, and Ian Millard. Managing Co-reference on the Semantic Web. 2009.

[gkg12] Introducing the Knowledge Graph: things, not strings. http://googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html, May 2012. The Official Google Blog.

[GMC10] B.H. Good, Y.A. De Montjoye, and A. Clauset. Performance of Modularity Maximization in Practical Contexts. *APS Physical Review E*, 81(4):046106, 2010.

[Gos11] Johannes Gosda. Überlappendes Clustering annotierter Graphen durch Motif-Perkolation. Master's thesis, Hasso Plattner Institute, 2011.

[Gre08] Steve Gregory. A Fast Algorithm to Find Overlapping Communities in Networks. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2008.

[Grü11] Toni Grütze. Holistic Concept Matching in the Web of Data. Master's thesis, Hasso Plattner Institute, 2011.

[HB11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 1 edition, 2011. `http://linkeddatabook.com/editions/1.0/`.

[HCQ11] Wei Hu, Jianfeng Chen, and Yuzhong Qu. A Self-training Approach for Resolving Object Coreference on the Semantic Web. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 87–96, 2011.

[HCZQ11] Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu. How Matchable are four thousand Ontologies on the Semantic Web. In *Proceedings of the Extended Semantic Web Conference (ESWC))*, pages 290–304, 2011.

[HEBR11] Maryam Hazman, Samhaa R. El-Beltagy, and Ahmed Rafea. A Survey of Ontology Learning Approaches. *International Journal of Computer Applications*, 22(8):36–43, 2011.

[HHK+10] Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, Kai-Uwe Sattler, and Jürgen Umbrich. Data Summaries for On-Demand Queries over Linked Data. In *Proceedings of the International World Wide Web Conference (WWW)*, 2010.

[HHM+10] Harry Halpin, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. When owl:sameAs isn't the Same: An Analysis of Identity Links on the Semantic Web. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2010.

[HHP+10] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. Weaving the Pedantic Web. In *Proceedings of the Linked Data on the Web workshop at the WWW Conference (LDOW)*, 2010.

[HHU+12] Aidan Hogan, Andreas Harth, Jürgen Umrich, Sheila Kinsella, Axel Polleres, and Stefan Decker. Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 9, 2012.

[Hic12] Ian Hickson. HTML Microdata. `http://www.w3.org/TR/microdata/`, October 2012. W3C Working Draft.

[HKL+09] Oktie Hassanzadeh, Anastasios Kementsietsidis, Lipyeow Lim, Renée J. Miller, and Min Wang. A Framework for Semantic Link Discovery over Relational Data. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 2009.

[HMZ10] Peter Haase, Tobias Mathäß, and Michael Ziller. An evaluation of approaches to federated query processing over linked data. In *Proceedings of the International Conference on Semantic Systems (I-SEMANTICS)*, 2010.

[HNST11] Melanie Herschel, Felix Naumann, Sascha Szott, and Maik Taubert. Scalable Iterative Graph Duplicate Detection. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, (preprint), 2011.

[HPUZ10] Aidan Hogan, Axel Polleres, Jürgen Umbrich, and Antoine Zimmermann. Some entities are more equal than others: Statistical Methods to Consolidate Linked Data. In *Proceedings of the Workshop on New Forms of Reasoning for the Semantic Web: Scalable & Dynamic*, 2010.

[HQ08] Wei Hu and Yuzhong Qu. Falcon-AO: A practical ontology matching system. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 6(3):237–239, 2008.

[HS12] Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. `http://www.w3.org/TR/sparql11-query/`, July 2012. W3C Working Draft.

[HSBW12] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence Journal, Special Issue on Artificial Intelligence, Wikipedia and Semi-Structured Resources*, 2012.

[HUH+12] Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker. An empirical survey of Linked Data conformance. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 14:14–44, 2012.

[HVTH12] Bernadette Hyland, Boris Villazón-Terrazas, and Michael Hausenblas. Best Practices for Publishing Linked Data. `https://dvcs.w3.org/hg/gld/raw-file/default/bp/index.html`, October 2012. W3C Editor's Draft.

[HXM+09] Oktie Hassanzadeh, Reynold Xin, Renée J. Miller, Anastasios Kementsietsidis, Lipyeow Lim, and Min Wang. Linkage Query Writer. *Proceedings of the VLDB Endowment (PVLDB)*, 2:1590–1593, 2009.

[Hyl11] Bernadette Hyland. Cookbook for Open Government Linked Data. `http://www.w3.org/2011/gld/wiki/Linked_Data_Cookbook`, December 2011.

[HZU+12] Aidan Hogan, Antoine Zimmermann, Jürgen Umbrich, Axel Polleres, and Stefan Decker. Scalable and Distributed Methods for Entity Matching, Consolidation and Disambiguation over Linked Data Corpora. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 10:76–110, January 2012.

[IB12] Robert Isele and Christian Bizer. Learning Expressive Linkage Rules using Genetic Programming. *PVLDB*, 5(11):1638–1649, 2012.

[IJB10] Robert Isele, Anja Jentzsch, and Christian Bizer. Silk Server - Adding missing Links while consuming Linked Data. In *Workshop on Consuming Linked Data*, 2010.

[IJB11] Robert Isele, Anja Jentzsch, and Christian Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. In *Proceedings of the International Workshop on the Web and Databases (WebDB)*, 2011.

[IvdMSW07] Antoine Isaac, Lourens van der Meij, Stefan Schlobach, and Shenghui Wang. An Empirical Study of Instance-Based Ontology Matching. In *Proceedings of the International Semantic Web Conference (ISWC)*, volume 4825, pages 253–266, 2007.

[JAaS+10] Cliff Joslyn, Bob Adolf, Sinan al Saffar, John Feo, Eric Goodman, David Haglin, Greg Mackey, and David Mizell. High Performance Semantic Factoring of Giga-Scale Semantic Graph Databases. Contribution to Semantic Web Challenge at ISWC, 2010.

[JC97] J.J. Jiang and D.W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proceedings of the Conference on Research in Computational Linguistics*, pages 19–33, 1997.

[JHL11] Yookyung Jo, John E. Hopcroft, and Carl Lagoze. The Web of Topics: Discovering the Topology of Topic Evolution in a Corpus. In *Proceedings of the International World Wide Web Conference (WWW)*, 2011.

[JHS+10] Prateek Jain, Pascal Hitzler, Amit P. Sheth, Kunal Verma, and Peter Z. Yeh. Ontology Alignment for Linked Open Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2010.

[JHY+10] Prateek Jain, Pascal Hitzler, Peter Z. Yeh, Kunal Verma, and Amit P. Sheth. Linked Data is Merely More Data. In *AAAI Symposium on Linked Data Meets Artificial Intelligence*, 2010.

[JMSK09] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. Ontology Matching with Semantic Verification. *Journal of Web Semantics, Science, Services and Agents on the World Wide Web*, 7:235–251, 2009.

[JYV+10] Prateek Jain, Peter Z. Yeh, Kunal Verma, Reymonrod G. Vasquez, Mariana Damova, Pascal Hitzler, and Amit P. Sheth. Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton. In *Proceedings of the Extended Semantic Web Conference (ESWC))*, 2010.

[Kar72] Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`, February 2004. W3C Recommendation.

[Kny12] Eyk Kny. Erweiterung und Optimierung eines Graph-Clustering-Verfahrens. Master's thesis, Hasso Plattner Institute, 2012.

[Kos00] Donald Kossmann. The State-of-the-art in Distributed Query Processing. *ACM Computing Surveys*, 32:422–469, 2000.

[KR10a] Hanna Köpcke and Erhard Rahm. Frameworks for Entity Matching: A Comparison. *Data and Knowledge Engineering (DKE)*, 69:197–210, 2010.

[KR10b] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data and Knowledge Engineering (DKE)*, 69:197–210, 2010.

[KS05] Yannis Kalfoglou and Marco Schorlemmer. Ontology Mapping: The State of the Art. In Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, editors, *Semantic Interoperability and Integration*, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[KSRC09] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective Annotation of Wikipedia Entities in Web Text. In *Proceedings of the ACM SIG Knowledge Discovery and Data Mining Conference (SIGKDD)*, 2009.

[KTF09] U. Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 229–238, 2009.

[KTR10] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of Entity Resolution Approaches on Real-world Match Problems. *Proceedings of the VLDB Endowment (PVLDB)*, 3:484–493, 2010.

[KTR12] Lars Kolb, Andreas Thor, and Erhard Rahm. Load Balancing for MapReduce-based Entity Resolution. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 618–629, 2012.

[LGK+12] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. *Proceedings of the VLDB Endowment (PVLDB)*, 5(8):716–727, 2012.

[LJC+09] Yang Liu, Xiaohong Jiang, Huajun Chen, Jun Ma, and Xiangyu Zhang. MapReduce-Based Pattern Finding Algorithm Applied in Motif Detection for Prescription Compatibility Network. In *Proceedings of the of the International Symposium on Advanced Parallel Processing Technologies*, 2009.

[LJPD+12] Simon Lacoste-Julien, Konstantina Palla, Alex Davies, Gjergji Kasneci, Thore Graepel, and Zoubin Ghahramani. SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases. *Computing Research Repository*, abs/1207.4525, 2012.

[LLM10] Jure Leskovec, Kevin J. Lang, and Michael Mahoney. Empirical Comparison of Algorithms for Network Community Detection. In *Proceedings of the International World Wide Web Conference (WWW)*, 2010.

[LS10] Jimmy Lin and Michael Schatz. Design Patterns for Efficient Graph Algorithms in MapReduce. In *Workshop on Mining and Learning with Graphs*, 2010.

[LTLL09] Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(8):1218–1232, 2009.

[MAB+10] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the ACM SIG Management Of Data Conference (SIGMOD)*, pages 135–146, 2010.

[McK81] Brendan D. McKay. Practical Graph Isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[MCRE09] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining Cohesive Patterns from Graphs with Feature Vectors. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 593–604, 2009.

[MEA12] Fadi Maali, John Erickson, and Phil Archer. Data Catalog Vocabulary (DCAT). `http://www.w3.org/TR/vocab-dcat/`, April 2012. W3C Working Draft.

[MGM07] Scott Meyer, Jutta Degenerand John Giannandrea, and Barak Michener. A Platform for Scalable, Collaborative, Structured Information Integration. In *International Workshop on Information Integration on the Web*, 2007.

[MP10] Fredrik Manne and Md. Mostofa Ali Patwary. A Scalable Parallel Union-Find Algorithm for Distributed Memory Computers. pages 186–195, 2010.

[MP12] Peter Mika and Tim Potter. Metadata Statistics for a Large Web Corpus. In *Proceedings of the Linked Data on the Web workshop at the WWW Conference (LDOW)*, 2012. Version as of April 14, 2012.

[MPR+10] Pierre-Nicolas Mougel, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-Francois Boulicaut. Constraint-Based Mining of Sets of Cliques Sharing Vertex Properties. In *Workshop on Analysis of Complex NEtworks ACNE co-located with ECML PKDD*, 2010.

[MS01] Alexander Maedche and Steffen Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[MSOI+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298:824–827, 2002.

[NA11] Axel-Cyrille Ngonga Ngomo and Sören Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *Proceedings of the of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[NG04] M. E. J. Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *APS Physical Review E*, 69:026113+, 2004.

[NGPC11] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. Encyclopedic Knowledge Patterns from Wikipedia Links. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2011.

[NH10] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Morgan & Claypool, 2010.

[NL12] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. EAGLE: Efficient Active Learning of Link Specifications using Genetic Programming. In *Proceedings of the Extended Semantic Web Conference (ESWC))*, 2012.

[NMS10] M. Niepert, C. Meilicke, and H. Stuckenschmidt. A Probabilistic-logical Framework for Ontology Matching. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[NNMS10] Jan Noessner, Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. Leveraging Terminological Structure for Object Reconciliation. In *Proceedings of the Extended Semantic Web Conference (ESWC))*, pages 334–348, 2010.

[NRZW11] Xing Niu, Shu Rong, Yunlong Zhang, and Haofen Wang. Zhishi.links results for OAEI 2011. In *Proceedings of the Workshop on Ontology Matching*. CEUR-WS.org, 2011.

[NSD+01] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating Semantic Web contents with Protege-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[NSW+11] Xing Niu, Xinruo Sun, Haofen Wang, Shu Rong, Guilin Qi, and Yong Yu. Zhishi.me: Weaving Chinese Linking Open Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 205–220, 2011.

[OV11] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Springer, 3rd edition, 2011.

[PDFV05] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. *Nature*, 435:814–818, 2005.

[PHHD10] Axel Polleres, Aidan Hogan, Andreas Harth, and Stefan Decker. Can we ever catch up with the Web? *Semantic Web*, 1(1,2):45–52, April 2010.

[PHM10] Eelco Plugge, Tim Hawkins, and Peter Membrey. *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing*. Apress, 2010.

[PIN+12] George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. Beyond 100 million entities: Large-scale Blocking-based Resolution for Heterogeneous Data. pages 53–62, 2012.

[PINF11] George Papadakis, Ekaterini Ioannou, Claudia Niederée, and Peter Fankhauser. Efficient Entity Resolution for Large Heterogeneous Information Spaces. pages 535–544, 2011.

[POM09] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in Networks. *Notices of the American Mathematical Society*, 56, 2009.

[PS08] Eric Prud'Hommeaux and Andy Seaborne. SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`, January 2008. W3C Recommendation.

[PSPLPMM10] Arnau Padrol-Sureda, Guillem Perarnau-Llobet, Julian Pfeifle, and Victor Muntés-Mulero. Overlapping Community Search for Social Networks. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2010.

[Rah11] Erhard Rahm. Towards Large-Scale Schema and Ontology Matching. In *Schema Matching and Mapping*, chapter 1, pages 3–27. Springer Berlin / Heidelberg, 2011.

[RB01] Erhard Rahm and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal*, 10:334–350, 2001.

[RDG11] Vibhor Rastogi, Nilesh Dalvi, and Minos Garofalakis. Large-scale Collective Entity Matching. *Proceedings of the VLDB Endowment (PVLDB)*, 4:208–218, 2011.

[RH09] Leonardo A Ribeiro and Theo Härder. Efficient Set Similarity Joins Using Min-prefixes. In *Advances in Databases and Information Systems*, pages 88–102. Springer Berlin / Heidelberg, 2009.

[RLDZ11] Chuitian Rong, Wei Lu, Xiaoyong Du, and Xiao Zhang. Efficient Duplicate Detection on Cloud using a new Signature Scheme. In *Proceedings of the international conference on Web-age information management*, pages 251–263, 2011.

[SAS11] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. In *Proceedings of the VLDB Endowment (PVLDB)*, volume 5, pages 157–168, 2011.

[Sch07] S. Schaeffer. Graph Clustering. *Computer Science Review*, 1:27–64, 2007.

[SD06] Parag Singla and Pedro Domingos. Entity Resolution with Markov Logic. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2006.

[SE05] Pavel Shvaiko and Jérôme Euzenat. A Survey of Schema-Based Matching Approaches. *JDS*, IV:146–171, 2005.

[Sea11] Andy Seaborne. SPARQL 1.1 Query Results JSON Format. `http://www.w3.org/TR/sparql11-results-json`, September 2011. W3C Working Draft.

[SH13] Steve Speicher and Michael Hausenblas. Linked Data Platform 1.0. `http://www.w3.org/TR/ldp/`, March 2013. W3C Working Draft.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago - A Core of Semantic Knowledge. In *Proceedings of the International World Wide Web Conference (WWW)*, 2007.

[SL90] Amit P. Sheth and James A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22:183–236, 1990.

[SLK+12] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Mark Birbeck. JSON-LD Syntax 1.0. `http://www.w3.org/TR/json-ld-syntax`, July 2012. W3C Working Darft, see also `http://json-ld.org`.

[SMB10] Elena Simperl, Malgorzata Mochol, and Tobias Bürger. Achieving Maturity: the State of Practice in Ontology Engineering in 2009. *International Journal of Computer Science and Applications*, 7(1):45–65, 2010.

[SMZ12] Arlei Silva, Wagner Meira, Jr., and Mohammed J. Zaki. Mining Attribute-structure Correlated Patterns in Large attributed Graphs. *Proceedings of the VLDB Endowment (PVLDB)*, 5(5):466–477, 2012.

[SPFW13] Andy Seaborne, Axel Polleres, Lee Feigenbaum, and Gregory Todd Williams. SPARQL 1.1 Federated Query. `http://www.w3.org/TR/sparql11-federated-query/`, March 2013. W3C Recommendation.

[Sta09] Rudi Staab, Steffen; Studer, editor. *Handbook on Ontologies*. Springer, 2009.

[TZS10] Duc Thanh Tran, Lei Zhang, and Rudi Studer. Summary Models for Routing Keywords to Linked Data Sources. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2010.

[Val90] Leslie G. Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8):103–111, 1990.

[VBGK09] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2009.

[W3C09] W3C OWL Working Group. OWL 2 Web Ontology Language. `http://www.w3.org/TR/owl2-overview`, October 2009. W3C Recommendation.

[WCRM09] Michael L. Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. An Entity Based Model for Coreference Resolution. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2009.

[WGM12] Steven Euijong Whang and Hector Garcia-Molina. Joint Entity Resolution. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2012.

[Wil12] Gregory Todd Williams. SPARQL 1.1 Service Description. `http://www.w3.org/TR/sparql11-service-description/`, January 2012. W3C Working Draft.

[WLB12] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology Learning from Text: A Look Back and Into the Future. *ACM Computing Surveys*, 44(4):20:1–20:36, 2012.

[Woo11] David Wood, editor. *Linking Government Data*. Springer, 2011.

[WPWCM11] Marcin Wylot, Jigé Pont, Mariusz Wisniewski, and Philippe Cudré-Mauroux. dipLODocus[RDF] - Short and Long-Tail RDF Analytics for Massive Webs of Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2011.

[WRSM08] Wensheng Wu, Berthold Reinwald, Yannis Sismanis, and Rajesh Manjrekar. Discovering Topical Structures of Databases. In *Proceedings of the ACM SIG Management Of Data Conference (SIGMOD)*, 2008.

[WWL+10] Chaokun Wang, Jianmin Wang, Xuemin Lin, Wei Wang, Haixun Wang, Hongsong Li, Wanpeng Tian, Jun Xu, and Rui Li. MapDupReducer: Detecting Near Duplicates over Massive Datasets. In *Proceedings of the ACM SIG Management Of Data Conference (SIGMOD)*, pages 1119–1122, 2010.

[XKS13] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping Community Detection in Networks: the State of the Art and Comparative Study. *ACM Computing Surveys*, 45(4), 2013. see also Computing Research Repository arXiv:1110.5813.

[XWLY08] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient Similarity Joins for Near Duplicate Detection. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 131–140, 2008.

[YCH+11] Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas Huang. Geographical Topic Discovery and Comparison. In *Proceedings of the International World Wide Web Conference (WWW)*, 2011.

[YJ06] Cong Yu and H. V. Jagadish. Schema Summarization. In *Proceedings of the VLDB Endowment (PVLDB)*, pages 319–330, 2006.

[YPS11] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. Summary Graphs for Relational Database Schemas. *Proceedings of the VLDB Endowment (PVLDB)*, 4(11):899–910, 2011.

[ZBGAA12] Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad Alkhouja. Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce. In *Proceedings of the International World Wide Web Conference (WWW)*, 2012.

[ZCY09] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph Clustering based on Structural/Attribute Similarities. *Proceedings of the VLDB Endowment (PVLDB)*, 2(1):718–729, 2009.

[ZN10] Amal Zouaq and Roger Nkambou. A Survey of Domain Ontology Engineering: Methods and Tools. In *Advances in Intelligent Tutoring Systems*, pages 103–119. Springer, 2010.

[ZRM+12] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality Assessment Methodologies for Linked Open Data. *Semantic Web*, 2012.

All links were last followed in March 2013.