FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Exploratory Search on Expert Knowledge Graphs

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Armin Friedl, BBSc.
Matrikelnummer 1053597

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Mitwirkung: PhD Marta Sabou

Wien, 30. November 2017

Armin Friedl

Stefan Biffl

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Exploratory Search on Expert Knowledge Graphs

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Armin Friedl, BBSc.
Registration Number 1053597

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Assistance: PhD Marta Sabou

Vienna, 30th November, 2017

Armin Friedl

Stefan Biffl

Erklärung zur Verfassung der Arbeit

Armin Friedl, BBSc.
Hasenstraße 38/5, 3430 Tulln

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. November 2017

Armin Friedl

Acknowledgements

First, I would like to thank my advisor Professor Stefan Biffl. His willingness to take over the patronage of this diploma thesis is what made it possible in the first place. At every step of the process he was within reach, answering any questions I had.

Furthermore, I want to thank Marta Sabou. The role she played for me during the course of this diploma thesis cannot be overrated. She went above and beyond what anyone could expect from an advisor. Her positive attitude and continuous support eased me through the most difficult times during this last journey of my studies. Her expertise and technical advice was of great help and always on point. I cannot imagine a better advisor and mentor, neither professionally nor personally.

I also want to take this opportunity to express my gratitude to my family. To my parents, for providing a secure foundation from where I could set out to the great unknown. They never failed to let me feel warm and welcomed and were as much a role model as they covered my back. And to my two lovely and caring sisters for the joy of growing up with them.

Finally, I want to thank all those that accompanied me through any step of the way. Unfortunately every acknowledgement has to reach an end and there will never be enough room to do all of them the justice they deserve. I am however eternally greatful for every single one of them and the roles they played in my life.

Kurzfassung

Die Erkundung unserer Umgebung ist ein wichtiger Bestandteil der menschlichen Aktivtät. Beginnend in unserer frühesten Kindheit, wird diese mit lernen, kognitiver Entwicklung, sozialem Verhalten und Anpassungsfähigkeit assoziiert. Obwohl ein zentraler Bestandteil unseres Verhaltens, wird der Erkundung kein angemessener Stellenwert im derzeit dominierenden Such-paradigma beigemessen, welches sich auf die einmalige Extrahierung von Elementen zu einer gegebenen Menge von Schlüsselwörter konzentriert.

Forschung im Bereich der *explorativen Suche* versucht dies zu ändern, indem ein Augenmerk auf interaktives Suchverhalten wie das Lernen und die Erkundung der Suchergebnisse gelegt wird. Für Maschinen ist es jedoch schwierig solche komplexen menschlichen Verhaltensweisen zu unterstützen. Dennoch hätte ein exploratives Suchsystem einen enormen Einfluss. Unternehmen in der ganzen Welt haben Schwierigkeiten ihr intellektuelles Kapital zwischen Mitarbeitern zu transferieren. Ein System, welches Mitarbeiter bei der Exploration und dem Erlernen dieses impliziten Wissens hilft, ist nicht nur ein finanzieller Vorteil, sondern trägt auch dazu dabei, dass Wissen nicht verloren geht.

Obwohl kürzlich vielversprechende Ergebnisse im Bereich des Semantic Web publiziert wurden, ist wenig über mögliche Basisalgorithmen für explorative Suchsysteme bekannt. Einer der hervorstechendsten Merkmale von Daten im Semantic Web ist deren Graph-Struktur. Die Diplomarbeit untersucht zwei Arten von Graphalgorithmen und deren Eignung für explorative Suche: Zentralitäts- und Ähnlichkeitsmetriken. Drei Algorithmen wurden im kürzlich publizierten Gather-Apply-Scatter Modell für die Verwendung mit Triple Stores neu formuliert. Zusätzlich wird eine neue Variation der auf Information Content basierenden Ähnlichkeitsheuristiken vorgeschlagen. Eine kürzlich publizierte und speziell für das Semantic Web entwickelte Ähnlichkeitsheuristik rundet den Mix ab.

Die Algorithmen wurden auf eine Unterart von semantischen Graphen — Expert Knowledge Graphs — angewandt, welche domänenspezifisches aber komplexes Expertenwissen enthalten. Die Ergebnisse von zwei verschiedenen Expert Knowledge Graphs wurden statistisch, wie auch qualitativ, in Hinblick auf explorative Suche evaluiert und verglichen. Während die Zentralitätsmetriken relevante Knoten effektiv herausfiltern konnten, war die Differenzierung weniger relevanter Knoten kaum gegeben. Die Ähnlichkeitsmetriken zeichneten sich durch hohe Anforderungen an die Struktur des modellierten Wissens aus. Waren diese erfüllt, so wies die neu vorgeschlagene Ähnlichkeitsheuristik die vielversprechendsten Resultate aus.

Abstract

Exploration of objects and the environment around us is an important part of human activity. Starting in our early infancy, it is closely associated with learning, cognitive development, education, social behavior and adaptability. Although exploration is such a vital trait, it did not receive proportional attention in today's dominating lookup-based search paradigms, which are typically only focused on the one-off retrieval of a set of items matching a set of keywords.

Research in *exploratory search* set out to change that by including interactive search activities like learning and investigation of the result space. But supporting such elaborate human traits is notoriously hard for machines to do. However, being able to build a system that assists users in exploratory search would have a tremendous impact. Businesses around the world are struggling with the transfer of hard-built intellectual capital between their employees. Helping them to explore and learn this expert knowledge is not only financially beneficial, but also prevents knowledge from getting lost or stale.

Recent advances using semantic web technologies show promising results. Still, little is known about which algorithms could be used at the core of an exploratory search system. One of the distinguishing properties of data specified via semantic web standards is their graph structure. This diploma thesis investigates two groups of graph algorithms and their eligibility for exploratory search: centrality and similarity metrics. To that end, three well-known algorithms were redesigned in the recently published Gather-Apply-Scatter graph processing paradigm for use in triple stores. Additionally, a novel variation of information content based similarity metrics is suggested. The mix is completed by a recently published similarity heuristic specifically designed for the semantic web.

The algorithms were applied to a special kind of semantic graphs — Expert Knowledge Graphs — featuring enterprise-scale, domain-specific but complex expert knowledge. The results from two different expert knowledge graphs were evaluated and compared statistically, as well as qualitatively, particularly with respect to exploratory search. While the evaluated centrality metrics turned out effective in suggesting high-profile nodes, they fell short on distinguishing between less important nodes. The tested similarity metrics were found to make high demands on the structuring of the encoded knowledge. If met, the newly suggested similarity heuristic turned out to yield the most promising results.

Contents

CHAPTER 1

Introduction

Exploration of objects and the environment around us is an important part of human activity. Starting in our early infancy, it is closely associated with learning, cognitive development, education, social behavior and adaptability [AB70, WM76, Haz82, Ruf84]. Although exploration is such a vital trait, it did not receive proportional attention in what is probably the most dominating search paradigm today: lookup-based information retrieval [Mar06].

Lookup-based information retrieval is often associated with keyword-based search systems such as Google[1], Bing[2], or Yahoo[3]. However, as an umbrella term it also contains more formal techniques such as SQL-queries conducted on relational databases. All of them have in common that the user must have a rather thorough understanding of the search domain and the items he or she wants to retrieve [Mar06]. Therein lies also a subtle difference between Exploratory Search (ES) and lookup-based information retrieval. The existence of an item or answer in lookup-based systems is already known (or at least presumed) beforehand. The exploratory searcher on the other hand has no, or only a very vague, conception of the items he or she wants to explore. It is the task of the Exploratory Search System (ESS) to assist the searcher in exploring the domain of interest and suggesting relevant items [WR09].

Figure 1.1 shows one conceptual overview of ES according to [Mar06]. *Learn* and *Investigate* are the defining activities of ES. However, they do not only depend on each other. Interactions with the related activity *Lookup* are common too. The main problem every ES systems tries to solve is to either assist the user during these activities or to enable them in the first place.

[1] `https://www.google.com`
[2] `https://www.bing.com`
[3] `https://www.yahoo.com`

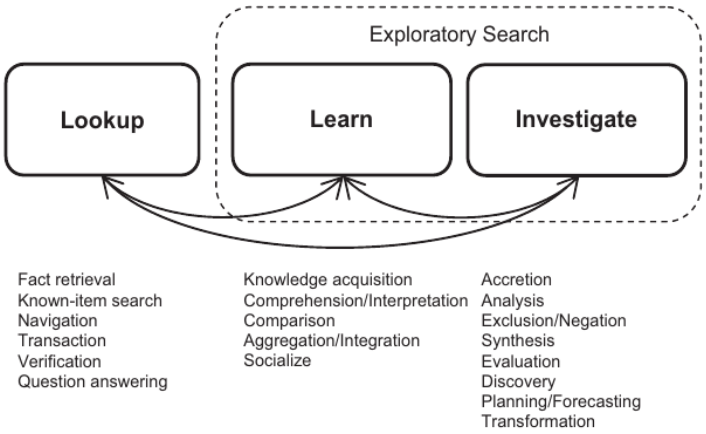


Figure 1.1: *Learn* and *Investigate* as core activities of ES. Arrows indicate interactions between the core activities as well as with the related activity *Lookup*. Figure taken from [Mar06].

As already mentioned shortly in the abstract, one potential use case of exploratory search is to support businesses and their employees to transfer knowledge accumulated over time. Being able to exploit and manage such intellectual capital is a competitive advantage for companies [KRSV14, AK12].

But this example stands for a much larger problem. The sophistication and accumulation of knowledge in our modern world seems to accelerate only ever faster. Although the legend of the last polymath is attributed to many, they all have one thing in common — they are long dead. To keep up with this development it may be even essential in the future to have assistance in exploring and learning a field of expertise to an expert-level degree.

But already today, for many constant learning and investigation of expert knowledge is a integral part of their lives. An ESS can help in bringing order to complex expert knowledge and making it more accessible. It can also assist the aspiring expert to learn more efficient in a more enjoyable way.

The increasingly widespread availability of semantic data opens new perspectives on ESS. In particular, it has been argued that semantic elements are well suited to support exploration [DLT+13, WKW+10]. Several W3C Recommendations (e.g. Resource Description Framework (RDF) [CWL14] and Web Ontology Language (OWL) [MW04]) specify means to capture data and its semantics in machine-processable formats. Leveraging these semantic web technologies in ESS research became popular recently [MG14].

However, most research in this area concentrates on encyclopedic knowledge graphs like Freebase[4] or DBpedia[5] [MG14]. But semantic web technologies are well suited for

[4] `http://freebase.com`
[5] `https://dbpedia.org`

encoding small-scale, complex and domain-specific expert knowledge too.

An example are Expert Knowledge Graphs (EKGs) such as the STAR knowledge base created at TU Vienna in cooperation with Siemens [MES+17]. This knowledge base encodes Software Architectural Knowledge collected over time at Siemens. Originating from research on STAR, the question arose how the encoded knowledge could be leveraged for ES. A seemingly simple question for, as it turned out, a deep, elaborate and largely unsolved problem. Framed by the STAR research the following diploma thesis tries to contribute a small part to the vast topic of ES.

1.1 Aim of the Work

ES has an „[...] unclear and open-ended definition“ [PGGT17]. Due to the intricate nature of exploratory search, attempts at conceptualizing and defining ES on a generic level are usually quite elaborated [PGGT17, WR09]. The very first question that arises is therefore what ES actually is. A conclusive definition of ES does not seem viable currently. However, in order to gain a better understanding of ES various approaches are reviewed and integrated in Section 2.2.

One specialty of the STAR KB was the fact that there existed already a faceted user interface developed in-house by Siemens. The fact that the user-facing part ought not to be altered outside Siemens sprouted a research question (RQ) that was not yet addressed much in current literature:

RQ–I. *How can exploratory search on semantic web technologies be supported algorithmically?*

Hence, the focus of the diploma thesis is on algorithmic methods for exploratory search systems (in contrast to view-based approaches as in [MG14]). This excludes research that is concerned with how to represent the data to the user for exploration (e.g. force-directed layouts [DLT+13], or timelines for temporal data [HMK05]).

One proposed solution investigated in this thesis builds upon a special property of semantic web technologies: The immediate representability as a graph structure. Several heuristics were developed in previous research to estimate the importance of, or similarity between, vertices in a graph. Some of these algorithms are reviewed for ES in Section 2.3. Subsequently an in-depth examination of selected algorithms is provided in Section 2.4.

Most of the research on ES in the semantic web is in the form of case studies bound to a single ontology only [MRDNDS10, Res95, Pas10]. Although this diploma thesis is rooted in the STAR KB too, the hope was to find generic methods applicable to various similar datasets. To this end two related questions are answered in this diploma thesis:

RQ–II. *What are the characteristics of datasets like STAR that are required by the algorithms?*

In order to answer this question the notion of EKGs is introduced in Section 2.1. EKGs are the accumulation of various concepts and terms from existing literature that were

concentrated in a definition for a class of knowledge graphs defined via semantic web technologies.

Additionally in Chapter 5 a close look at the results of the algorithms conducted on two different datasets is taken. In particular, those cases where the results were not satisfactory hint at preconditions that must be fulfilled by the datasets in order to be supported by the algorithms.

Being able to state requirements that must be fulfilled by the datasets is important. However, the closely connected question arises how well algorithms can be ported to other datasets. This is a question seldom considered in current literature and leads directly to the next research question investigated in this diploma thesis.

RQ–III. *To what extent is it possible to transfer the investigated methods to different datasets?*

In order to answer this question, MusicPinta [DLT+13] was chosen as a second EKG (besides STAR) for investigating the portability of the implemented algorithms to different EKGs. MusicPinta itself is an aggregation of information about musical instruments, artists, pieces of music and customer reviews from various sources. Consequently, the evaluation in Chapter 5 was conducted on both datasets which confirmed the portability of the algorithms to different EKGs.

One of the major components in the semantic web ecosystem are triple stores. Triple stores are datastores implementing specifications of the semantic web stack like inference, consistency checks, and the SPARQL Protocol and RDF Query Language (SPARQL). Especially considering that most EKGs like STAR are used in an applied setting, portability may be also seen as being able to work in the prevalent environment these datasets are embedded in. Hence, one goal was to build upon, and integrate with, existing triple stores.

The intended means to access a triple store is via the provided SPARQL interface. However, due to a lack of a loop construct many algorithms cannot be expressed via SPARQL. Also, SPARQL is a graph query language not well suited for algorithms demanding graph processing features.

Current ES research falls largely into two categories with respect to triple stores. Either the proposed approaches are chosen to be well suited for a SPARQL based solution. Or special prototypes are developed that effectively circumvent triple stores, thereby also loosing their advantages. Unsatisfied with both of these solutions, the following research question arose:

RQ–IV. *How can we implement the proposed algorithms based on a common triple store?*

Some of the proposed algorithms are well suited for a SPARQL based implementation and can therefore leverage the common query interface of triple stores. However, many of the general graph algorithms need efficient access to the graph structure integral to every EKGs. Since triple stores can be considered black-boxes only accessible through defined interfaces like SPARQL, this access has to be provided by the triple store.

One potential solution was found in the Gather Apply Scatter (GAS) computational model discussed in Section 2.5. At least one common triple store (Blazegraph [6]) implements this interface. In order to use it, the graph algorithms had to be reformulated in this new paradigm. The newly suggested implementations are shown in Chapter 4.

Finally, the proposed solutions had to be evaluated too. Hence, the last major research questions answered in this diploma thesis is:

RQ–V. *Which of the proposed algorithms work best? What are potential shortcomings and why?*

To address these questions a statistical analysis of the computations yielded by the implemented metrics was conducted in Chapter 5. To circumvent single observation outliers the evalutations were performed on two different EKGs. Indeed, several interesting results were found that could also spawn topics for further research in this area. Although covering only a narrow notion of „best“, the results also indicate at an answer to this last research question.

1.2 Contributions

By answering the aforementioned research questions, this diploma thesis provides several contributions to advance the current state of ES on EKGs:

- The identification of algorithmic methods (metrics) that could potentially support ES.
- An attempt at transferring the wealth of previous research on graph algorithms to triple stores for ES in the semantic web via the GAS computational model.
- The first formulation of the Load and Accessibility (ACS) algorithms in GAS.
- An attempt at fostering the generalisation of ES methods over various datasets in the semantic web by defining dataset-independent charactersitics of EKGs.
- A novel suggestion for an information content based similarity metric
- An statistical and qualitative analysis of the suggested metrics evaluated on two different datasets.

1.3 Structure of the Work

This diploma thesis is structured as follows. Chapter 2 introduces the concepts used during the rest of the work. This includes (i) the definition of EKGs by extracting important characteristics of STAR and conglomerating various existing terms and definitions

[6] `https://www.blazegraph.com/`

in Section 2.1, (ii) a review of existing literature about ES in general and ES in the context of semantic web technologies in particular in Section 2.2, (iii) an overview of some metrics that are specifically designed, or could be leveraged, for ES (Section 2.3), (iv) an in-depth presentation of selected metrics in Section 2.4, (v) a presentation of the GAS paradigm for graph algorithms in Section 2.5.

In Chapter 3 the approach to implement and evaluate the metrics is presented and discussed. Chapter 4 then presents the implementation of the selected metrics, particularily with respect to their reformulation in GAS. The results obtained by running the metrics on two different EKGs are analyzed statistically in Chapter 5.

In the second to last section, Chapter 6, the research questions are revisted again and discussed with respect to the results obtained during this diploma thesis. Finally, Chapter 7 summarizes the results and provides an outlook for future research topics in this area.

CHAPTER 2

State of the Art

2.1 Expert Knowledge Graphs

When we refer to Expert Knowledge Graphs (EKGs), we refer to a concept that is not yet defined as such in current literature. During this section various related terms from literature are integrated to arrive at a useful notion of EKGs. Additionally, the technological basis — semantic web technologies — and their relation to EKGs are discussed during this section.

The first perspective taken is that of human experts. The earliest modern study about human experts is the analysis of expert agriculture judges by Hughes in 1917. Unfortunately the original study became unavailable or lost. However, a later reanalysis of the data pointed out the low correlation between expert judgement and actual crop yields [Wal23]. This in turn sparked loads of research pinpointing various deficies of experts and expert knowledge [Sha92].

There are probably more questions than answers when it comes to expert knowledge. Nevertheless, two general — and usually agreed upon — properties may be claimed here. First, expert knowledge is domain specific and constrained in its scope. Experts do not neccessarily exhibit specific traits outside their field of expertise [Sha92]. Second, knowledge of an expert within his or her domain of expertise is usually perceived as highly structured, detailed and interconnected [Hof98].

Adapted to EKGs these two properties of expert knowledge can be summarized as follows. EKGs are highly *specific*, often encoding a single, narrowly defined domain. Furthermore, they are comparatively *small* (compared to e.g., general knowledge graphs). EKGs are also deeply structured with sophisticated and *detailed interconnections* between their domain concepts.

This definition of EKGs also adheres to an advantageous technical perspective. Due to their constrained size (in comparison to web-scale general knowledge graphs) they

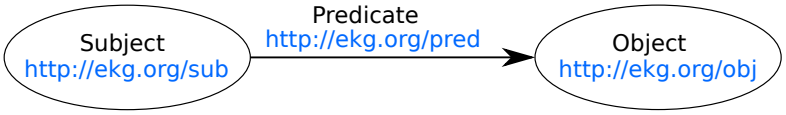


Figure 2.1: Model of an RDF Triple consisting of a subject, a connecting predicate and an object. All of them assigned to a unique label IRI.

allow for more demanding processing than would be possible if scaling would be of great concern. The detailed structure potentially gives rise to the possibility of extracting richer information.

Another important concept of EKGs is the graph structure the knowledge is encoded in. Definitions of graphs can be found in every introductory text about the subject (e.g., [Wil96]). In particular, EKGs are directed multi-graphs:

Definition 2.1.1 (Directed multi-graph)**.** A *graph* $G = (V, E)$ consists of a non-empty, finite set $V = V(G)$ of elements called *vertices* (or *nodes*), and a finite set $E = E(G)$ of *edges*. An edge $e \in E(G)$ is a pair of vertices $v_1, v_2 \in V(G)$. An ordered pair of vertices $(v_1, v_2) \in E(G)$ with $v_2, v_2 \in V(G)$ is called a *directed* edge. An unordered pair of vertices $\{v_1, v_2\} \in E(G)$ with $v_1, v_2 \in V(G)$ is called an *undirected* edge. If all edges $e \in E(G)$ of a graph G are ordered (unordered), G is called directed (undirected). A *multi-graph* $G' = (V', E')$ consists of a finite multi-set $E' = E'(G')$ of edges. That is, it allows for multiple edges between any two nodes $v_1, v_2 \in V'(G')$.

Although EKGs could be left at the general notion given above, in this thesis they are also tightly bound to a particular technology: the semantic web stack. On the knowledge engineering level different specifications can be used for encoding knowledge. Examples are the OWL [BCG+12] for (asserting) ontologies or Simple Knowledge Organization System (SKOS) [BM09] for (unasserting) knowledge organization schemes[1].

The higher level specifications mentioned above are built upon the RDF model [CWL14]. The core structure of RDF consists of subject-predicate-object triples Figure 2.1. A set of triples naturally corresponds to a directed multi-graph (see Definition 2.1.1). Every node and edge is assigned a label. For non-blank nodes the label must be an Internationalized Resource Identifier (IRI) as defined in [DS05].

Note that, already from the definition of a multi-graph, the assigned labels have to uniquely identify (i) a vertex globally, and (ii) an edge within the edge set between two vertices[2]. However, IRIs do not uniquely identify edges globally. Which is also why we did not define the graph a *labeled* directed multi-graph to avoid confusion with mathematical definitions.

[1] SKOS schemes, in contrast to OWL, do not assert facts or axioms about the world, but rather just state facts about the particular encoded knowledge organization scheme. For the lack of a concise term we call this unasserting. For a discussion of the differences see [BM09].

[2] Due to the vertex set $V(G)$ and any edge set $e \in E(G)$ being sets

Due to its simplicity, RDF is a very general model. Flexible enough to represent various information. In particular, it is flexible enough to provide the basis for OWL itself, ontologies defined via OWL, and also further schemes leveraging the already defined RDF representation of OWL to define different semantics for knowledge representation such as SKOS. Note that all of these specifications, by having a defined representation in RDF, are automatically representable as a graph structure too.

Taken together, an Expert Knowledge Graph is then defined as:

Definition 2.1.2 (Expert Knowledge Graph)**.** An Expert Knowledge Graph is any knowlege-encoding directed multi-graph based on the RDF model and semantic web technologies. EKGs are specific, covering a narrowly defined domain of expertise. They are small enough in size to allow for complex processing procedures to run in reasonable time. Furthermore, EKGs are deeply structured with meaningful and detailed interconnections between their domain concepts.

This definition is still purposefully open-ended. However, it should capture a class of well-behaving knowlege graphs (with respect to processing methods), while still representing a useful subset of all possible ones. The first two properties help to concentrate on a specific technological stack. Every knowledge base defined in the semantic web stack fulfills these properties (being based on RDF and in turn representable as directed multi-graph).

The size properties vary with advances in technology. It is useful to allow for investigations of demanding methods without having to care for web-scale scaling issues. Nevertheless, especially in the semantic web context, scaling is a desirable property.

The deep structure and meaningful, detailed interconnections are necessary to allow for extraction of information. It is related to the folkloric dictum in machine learning: "Garbage in, garbage out". That is, a high-qualitative data basis with enough information to work on is a basic necessitiy of almost all algorithms. What is an acceptable level of quality may vary with the method in question and is a useful property to investigate. It is, however, not a major concern for algorithms working on EKGs since a qualitative data basis is axiomatically assumed.

One shortcoming of a majority of current research on ES in the semantic web is the focus on special-purpose prototypes applicable to a single dataset only. Properties of EKGs as defined above are often implicitely assumed. We deem this lack of generalizability a gap in current research. By operationalizing a notion of EKGs, the investigations in this diploma thesis are made independent from a specific dataset, relying only on a small amount of assumptions.

2.2 Exploratory Search

As already discussed in Chapter 1, Exploratory Search (ES) is oftentimes contrasted with lookup-based information retrieval, the currently predominant search paradigm. In the

first part of this section, research that tries to conceptualize ES is presented. Towards the end, current literature that attempts to exploit semantic web technologies for exploratory search is reviewed. Finally, the area of ES considered during this thesis is narrowed down.

Figure 2.2 shows the conceptualization by [Mar06] again. In this model ES consists of the two main activities learn and investigate. These activities are interrelated as learning is not cleanly separated from investigation and vice versa. Also, lookup based search can be an integral part for both of these activites. The activities are further refined by terms derived from Bloom's taxonomy of educational objectives [BEF+56].

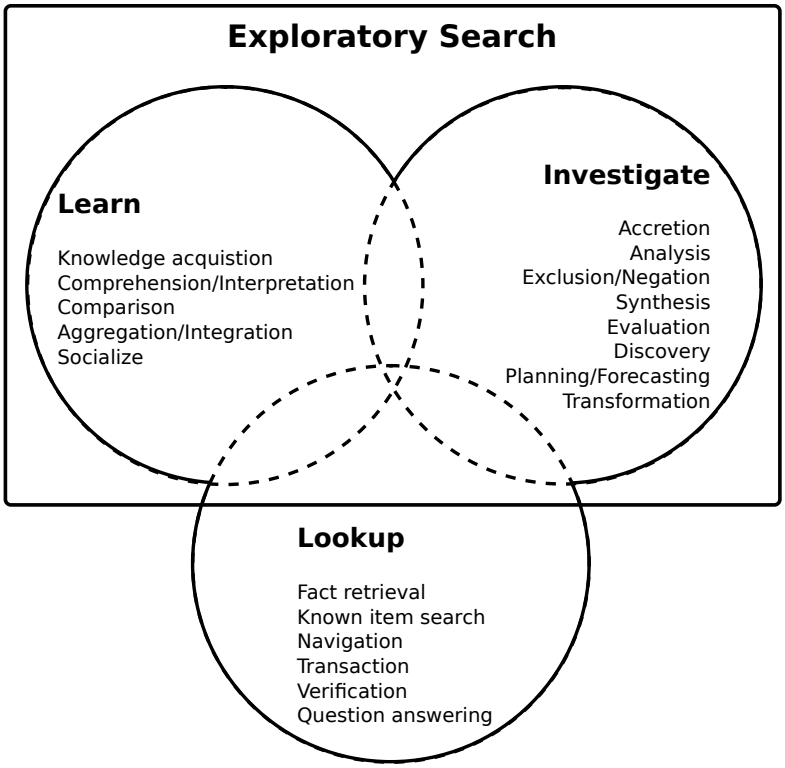


Figure 2.2: Learn and Investigate as core activities of ES. Intersections indicate overlaps between the core activities as well as with the related activity Lookup. Figure after [Mar06].

Although this model is conceptually simple, [WR09] criticizes that it does not capture the *process* of an exploratory search well enough. A pictorial model of this process was suggested as being akin to *berry picking*. In this analogy the search process consists of finding and selecting small bits of information (just like berries on a bush). These small bits constantly inform the search process to find new and refined bits of information [Bat89].

This is arguably also how lookup-based search systems are used. New lookups are constantly conducted after refining and changing keywords according to newly found information. An example is literature search where a searcher may start with entering

some general keywords about the topic of interest. After learning about prominent authors in the field, the searcher looks up papers from these authors. ISBN and DOI numbers retrieved from the bibliography sections may then be used to explicitly look up further literature, etc. [Bat89]. Although this process can be simulated with lookup-based information retrieval systems, it is not only a time-consuming but also a manual and laborious activity. Ideally, an ESS would offload some of this work from the user to the system.

The question may arise if there is even a difference between lookup-based and exploratory search. A recent study tried to answer that. Users were asked to conduct different tasks on a keyword-based search system. The tasks were selected according to Marchionini's model already presented in Figure 2.2. The predicted categorization of search and their accompanying key tasks can be found in Table 2.1. A total of four different search categories were suggested. Moreover, the transition from lookup to ES is suggested to be a smooth one. Starting from core lookup, borderline lookup is already on the edge to borderline exploratory. Core exploratory captures only clearly identifiable exploratory tasks. However, according to the tested assumption the four categories are still distinguishable [AGJ+16].

Several variables like gaze distribution, completion time and scroll depth were measured. Whereas gaze distribution showed no significant differences, completion time and scroll depth were significantly higher for exploratory tasks. These results suggest that there is indeed a difference between lookup and ES and that the current theoretic conceptualizations resemble these differences [AGJ+16].

Goals	Complexity	Search Category	Key Tasks
Precise	Low	Core lookup	Fact-finding, Navigation
Precise	High	Borderline lookup	Question answering
Open-ended	Low	Borderline exploratory	Comparison
Open-ended	High	Core exploratory	Knowledge acquisition, Planning

Table 2.1: Distinguishable categories of search from core lookup to core exploratory, together with their respective goals, complexity and key tasks. A priori predicted and empirically validated by [AGJ+16].

In the aforementioned study, lookup and ES were operationalized along two dimensions: Complexity and Goals. Whereas complexity is also a property of the ES task, goals is a property of the user. The user is also an equally important task for a model of ES extracted by a survey on ES literature conducted in [PGGT17].

In their survey Palagi, et al. [PGGT17] extracted eleven high level characteristics commonly used in various studies about definitions and models of ES. They divide these characteristics into two larger, overlapping domains considered in ES systems: The ES task itself and the user domain. This conceptualization of ES is depicted in Figure 2.3.

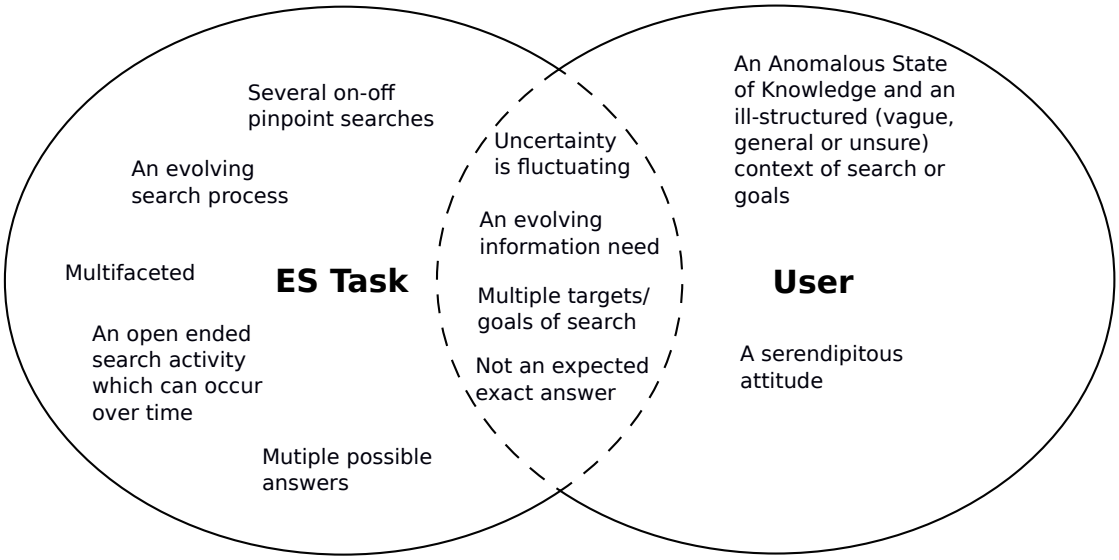


Figure 2.3: High-level characteristics of ES commonly used in various ES models categorized into two larger, overlapping domains. After a figure in [PGGT17]'s review study.

Since most of the terms used are intuitively clear, only the following terms that may not be familiar to the reader are explained here: (i) Multifaceted, (ii) Anomalous State of Knowledge and, (iii) Serendipitous attitude. For a complete description the interested reader may refer to [PGGT17].

Multifaceted search is a property of the user interaction. During exploratory search, the user repeatedly selects facets that restrict the search space to a subset. By adding facets the search space can be narrowed down until a specific information is found. Also, different subsets of facets can be used to explore the search space. Note how a multifaceted search paradigm does not demand specific domain knowledge from the user [PGGT17].

Anomalous State of Knowledge refers to the state the user is assumed to be in when starting the ES. Although the user has a motivation to start the search, a precise idea of what he or she is actually looking for is lacking. Also, although the user has a vague objective, a definite plan to attain it is missing in the user's mind [PGGT17].

The last term explained here is the serendipitous attitude an exploratory searcher is assumed to exhibit. The user is open to surprise and tends to pay attention to it. This open-mindedness allows to exploit unexpected elements for further discovery and adaption of search strategies or goals [PGGT17].

ES can be seen as even much broader than what was discussed so far. For example the problem context and search behavior can be taken into account too [WR09]. However, the main terms and concepts of ES are now established. Also, as stated in [PGGT17]:

"[ES] is a loosely defined concept as its definition is not stable and continues to evolve every time new systems are being developed". Hence, a conclusive definition of ES does not seem to be viable currently.

However, the conceptualizations and models discussed so far give rise to a good enough working definition for research on the various sub-problems of ES. Marie and Gandon [MG14] reviewed linked data based ES systems and how the characteristics of ES translated to features in these systems.

According to their survey most systems concern themselves with different visualizations of the semantic graph. Faceted search systems are among the most prominent of these visualizations. In addition to mere presentation of the data, they also allow the user to restrict the search space and explore the data by selecting different subsets of facets.

As an example of a faceted search system *mSpace* is presented here. Faceted search systems try to map the multi-dimensional data into a two-dimensional view. In mSpace facets for the domain of music are grouped into different facet groups like eras, composers, and pieces. Facet groups are ordered horizontal columns containing the facets of each group. The order is significant. That is, the left-most column (e.g., eras) is the most significant one [SWRS06].

Facets selected on the left (e.g., classical era) restrict possible facets to the right (e.g., only composers of the classical era are shown). Additionally, the columns can be reordered and different groups can be selected (e.g., Albums instead of Places). An exemplary state of the facet visualization is depicted in Figure 2.4. One drawback of the mSpace system is that a new RDF model for every dataset has to be manually specified in order to guide the system which groups and facets should be shown [SWRS06].

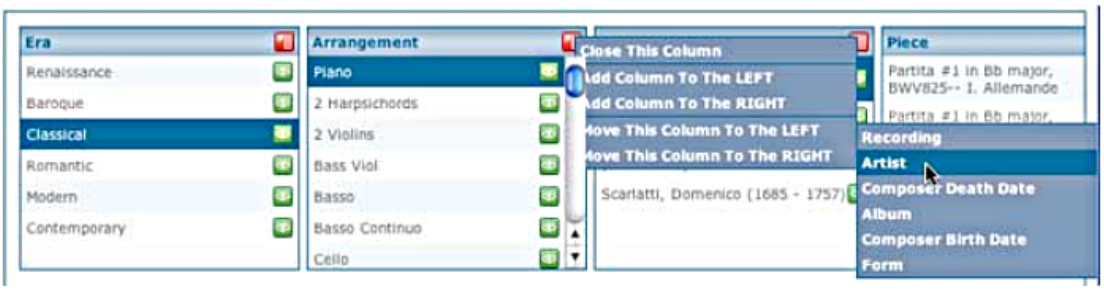


Figure 2.4: Exemplary user interface of a faceted search system. Taken from mSpace [SWRS06].

Faceted search systems are probably the most prevalent in current exploratory search systems on semantic web technologies. The reason for that is probably that the structure of semantic web ontologies defined via OWL or RDF Schema (RDFS) consist of classes and properties which can be rather directly mapped to facets.

facet for example lifts the requisite for manually defining the mapping between the dataset, facet groups and facets. Instead, it automatically extracts that information from

RDFS classes and properties [HvOH06]. Other systems investigate different visualization paradigms. For example *gFacet* visualizes the facets in a graph visualization instead of hierarchical columns [HZL08].

Although user interaction and interface design is an important part of ES systems, the question is if semantic graphs can be leveraged for ES systems that go beyond exposing classes and properties via facets.

Indeed, several prototypes where developed that investigate other parts of ES. *DBpedia Ranker* uses external services like the amount of search results returned by Google to estimate the similarity between entities in the semantic graph [MRDNDS10]. However, the question remains how the semantic graph itself (and not external lookup-based services) can be leveraged for ES beyond providing a convenient basis for facets.

A recommender system based on similarity was suggested by [DNMO+12]. This work was selected as being representative of a class of prototypes that are tightly bound to one specific dataset. Similarity in this prototype is calculated by leveraging specific, predefined properties of movies in e.g. DBpedia like `dbpedia-owl:director`. However, these properties have to be manually re-defined for each domain. Depending on the data this may not be possible.

A short look at recommendation systems that do not leverage semantic web technologies should be taken here too. An overview of current recommender systems is given in [AT05]. These systems are tailored to specific use cases. An example are recommendations based on user ratings. These techniques can be used if user ratings (and accompanying data about users) are encoded in a semantic graph. However, the domain of semantic graphs are potentially more general and can contain almost any information. The goal of classical recommender systems diverges from the needs of ES on semantic graphs. Depending on the dataset they can still be a rich resource for specialized ES systems on semantic graphs too.

One way to leverage semantic web technologies for enhanced ES is to use the graph representation of the information. Research on graphs have a long history and yielded a huge amount of algorithms and suggestions for various tasks. However, bringing this wealth of knowledge over to the semantic web world for ES is not a trivial task.

The intended access to the semantic web data for applications is by using the SPARQL interface of triple stores. SPARQL, being a declarative querying language, does not allow for more involved processing procedures. For example, there is no loop or recursion construct defined in SPARQL (besides property paths which have restricted use cases) [HS13]. Algorithms with statically unknown bounds are therefore prohibited. This is also reflected in previous research. The majority of ES systems on semantic web technologies is concentrated on user interface and interaction design [MG14].

In order to develop common graph algorithms that can be leveraged for exploratory search, the underlying graph structure exposed by RDF has to be available to the algorithms. One way is to circumvent this problem by developing prototypes that read the RDF data

into a graph structure. However, the facilities and advantages (e.g., consistency checks, SPARQL queries, inferences) that are provided by widely available triple stores are then lost or would have to be reimplemented.

Previous work did not spend enough attention to transferring the wealth of knowledge accumulated in graph algorithm research to commonly used triple stores. However, the requirements of graph algorithms demand special provisions that are currently not widespread among these triple stores (see also Section 4.2). The model used during this diploma thesis is the GAS paradigm presented in Section 2.5. But first a discussion of graph algorithms and selected metrics that are potentially useful for ES are presented in Section 2.3 and Section 2.4

2.3 Classification and Overview of Metrics

Research on graph algorithms has a long history and a variety of measurements were developed that try to capture different features of the graph [New10]. Due to the sheer amount only a small part of measurements that are potentially useful for ES can be reviewed here.

The Resource Description Framework (RDF) is the fundamental representation format underlying the semantic web technology stack. As already discussed in Section 2.1, RDF directly translates to a graph model. It therefore seems useful to transfer the knowledge aggregated in graph algorithms to triple stores containing the semantic knowledge graph. However, many of the general graph algorithms below do not lean themselves well towards a SPARQL based algorithms.

Current research in ES based on semantic web technologies does not address this problem. Sometimes the main focus is a different one, such as user interface and interaction design. Other times, research invents novel measurements that lean themselves well towards SPARQL based processing (e.g., [Pas07]). Yet other research loads the RDF data into custom data structures to support a prototypical implementation, sacrificing the advantages of triple stores which support techniques such as inferences and consistency checks (e.g., [FSSS09]).

From a viewpoint of EKGs that are deployed on standard triple stores, none of these solutions seems satisfying. Additionally, a way to transfer algorithms developed for general graphs to the semantic web would open up the floodgates to a wealth of research that can be advantageous for ES.

The first category of graph algorithms discussed in this section are so-called centrality metrics. An overview is given in Table 2.2 on page 16. They generally try to estimate the importance of a node in the graph as a whole.

The centrality of a node for ES is useful for at least the following two reasons:

- As already mentioned in 1, in comparison to lookup-based information retrieval, the exploratory searcher is not assumed to be familiar with the domain. It is therefore

important to provide a starting point in the exploration of the knowledge graph. Centrality metrics can provide just that. The more central a node is, the more importance it bears in the graph. The notion of importance, however, varies with the different metrics. Selecting the best fit (or a combination thereof) may be subject to the specific use case.

- Furthermore, centrality metrics can also provide guidance during further exploration of the knowledge graph. Suggestions for exploration worthy nodes can be weighted according to their centrality. An example is the manifold faceted search systems suggested by research in semantic databases (e.g. [AGK+16, HZL08, HvOH06]). Although they are tailored to a specific dataset, all of them could integration the suggested centrality measures to weight instances and facets and present them to the user accordingly.

Metric	Description	Reference
PageRank	Estimates the probability of arriving at a vertex when randomly following edges. Calculated by vertices spreading commodity (current PageRank) equally along outgoing edges until a convergence criteria is met.	[BP98]
Katz	Estimates the importance of a vertex for information flow through the graph. Calculated by accumulating commodity from direct and indirect neighbors proportional to their distance.	[Kat53]
HITS	Extension of PageRank that assumes two influential kinds of nodes: Hubs (linked to by authoritative nodes) and Authorities (linked to by hubs). Targeted at extracted subgraphs according to, for example, a query. Also, able to assign high rank to lesser-linked nodes, still deemed influential (in contrast to PageRank)	[Kle99]
Betweenness	Nodes deemed more influential the more shortest paths between any two nodes they intercept. Extensions and variants including Load, Stress, Group betweeness, and Edge betweeness	[Bra08]
Accessibility	Two kinds of influential nodes: Nodes that can be quickly accessed from anywhere in the graph (Inward Accessibility) and nodes from which a large part of the graph can be quickly accessed (Outwards Accessibility).	[TdFC08]

Table 2.2: Centrality Metrics: Measure the influence of a node in the (semantic) graph.

Similarity metrics are the second kind of graph algorithms advantageous to ES that

are investigated in this diploma thesis. An overview of metrics is given in Table 2.3 on page 18. Note that the table also contains a similarity metric designed to allow a simple expression in various SPARQL queries. Although a new approach to calculation of general graph algorithms on RDF graphs is suggested in this diploma thesis, the triple store foundation still allows for standard semantic web technologies to be leveraged.

It is easy to imagine possible use cases of similarity metrics in exploratory search. An exploratory searcher is assumed to explore the semantic graph without a clear future path in his or her mind. Similarity metrics can help guide the exploratory searcher from one concept to the next, thereby offering a path through the expert knowledge thicket.

In particular similarity metrics can potentially address at least the following characteristics of ES systems as suggested in Figure 2.3:

An evolving search process. By repeatedly suggesting related entities in the EKG or depicting similarity between entities, the exploratory searcher can build a similarity map in his or her mind. The search process can be informed by that information, thereby constantly evolving in sophistication.

Multiple possible answers. Due to the system knowing about related entities (and potentially also the kind of relatedness as defined by the metrics), similar entities that could be part of a more involved answer or provide an alternative answer can be suggested.

A serendipitous attitude. By suggesting unrelated items (e.g., the inverse relatedness), surprising new entities can be suggested leading to serendipitous discoveries.

Metric	Description	Reference
Distance	Assumed that the closer two nodes are the more similar they should be. Estimated by the length of the shortest path between them. Extensions for groups of nodes: Conjunctive (Normalized sum of individual distances), Disjunctive (Minimum of individual distances). In original formulation only is-a relations are considered since others may have arbitrary positive or negative influences.	[RMBB89]
SimRank	Estimates similarity by how many steps two random walkers starting at two nodes need (on average) until they meet.	[JW02]
SSDM	Based on SimRank but follows only equal properties. Directed edges can be traversed in both directions (from source to target or vice-versa), but both walkers must take the same direction.	[OPSPL11]

LDSD	Linked Data Semantic Distance. Defines different link counts like direct, incoming links or indirect, outgoing links. Similarity is estimated by combining the basic link-counting formulas in various ways	[Pas07]
Information Content	Makes use of information theory for similarity measures. Similarity is estimated by the information content of features two nodes have in common. In the original formulation indicated by most specific concept that subsumes them both. Works with detailed class structure only. Various extensions exist [Lin98, SDRL06, CY11]	[Res95]
Binary Similarity	A general class of similarity metrics amount of features possessed by two nodes are combined in various ways. E.g. $sim_{jacard}(i,j) = \frac{a}{a+b+c}$ where $a = \|P : P(i) \wedge P(j)\|$, $b = \|P : P(i) \wedge \neg P(j)\|$, $c = \|P : \neg P(i) \wedge P(j)$. Survey of 76 binary similarity metrics.	[sChC10]

Table 2.3: Similarity Metrics: Measure the relatedness between two nodes in the (semantic) graph.

2.4 Selected Metrics

Five metrics were selected to represent the two large categories discussed in Section 2.3: Centrality and Similarity. The selected metrics are presented in this chapter on a conceptual level. For the adaption to the GAS model please refer to Section 4.2.

Section 2.4.1, Section 2.4.2 and Section 2.4.3 take a look at the three centrality metrics. The two similarity metrics presented in Section 2.4.4 and Section 2.4.5 fulfill a different purpose. They measure similarity between two arbitrary nodes in the graph. Given two nodes they estimate how strongly related the nodes are.

Again, both of these similarity metrics can support an exploratory searcher during his or her path through the knowledge domain. Given the node the exploratory search is currently investigating, closely related nodes can be calculated and suggested for further exploration.

Another opportunity would be to purposefully suggest nodes that are as unrelated as possible (the inverse relatedness) to encounter surprising, new knowledge. The suggestion of such unrelated (also called peculiar or serendipitous knowledge) was also discussed in previous research and found as a useful addition for an exploratory searcher [NPG+17, BPCF17].

2.4.1 PageRank

Probably one of the most popular centrality metrics is PageRank (PR). Intuitively, the goal is to estimate the probability of a random surfer to arrive at a vertex. The random surfer is assumed to follow the directed links of the graph, hopping from vertex to vertex. Additionally, the surfer may jump to an arbitrary, non-adjacent vertex. Eventually the surfer will stop, staying at some vertex u. The PageRank $PR(u)$ of a vertex u reflects the probability that the surfer is at node u when coming to a halt [PBMW99].

The PageRank $PR(v_i)$ of a vertex v_i in a graph $G = (V, E)$ is mostly determined by the PR of its incoming vertices $PR(v_j)|(v_j, v_i) \in E(G)$ scaled by the number of outgoing edges $out(v_j)$. Additionally, the probability for leaving/arriving at v_i by a "remote jump" to/from a non-adjacent vertex has to be factored in. In the simplest case this is done by means of a damping factor d capturing the probability of a remote jump.

The iterative formula for calculating the PR of vertex v_i adapted from [BP98] is then:

$$PR(v_i) = (1 - d) + d \times \sum \frac{PR(v_j)}{out(v_j)} \quad \forall v_j|(v_i, v_j) \in E \tag{2.1}$$

PR has multiple alternative formulations, the one we concentrate on here is the algorithmic one adapted from [BP98, PBMW99] and shown in Algorithm 2.1. In the beginning PageRanks are set to some initial distribution S. A simple (but effective) example is to set the initial PR to $\frac{1}{|V|}$ where $|V|$ is the number of vertices in the graph.

Afterwards the PR algorithms runs until a convergence criteria ($delta \leq \epsilon$) is met. Alternatively, other stopping criteria could be defined such as a predetermined number of rounds.

The new PageRank (prn) of every vertex is then calculated based on the previous PageRanks (pr) of its neighborhood. Note that the reset probability $r = 1 - d$ (capturing the probability to arrive at the vertex by a remote jump) is scaled by the number of vertices in the graph. Although not part of the original formula stated in [BP98], this is a common normalization factor not affecting the algorithm meaningfully.

2.4.2 Accessibility

ACS is based on a similar notion of random walks as PR. However, the semantics for estimating centrality of a vertex are quite different. When talking about accessibility one has to distinguish between inward accessibility and outward accessibility [TdFC08].

While inward accessibility is mentioned in this section shortly too, the main focus is on outward accessibility since the semantics diverge more from the already discussed PR and similar centrality metrics. However, an algorithm than can calculate outward accessibility is, with slight modifications, also able to calculate inwards accessibility.

From a birds view, outward accessibility quantifies the diversity of the access of the remainder vertices starting from some vertex in the graph in a finite number of steps.

Algorithm 2.1: PageRank adapted from [PBMW99]

Input: A directed graph $G(V, E)$
Output: The PageRank for every vertex

$pr \leftarrow S$
repeat
  foreach $v_i \in V$ **do**
    $prn[v_i] \leftarrow \frac{1-d}{|V|} + d \sum_{v_j \in \text{inV}(v_i)} \frac{pr[v_j]}{|\text{outE}(v_i)|}$
  end
  $delta = ||prn - pr||_1$
  $pr \leftarrow prn$
until $delta \leq \epsilon$

Inwards accessiblity on the other hand quantifies the frequency of access of a vertex starting from the remainder vertices in a finite number of steps [TdFC08].

Although the calculations differ substantially and will likely yield different results on a given graph, on the level of semantics the inwards accessibility is quite similar to PR: A vertex that is frequently accessed should also have a rather high probability for a random walker to end up in.

A vertex with high outward accessibility allows a random walker starting at this vertex to explore a large and diverse part of the graph after a finite period of time [TdFC08]. This is an important property for ES since it allows to suggest vertices that foster a quick and diverse overview of the graph during traversal. Conversely, if a concept should be explored in-depth it may yield vertices that easily distract the searcher.

For calculating accessibility only self-avoiding walks are considered. A self-avoiding walk is a finite sequence of consecutive edges contains no repeated edges or vertices. The transition probability $P_h(i, j)$ is the probability that a random walker stops at vertex i after h steps in a self-avoiding walk starting from vertex j. The diversity entropy signature E_h of a vertex i is defined by [TdFC08]:

$$E_h(\Omega, i) = -\sum_{j=1}^{N} \begin{cases} 0 & \text{if } P_h(j,i) = 0, \\ P_h(j,i) \log(P_h(j,i)) & \text{otherwise} \end{cases} \tag{2.2}$$

$$E_h(i, \Omega) = -\sum_{j=1}^{N} \begin{cases} 0 & \text{if } P_h(i,j) = 0, \\ (\frac{P_h(i,j)}{N-1}) \log(\frac{P_h(i,j)}{N-1}) & \text{otherwise} \end{cases} \tag{2.3}$$

where Ω is the set of vertices except i.

Finally, the outwards accessibility $OA_h(i)$ and inwards accessibility $IA_h(i)$ of a vertex i

are defined as [TdFC08]:

$$OA_h(i) = \frac{\exp(E_h(\Omega, i))}{N-1} \tag{2.4}$$

$$IA_h(i) = \frac{\exp(E_h(i, \Omega))}{N-1} \tag{2.5}$$

Hence, in comparison to other metrics, the calculation of accessibility is more involved. A potential algorithm has to overcome at least the following hurdles for calculating $OA_h(i)$ for a single vertex i: (i) ensuring self-avoiding walks, (ii) determining the transition probability for a multi-hop neighborhood (iii) calculating E_h in a non-local neighborhood with optimization (ignoring nodes that are non-reachable in h hops), or over all vertices without optimization (iv) calculating $OA_h(i)$ presuming knowledge of the number of vertices in the graph

2.4.3 Load

Load is a metric set in the comparatively long heritage of betweenness centrality. Before discussing load in particular, a short discussion of betweenness is provided first.

Interest in measuring the centrality of a vertex in a graph structure stems from a variety of fields. The first explicit mention of betweenness probably originates from the importance of people in a group (a social network, representable as graph) [Bav48]. Betweenness was later picked up and sculpted by [Fre77] to its currently most authoritative definition.

The intuition behind betweenness is rather simple though it may vary depending on the domain. For example in a communication network, a node that relays most communication has more power in the network than a vertex that just sends or receives message [Fre77]. Betweenness additionally assumes that under standard conditions communication always tries to take the shortest, most efficient way.

The importance of a vertex v in a network (its centrality) is estimated by how many shortest (geodesic) paths v intercepts. The betweenness centrality of a given node v in a graph can then be defined as [Fre77, Bra01]:

$$Btw(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.6}$$

where σ_{st} is the number of shortest paths from vertex s to vertex t with $\sigma_{ss} = 1$. $\sigma_{st}(v)$ is then the number of shortest paths from s to t intercepted by some vertex v.

As with other metrics this basic definition of betweenness spawned a wealth of variants (for a selection see [Bra01]). Anecdotally, their semantics and definitions start to become so similar that their authors occasionally mistake them for each other [New06, Bra01].

[New06] suggested an algorithmic approach to a betweenness variant called Load. This is also the representative of betweenness metrics used in this thesis. The basic idea of Load

is to distribute a commodity along the shortest paths from any vertex v to any vertex w. Those nodes that accumlate most commodity have the highest load. An simplified algorithm after [New06] and [Bra08] is shown in Algorithm 2.2.

Algorithm 2.2: Load

Input: A directed graph $G(V, E)$, shortest paths between vertices
Output: Load of all vertices

```
1 for v ∈ V do δ[v] ← 1                    /* load initialized to 1 */
2 for v ∈ V do
3 |   for w ∈ V do
  |   |   /* Pred(v,w) is the set of predecessors of vertex v
  |   |      along all shortest paths from w to v ordered after
  |   |      their ascending distance from v                    */
4 |   |   for p ∈ Pred(v,w) do δ[p] ← δ[p] + δ[v]/|Pred(v,w)|
5 |   end
6 end
```

Although the calculation of Load seems simple, note how a potential algorithm has to efficiently (i) calculate all shortest paths between any combination of vertices v and w in the graph, and (ii) spread the Load equally between all predecessors along any shortest path of every combination of vertices v and w in the graph..

2.4.4 Information Content

One class of metrics for ES is based on information content commonly defined as the negative log-likelihood $-\log(p(c))$ where $p(c)$ is the probability of c. Probably the first attempt to apply information content as a way to estimate semantic similarity can be found in [Res95]. This is also the one used for application to EKGs. Since the original paper has no notion of semantic web technologies, some terms are adapted to the vocabulary used in semantic web based knowledge graphs.

Resnik's reasoning was based on taxonomies. Normally taxonomies exhibit highly sophisticated and deep subsumption structures. Although semantic knowledge graphs in general, and EKGs in particular, can potentially exhibit such structures too, they are not obliged to. In cases where such deeply nested classes are not encoded in the EKG, information content based metrics (as defined below) are not applicable.

The underlying idea behind information content metrics is that two concepts are similar to the extent to which they share information in common. Given two resources the information content of the concepts that subsume them both is used as heuristic for their similarity [Res95]. In case of an expert knowledge graph this means classes (and super-classes) that contain both given resources.

Information content itself is based on the probability of encountering a class, denoted as function $p : \mathcal{C} \rightarrow [0, 1]$ where $\mathcal{C}$ is a class. One possibility to calculate the similarity of two resources r_1, r_2 in the EKG is then to take the maximum information content of all classes that subsume them both, i.e [Res95]:

$$\text{sim}(r_1, r_2) = \max_{c \in S(r_1, r_2)} [-\log(p(c))] \tag{2.7}$$

where $S(c_1, c_2)$ is the set of classes that subsume (or contain) both resources c_1 and c_2.

Another possibility is to take a weighted sum. Potentially, this allows an expert to assign different weight to classes based on external judgment. The formula according to [Res95] is then defined as:

$$\text{sim}(r_1, r_2) = \sum_{c \in S(r_1, r_2)} \alpha(c)[-\log(p(c))] \tag{2.8}$$

where $\sum_{c \in S(r_1, r_2)} \alpha(c) = 1$ and represents the weighting factor. Alternatively, it can just be set to $\alpha(c) = \frac{1}{|S(r_1, r_2)|}$ for equal weighting.

The second formulation can be more suitable for EKGs that have a sophisticated, but non-hierarchical structure (in contrast to taxonomies that typically have a tree-like, strictly hierarchical shape). However, the EKGs still has to fine-grained classes with high discriminatory power. Otherwise, the sets $S(r_1, r_2)$ will be mostly the same (most resources are subsumed by the same, few classes) for any two resources r_1, r_2.

2.4.5 Linked Data Semantic Distance

In contrast to centrality metrics that try to estimate the importance of a vertex in relation to the whole graph, Linked Data Semantic Distance (LDSD) tries to measure the relatedness between two given vertices [Pas07, PD10].

LDSD is not a single, strictly defined measure. Instead, it can be seen as a family of measures that are defined by combining the same basic ingredients in different ways. Another variation from other metrics discussed so far is that LDSD does not have any overarching semantic reasoning. Instead, the goal is to measure relatedness. The underlying idea is, that relatedness correlates with the amount of various kinds of links between them. The effectiveness is tested by empirical observation [Pas07].

Since the various LDSD measures in [Pas07] are different linear combinations of the same ingredients, we will concentrate only on one of them. The LDSD variation (henceforth just called LDSD) discussed here is the one determined most effective on a musical domain [PD10]. This need not hold true for every conceivable EKG. However, the implementation discussed in Chapter 4 can be easily adapted by changing the combination of the basic ingredients.

Two different kinds of basic link counting formulas can be distinguished in Passant's [Pas07] model: direct distance formulas consisting only of 1-hop neighborhood calculations and indirect distance which considers at most two hops.

The following direct distance formulas are defined [Pas07, PD10]:

C_d The number of direct and distinct links between resources.

- $C_d(l_i, r_a, r_b)$ Equals 1 if $< r_a\ l_i\ r_b >$ is a triple in the semantic graph, 0 otherwise.
- $C_d(n, r_a, r_b)$ The total number of links l_i for which $C_d(l_i, r_a, r_b) = 1$
- $C_d(l_i, r_a, n)$ The total number of resources r_b for which $C_d(l_i, r_a, r_b) = 1$

The following indirect distance formulas are defined [Pas07, PD10]:

C_{io} The number of indirect and distinct outgoing links between resources

- $C_{io}(l_i, r_a, r_b)$ Equals 1 if there exists a resource r such that both $< l_i\ r_a\ r >$ and $< l_i\ r_b\ r >$ are triples in the semantic graph.
- $C_{io}(n, r_a, r_b)$ The total number of distinct links l_i for which $C_{io}(l_i, r_a, r_b) = 1$
- $C_{io}(l_i, r_a, n)$ The total number of resources r_b for which $C_{io}(l_i, r_a, r_b) = 1$

C_{ii} The number of indirect and distinct incoming links between resources

- $C_{ii}(l_i, r_a, r_b)$ Equals 1 if there exists a resource r such that both $< l_i\ r\ r_a >$ and $< l_i\ r\ r_b >$ are triples in the semantic graph.
- $C_{ii}(n, r_a, r_b)$ The total number of distinct links l_i for which $C_{ii}(l_i, r_a, r_b) = 1$
- $C_{ii}(l_i, r_a, n)$ The total number of resources r_b for which $C_{ii}(l_i, r_a, r_b) = 1$

With these basic ingredients, the LDSD used in this thesis is then defined as suggested by [PD10]:

$$\mathrm{LDSD}(r_a, r_b) = \frac{1}{1 + \sum_i \frac{C_d(l_i,r_a,r_b)}{1+\log(C_d(l_i,r_a,n))} + \sum_i \frac{C_d(l_i,r_b,r_a)}{1+\log(C_d(l_i,r_b,n))} + \sum_i \frac{C_{ii}(l_i,r_a,r_b)}{1+\log(C_{ii}(l_i,r_a,n))} + \sum_i \frac{C_{io}(l_i,r_a,r_b)}{1+\log(C_{io}(l_i,r_a,n))}} \tag{2.9}$$

2.5 Gather–Apply–Scatter

Gather Apply Scatter (GAS) is one possible computational model for graph processing. It is particularly well suited for large graphs that are distributed among several computing nodes. The mechanisms of this paradigm are presented in this section. Additionally, connections to other models are provided.

GAS is based upon the "think like a vertex" philosophy popularized by Pregel, another graph computation model [MAB+10]. Pregel consists of vertex-local computations that are synchronized and distributed in supersteps. In contrast to GAS basically all work is

done in a node-local `compute` method. However, also Pregel fundamentally executes the GAS steps. GAS just breaks them out and explicitly invokes them in each round. Additionally, the main difference between Pregel and GAS is that vertices in GAS can operate only on the directly connected (adjacent) neighborhood.

A GAS algorithm has three distinct phases constituting a round: Gather, Apply and Scatter. After every phase there is an implicit barrier for synchronization. Other than that, the operations can be processed in parallel. After a GAS round control is transferred to a super-step which is responsible to coordinate and redistribute the local computations. All three phases will be discussed in order during the following.

Gather is probably the most involved and restrictive operation. It extracts and aggregates information from adjacent vertices as well as their respective connecting edges into a generalized sum [GBL+12]:

$$\Sigma \leftarrow \bigoplus_{v \in N(u)} g(D_u, D_{(u,v)}, D_v) \tag{2.10}$$

where $N(u)$ denotes the neighborhood of vertex u, D_u and D_v denotes the vertex state associated with the vertices u and v respectively, and $D_{(u,v)}$ denotes the edge state associated with the edge (u, v). $g(...)$ stands for the Gather function and $\oplus$ for the generalized sum (the aggregation) of the gathered information.

Hence, the Gather phase actually consists of two operations. First, the gathering of data ($g(...)$). Second, the aggregation of the gathered data ($\oplus$). However, if makes sense to view them on a conceptual level as one edge-parallel operation. Subsequently, if we speak about Gather[3] or the Gather phase, both operations (gather and sum) are combined. In case gather or sum is used, it is about the particular operation.

The gathering operates on state associated with vertices and edges (D_u,... where D stands for **D**ata). State can be transformed during subsequent phases and accessed again in the next round. Only the direct neighborhood of a vertex can be potentially accessed during gather. The gather edges are specified in the GAS program and consist either of in-edges, out-edges or both.

The sum operation takes the result of gather and aggregates them to a single result. Since there is no guarantee in which order the gather results are provided, sum has to be associative and commutative [KVH16].

The result Σ of the Gather phase is then available to the Apply phase for transforming the state of the vertex [GBL+12]:

$$D_u^{new} \leftarrow a(D_u, \Sigma) \tag{2.11}$$

[3]Note the capital G

The scatter phase can then update the state of adjacent edges and activate vertices for the next GAS round. Scatter edges are specified in the GAS program and consist either of in-edges, out-edges or both. Again (and in contrast to e.g. Pregel) only the directly connected (adjacent) neighborhood can potentially be accessed and activated.

$$\forall v \in N(u) : (D_{(u,v)} \leftarrow s(D_u^{new}, D_{(u,v,)}, D_v)) \tag{2.12}$$

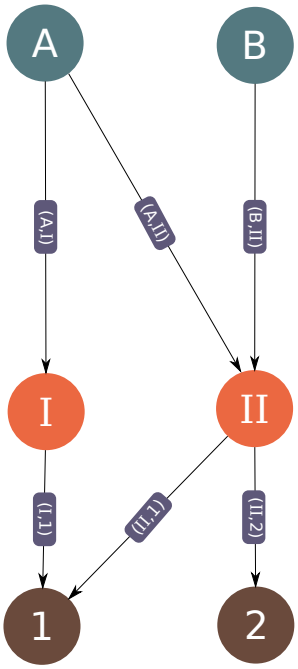
For illustration purposes, a complete GAS round is shown in Figure 2.5. In the figure ○ denotes a vertex state, whereas ▭ denotes an edge state. Colors denote for a particular state at a round r.

Figure 2.5a shows the graph at some round $r = r_0$. Vertices and edges are already in some state before the next round r_1 starts. In case this is the first round, vertex and edge state are initialized to a starting state.

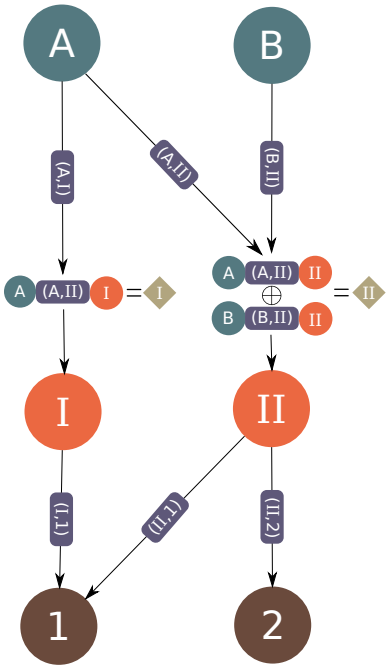
The active vertices in the example round r_1 are I and II. During the Gather phase Figure 2.5b edge and vertex states connected by incoming edges to either I or II are gathered and summed. The result of the sum operation is denoted by ◇. Note how the gather operation and an associative and commutative sum can be parallelized along the incoming edges of a vertex.

In Figure 2.5c the result ◇ of the Gather phase is processed locally for each vertex. This operation also updates the state of the vertex (denoted by a changing color). The updated state is accessible in the next GAS round.

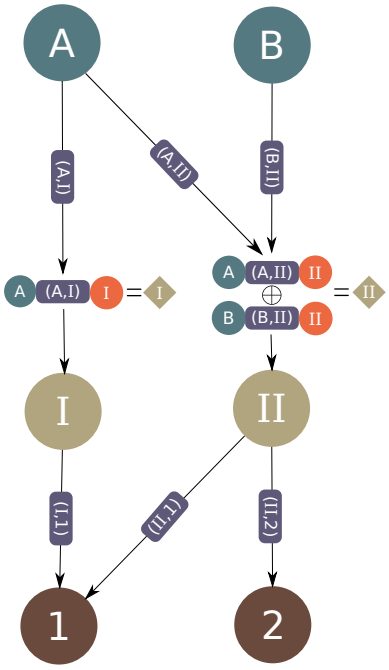
Finally the scatter phase allows to update the edge state of scatter edges. In the example in Figure 2.5d the out edges of the vertices I and II are used for scattering. Additionally, during the scatter phase adjacent vertices can be activated for the next GAS round (e.g. vertices 1 and 2 for currently active vertex II). In case there are no more active vertices scheduled for the next round GAS terminates.



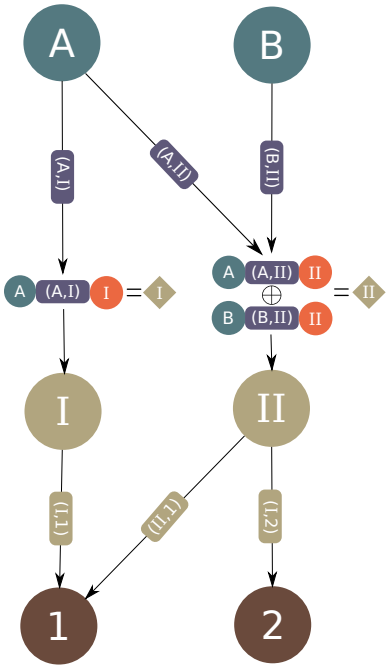
(a) Init. Initial state before the start of a new round or after initialization.



(b) Gather. Merging of vertex and edge states along gathering edges.



(c) Apply. Integrating results from Gather into the vertex states.



(d) Scatter. Integrating information from the new vertex states into the state of scatter edges.

Figure 2.5: Illustration of the GAS phases. Distribution of new information shown by the flow of ocher through the graph.

CHAPTER 3

Methodology

During the last sections all the notions needed for the rest of this diploma theses were established. Besides reviewing current literature for models of Exploratory Search (ES), the term Expert Knowledge Graph (EKG) was synthesized from properties of STAR and notions in the literature too.

Furthermore, several metrics potentially usable for ES where classified. A selection of these metrics for assessing centrality of a node and similarity between nodes was examined in more detail. Finally, the Gather Apply Scatter (GAS) model was introduced as an abstract computational model for graph algorithms.

The next two sections (Chapter 4 and Chapter 5) contain the main part of this diploma thesis. An overview of the methodological approach taken during these chapters is discussed in this section.

The centrality metrics originate from general graph algorithms. In order to implement them on a triple store the GAS computational model was leveraged. Blazegraph, a triple store, supports an Application Programming Interface (API) for this model already. However, in order to use it, the centrality algorithms had to be adapted.

The theory and reasoning behind the algorithms was already discussed in the previous sections. In Chapter 4 the reformulation as GAS steps is shown conceptually. The algorithms were implemented in Java based on the API provided by Blazegraph.

The similarity metrics differ from this approach. Information content based on PageRank (Information Content/PageRank (ICPR)) is a novel suggestion for an information content based algorithm. It leverages the PageRank (PR) implementation and a SPARQL query embedded in a simple controller program. This was again implemented on Blazegraph, this time using the RDF4J API and the SPARQL interface.

The final similarity algorithm, Linked Data Semantic Distance (LDSD), is again of a different nature than the previously described ones. This metric was specifically designed

for ES on triple stores by leveraging their SPARQL capabilities. The respective SPARQL queries were reimplemented from the description of the LDSD formula as given in the previous chapters. Again a controller program was used leveraging the RDF4J API and SPARQL.

The evaluation in Chapter 5 was conducted on two different EKGs: STAR and MusicPinta. On the one hand this ought to showcase the portability of the algorithms to various EKGs. On the other hand the evaluation on two datasets strengthens the arguments by avoiding single observation outliers.

For the evaluation common descriptive statistics are used. Additionally, the correlation between centrality metrics is calculated and discussed. Violin and scatter plots allow for a visual examination of the centrality results. Histograms were used as visualizations for the similarity results. Furthermore, a small subset of nodes together with their computed value is tabulated for each metric. This allows for a simple qualitative analysis.

CHAPTER 4

Implementation

In Chapter 3 a theoretical foundation was built. This chapter describes the realization of a concrete prototype. One of the main goals of the implementations was to run in an environment that could be found in a real world setting too. Furthermore, the used data was not purpose-built for the implementation. Instead, independently created knowledge graphs were used. Conversely, the prototype was not special-fitted to the data either.

Section 4.1 contains some more specific information about the technical environment the algorithms were implemented in. Section 4.2 contains pseudo-code and figurative illustrations for the formulations of the algorithms discussed in Section 2.4.

4.1 Environment

Expert Knowledge Graphs (EKGs) are a special subset of knowledge graphs captured via semantic web technologies as defined in Section 2.1. EKGs — especially if originating from a business setting like the STAR KB — are oftentimes deployed on off-the-shelf components. A tailor-made environment to support the suggested prototype is therefore precluded. The algorithms are purposefully built on commonly available and widely used components. This includes the underlying triple store as well as various APIs.

While this may seem like common practice, it is comparatively unusual in existing literature. As an example, many metrics are based on the concept of random walkers[1] traversing the graph structure. However, none of the widely available triple stores provides a flexible enough access to the graph structure. Implementing sophisticated random walker algorithms is prohibitively complex.

[1] This should not be confused with algorithms such as `PageRank` which additionally provide a concise mathematical formulation.

Blazegraph [2] was chosen as triple store. Besides commercial grade maturity it provides possibilities to overcome some of the shortcomings when it comes to calculating the suggested metrics by providing a Gather Apply Scatter (GAS) interface. GAS can be used for accessing and processing the graph structure directly (see also Section 2.5). Blazegraph was deployed as an embedded instance running in the same JVM as the prototype itself.

For pragmatic reasons, Java with JDK 1.8 running on an Oracle JVM was chosen as programming environment. Blazegraph itself runs on the JVM and provides the best integration within this environment.

The results in section Chapter 5 were gained from experiments conducted on a 64-bit Intel Core i5-4200U processor with 2.6 GHz, 2 cores and 4 hardware threads via Hyper-Threading. The operating system used was a Fedora Linux 26 with Kernel 4.12.9.

4.2 Implementation

In this section the implementation of three centrality measures and two similarity measures are described. Due to the high computational requirements, the centrality measures were implemented by means of the Gather Apply Scatter (GAS) model. GAS is an abstract computational model for graph processing consisting of the three distinct phases it is name after. Each phase can be specified by the algorithm developer and is executed by, for example, a triple store.

Another, and also the more widespread, means to access a triple store is via its SPARQL interface. SPARQL is a query language specifically designed for the semantic web. The similarity measures could be implemented externally by querying the triple store through SPARQL and processing the resulting triples. Therefore, the SPARQL interface to the triple stores was instead of GAS.

4.2.1 PageRank

PageRank (PR) is probably one of the best researched centrality measures. It possesses a particular elegant definition allowing for simple formulations in manifold models. This is also shown by a review of programming model demonstrations in [KVH16] where PR appears in more than twice as many demonstrations than shortest paths, the second most popular measure.

PR is also one of the prime examples of an algorithms leaning itself well towards a GAS-style formulation. It almost naturally fits into this model and was already presented as an example in the genesis paper of GAS [GBL+12]. GAS is an abstract computational model consisting of the three phases Gather, Apply and Scatter. During each phase computations can be conducted that alter well specified states of vertices or edges.

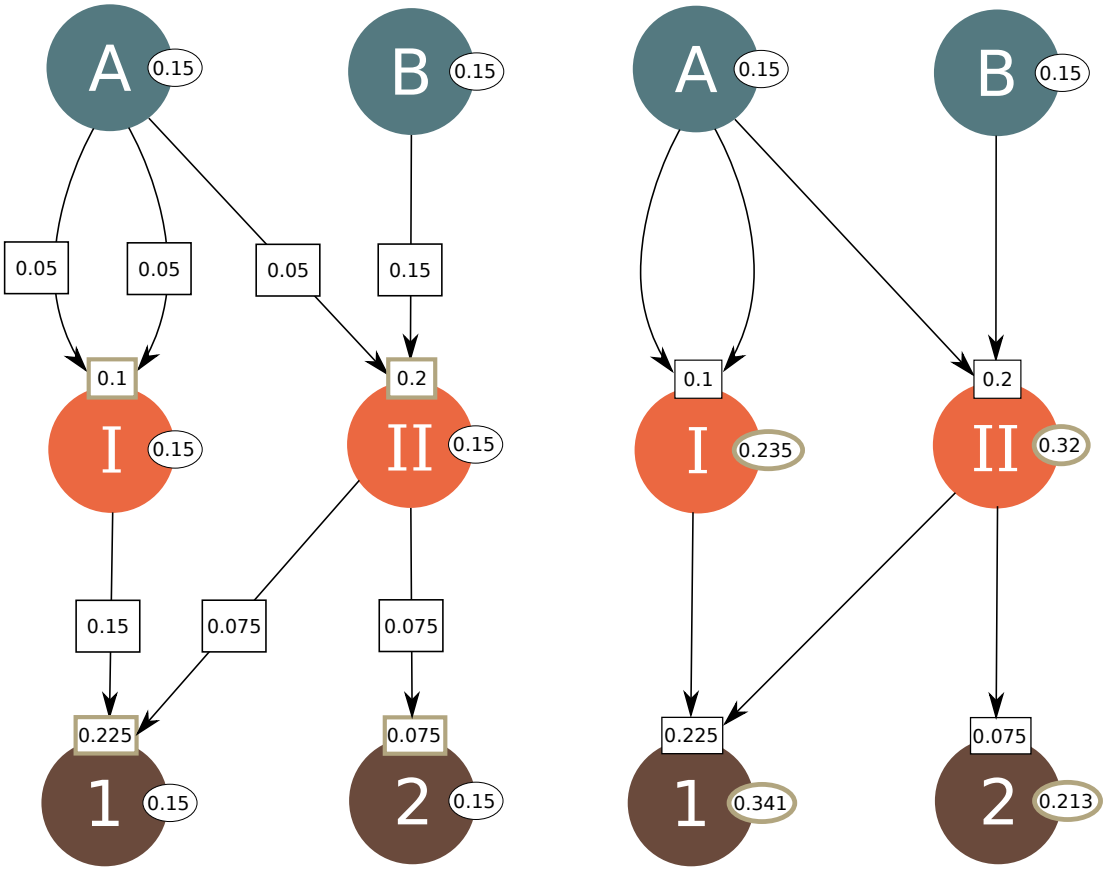
[2] https://www.blazegraph.com/

A conceptual overview of the most important phases is given in Figure 4.1 and Algorithm 4.1. The shown pseudocode is taken (and slightly adapted) from [GBL+12] as there are no major differences to the implemented algorithm.

The gather and sum phase are depicted in Figure 4.1a. Before the first round all nodes are initialized with the reset probability $rp = 1 - dp = 1 - 0.85 = 0.15$, where dp is the damping factor commonly defined as $dp = 0.85$. During the gather phase the current weighted PR (the reset probability in the first round) of incoming adjacent vertices is collected. In the sum phase these the PageRanks of neighboring vertices are added together (see also Line 1 to Line 6 in the pseudocode).

During the scatter phase the new PR of the vertex is calculated from the previously gained sum of (weighted) neighboring PageRanks. As can be seen on Line 8, the calculation is rather simple: $newpagerank = (1 - dp) + dp * sum = 0.15 + 0.85 * sum$ where dp denotes the damping factor commonly set to $dp = 0.85$. Figure 4.1b shows the vertex states after this step. For example the new PR of vertex 1 is calculated by $newpagerank = 0.15 + 0.85 * 0.225 = 0.341$ with the gathered weighted sum of adjacent vertices equaling 0.225.

In order for the algorithm to terminate, a convergence criteria has to be defined. The criteria can be either a particular number of rounds, or a minimum amount of (weighted) change between two calculated PageRanks of a vertex. The criteria shown in the pseudocode is of the second kind. In Line 13 the delta is calculated. During the scatter phase all adjacent vertices along outgoing edges are activated for the next round. That is, all vertices that can potentially receive an updated PR. If the delta is below a defined minimum (ϵ), none of the adjacent vertices is activated for the next round (the amount of change is not worth propagating). Note however that the adjacent vertices can be activated by another vertex with a larger change than ϵ.



(a) Gather-Sum phase calculating the weighted sum of incoming edge importance

(b) Vertex states after the Apply phase containing the new PageRank

Figure 4.1: Illustration of a Gather-Sum and Apply phase for PageRank. Calculations and state changes in each phase denoted by ocher colored border.

Algorithm 4.1: GAS PageRank after [GBL+12]

Input: A directed graph $G(V, E)$
Output: The PageRank for every vertex

```
1  function gather(D_u, D_(u,v), D_v) is
2  |   return D_v.rank/out(v)
3  end
4  function sum(left, right) is
5  |   return a + b
6  end
7  function apply(D_u, sum) is
8  |   newrank = 0.15 + 0.85 * sum
9  |   D_u.delta = (newrank − D_u.rank)/out(u)
10 |   D_u.rank = newrank
11 end
12 function scatter(D_u, D_(u,v), D_v) is
13 |   if |D_u.delta| > ε then activate(v)
14 |   return delta
15 end
```

4.2.2 Accessibility

This chapter presents the first formulation of Accessibility (ACS) in the Gather Apply Scatter (GAS) paradigm. GAS is an abstract computational model consisting of the three distinct phases: Gather, Apply and Scatter. During each phase computations can be conducted that alter well specified states of vertices or edges (see also Section 2.5).

To the best of the authors' knowledge, it is the first GAS-formulation of an algorithm that (i) deploys a non-trivial reduction operation, (ii) operates on a non-local neighborhood, and (iii) encodes an involved, multi-step, non-uniform procedure.

ACS — in contrast to e.g. PR — does not naturally fit into the GAS paradigm. Although the basic calculations are the same as in the original definition, the procedure to get the required ingredients had to be fundamentally redefined.

Although slightly hidden in the conceptual view of GAS, one of the most critical points is the design of the sum function. It has to be associative and commutative [KVH16]. Moreover, it has to be homomorph with sum being defined for the domain as well as the range of the function. In Blazegraph's GAS implementation it even has to be a function $sum : \mathcal{T}^2 \to \mathcal{T}$ with $\mathcal{T}$ being a type. That is, the argument and return types of the sum function are the same.

The obstacle with more involved algorithms like ACS is that not all needed information be reduced to a simple type like double, integer or string. How to merge congregate types in the discussed way, without loosing information necessary, is often not obvious or wanted. The suggested solution is to lift any congregate containing the needed information to a bag container abstraction. Merging two bags fulfills all needed properties. This idea is also used in the GAS implementation of ACS.

In our implementation linked lists are actually underlying the bag container. This provides constant merge-time (just rewiring pointers) and linear time traversal. However, later usages cannot depend on the ordering of elements (although a particular order is given in the linked list). Otherwise, the merge operation would lose commutative-ness.

Although strictly speaking, the gather and sum phase are distinct, they are often considered together. In this vein Figure 4.2 shows both phases together. During gathering, the state of incoming neighbors is wrapped into single element lists. Sum then concatenates these single element lists to a flat list of incoming neighborhood states.

For example in the graph in Figure 4.2 during the gathering phase of vertex II, the states of both adjacent vertices A and B are collected into single-element lists. This wrapping is parallelized along the edges and can be seen as edge information as depicted in the figure. During sum the single element lists `(A . nil)` and `(B . nil)` are merged into `(B . (A . nil))`. Note, that RDF also allows multiple edges between vertices. These edges are not handled differently. For example Vertex I's gather-sum phase results in `(A . (A . nil))`. Multiple edges are also the reason why a bag abstraction instead of a set abstraction has to be used.

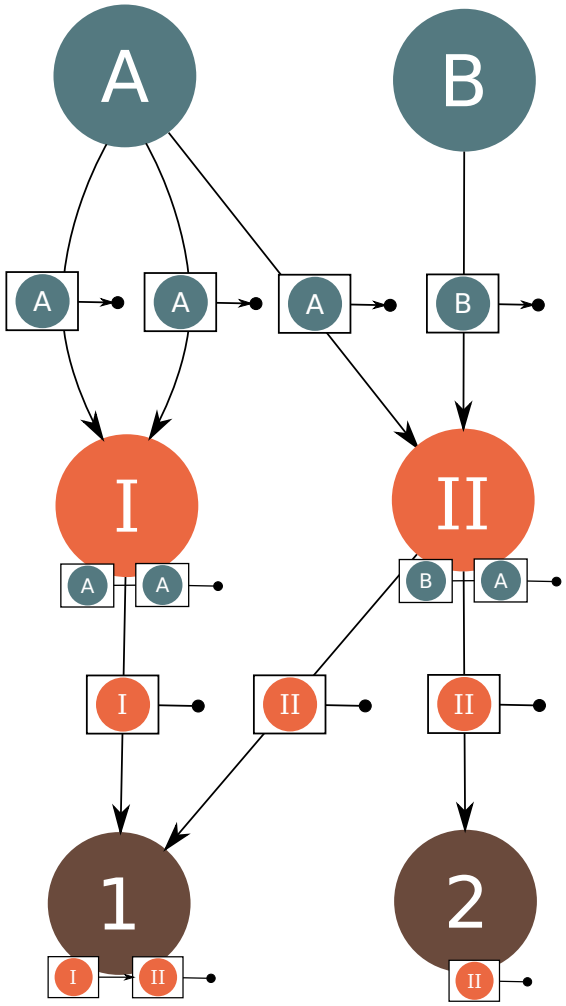


Figure 4.2: Gather-Sum phase creating and concatenating lists of neighborhood states.

Figure 4.3 shows the vertex states after the first (Figure 4.3a) and after the second (Figure 4.3b) round of apply. During apply the gathered information is transformed to a new data structure and stored as vertex state for the next GAS round. The data structure has to hold enough information about the transitions to avoid loops (only self-avoiding walks are allowed) and to calculate the transition probability of paths.

For each possible starting vertex a map consisting of the starting vertex and a pointer to a set of intermediary path vertices is created. Note that, a set is used since vertices are not allowed to be visited more than once (leading to a loop).

For example in the first round, vertex II stores its adjacent nodes A and B as starting vertices and puts itself in the bag of (intermediary) path vertices.Furthermore,e the transition probability is stored with the transition. The transition probability is calculated by $tp(A, II) = inEdges(A, II) * \frac{1}{out(A)} = 1 * \frac{1}{3} = 0.33$. This yields the exact transition

probability compared to an approximated probability as suggested by the original ACS definition (see Section 2.4.2). After the first round the vertex states contain all single-edge traversal paths with their probabilities ending in the vertex where they are stored.

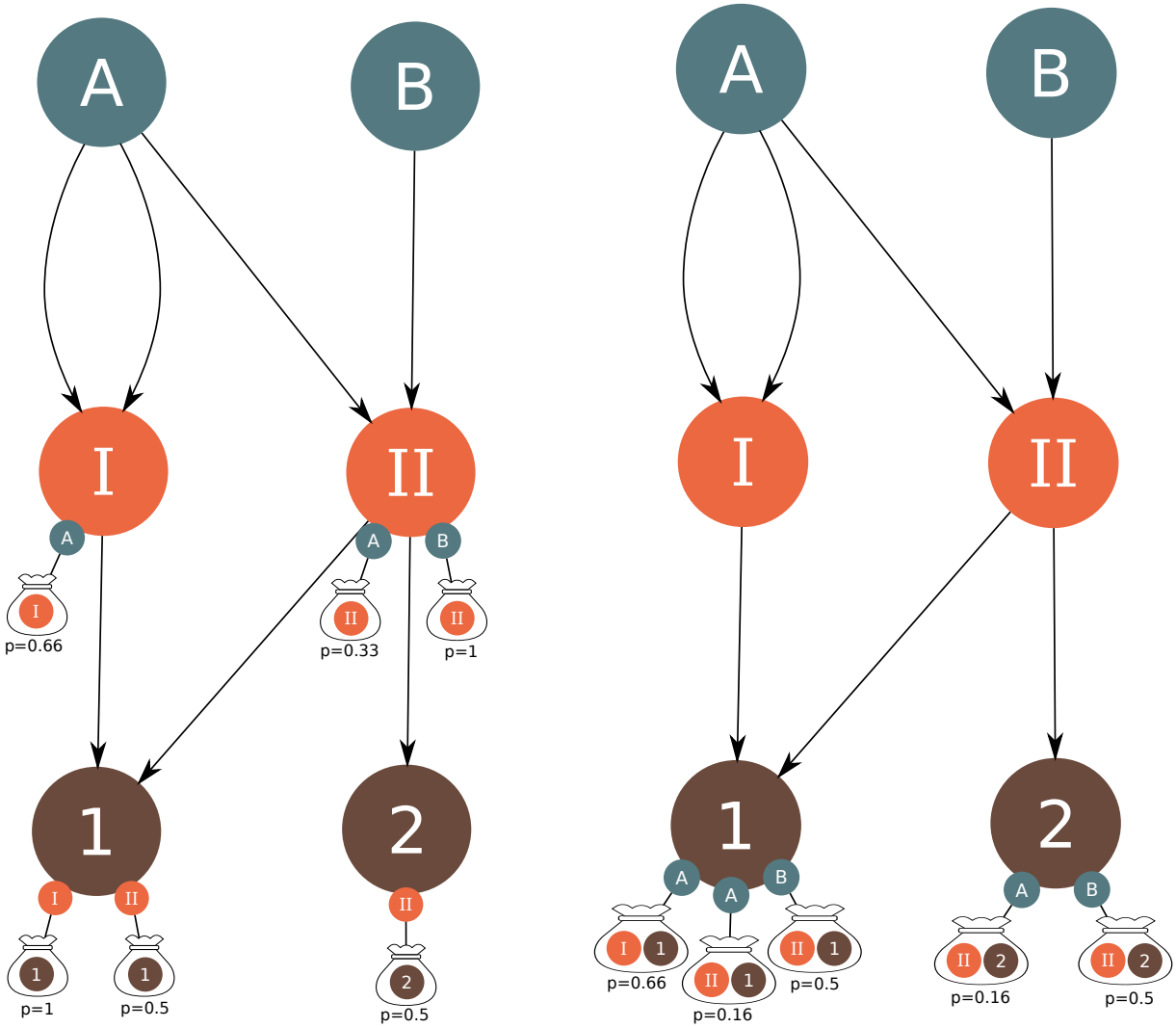
In the second GAS round these single-edge traversal paths are distributed again during the gather phase as sen in Figure 4.2. During the second apply depicted in Figure 4.3b all two-edges traversal paths are calculated.

For example vertex 2 retrieves the single-edge traversal paths from vertex II during the second gather. It then checks if it is not already in the set of path edges for any retrieved traversal. If not, it adds itself creating a new two-edges traversal. Otherwise, the traversal is ignored since only self-avoiding walks are allowed. The probability for the traversal from A to 2 is calculated by $tp(A, 2) = tp(A, II) * \frac{1}{out(II)} = 0.66 * \frac{1}{2} = 0.16$.

Algorithm 4.2 shows the previously described gather, sum, apply phases in pseudo code. The gather and sum phases are only responsible for transferring neighborhood states and combining them in an associative, commutative, homomorphic way by lifting the state to a bag containing the state(s).

The apply phase is more involved. Line 8 associates adjacent vertex states with the number of incoming edges. This is needed for the calculation of transition probabilities (see for example the transitions from node A to I in Figure 4.2 and Figure 4.3). Afterwards the transition probability for every adjacent vertex then calculated. In the first round (Line 11 to Line 13) all possible (single-element) transitions have to be initialized. All other rounds just extend these transitions one edge per round until the maximum number of rounds, carefully avoiding loops.

The scatter phase was not discussed yet. However, it just activates all adjacent vertices on outgoing paths of a vertex (i.e. all vertices that can potentially extend the saved transition) for the next GAS round. Except the last round is reached, then the stored vertex states are reduced.



(a) Vertex states after first apply round

(b) Vertex states after second apply round

Figure 4.3: Apply phases storing start vertices with bags of intermediate path vertices together with the transition probability of the paths

Algorithm 4.2: GAS Accessibility

Input: A directed graph $G(V, E)$, maximum path length *len*
Output: $\forall v_i \in V(G)$: All self-avoiding walks of length *len* ending in v_i

```
1  function gather(D_u, D_(u,v), D_v) is
2  |  return cons(D_v, nil)                          /* single-element list */
3  end
4  function sum(left, right) is
5  |  return append(left, right)
6  end
7  function apply(D_u, sum) is
8  |  gc ← groupCount(sum)                           /* count multi-edges */
9  |  foreach key in gc.keys do
10 |  |  transProb ← gc[key] / outEdges(key)         /* transition probability */
11 |  |  if round ≡ 0 then
12 |  |  |  if key ≡ D_u then continue;              /* no self-loops */
13 |  |  |  else mkTrans(D_u, key, set(D_u), transProb)
14 |  |  else
15 |  |  |  foreach trans in key.transitions do
16 |  |  |  |  if D_u ∈ trans.elems then continue;
17 |  |  |  |  else
18 |  |  |  |  |  fullTransProb ← trans.transProb * transProb
19 |  |  |  |  |  pathBag ← add(trans.elems, D_u)
20 |  |  |  |  |  mkTrans(D_u, trans.key, pathBag, fullTransProb)
21 |  |  |  |  end
22 |  |  |  end
23 |  |  end
24 |  end
25 end
26 function scatter(D_u, D_(u,v), D_v) is
27 |  if round < len then activate(D_v)
28 |  else reduce(D_u)
29 end
```

The reduction phase was not yet discussed. So far the GAS algorithm collected transition paths and their probabilities. In a last step these probabilities have to be reduced to the diversity entropy signature $E_h(\Omega, i)$ and finally to the outwards accessibility $OA_h(i)$ [TdFC08]. The reduction steps are shown in Algorithm 4.3. First, from the transition probabilities $E_h(\Omega, i) = -\sum_{j=1}^{N} tp(j,i) * \log(tp(j,i))$, with i being the current node, is calculated in Line 1 to Line 5. The outwards accessibility is then defined as $OA_h(i) = \frac{\exp E_h(\Omega,i)}{N-1}$ with N being the number of vertices in the graph and i the current node (Line 7).

Algorithm 4.3: Accessibility Reducer

Input: Vertex state data D_u for vertex u
Output: Outward accessibility for vertex u

```
1 forall trans do
2 |   if size(trans.pathBag) < len then continue;
3 |   eh ← −trans.transProb ∗ log2(trans.transProb)
4 |   ehSum ← ehSum + eh
5 end
6 if ehSum ≢ null) then
7 |   D_u.accessibility ← exp(ehSum) / |V|−1        /* G = (V, E) */
8 end
```

4.2.3 Load

The third centrality metric adapted to Expert Knowledge Graphs (EKGs) (and triple stores in general) is Load. Structurally the implemented algorithm follows the formulation in [New01]. However, there is a lot of confusion about the relation between Load, Betweenness and similar metrics [Bra08, New06]. The original formulation by [New01] (although coined as Betweenness) actually encodes Load on undirected graphs [Bra08]. Since expert knowledge graphs are directed, it had to be slightly adapted as suggested in [Bra08].

As for the previous algorithms the Gather Apply Scatter (GAS) computational model was used for the implementation. GAS is an abstract computational model consisting of the three distinct phases Gather, Apply and Scatter. During each phase computations can be conducted that alter well specified states of vertices or edges (see also Section 2.5).

Load (as many other Betweenness measures) is based on shortest paths. The unique challenge for GAS was that it consisted of two consecutive phases:

1. Calculation of all shortest paths from a given node
2. Distribution of a commodity along the shortest paths

Figure 4.4 depicts these two phases for node 1. During the first phase in Figure 4.4a the incoming edges are traversed in a breadth-first manner. The current depth of the breadth search traversal is stored in the vertex states for bookkeeping means.

The second phase (Figure 4.4b) then starts at the nodes farthest away from the starting node and traverses the shortest paths back to the origin. During this traversal the Load is distributed to the encountered vertices.

At the start of the second phase all vertices are initialized with a Load of 1. Starting from the vertices farthest away the Load of a vertex distributed to its predecessors along the shortest paths. In case a vertex has multiple predecessors its load is equally distributed among its predecessors.

Note that only the number of predecessors are counted for the distribution of Loads, not the number of edges. For example node A in Figure 4.4b distributes an equal load of $\frac{load(A)}{|Predecessors(A)|} = \frac{1}{2}$ to its predecessors although there are two edges to node I. Alternatively, A could distribute a load of $\frac{load(A)}{outEdges(A)} * 2$ to node I and $\frac{load(A)}{outEdges(A)}$ to node II. This is mostly a design decision and can be adapted by minor changes to the algorithm.

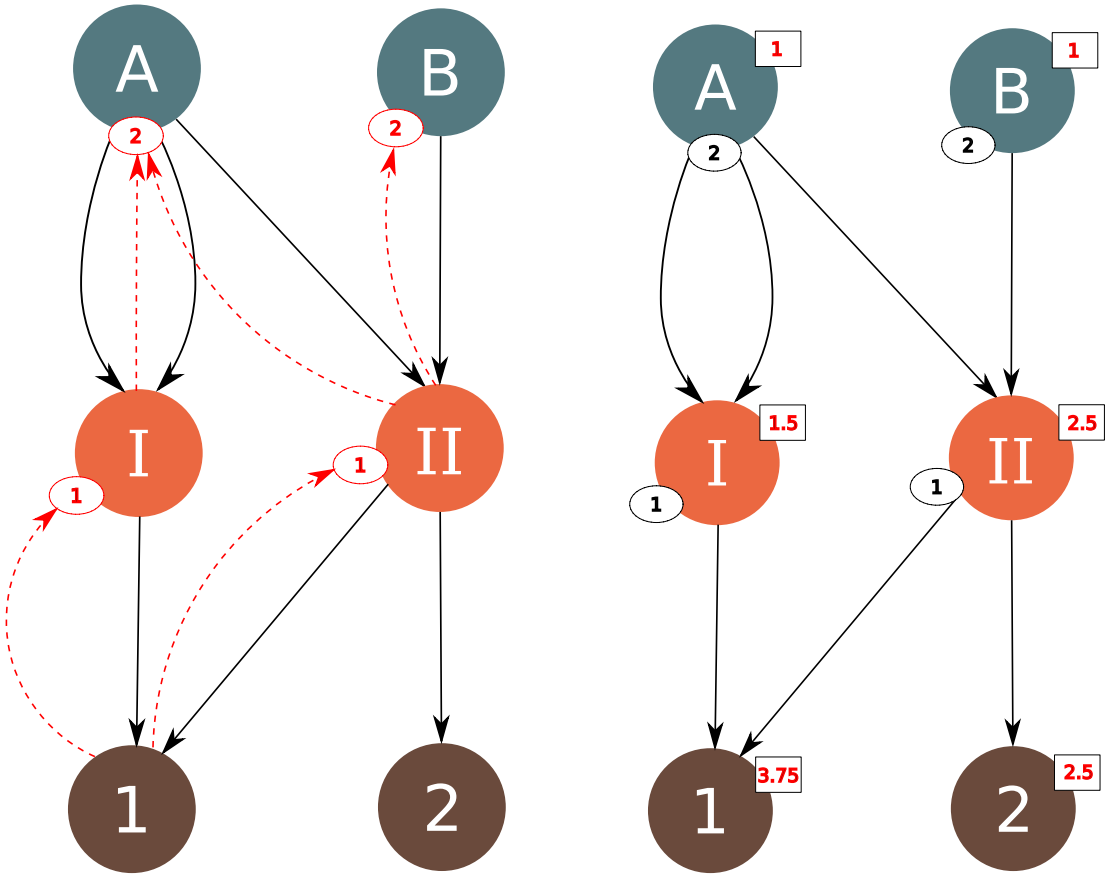
Load has two peculiarities when formulating the algorithm in the GAS paradigm. First, the already discussed two consecutive phases. Second, there is no benefit in using the gather and apply phases except additional complexity. Instead, all needed information can be distributed in what is called a push-style scatter. That is, all information is pushed to the next activated vertex during the scatter phase.

The switch between the two phases (breadth-first shortest path search and Load distribution) is done by intercepting at the barrier after the scatter phase. After every phase in GAS a barrier is introduced to synchronize the distributed computing of the phases [GBL+12]. Blazegraph's GAS implementation provides standard hook at this point to decide if a new round should be started. This hook is used to switch phases if the breadth-first shortest path search is over. That is, no more vertices can be reached.

The corresponding pseudocode can be found in Algorithm 4.4. In contrast to the other presented GAS implementations, the Single Node Load does not need gather and apply phases. Instead, all information is distributed during the push-style scatter.

Line 3 to Line 11 show the scatter during the breadth-first shortest paths search. If the adjacent vertex does not have a distance assigned yet, the distance of the current vertex plus one edge is the shortest possible path length from the origin node (breadth-first). Only the first vertex that encounters a yet unassigned vertex schedules that vertex for the next GAS round. In case another vertex at the same distance encounters is adjacent too, it is only added to the predecessor set. During the load phase (Line 12 to Line 17) the loads are then distributed as already described previously.

The function `nextRound` is called after the scatter phase of a GAS round has finished deciding whether a new round should be started. In this case, it is also responsible



(a) First phase of Load, calculating all shortest paths for node A by traversing the graph in a breadth first manner.

(b) Vertex states after second phase of Load. A commodity is distributed along all shortest paths for any given node.

Figure 4.4: Illustration of the shortest path calculation and commodity distribution phases for LOAD via two subsequent GAS phases.

to switch between the breadth-first and load phases. In case GAS is currently in the breadth-first state and there are no more vertices left (breadth-first has traversed all reachable nodes), the phase is set to load and the scatter edges are reversed: As can also be seen in Figure 4.4 during the breadth-first phase scattering happens along the incoming edges starting from the origin node whereas during the load is distributed along the outgoing edges towards the origin node.

Algorithm 4.4: Single Node Load

Input: A starting node
Output: Load for reachable nodes

```
1  function scatter(D_u, D_(u,v), D_v) is
2      switch phase do
3          case BFS do
4              if D_v.distance ≡ −1 then
5                  D_v.distance = D_u.distance + 1
6                  D_v.predecessors.add(u)
7                  schedule(v)
8              else if D_v.distance ≡ D_u.distance + 1 then
9                  D_v.predecessors.add(u)
10             end
11         end
12         case LOAD do
13             if v ∈ D_u.predecessors then
14                 D_v.load = D_u.load / |D_u.predecessors|
15                 schedule(v)
16             end
17         end
18     end
19 end
20 function nextRound (gas) is   /* Called after scatter finished */
21     if phase ≡ BFS ∧ gas.schedule ≡ null then
22         phase = LOAD
           /* The currently activated vertices (frontier) are the
              ones farthest away from the origin                  */
23         gas.schedule = gas.frontier
24         scatterEdges = OutEdges      // Swap scatter orientation
25         return true                  // Continue GAS
26     else if phase ≡ LOAD ∧ gas.schedule ≡ null then
27         return false                 // Stop GAS
28     end
29 end
```

4.2.4 Information Content

The implementation of information content as defined by [Res95] measures the similarity between two nodes. The major drawback of this method is that it presumes a deep class hierarchy. This is the case in taxonomies and can be the case in semantic graphs. However, depending on the domain this may not be how an Expert Knowledge Graph

(EKG) is constructed. In such knowledge bases, information content based measures are not applicable since they are not able discriminate enough between the similarity of nodes.

However, information content based similarity measures are also comparatively efficient and simpler to implement. A formulation in SPARQL is possible [3] making it available to triple stores that do not support more sophisticated triple processing capabilities like GAS.

Although there are many alternative formulation of information content based measures, they are generally based on two key ingredients:

1. The capability to extract common super classes of the given instances, and
2. The calculation of a probability for a given class

.

The implementation described here did not use a pure SPARQL formulation but a hybrid implementation with SPARQL queries for retrieving relevant data from the triple store and calculations done in Java. This allows for a very concise formulation of this metric. Furthermore, a novel suggestion for the calculation of class probabilities is provided.

The SPARQL query for retrieving common super-classes of two nodes is given in Algorithm 4.5. It is relatively simple due to the possibility of using property paths in SPARQL 1.1 [HS13]. Property paths provide a kind of restricted traversal with statically unknown bound. Note however that this is different from full recursive capabilities or a loop construct, which are not available in SPARQL. Also, property paths are not available in previous SPARQL versions (see [PS08]), yielding common super-class calculations via a SPARQL-only approach impossible for triple stores not supporting SPARQL 1.1.

The algorithm retrieves all common super-classes of two given nodes designated as <a> and <b>. The union is necessary to also allow for computing the similarity of two class-nodes. Furthermore, standard classes defined by the w3 consortium [4] are excluded as they are not defined by the EKG and bear no relevant domain information (but could potentially spoil the similarity calculations).

The second ingredient for information content based measures is a function $p : \mathrm{C} \to [0, 1]$ associating probabilities to classes. The function should reflect "the probability of encountering an instance of concept c" [Res95]. However, the probability calculation in [Res95] is not suitable for EKGs as it presumes a taxonomy and effectively a tree structure.

There are many possibilities to defining the probability function. One simple way would be to assign for each class the number of nodes belonging to it (or a subclass) divided

[3] No inherent recursive computations with statically unknown bounds

[4] E.g. rdfs:Class which is a rdf:type of all RDF classes by definition [BG14]

Algorithm 4.5: SPARQL Super-Classes

Input: Nodes <a> and <b> for which common super-classes should be found
Output: The common super-classes of <a> and <b>

```
1  SELECT ?sub WHERE
2  {
3      {
4        <a> rdf:type/rdfs:subClassOf* ?sup .
5        <b> rdf:type/rdfs:subClassOf* ?sup .
6      }
7      UNION
8      {
9        <a> rdfs:subClassOf/rdfs:subClassOf* ?sup .
10       <b> rdfs:subClassOf/rdfs:subClassOf* ?sup .
11     }
12     FILTER (!STRSTARTS (str (?sup), "http://w3.org"))
13 }
```

by the number of total nodes, i.e. $\frac{nodes((C))}{N}$ with N the total number of nodes. A more sophisticated choice would be the probability of a surfer to arrive at a class by randomly following edges. PR calculates this probability, but other centrality measures could potentially be used (with different semantics) too.

Although PR is comparatively efficient it is still not suitable for real-time calculations. Therefore, in the current implementation it is computed beforehand for every node. The probability function is then a simple lookup of the precomputed PR for a given class.

The final information content based algorithm with PR is given in Algorithm 4.6. It retrieves both discussed ingredients in computes the final similarity between two nodes with the given common super-classes. Note that the similarity is defined like in [Res95] as the maximum information content. However, different definitions can be used with minor adaptions too (e.g. the average).

Algorithm 4.6: ICPR

Input: A list of classes (sups) and a lookup table of PageRanks for classes
Output: The information content based similarity between nodes with common super-classes (sups)

```
1 sim = 0.0
2 for sup in sups do
3 |   tmpsim = − log2 (pageRank.get(sup))
4 |   if tmpsim > sim then sim = tmpsim                    /* maximum */
5 end
6 return sim
```

4.2.5 LDSD

Linked Data Semantic Distance (LDSD) is another measure estimating similarity between two nodes. Again, similar to the information content measure in Section 4.2.4, the definition provides an example of an algorithm that can be computed efficiently by querying the triple store through SPARQL and processing the results in Java. A GAS style formulation is therefore not necessary.

The reason for this is that only a node its neighborhood with at most one indirection is considered for the computations. That is, similarity can be calculated only between a node <a> and all nodes <b> connected to <a> by at most two edges. Given a node <a>, the algorithm always computes the similarity of all possible <b> nodes at once. If needed the similarity between <a> and a particular <b> can be queried afterwards.

The algorithm consists of several SPARQL queries that are then combined to form the final formula for LDSD (see also Section 2.4.5). The first query in Algorithm 4.7 retrieves all relevant nodes for a given node. That is, all connected nodes that are at most two edges away and are relevant for future calculations.

The following algorithms are SPARQL queries calculating values for the functions as defined in [PD10]:

- The direct and distinct links between resources r_a and r_b: $C_d(l_i, r_a, r_b)$ in Algorithm 4.8
- The indirect and distinct incoming links between resources r_a and r_b: $C_{ii}(l_i, r_a, r_b)$ in Algorithm 4.9
- The indirect and distinct outgoing links between resources r_a and r_b: $C_{io}(l_i, r_a, r_b)$ in Algorithm 4.10

Algorithm 4.7: SPARQL LDSD Neighborhood (ldsd-neighborhood)

Input: Nodes <a> and <b> for which common super-classes should be found
Output: The common super-classes of <a> and <b>

```
1  SELECT DISTINCT ?b WHERE
2  {
3    {
4      { <a> ?l1 ?b .}
5      UNION
6      { ?b ?l2 <a> .}
7    }
8    UNION
9    {
10     {
11       ?nii ?lii <a> .
12       ?nii ?lii ?b .
13     }
14     UNION
15     {
16       <a> ?lio ?nio .
17       ?b ?lio ?nio .
18     }
19   }
20 }
```

Additionally, the respective extensions to calculate total number of resources for functions as defined in [PD10]:

- The total number of resources n linked directly to r_a via l_i: $C_d(l_i, r_a, n)$ in Algorithm 4.11
- The total number resources n linked indirectly to r_a via the incoming edge l_i: $C_{ii}(l_i, r_a, n)$ in Algorithm 4.12
- The total number of resources n linked indirectly to r_a via the outgoing edge l_i: $C_{io}(l_i, r_a, n)$ in Algorithm 4.13

Algorithm 4.8: SPARQL C_d links (cdlab)

Input: Nodes <a> and <b>
Output: A list of direct and distinct links between <a> and <b>

```
1  SELECT ?l WHERE { <a> ?l <b> . }
```

Algorithm 4.9: SPARQL C_{ii} links (ciilab)

Input: Nodes <a> and <b>
Output: A list of indirect and distinct incoming links shared between <a> and <b>

```
1 SELECT ?l WHERE
2 {
3     _:x ?l <a> .
4     _:x ?l <b> .
5 }
```

Algorithm 4.10: SPARQL C_{io} links (ciolab)

Input: Nodes <a> and <b>
Output: A list of indirect and distinct outoing links shared between <a> and <b>

```
1 SELECT ?l WHERE
2 {
3     <a> ?l _:x .
4     <b> ?l _:x .
5 }
```

Algorithm 4.11: SPARQL C_d resource count (cdlan)

Input: Node <a>
Output: The number of resources b linked directly to <a> per link type l

```
1 SELECT ?l (COUNT (DISTINCT ?b) AS ?n) WHERE
2 { <a> ?l ?b . }
3 GROUP BY ?l
```

Algorithm 4.12: SPARQL C_{ii} resource count (ciilan)

Input: Node <a>
Output: The number of resources b linked indirectly to <a> per incoming link type l

```
1 SELECT ?l (COUNT (DISTINCT ?b) AS ?n) WHERE
2 {
3     _:x ?l <a> .
4     _:x ?l ?b .
5 }
6 GROUP BY ?l
```

Algorithm 4.13: SPARQL C_{io} resource count (ciolan)

Input: Node <a>

Output: The number of resources b linked indirectly to <a> per outgoing link type l

```
1 SELECT ?l (COUNT (DISTINCT ?b) AS ?n) WHERE
2 {
3       <a> ?l _:x .
4       ?b ?l _:x .
5 }
6 GROUP BY ?l
```

Finally, with these SPARQL query the LSDS measure can be calculated for a given node <a> and all relevant neighborhood nodes <b> by Algorithm 4.14.

Algorithm 4.14: LDSD

Input: Node a
Output: A map containing the LDSD for all relevant neighborhood nodes

```
1  neighborLSDS = map()      /* neighborLSDS is an empty map */
2  neighbors = ldsd-neighborhood(a)
3  linkCdlan = cdlan(a)
4  linkCiilan = ciilan(a)
5  linkCiolan = ciolan(b)
6  for b in neighbors do
7  |  sum_cd = sum(cdlab(a,b), linkCdlan)
8  |  sum_cii = sum(ciilab(a,b), linkCiilan)
9  |  sum_cio = sum(ciolab(a,b), linkCiolan)
   |  /* Reusing same methods with a and b switched          */
10 |  sum_cd_b = sum(cdlab(b,a), cdlan(b));
11 |  lsds = 1/(1+sum_cd+sum_cd_b+sum_cii+sum_cio)
12 |  neighborLSDS.put(b, lsds)
13 end
14 return (neighborLSDS)
15 function sum (links, linkCxlan) is
16 |  sum = 0
17 |  for l in links do
18 |  |  sum += 1/(1+log(linkCxlan.get(l)))
19 |  end
20 |  return sum
21 end
```

CHAPTER 5

Results

During this chapter the metrics are evaluated in order to answer the research questions stated in the beginning. All of the evaluations were carried out on two different graphs: STAR and MusicPinta. This ought to reduce the chance that the conclusions are prone to a coincidental outlier result. Additionally, for each metric a qualitative discussion of the suggested nodes and a statistical-quantitative investigation was performed.

The evaluation is split according to the classification established in Section 2.3. That is, Section 5.1 will concentrate on the centrality metrics PageRank (PR), Accessibility (ACS) and Load. Section 5.2 on the other hand evaluates the similarity metrics Linked Data Semantic Distance (LDSD) and Information Content/PageRank (ICPR).

5.1 Centrality Metrics

In this section results from the centrality metrics as discussed in Section 4.2 are shown. Namely, results for Accessibility (ACS), Load (LOAD) and PageRank (PR) are examined. Since ACS is parameterized by the path length, results for (i) a path length of exactly four, denoted as ACS(4), as well as (ii) the averaged ACS for path lengths between one and four, denoted as ACS(1-4) are used. Where applicable results are normalized to a range between $[0, 1]$ to allow for better comparability.

The evaluation was carried out with Expert Knowledge Graphs (EKGs) and Exploratory Search (ES) in mind. EKGs are a special kind of knowledge graph leveraging in semantic web technology. Furthermore, they consist of specialized, domain-constrained knowledge, and sophisticated knowledge as an expert would have in his or her domain of expertise (see Section 2.1). ES is a broader view on search than previous search paradigms provides. In addition to lookup it takes into account activities like learning and investigation too (see Section 2.2). Exploratory Search Systems (ESSs) are software systems built for assisting an exploratory searcher.

STAR

In the first part of this section centrality metrics calculated from the STAR EKG are examined. Table 5.1 on page 54 shows the five most influential nodes for each of the metrics with respect to the whole STAR graph. All algorithms work very well for filtering out less informative nodes that are contained in the graph. All the concepts in Table 5.1 on page 54 may also be assessed by a human evaluator as good starting points for an ES.

Metric	Node	Value
ACS (4)	context#AuthorTag	1
	specific#Domain	0.84776
	context#ContextInformationElement	0.81734
	context#Role	0.76283
	instance#DesignPattern_modal_window	0.49041
ACS (1-4)	specific#UseCase	1
	generic#DesignTactic	0.02967
	context#Role	0.02393
	context#ProjectPhase	0.02393
	context#AuthorTag	0.01835
LOAD	instance#ArchitecturalQuality_scalability	1
	instance#ArchitecturalQuality_security	0.99651
	instance#ArchitecturalQuality_safety	0.86639
	instance#ArchitecturalQuality_reliability	0.81948
	instance#ArchitecturalQuality_usability	0.76662
PR	context#ContextInformationElement	1
	generic#ArchitecturalKnowledgeElement	0.85672
	specific#Domain	0.42875
	context#Project	0.28875
	context#Author	0.21385

Table 5.1: The five most influential nodes selected by centrality metrics with respect to the whole STAR EKG.

Indeed, considering a faceted search interface, the user could reasonably select any of the suggested nodes for further exploration of the domain. Nodes in the graph that are less informative or represent dead ends are successfully filtered out. Note also, that a ESS could support an exploratory search by iteratively and semantically refining facets. If the user selects `context#AuthorTag` the most central nodes contained in this class (that are `rdf:type context#AuthorTag`) could be further explored. However, given that this diploma thesis investigates core algorithms only, the concrete usage is at the discretion of the ESS.

Whereas most metrics exhibit a decent decline among the highest ranking nodes, the

averaged ACS(1-4) puts a disproportional emphasis on the dominant node with a stark decrease in value of the second-to-highest ranking node.

But the summary statistics in Table 5.2 on page 55 hint that all the metrics are skewed. Already at the third quartile the numerical values are roughly 1/15 for LOAD down to $1/10^3$ for ACS(1-4). At some point after the first few highest ranking nodes all of them seem to exhibit a steep decrease with most of the values clustering at comparatively low levels.

	ACS (4)	ACS (1-4)	LOAD	PR
mean	0.01539	0.00182	0.06154	0.01260
std	0.06503	0.03179	0.09384	0.04502
min	0	0	0	0
25%	7.1e-06	0.00065	0.01747	0.00051
50%	1.0e-05	0.00069	0.04491	0.00222
75%	0.00670	0.00082	0.06676	0.02379
max	1	1	1	1

Table 5.2: Descriptive summary statistics of the centrality metrics as computed on the STAR EKG.

Figure 5.1 shows the distribution of values for every centrality metric. Although all metrics, with exception of ACS(1-4), distinguish well between the higher ranking metrics, most results cluster around very low numerical values. LOAD exhibits the most favorable behavior in that regard, distributing more nodes evenly along the possible value-range than any other metric.

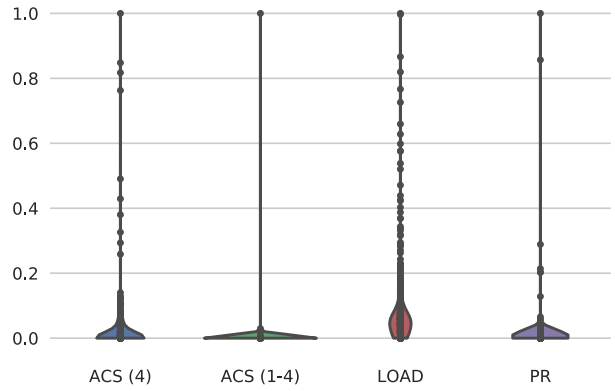


Figure 5.1: Violin plots of the centrality metrics as computed on the STAR EKG showing the data points as well as the kernel density estimation.

Judging from Table 5.1 on page 54 the metrics also do not seem to exhibit homogeneous results. Indeed, the sole noticeable Pearson correlation is between ACS (path length = 4) and PR with a moderate correlation of $r = 0.44$. All other correlations between any pair of metrics are in-betwen the $[-0.01, 0.01]$ range, hence not noteworthy. To the contrary, the results rather hint at independence. Scatter plots for the metrics are shown in Figure 5.2. They confirm the low correlation and skewed distribution of the metrics too.

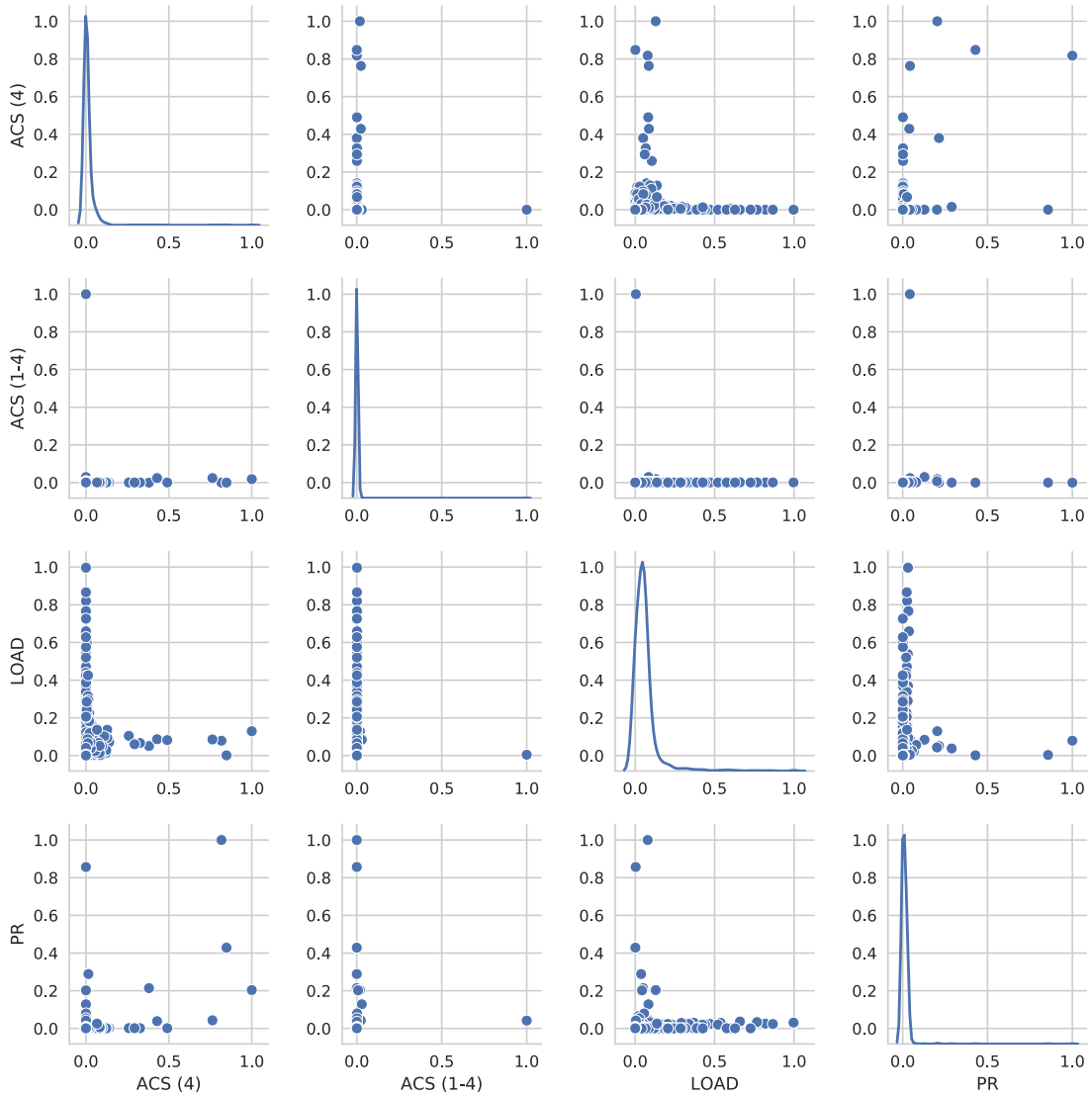


Figure 5.2: Scatter plots of the centrality metrics, as computed on the STAR EKG, showing the data points of the centrality metrics plotted against each other as well as the kernel density estimation for each metric on the diagonal axis.

However, especially from the point of view of a human exploratory searcher, it is not necessarily reasonable to interpret the degree of difference between values as suggested by the results. That is, an interval scale may not be appropriate. Indeed, presuming only an ordinal scale the Spearman correlation between any pairs of metrics suggests a tighter coupling. All correlations, except between ACS(4) and PR, are at least weak up to strong according to common interpretation (see Table 5.3 on page 57).

	ACS (4)	ACS (1-4)	LOAD	PR
ACS (4)	1	0.252593	0.394421	-0.092467
ACS (1-4)	0.252593	1	0.468736	0.775616
LOAD	0.394421	0.468736	1	0.340784
PR	-0.092467	0.775616	0.340784	1

Table 5.3: Spearman correlations between any pair of centrality metrics as computed on the STAR EKG.

MusicPinta

In order to judge the previous results from more than a single observation, the algorithms were conducted on a second dataset too. MusicPinta, which is a mesh from many different datasets about music, also contains many scrub nodes that cannot be considered expert knowledge such as hash values used for fingerprinting songs or invalid download links. The MusicPinta dataset was hence pruned to the EKG about instruments for the investigation.

Due to peculiarities of the selection chosen from the MusicPinta dataset the metrics revealed a problem already seen at the averaged ACS(1-4) metric on the STAR EKG: An overemphasis of a dominant concept to the disadvantage of all others. Concretely, due to the selection of the MusicPinta subgraph, the abstract concept `Instrument` (represented by the node `musicbrainz:instrument/14`) is arguably the most important node in the graph and also designated as such by various metrics.

However, due to the stark decrease in value of the second-to-highest ranking node, this node also represents an extreme outlier leading to a dominating node in the following examination which would hide other effects. For cosmetic reasons, `Instrument` was therefore left out in the following examination [1]. However, this bears no consequences for the stated arguments.

Table 5.4 on page 59 shows the five most influential nodes and their respective values as calculated by the centrality metrics. Since the IRIs in the MusicPinta dataset bear little information for a human examiner, the respective labels are given too (if available). Note that all metrics are successful in providing a variety of fruitful suggestions for further exploration. They do not only filter out dead ends but also nodes that a human assessor would probably deem less important when examining the MusicPinta EKG.

The summary statistics in Table 5.6 on page 60 exhibits a similar pattern to the one already discovered on the STAR EKG. This suggests that the results are largely independent from any particular dataset but rather tied to the algorithms themselves. Again a stark decrease in values at or before the third quartile can be observed. An ESS incorporating these algorithms may take this into account, using only the suggested high-profile nodes.

[1] Note that, especially if an ordinal scale is presumed, a potential ESS is not affected by the dominating node.

Metric	Node	Label	Value
ACS (4)	musicbrainz:instrument/69	String instruments	1
	musicbrainz:instrument/322	Struck string instruments	0.87688
	musicbrainz:instrument/15	Wind instruments	0.75184
	musicbrainz:instrument/180		0.43062
	musicbrainz:instrument/329	Electric Piano	0.34246
ACS (1-4)	musicbrainz:instrument/75	Guitars	1
	musicbrainz:instrument/322	Struck string instruments	0.26965
	musicbrainz:instrument/159	Electronic instruments	0.02216
	musicbrainz:instrument/123		0.00506
	dbpedia:Huqin	Huqin	0.00461
LOAD	musicbrainz:instrument/124	Percussion instruments	1
	musicbrainz:instrument/229	Guitar	0.82813
	musicbrainz:instrument/302	Plucked string instruments	0.76563
	musicbrainz:instrument/75	Guitars	0.70313
	musicbrainz:instrument/233	Reeds	0.69531
PR	musicbrainz:instrument/69	String instruments	1
	musicbrainz:instrument/15	Wind instruments	0.67156
	musicbrainz:instrument/229	Guitar	0.58000
	dbpedia:Guitar	Guitar	0.57975
	musicbrainz:instrument/124	Percussion instruments	0.51585

Table 5.4: The five most influential nodes selected by centrality metrics with respect to the whole MusicPinta EKG.

	ACS (4)	ACS (1-4)	LOAD	PR
mean	0.01141	0.00142	0.04829	0.01236
std	0.09260	0.03272	0.08082	0.04574
min	0	0	0	0
25%	3.2e-09	2.7e-08	0	0
50%	3.2e-09	2.7e-08	0.03906	0.00380
75%	3.2e-09	0.00017	0.06250	0.00817
max	1	1	1	1

Table 5.5: Descriptive summary statistics of the centrality metrics as computed on the MusicPinta EKG.

The violin plots in Figure 5.3 confirm the observations made so far. Although, the centrality metrics provide a useful distinction among the highest-ranking nodes most others group around low numerical values. Again, LOAD exhibits the most favorable

behavior in that regard, distributing more nodes evenly along the possible value-range than any other metric.

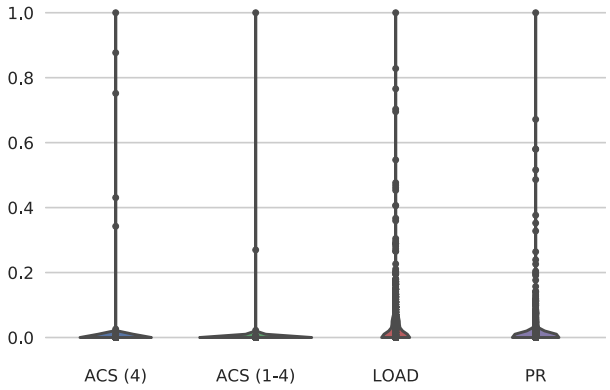


Figure 5.3: Violin plots of the centrality metrics as computed on the MusicPinta EKG showing the data points as well as the kernel density estimation.

Given that a degree of difference is not necessarily interpretable only the Spearman correlation were calculated for comparison. The correlations between the metrics on the MusicPinta EKG are given in Table 5.6 on page 60. However, they are harder to interpret with respect to the previous results on the STAR EKG. Despite being generally higher, some combinations (in particular ACS(4) to PR and ACS(1-4) to PR) show a diverging pattern compared to the previous EKG. The reason for that is unclear and subject to future investigations.

	ACS (4)	ACS (1-4)	LOAD	PR
ACS (4)	1	0.618032	0.682259	0.508396
ACS (1-4)	0.618032	1	0.694103	-0.123142
LOAD	0.682259	0.694103	1	0.125202
PR	0.508396	-0.123142	0.125202	1

Table 5.6: Spearman correlations between any pair of centrality metrics as computed on the MusicPinta EKG.

The scatter plots in Figure 5.4 also bear no surprises compared to the ones from the STAR EKG. Although slightly closer correlated, they show no clear connection between the metrics. The results within each of the metrics are again highly skewed towards low values as can be seen on the diagonal axis in Figure 5.4.

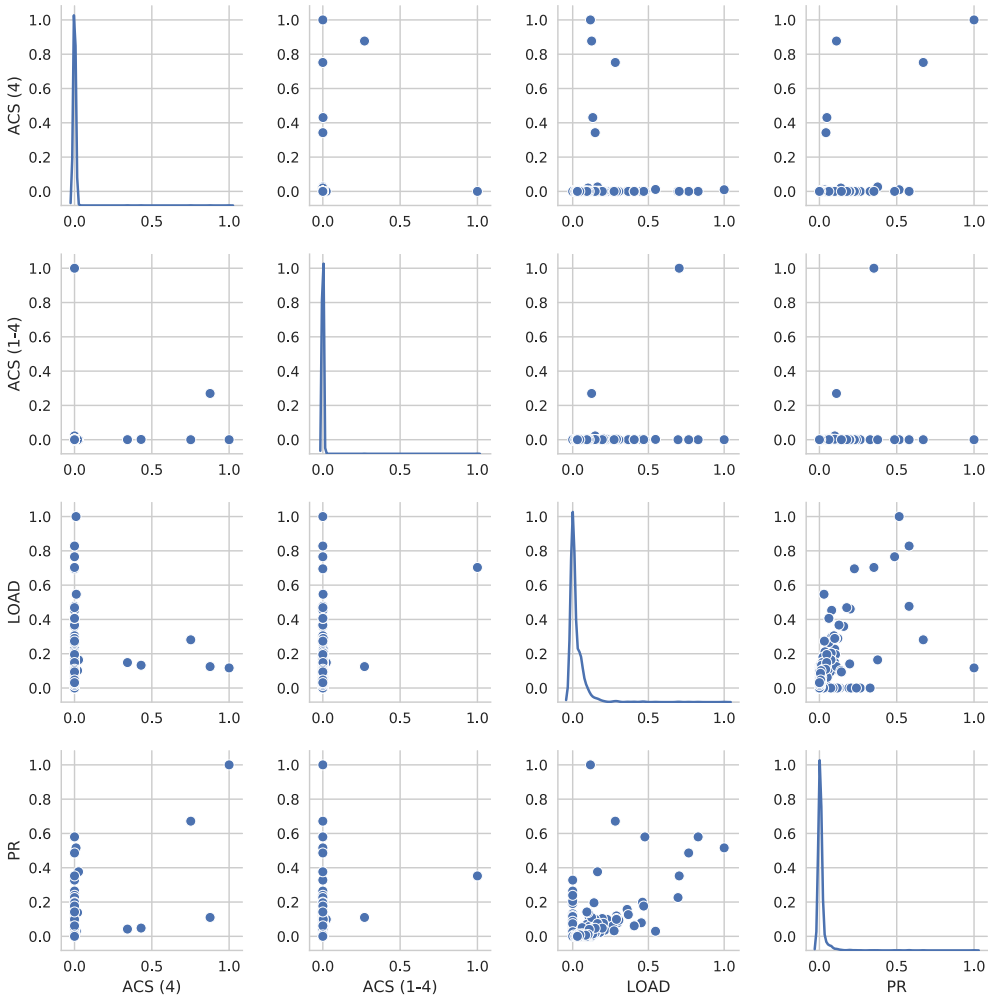


Figure 5.4: Scatter plots of the centrality metrics as computed on the MusicPinta EKG, showing the data points of the centrality metrics plotted against each other as well as the kernel density estimation for each metric on the diagonal axis.

5.2 Similarity Metrics

For the evaluation of similarity metrics the top node of each centrality metric (see Section 5.1) was chosen and the similarity to all other nodes in the Expert Knowledge Graph (EKG) was computed. Namely, results for Information Content/PageRank (ICPR), the newly suggested information content metric based on PageRank (PR), and Linked Data Semantic Distance (LDSD), the similarity metric specifically designed for the semantic web, are examined during this section (for implementation details see Section 4.2).

The evaluation was carried out with Expert Knowledge Graphs (EKGs) and Exploratory Search (ES) in mind. EKGs are a special kind of knowledge graph leveraging in semantic

web technology. Furthermore, they consist of specialized, domain-constrained knowledge, and sophisticated knowledge as an expert would have in his or her domain of expertise (see Section 2.1). ES is a broader view on search than previous search paradigms provides. In addition to lookup it takes into account activities like learning and investigation too (see Section 2.2). Exploratory Search Systems (ESSs) are software systems built for assisting an exploratory searcher.

The first part of the analysis again concentrates on the STAR EKG. In the second part the results from STAR are compared to results of the same evaluation conducted on the MusicPinta EKG. This approach was chosen in order to mitigate a potential outlier observation and show the portability of the metrics to various datasets.

STAR

Table 5.7 on page 63 and Table 5.8 on page 63 show the three highest ranking nodes (according to their LDSD and ICPR value respectively) for each of the top centrality nodes calculated in Section 5.1. There are several things of note when examining the tables.

First, the similarity nodes in Table 5.7 on page 63 partially consist of properties. From a naive view-point these are ought to materialize as edges in the EKG, therefore not occur in results of LDSD or ICPR. However, properties defined in OWL are also subjects (read nodes) in other triples[2]. Depending on the ESS built upon these algorithms such a duality may be desired or not. In this evaluation the properties were kept as nodes too. The advantage of this approach is that similarity (or relative importance) is assigned to edges too [3], which can be leveraged in an ESS. However, it would be straight-forward to filter out properties with an appropriate SPARQL query.

Second, whereas the centrality algorithms at least distinguished well between the highest ranking nodes, this cannot be said of the similarity metrics. In fact, when filtering out properties, there would be no difference in similarity values in the given tables. As already discussed, the similarity metrics depend on a differentiating class hierarchy to expose a high discriminative power. If the ontology is more homogeneous, the similarity between nodes tends to fall into one out of a few discrete categories of values. This pattern can be observed on the STAR EKG too.

From the perspective of an ESS, the suggested nodes in Table 5.7 on page 63 and Table 5.8 on page 63 can, for example, be used to facilitate further exploration after the user has explored an initial node suggested by one of the centrality metrics. Under this perspective, the nodes emphasized by LDSD may also be chosen by a human assessor for further exploration. However, the nodes emphasized by ICPR do not seem particularly well chosen considering the STAR EKG.

[2]Stating something to be a property already necessitates this — at least implicitly. Assigning a label to a property would be another common case.

[3]Note that in the average case ICPR implicitly filters out properties by construction.

Node	Value
context#AuthorTag	
generic#hasMethodTag	1.00000
instance#AuthorTag_technical_realization_plan	0.71366
instance#AuthorTag_cost-benefit_analysis	0.71366
context#ContextInformationElement	
generic#hasContext	1.00000
instance#ProjectPhase_architecture	0.71677
instance#AuthorTag_software_analysis	0.71677
instance#ArchitecturalQuality_scalability	
generic#ArchitecturalQuality	1.00000
instance#AuthorTag_scalability	1.00000
generic#ArchitecturalKnowledgeElement	1.00000
specific#UseCase	
specific#hasUseCaseRequirement	1.00000
specific#containedWithin	1.00000
instance#UseCase_sad_generation	0.71659

Table 5.7: The three highest ranking LDSD-nodes for each of the top centrality nodes in the STAR EKG

Node	ICPR
context#AuthorTag	
instance#Domain_ar-103___kpi_computation	1.00000
instance#Domain_ar-102___tool_integration	1.00000
instance#Domain_ar-021___privacy_monitoring	1.00000
context#ContextInformationElement	
instance#Domain_ar-103___kpi_computation	1.00000
instance#Domain_sr-01_01	1.00000
instance#Domain_ar-024___data_and_process_auditing	1.00000
instance#ArchitecturalQuality_scalability	
instance#Domain_ar-104_2___variable_filtering	1.00000
instance#Domain_ar-105___reports_and_analytics	1.00000
instance#Domain_ar-08___rapid_specification_authoring	1.00000
specific#UseCase	
instance#Domain_ar-102___tool_integration	1.00000
instance#Domain_ar-101___dwh_querying	1.00000
instance#Domain_ar-02___incentives_for_architects	1.00000

Table 5.8: The three highest ranking ICPR-nodes for each of the top centrality nodes in the STAR EKG

Summary statistics are given in Table 5.9 on page 64. Due to space constraints the following abbrevations were used:

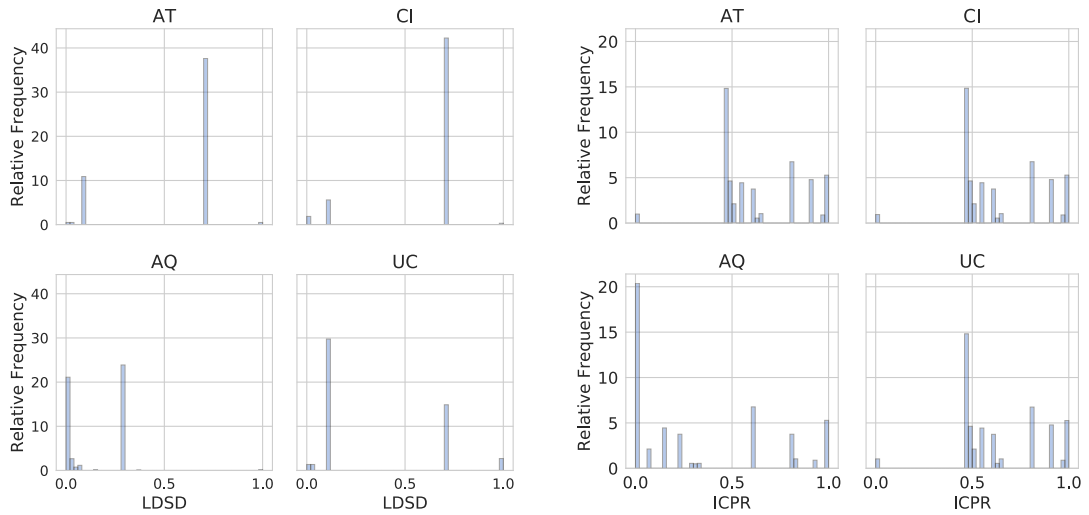
- AT... context#AuthorTag
- CI... context#ContextInformationElement
- AQ... instance#ArchitecturalQuality_scalability
- UC... specific#UseCase

Contrary to the impression from the top similar nodes in 63 and 63 the summary statistics show that, over all nodes, LDSD actually discriminates worse than ICPR with at least 50% of AT and CI and at least 25% of UC having the same values. In fact, it seems like LDSD is not usable for assessing discriminating values for medium- to low-profile nodes in many cases.

	LDSD				ICPR			
	AT	CI	AQ	UC	AT	CI	AQ	UC
mean	0.56755	0.62382	0.15474	0.32922	0.64082	0.64146	0.32844	0.64020
std	0.27144	0.22931	0.14558	0.32800	0.21219	0.21134	0.3715	0.21304
min	0	0	0	0	0	0	0	0
25%	0.71366	0.71677	0.01609	0.10308	0.47992	0.47992	0	0.47992
50%	0.71366	0.71677	0.07053	0.10308	0.55380	0.55380	0.14206	0.55380
75%	0.71366	0.71677	0.29288	0.71659	0.80185	0.80190	0.61900	0.80185
max	1	1	1	1	1	1	1	1

Table 5.9: Descriptive summary statistics of the similarity metrics for each of the top centrality metrics as computed on the STAR EKG.

Figure 5.5 shows a more detailed picture of the distribution of LDSD and ICPR values. Although ICPR contains more nodes in the top range (ICPR=1), the resolution over all possible values is finer than for LDSD. However, due to the dependence on discriminating elements in the ontology (like a fine-grained class hierarchy), both of the investigated similarity metrics exhibit a low sampling rate yielding only few, discrete categories of values.



(a) Histograms of LDSD values for the top centrality nodes in the STAR EKG

(b) Histograms of ICPR values for the top centrality nodes in the STAR EKG

Figure 5.5: Histograms of similarity values for the top centrality nodes in the STAR EKG

MusicPinta

For comparison and in order to reassure the observations made on the STAR EKG, the same similarity evaluation as before was repeated on the MusicPinta EKG. In fact, MusicPinta has a slightly more discriminating class structure which is why the similarity metrics could be expected to yield more satisfactory results. However, also the MusicPinta EKG does not come close to the optimal case of a thesauri-like class structure.

Despite this, the top similarity nodes for each of the most central nodes as selected in Section 5.1 show a surprisingly similar pattern on the MusicPinta EKG as for the STAR EKG (see Table 5.10 on page 66 and 66).

Again, LDSD seems to discriminate better among the top nodes than ICPR. In fact, there are several nodes with a top ICPR score (ICPR=1) such that the particular selection of the three nodes among the top nodes shown in Table 5.11 on page 66 is an artifact of the ordering in the evaluation program. This also explains the uniformity of the ICPR selection. Most of the musicbrainz:instrument/[0-9]+ instruments have the same ICPR score for each of the top centrality nodes. The uniform selection is due to the (same) ordering of equal values along an independent dimension in the evaluation program.

Node	Value
musicbrainz:instrument/124	
mo:Instrument	1.00000
dbpedia:Kontra	0.21856
dbpedia:Bandone%C3%B3n	0.21856
musicbrainz:instrument/69	
mo:Instrument	1.00000
dbpedia:Kontra	0.22383
dbpedia:Bandone%C3%B3n	0.22383
musicbrainz:instrument/75	
leeds:Media.owl#3c53d830-e548-11e2-b8b2-bc305beb9a1f	1.00000
leeds:Media.owl#3ad2a150-e546-11e2-b8b2-bc305beb9a1f	1.00000
leeds:Media.owl#f6eff330-e549-11e2-b8b2-bc305beb9a1f	1.00000

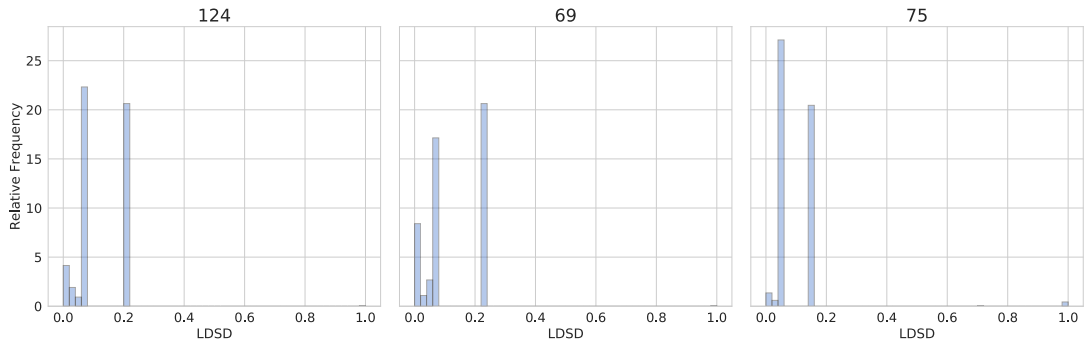
Table 5.10: The three highest ranking LDSD-nodes for each of the top centrality nodes in the MusicPinta EKG

Node	Label	Value
musicbrainz:instrument/124	Percussion instruments	
musicbrainz:instrument/102	Hardingfele	1.00000
musicbrainz:instrument/36	Tenor saxophone	1.00000
musicbrainz:instrument/309	Rebab	1.00000
musicbrainz:instrument/69	String instruments	
musicbrainz:instrument/102	Hardingfele	1.00000
musicbrainz:instrument/36	Tenor saxophone	1.00000
musicbrainz:instrument/309	Rebab	1.00000
musicbrainz:instrument/75	Guitars	
musicbrainz:instrument/103	Hurdy gurdy	1.00000
musicbrainz:instrument/36	Tenor saxophone	1.00000
musicbrainz:instrument/309	Rebab	1.00000

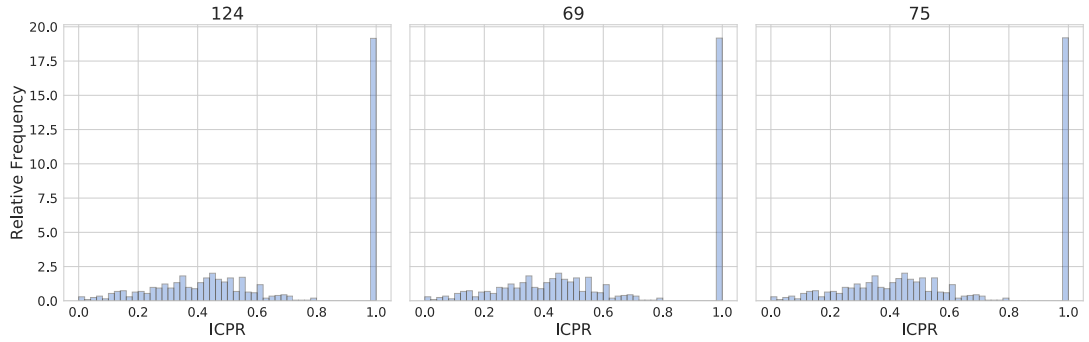
Table 5.11: Three of the highest ranking ICPR-nodes for each of the top centrality nodes in the MusicPinta EKG

The summary statistics in Table 5.12 on page 68 show a similar pattern compared to the STAR EKG too. Again, for space constraints, the similarity nodes where abbreviated by their unique number in their musicpinta:instrument/[0-9]+ uri pattern. Many similarity assignments have the same numerical value. However, the uniformity in the top quartile of ICPR stands out from the ICPR values obtained on the STAR EKG.

The question that may arise at this point is why the similarity metrics seem to work just equally well if not worse on the MusicPinta EKG as on the STAR EKG, despite the proposed more discriminating structure of MusicPinta. Indeed the nature of the similarity metrics is not yet fully revealed as Figure 5.6 shows.



(a) Histograms of LDSD values for the top centrality nodes in the MusicPinta EKG



(b) Histograms of ICPR values for the top centrality nodes in the MusicPinta EKG

Figure 5.6: Histograms of similarity values for the top centrality nodes in the MusicPinta EKG

LDSD exhibits only slight improvements over the STAR EKG. However, ICPR discloses an exceptionally well discriminating, and approximately normally distributed, pattern in the low to medium range. In fact, the only outlier is the top ICPR value (ICPR=1) with a large cluster of nodes assigned to it. Another peculiarity is the uniformity of all three ICPR plots in Figure 5.6b.

One explanation for the results of the analysis so far is that, although as a whole the MusicPinta EKG possesses a well discriminating class hierarchy, the subgraph induced by the `musicbrainz` sub-URIs does not. Hence, the ICPR algorithm is not able to distinguish between `musicbrainz` instruments, as there are no distinguishing features in the data which are potentially exploitable by ICPR.

To take a closer look at this new hypothesis, the MusicPinta EKG was further pruned and all nodes with a `musicbrainz` sub-URI were removed. In addition, the URIs of the form leeds:Media.owl#.* were removed since they bear no meaning in the graph. Note that after this final pruning the remaining EKG consisted almost solely of dbpedia instruments.

	LDSD			ICPR		
	124	69	75	124	69	75
mean	0.12185	0.12101	0.09718	0.62868	0.62888	0.62897
std	0.08851	0.09447	0.10366	0.31902	0.31912	0.31926
min	0	0	0	0	0	0
25%	0.06315	0.06947	0.04370	0.36077	0.36046	0.36009
50%	0.06315	0.06947	0.04370	0.54553	0.54625	0.54649
75%	0.21856	0.22383	0.15571	1	1	1
max	1	1	1	1	1	1

Table 5.12: Descriptive summary statistics of the similarity metrics for each of the top centrality metrics as computed on the MusicPinta EKG.

After pruning, the same similarity analysis as before was conduced again. That is, the most central nodes, as recalculated by the various centrality metrics on the pruned graph, were selected and the similarity values were then computed again.

Whereas LSDS did not show much of an improvement, ICPR could benefit from the new graph. Figure 5.7 shows the new ICPR histogram. As expected, the dominant cluster at ICPR=1 disappeared. Furthermore, the remaining values exhibit a (slightly skewed) normal distribution with many distinct values. Additionally, the uniformity of the three similarity calculations was replaced by more unique patterns for each of the top centrality nodes.

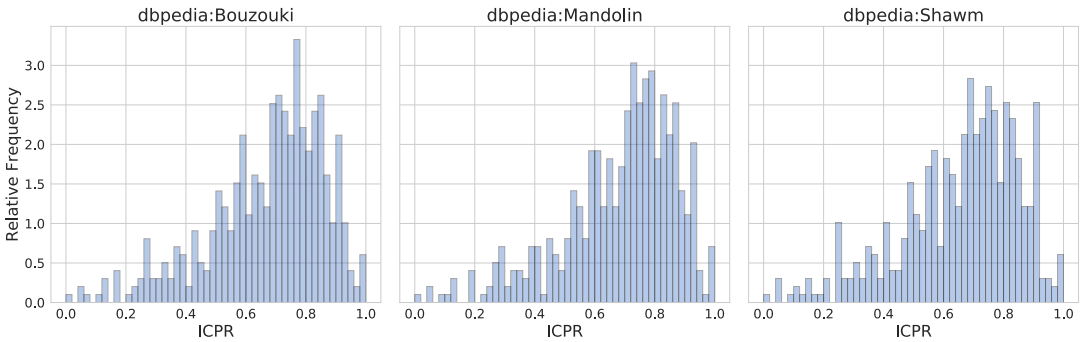


Figure 5.7: Histograms of similarity values for the top centrality nodes in the MusicPinta EKG after pruning the MusicBrainz nodes.

CHAPTER 6

Discussion and Conclusion

This diploma thesis investigated Exploratory Search (ES) on Expert Knowledge Graphs (EKGs). EKGs are a special kind of knowledge graph leveraging in semantic web technology. Furthermore, they consist of specialized, domain-constrained knowledge, and sophisticated knowledge as an expert would have in his or her domain of expertise (see Section 2.1). ES is a broader view on search than previous search paradigms provides. In addition to lookup it takes into account activities like learning and investigation too (see Section 2.2). Exploratory Search Systems (ESSs) are software systems built for assisting an exploratory searcher.

The field of ES is still a largely undetermined topic. Although several approaches exist (see Section 2.2), a clear direction is missing and its definition is currently evolving constantly. Furthermore, ES is a vast topic spanning various fields from human centered approaches like psychology to approaches leaning more towards engineering approaches in computer science. Any attempt to tackle this problem can only hope to contribute a small part to this topic.

ES also found its way into the semantic web research already. However, current approaches are either tailor made solutions applicable only to a single ontology, concentrate solely on the user interfacing part of an ESS or handle the problem on a more theoretic level. Only few research tries to investigate possible algorithmic fundamentals of an ES (Section 2.2).

In contrast this diploma thesis tries to highlight particularly this part of an ESS. Two similarity metrics, namely Linked Data Semantic Distance (LDSD) and Information Content/PageRank (ICPR) have been implemented. ICPR is newly suggested information content metric based on PageRank (PR), LDSD is a similarity metric specifically designed for the semantic web. Furthermore, the three centrality algorithms PageRank (PR), Accessibility (ACS), and Load have been implemented via Gather Apply Scatter (GAS). Centrality metrics try to estimate the importance or the influence of a node in the graph. GAS is an abstract computational model consisting of the three distinct

phases Gather, Apply and Scatter. During each phase computations can be conducted that alter well specified states of vertices or edges (see also Section 2.5).

The approach taken in this diploma thesis is novel in several ways that will be discussed during this section. In addition, the results obtained from conducting several algorithms (see Section 4.2 and Chapter 5) are revisited again and conclusions from these results will be drawn.

The main research question examined in this diploma thesis was:

RQ–I. *How can exploratory search on semantic web technologies be supported algorithmically?*

The suggested solution are metrics assessing the centrality of a single node and the similarity between two nodes in the semantic graph. The selected metrics vary in their nature and show different approaches how algorithms can be used for ES on EKGs.

The centrality metrics stem from general graph algorithms that were not designed with ES or semantic web technologies in mind. One of the major advantages of semantic web technologies compared to other data models is their immediate representation as a graph. Research on graph algorithms has a long tradition already. Although it may seem like a natural fit, almost no current research tries to leverage the accumulated knowledge in both of these fields to advance ES. The transfer into the realm of EKGs shown in this diploma thesis offers a procedure that can be used for many other general graph algorithms too.

Centrality metrics estimate the importance of a node. The notion of importance differs from metric to metric. This can be used in various ways for exploratory search. For example, an important node can be suggested as a starting point for exploration to a user unfamiliar with the EKG domain. As part of a faceted search system, facets can be selected and/or ordered by node centrality too.

The first similarity metric, information content based on PR (ICPR) is a novel suggestion for an information content based metric. It leverages PR for associating probabilities to classes. PR is known to converge fast and, in concert with the suggested approach for centrality metrics, is also readily available.

LDSD is a similarity metric specifically designed with ES and semantic web technologies in mind. This kind of algorithms is comparatively easy to implement since it uses the SPARQL capabilities of triple stores without the need for further provisions.

A potential shortcoming of LDSD with respect to other metrics like the discussed graph algorithms is the lack of theoretic reasoning. LDSD was developed by empirically evaluating various combinations of the basic ingredients (C_d, C_{ii}, etc. See Section 4.2.5). However, the only evidence that this metric is advantageous for ES is given by an empirical user study.

The second research question was concerned with transferring the investigated methods to different datasets:

RQ–II. *What are the characteristics of datasets like STAR that are required by the algorithms?*

In Section 2.1 several characteristics of EKGs were proposed that can be presupposed by the algorithms. Some of them, like a graph representation, are immediately given by datasets specified via semantic web technologies.

However, others are less widespread. One of the defining characteristics of EKGs are their constrained size (expert knowledge is rather small and complex than shallow and huge). Indeed, despite the GAS computational model allows for parallelization, it also puts severe constraints on the algorithms, like immediate vicinity of the gather and scatter nodes. This results in demanding workaround like seen in Section 4.2. Also, most of the investigated algorithms are inherently demanding. Depending on the available computing power, a comparatively a constrained size is a necessary precondition.

A sophisticated structure of the graph is another defined characteristic of expert knowledge and consequently EKGs. The extend to which this is required was only revealed in Chapter 5. In fact, it seems like there is a need for less sensitive algorithms.

In particular the similarity metrics are very sensitive to the discriminative power of the class structure in the EKG. That is, if the class structure is shallow with few classes containing the majority of nodes, or subsets of nodes are all assigned to the same classes, the algorithms cannot distinguish well between the similarity of nodes. The result are few distinct similarity values assigned to large subsets of the nodes. While centrality metrics are subject to the same problem in principle, they were found to be more robust in that regard.

RQ–III. *To what extent is it possible to transfer the investigated methods to different datasets?*

The proposed centrality metrics, as implemented depend on no internals of the dataset beside the graph structure and the previously discussed characteristics. If the term portability is extended to the environment the datasets commonly reside in, then the triple store containing the data has to support a GAS interface. Currently the only triple store known to the authors to support this is Blazegraph. However, the hope is that other vendors will also recognize the need for direct access to the graph and implement a GAS interface.

The similarity metrics do not depend on internals of the datasets, besides the discussed characteristics, as well. However, they must be held in a triple store supporting SPARQL. This is a reasonable assumption, since triple stores are prevalent for data provision in the semantic web.

Indeed, in Chapter 5 the proposed metrics were also conducted on two different EKGs. Besides switching the datasets in the triple store no adaptions to the algorithms had to be made.

The third research question targeted to implementation on triple stores which are a major part of the semantic web stack. However, algorithms that do not align well with

SPARQL are implemented by bypassing triple stores in current literature. Hence, the question was:

RQ–IV. *How can we implement the proposed algorithms based on a common triple store?*

Since the similarity metrics leaned themselves well towards a SPARQL-based implementation, this question had to be answered only for the centrality metrics as examples of general graph algorithms.

Loading the raw RDF data into a specialized graph data structure is not enough as many of the defining advantages of semantic web ontologies, like triple inference and consistency checks, would be lost. Another possibility would be to use the SPARQL access triple stores commonly provide.

However, as already mentioned, SPARQL is not powerful enough to support most graph algorithms since a loop construct is missing. Continuously issuing queries to a SPARQL endpoint and processing the results externally is also no sustainable solution, since the demands of graph algorithms are oftentimes too high to be computed efficiently this way.

Currently, common triple stores provide almost no solutions for efficient access to the graph structure. One was found in the GAS API provided by Blazegraph. Besides being still only a preliminary API, the graph algorithms were reformulated in this paradigm.

Whereas PR is a natural fit for GAS (in fact GAS is an abstraction and generalization of the PR computation), the restrictions of GAS (Section 2.5) induce obstacles hard to come by in other algorithms. Workarounds often require to carry around a lot of state for bookkeeping. Also, GAS is still a rather novel paradigm such that there is not much experience in reformulating algorithms for GAS that could be built upon. Nevertheless, it was possible to reformulate all of the proposed algorithms in this paradigm.

Finally, the last research question addressed in this master thesis was:

RQ–V. *Which of the proposed algorithms work best? What are potential shortcomings and why?*

The prevalent shortcoming of centrality algorithms was their lack of differentiation in the medium to low value range. While the centrality algorithms worked well in selecting and distinguishing the top nodes, most of the nodes built a cluster at a very low value range. Many of them got assigned the same values. Depending on the ES system built upon the algorithms this might be enough (e.g., if the ES system is only interested in the top nodes).

With respect to differentiation of the nodes, LOAD did exhibit the most beneficial results. Although still subject to a cluster in the low value range, LOAD distributed more nodes along the whole value range than any other of the tested centrality metrics. This behavior could be observed on both datasets, hence it seems to be a property of the algorithm rather than properties of the data.

Another peculiarity could be observed for the averaged ACS for path lengths between one to four. This metric tends to overemphasize one or very few nodes, therefore, in

relation, dwarfing all other values. However, if only the rank of the nodes is considered, this is not an obstacle.

The rank correlation between the centrality metrics were mostly in the low to medium range. This hints at desirable properties of the metrics. On the one hand there might be a common component measured by all metrics (the fundamental importance of a node). On the other hand the metrics also seem to measure slightly different notions of importance, therefore being all relevant and cannot be substituted for one another. However, this hypothesis would demand further investigations for a final conclusion.

As already mentioned, the similarity metrics require a very discriminative class structure to work well. The similarity metrics suggested in this diploma thesis seem to be too sensitive to this requirement. ICPR was not able to distinguish between the most similar nodes at all. LDSD performed slightly better in the top range, tough still dissatisfying. Conversely, over the whole value range ICPR performed slightly better than LDSD. Both metrics reduced the similarities into only a few distinct categories.

Whereas the pattern for LDSD was similar on any of the tested datasets, ICPR exhibited a different behavior on the MusicPinta EKG. In fact, it turned out that the ICPR metric was able to differentiate very well between the similarity of the MusicPinta nodes. However, this was hidden by a particular subgraph of the MusicPinta EKG. After pruning this subgraph from the EKG, the results from ICPR were satisfying and clearly superior to LDSD. However, this also means that ICPR is sensitive to even parts of an EKG not having a high discriminative power.

CHAPTER 7

Summary and Future Work

During this diploma thesis algorithms for Exploratory Search (ES) based on semantic web technologies were proposed and investigated. ES is a broader view on search than previous search paradigms provides. In addition to lookup it takes into account activities like learning and investigation too. Exploratory Search Systems (ESSs) are software systems built for assisting an exploratory searcher.

The investigated algorithms fell into two categories: Metrics estimating centrality of a node and metrics estimating similarity between two nodes. Two similarity metrics, namely Linked Data Semantic Distance (LDSD) and Information Content/PageRank (ICPR) have been implemented. ICPR is newly suggested information content metric based on PageRank (PR), LDSD is a similarity metric specifically designed for the semantic web. Furthermore, the three centrality algorithms PageRank (PR), Accessibility (ACS), and Load have been implemented via Gather Apply Scatter (GAS). Centrality metrics try to estimate the importance or the influence of a node in the graph.

In order to be able to use triple stores, a major component of the semantic web stack, graph algorithms were reformulated in the GAS computational model. GAS is an abstract computational model consisting of the three distinct phases Gather, Apply and Scatter. During each phase computations can be conducted that alter well specified states of vertices or edges. The algorithms were then conducted on two different datasets. The STAR EKG and the MusicPinta EKG. Expert Knowledge Graphs (EKGs) are a special kind of knowledge graph leveraging in semantic web technology. Furthermore, they consist of specialized, domain-constrained knowledge, and sophisticated knowledge as an expert may possess in his or her domain of expertise

The centrality metrics worked reasonably well but exhibited less desirable properties beneath the most central nodes. Requirements of the similarity metrics were not met by the STAR EKG, leaving some room for improvements on the metrics. The novel ICPR algorithm suggested in this diploma thesis however worked very well on the MusicPinta

EKG after a confounding subgraph was pruned from the data set. LDSD did not match ICPR on the MusicPinta EKG, even after the pruning.

Before discussing possible future work, a look back to the beginning of this diploma thesis is taken here. There the use case of ES for supporting businesses in managing and exploiting intellectual capital was discussed. However, this example was also generalized to the overarching problem of becoming an expert in light of increasingly more sophisticated expert knowledge. Another stated use case of an ESS was also just to help the casual knowledge worker in his or her daily work with the information in this world.

Considering these examples the question arises how the results from this thesis have contributed to these larger future visions. The problem of creating a fully working ESS was not solved. Instead, only a small part (possible core algorithms) were investigated. However, the algorithms could be used already today as part of the backend of a faceted search interface, or some other user interface and -interaction methodology for ES.

Furthermore, graph algorithms were reformulated in the GAS model and applied to a common triple store. This also provides possible case study for implementing many more graph algorithms on triple stores. As a matter of fact, graph algorithms have a long history with many interesting results already. Show-casing how ESS for the semantic web can leverage this history is another small contribution to the overall vision.

Although just another similarity algorithm in an already rich selection, the newly proposed ICPR algorithm managed to improve upon LDSD in some cases. While a small contribution (and ICPR still not being perfect), the algorithm can be used in the backend of an ESS for improved similarity assessment.

Finally, a business looking to get an exploratory search system at a level of sophistication the research vision envisions is still out of luck. Provided in this diploma thesis are only small steps embedded in a much larger overarching task.

While this diploma thesis tried to advance ES on semantic web technologies in various novel ways, there are still several open questions that were purposefully not addressed here, as well as new questions sprouted during the course of this work.

Probably one of the hardest, overarching tasks for future research concerns the notion of ES itself. While there exists a wealth of research concerning theoretic models of ES, they often reside on a very abstract level leaving a gap between the models and the needs of research trying to use them. In the future it would be desirable to close this gap by operationalizing the high level terms used in theoretic models of ES. This would not only be useful in research of methods and algorithms supporting ES but also in their evaluation. This is of course a highly non-trivial problem.

A more accessible task for future research is trying to transfer knowledge gained in previous research to ES in the semantic web. There are still many graph algorithms that could be useful for ES. Reformulating and evaluating them in the context of semantic web technologies would be interesting.

In that vein future research could also look into other possibilities to access the graph structure inherent to semantic web technologies via triple stores. While GAS is one possibility, it is also quite restrictive (mainly due to the locality requirement of gather and scatter edges/nodes). This makes it hard to port algorithms developed in other areas over to triple stores. However, the immediate graph representation is a major advantage of the semantic web over other technologies like relational databases. Finding possibilities to make the underlying graph more accessible to applications built upon triple stores could be an important future contribution.

Another topic future work could look more into is the generalizability of approaches. Most of the current research on ES in the semantic web is effectively bound to a single ontology or application only. Extracting only the bits and pieces that are quintessential to these approaches could make them transferable to other datasets as well. The notion of EKGs introduced in this diploma thesis could be used for this task as well or refined in future research to capture more of the features needed for generic approaches.

More specific to this diploma thesis, future research could try to evaluate the results in a user centered study. While the synthetic analysis in this diploma thesis provides valuable insights, a user study could better assess the effectiveness from a user perspective.

However, this would also presume the last suggestion for future work given in this section. Future work could try to integrate the provided algorithmic basis into a fully fledged ESS. Whether via a faceted interface as suggested at several points during this thesis or with another interface and interaction paradigm. The metrics discussed here are potentially flexible enough to support various approaches. However, this is left for the future.

List of Figures

List of Tables

List of Algorithms

Acronyms

ACS Accessibility. 5, 19, 34, 35, 49, 51, 68

ACS(1-4) Averaged Accessibility with path length 1 to 4. 49–51, 53, 55

ACS(4) Accessibility with path length 4. 49, 53, 55

API Application Programming Interface. 29–31, 67, 68

EKG Expert Knowledge Graph. 2–9, 15, 17, 22, 23, 29–31, 39, 42, 43, 49–63, 65–69, 71–75

ES Exploratory Search. 1–5, 9–17, 20, 22, 29, 30, 49, 65, 66, 68, 71–73

ESS Exploratory Search System. 1, 2, 11, 50, 53, 54, 57, 65, 72

GAS Gather Apply Scatter. 4–6, 15, 18, 24–27, 29, 31, 32, 34, 35, 37, 39–41, 43, 45, 66–68, 71–73

ICPR Information Content/PageRank. 29, 49, 57, 59–63, 66, 68, 69, 71

IRI Internationalized Resource Identifier. 8, 53, 73

LDSD Linked Data Semantic Distance. 23, 24, 29, 30, 45, 49, 57, 59, 60, 62, 66, 68, 69, 71

OWL Web Ontology Language. 2, 8, 9, 13, 57

PR PageRank. 19, 20, 29, 32–34, 44, 49, 51, 53, 55, 57, 66, 68

RDF Resource Description Framework. 2, 8, 9, 13–15, 17, 35, 43, 67, 73

RDFS RDF Schema. 13, 14

SKOS Simple Knowledge Organization System. 8, 9

SPARQL SPARQL Protocol and RDF Query Language. 4, 14, 15, 17, 29, 30, 32, 43, 45, 48, 57, 66, 67

Bibliography

[AB70] Mary D. Salter Ainsworth and Silvia M. Bell. Attachment, exploration, and separation: illustrated by the behavior of one-year-olds in a strange situation. *Child Development*, 41(1):49–67, 1970.

[AGJ+16] Kumaripaba Athukorala, Dorota Głowacka, Giulio Jacucci, Antti Oulasvirta, and Jilles Vreeken. Is exploratory search different? a comparison of information search behavior for exploratory and lookup tasks. *Journal of the Association for Information Science and Technology*, 67(11):2635–2651, 2016.

[AGK+16] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarūnas Marciŭska, and Dmitriy Zheleznyakov. Faceted search over RDF-based knowledge graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37-38:55–74, 2016.

[AK12] Tatiana Andreeva and Aino Kianto. Does knowledge management really matter? linking knowledge management practices, competitiveness and economic performance. *Journal of Knowledge Management*, 16(4):617–636, 2012.

[AT05] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[Bat89] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424, 1989.

[Bav48] Alex Bavelas. A mathematical model for group structures. *Human Organization*, 7(3):16–30, 1948.

[BCG+12] Jie Bao, Diego Calvanese, Bernardo Cuenca Grau, Martin Dzbor, and Achille Fokoue et al. OWL 2 web ontology language. Technical report, W3C, December 2012. `http://www.w3.org/TR/2012/REC-owl2-overview-20121211`.

[BEF+56] Benjamin S Bloom, Max D Engelhart, Edward J Furst, Walker H Hill, and David R Krathwohl. *Taxonomy of educational objectives, Handbook I: The cognitive domain*, volume 19. David McKay Co Inc, 1956.

[BG14] Dan Brickley and Ramanathan Guha. RDF Schema 1.1. W3C recommendation, W3C, 2014. `http://www.w3.org/TR/2014/REC-rdf-schema-20140225`.

[BM09] Sean Bechhofer and Alistair Miles. SKOS simple knowledge organization system reference. W3C recommendation, W3C, August 2009. `http://www.w3.org/TR/2009/REC-skos-reference-20090818`.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107 – 117, 1998.

[BPCF17] Federico Bianchi, Matteo Palmonari, Marco Cremaschi, and Elisabetta Fersini. Combining serendipity and active learning for personalized contextual exploration of knowledge graphs. *Semantic Web*, 2017. Submitted for peer-review.

[Bra01] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[Bra08] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136 – 145, 2008.

[CWL14] Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C, 2014. `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225`.

[CY11] Valerie Cross and Xinran Yu. Investigating ontological similarity theoretically with fuzzy set theory, information content, and tversky similarity and empirically with the gene ontology. In S. Benferhat and J. Grant, editors, *Proceedings of the 5th International Conference on Scalable Uncertainty Management*, volume 6929 of *SUM'11*, pages 387–400, Berlin, Heidelberg, 2011. Springer-Verlag.

[DLT+13] Vania Dimitrova, Lydia Lau, Dhavalkumar Thakker, Fan Yang-Turner, and Dimoklis Despotakis. Exploring exploratory search: A user study with linked semantic data. In *Proceedings of the 2nd International Workshop on Intelligent Exploration of Semantic Data*. ACM Press, 2013.

[DNMO+12] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 1–8, New York, NY, USA, 2012. ACM.

[DS05] M. Duerst and M. Suignard. Internationalized resource identifiers (iris). RFC 3987, RFC Editor, January 2005. http://www.rfc-editor.org/rfc/rfc3987.txt.

[Fre77] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[FSSS09] Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab. Triplerank: Ranking semantic web data by tensor decomposition. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009: 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, pages 213–228, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[GBL+12] Joseph E. Gonzalez, Danny Bickson, Yucheng Low, Carlos Guestrin, and Haijie Gu. Powergraph: Distributed graphparallel computation on natural graphs. In *The 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012.

[Haz82] Nancy L. Hazen. Spatial exploration and spatial knowledge: Individual and developmental differences in very young children. *Child Development*, 53(3):826–833, 1982.

[HMK05] David Huynh, Stefano Mazzocchi, and David R. Karger. Piggy bank: Experience the semantic web inside your web browser. In *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, pages 413–430, 2005.

[Hof98] Robert R. Hoffman. How can expertise be defined? implications of research from cognitive psychology. In Robin Williams, Wendy Faulkner, and James Fleck, editors, *Exploring Expertise: Issues and Perspectives*, pages 81–100, London, 1998. Palgrave Macmillan UK.

[HS13] Steven Harris and Andy Seaborne. SPARQL 1.1 query language. W3C recommendation, W3C, March 2013. http://www.w3.org/TR/2013/REC-sparql11-query-20130321.

[HvOH06] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A browser for heterogeneous semantic web repositories. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo, editors, *The Semantic Web - ISWC 2006: 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006. Proceedings*, pages 272–285, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[HZL08] Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. gFacet: A browser for the web of data. In *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web*, volume 417 of *IMC-SSW'08*, pages 49–58, 2008.

[JW02] Glen Jeh and Jennifer Widom. Simrank: A measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[Kat53] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, Mar 1953.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.

[KRSV14] Aino Kianto, Paavo Ritala, John-Christopher Spender, and Mika Vanhala. The interaction of intellectual capital assets and knowledge management practices in organizational value creation. *Journal of Intellectual Capital*, 15(3):362–375, 2014.

[KVH16] Vasiliki Kalavri, Vladimir Vlassov, and Seif Haridi. High-level programming abstractions for distributed graph processing. *CoRR*, abs/1607.02646, 2016.

[Lin98] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning,*, ICML 1998, pages 296–304, 1998.

[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 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[Mar06] Gary Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[MES+17] J. Musil, F. J. Ekaputra, M. Sabou, T. Ionescu, D. Schall, A. Musil, and S. Biffl. Continuous architectural knowledge integration: Making heterogeneous architectural knowledge available in large-scale organizations. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 189–192, 2017.

[MG14] Nicolas Marie and Fabien Gandon. Survey of linked data based exploration systems. In *Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data*, volume 1279, pages 66–77. CEUR-WS.org, 2014.

[MRDNDS10] Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio. Ranking the linked data: The case of dbpedia. In Boualem Benatallah, Fabio Casati, Gerti Kappel, and Gustavo Rossi, editors, *Web Engineering: 10th International Conference, ICWE 2010, Vienna Austria, July 5-9, 2010*, pages 337–354, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[MW04] Deborah McGuinness and Christopher Welty. OWL web ontology language guide. W3C recommendation, W3C, 2004. `http://www.w3.org/TR/2004/REC-owl-guide-20040210`.

[New01] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical Review E*, 64:016132, Jun 2001.

[New06] M. E. J. Newman. Erratum: Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality [phys. rev. e 64, 016132 (2001)]. *Physical Review E*, 73:039906, Mar 2006.

[New10] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[NPG+17] Andrea Giovanni Nuzzolese, Valentina Presutti, Aldo Gangemi, Silvio Peroni, and Paolo Ciancarini. Aemoo: Linked Data Exploration Based on Knowledge Patterns. *Semantic Web*, 8:87–112, 2017.

[OPSPL11] Catherine Olsson, Plamen Petrov, Jeff Sherman, and Andrew Perez-Lopez. Finding and explaining similarities in linked data. In *Semantic Technology for Intelligence, Defense, and Security*, pages 52–59, 2011.

[Pas07] Alexandre Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *Linked AI: AAAI Spring Symposium Linked Data Meets Artificial Intelligence*. AIII, 2007.

[Pas10] Alexandre Passant. dbrec - music recommendations using dbpedia. In *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, pages 209–224, 2010.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[PD10] Alexandre Passant and Stefan Decker. Hey! ho! let's go! explanatory music recommendations with dbrec. In *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*, pages 411–415, 2010.

[PGGT17] Émilie Palagi, Fabien L. Gandon, Alain Giboin, and Raphaël Troncy. A survey of definitions and models of exploratory search. In *Proceedings of the 2017 ACM Workshop on Exploratory Search and Interactive Data Analytics, ESIDA@IUI 2017, Limassol, Cyprus, March 13, 2017*, pages 3–8, 2017.

[PS08] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C recommendation, W3C, January 2008. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115`.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 448–453, 1995.

[RMBB89] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, January 1989.

[Ruf84] Holly A. Ruff. Infants' manipulative exploration of objects: Effects of age and object characteristics. *Developmental Psychology*, 20(1):9–20, 1984.

[sChC10] Seung seok Choi and Sung hyuk Cha. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, pages 43–48, 2010.

[SDRL06] Andreas Schlicker, Francisco S. Domingues, Jörg Rahnenführer, and Thomas Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, 7(1):302, Jun 2006.

[Sha92] James Shanteau. The psychology of experts an alternative view. In George Wright and Fergus Bolger, editors, *Expertise and Decision Support*, pages 11–23, Boston, MA, 1992. Springer US.

[SWRS06] m.c. Schraefel, Max Wilson, Alistair Russell, and Daniel A. Smith. mspace: Improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM*, 49(4):47–49, April 2006.

[TdFC08] B.A.N. Travençolo and L. da F. Costa. Accessibility in complex networks. *Physics Letters A*, 373(1):89 – 95, 2008.

[Wal23] HA Wallace. What is in the corn judge's mind. *Journal of the American Society of Agronomy*, 15(7):300–304, 1923.

[Wil96] Robin J. Wilson. *Introduction to graph theory*. Longman, Harlow, 4th ed edition, 1996.

[WKW+10] Jörg Waitelonis, Magnus Knuth, Lina Wolf, Johannes Hercher, and Harald Sack. The path is the destination – enabling a new search paradigm with linked data. In *In Proc. of the Workshop on Linked Data in the Future Internet at the Future Internet Assembly, Dec 16–17, 2010, Ghent, Belgium, CEUR Workshop Proc*, 2010.

[WM76] Ann Weisler and Rober R. McCall. Exploration and play: Resume and redirection. *American Psychologist*, 31(7):492–508, 1976.

[WR09] Ryen W. White and Resa A. Roth. *Exploratory Search : Beyond the Query-Response Paradigm.* Morgan & Claypool Publishers, 2009.