NTNU
Norwegian University of
Science and Technology

Knowledge exploration in public linked data ontologies

Espen Albert

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Jon Atle Gulla, IDI
Co-supervisor: Cristina Marco, IDI

Norwegian University of Science and Technology
Department of Computer Science

Summary

The internet is constantly expanding across millions of web pages. Using the internet effectively is a hard skill to learn both for humans and machines. The Semantic Web is an attempt to standardize the data across the web making it accessible and usable. This thesis is meant as a practical guide to using knowledge graphs the main building blocks of the Semantic Web. A knowledge graph is a large data source with facts about millions of real and fictional entities. Wikidata and DBpedia are two publicly available knowledge graphs that we use to look at three major aspects: Understanding knowledge graphs, finding relevant information from knowledge graphs, and creating features from knowledge graphs. These three aspects will be explained by using the task of finding similar entities, specifically similar artists.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) as a part of the course TDT4900 Computer Science, Masters thesis, mandatory for the completion of the degree of Master of Science at NTNU. The work culminating in this report has been performed during the fall of 2016 and the spring of 2017 at the Department of Computer and Information Science(IDI) with Professor Jon Atle Gulla and Postdoctoral Fellow Cristina Marco as supervisors, and as a part of the larger SmartMedia project.

Acknowledgment

I would like to thank my supervisors for their expertise, guidance, and trust to give me freedom to explore and influence the task, this thesis would not see the light of day without your help.

E.A.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The introduction of the Internet has impacted how we organize and find information. The semantic web is a part of the Internet's evolution, it attempts to make information on the Internet accessible and understandable for humans and machines. Knowledge Graphs (KGs) are the main building blocks of the semantic web. These are huge graphs describing real and fictional entities: "things not strings" (Google Inc.). Since KGs describe millions of entities and relations, their application areas are numerous: enriching entities, finding similar entities, finding related entities, exploring relations between entities, answering questions about entities, etc. Schuhmacher and Ponzetto (2014); Pirrò (2015); Cheng et al. (2014); Yahya et al. (2016) However, using a KG is complicated by its large size. With millions or billions of statements KGs require a systematic approach for finding relevant information. Therefore, the goal of this thesis is to give an introduction on how to extract knowledge from a KG with a more in depth analysis on how to find similar entities.

1.2 Problem Outline

When extracting knowledge from KGs there are four main challenges:

1. Understanding the information structure
2. Extracting features
3. Ranking features
4. Presenting the knowledge to a person

The first issue when using a KG is to understand how the information is structured. The resource description framework (RDF) (details in sec. 2.1.1) is the only standard that exists across knowledge graphs. This standard enables KGs to use shared vocabularies.

These vocabularies are important since they make it possible to talk about the same thing by using the same identifiers. There exist efforts to standardize the vocabularies (e.g. schema.org); however, most KGs use their own ontology and identifiers for representing entities and their relations. The second challenge when using a KG is to create features from the RDF-statements. E.g. Di Noia et al. (2012) creates a recommender-system for movies where each movie is represented by a vector which is build by using the property of the statements in the movie domain. The third challenge concerns ranking the features. Also, by considering the data and user of the query this problem is further complicated. E.g. a recommender-system for similar artist might requires different ranking than a recommender-system for similar politicians and both systems most likely also depend on the user's preferences. Presenting the information to a human is the fourth challenge of using knowledge graphs. Although the RDF has formats which are understandable to humans (e.g. turtle syntax), these formats require existing competence and are limited in their expressive power.
This thesis focus mostly on the three first challenges: the information structure, the process of feature generation, and the process of feature ranking for Wikidata and DBpedia. Moreover, this thesis use the task of finding similar entities as the knowledge extraction task. Previous approaches often use the KG in conjunction with other information sources (e.g. search engines, social media, dictionaries, web pages)Yahya et al. (2016); Zhang et al. (2015); Pirrò (2015); Bi et al. (2015) These information sources often prove valuable since they can contain more information relating to the task (e.g. entity popularity or user preferences). On the downside, these information sources make the experiments much harder to replicate since they depend on inaccessible or non-deterministic sources. Additionally, most work use a combined set of features with an unique weighting scheme which makes it hard to determine the importance of the different features from the KG. By using features exclusively from the KG, analyzing one type of feature at a time, and testing on a gold standard dataset this thesis presents reproducible methods for extracting knowledge from the KG. Wikidata was selected based on the quality of the information (user created content with references) and the newness of the knowledge graph. DBpedia, on the other hand, is an older and more widely studied knowledge graph; therefore, a good candidate for comparison and knowledge extraction.

1.3 Research Goals and Questions

Providing as a guidance for the thesis, the following research questions have been defined:

RQ1: Which features are the most important to compute similarity and/or relatedness between two or more entities in Wikidata and DBpedia?

RQ2: How can the relationship between two or more given entities be described or labeled?

RQ3: How can similar/related entities be found in Wikidata and DBpedia?

RQ4: How can the context from news articles (containing the entity) be used in order to find similar/related entities?

RQ5: How can the informativeness of the properties of the entities in Wikidata and DBpedia be measured?

1.4 Research Contributions

This thesis contains three research contributions: First, a description and comparison of the data model, type- and property-hierarhcy of Wikidata and DBpedia (chapter 4). Secondly, paper 1: How to retrieve relevant information from knowledge graphs using common subsumers: Entity types, similar types, and entity requirements. The paper presents four methods for retrieving relevant information based on an entity's type and properties. And each method is tested on the Semantic Artists Similarity dataset Oramas et al. (2015) (chapter 5). Third, paper 2: Feature generation methods for Knowledge Graphs based on Common Subsumers and counts. The paper evaluates five feature generation methods for extracting and ranking information from the knowledge graph (chapter 6).

1.5 Thesis Structure

The rest of this thesis is structured as follows: Chapter 2 describes relevant background information. Chapter 3 gives an overview of related work areas relevant to the 4 challenges introduced in sec 1.2. Chapter 4 describes Wikidata and DBpedia. Chapter 5 and 6 present the two papers. And chapter 7 concludes the paper by summarizing the result and suggesting further work.

Chapter 2

Theoretical Background

This chapter focus on the underlying concepts for the report. The three themes are: the Semantic Web, the linked open data (LOD) project, and information retrieval.

2.1 The Semantic Web

> The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation
>
> Hendler and Berners-Lee (2010)

The goal of the semantic web, also known as Web 3.0 Shannon (2016), is to enhance the usability of the web by giving explicit meaning to data both in the real world and online. Linking data enables useful interaction with data for both humans and machines (Bizer et al. (2009)).

Example 1. *"Barrack Obama is the president of the US"*

Example 1 is understandable for a human being who is familiar with the concept of president and the two entities Barrack Obama and US. For a machine, this sentence is only 0s and 1s; therefore, the meaning is lost. The semantic web is a framework of making the semantic (meaning) of data understandable to a machine. By using the framework, we prepare the content, so the machine can perform complex tasks like location based information and question answering. Yahya et al. (2016). Figure 2.1 shows the structure of the semantic web. The next three sections describes four of these building blocks: RDF, how we represent data, RDFS and OWL, how we reason with data, and SPARQL, how we query the data.

2.1.1 RDF Resource Description Framework

RDF is the layer for representing information about anything, be it a real person or a web page. Written on top of XML, RDF uses Uniform Resource Identifiers (URIs) to

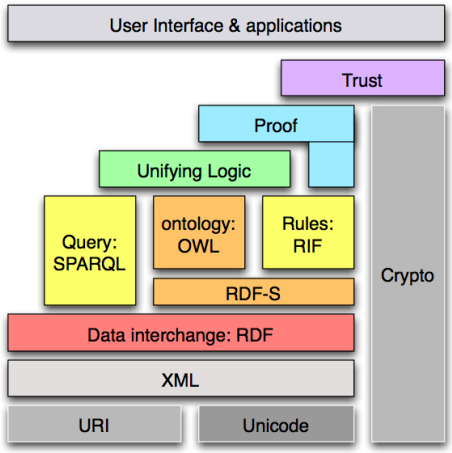

Figure 2.1: The semantic web stack. Figure source: `http://www.thefigtrees.net/lee/blog/2006/11/semantic_web_technologies_in_t.html`

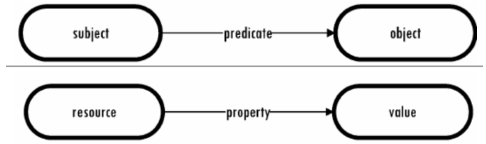

Figure 2.2: A statement or triple in RDF

disambiguate entities, meaning we know exactly which entity we are describing. Figure 2.3 shows example 1 as a RDF statement. Statements represent information and are split into three parts: (as seen in figure 2.2)

1. Subject/resource. Must be a full URI. For any property we label the set of subjects as the domain.
2. Predicate/property. Must be a full URI
3. Object/Value. Can be a full URI or literal (String, integer, date, etc.) For any property we label the set of objects as the range.

The bottom row of figure 2.3 shows a statement found in Wikidata, note how each component is a URI, another type of statements use a literal value instead of a URI in the range, height of a person is an example statement with a decimal literal value. URIs are not expected to return any content, different from an URL, it is merely an identifier; however, this is an desired attribute (see 2.2). Anyone can create URIs, to improve the organization of URIs, we use vocabularies, they contain common URIs (e.g. wd and wdt in fig. 2.3). URIs should be shared and reused (see Yu, 2014, Chap.2 p 34) because without common URIs data will be isolated instead of interlinked. There is a way of combining URIs that point to the same entity (owl:sameAs) but this should be avoided whenever

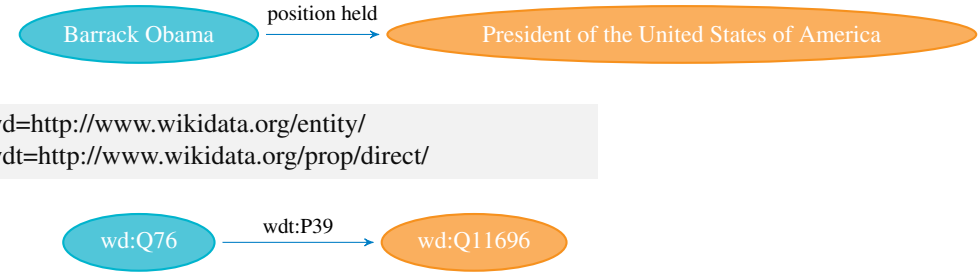

Figure 2.3: Example 1 as a RDF statement. Prefixes in grey. Exact statement at bottom, URIs switched with labels on top

possible to make it easier to reuse data. A collection of statements, also called triples, can be represented in a graph (e.g. fig 2.3) or in a file format: RDF/XML, is the standard, other types are turtle and n3.

2.1.2 RDF Schema - RDFS and The Web Ontology Language - OWL

The examples so far show instance data, also know as ABox statements Sazonau et al. (2015). An ontology use ground truths, also know as TBox statements to describe what information can be represented for a domain, e.g. which classes exist and which properties exist. Schema modeling is the process for building an ontology. RDFS[1] allows schema modeling (TBox statements) by expanding the range of keywords. Examples are: rdfs:subClassOf states that a class has a more general parent class(e.g. car-vehicle), rdfs:subPropertyOf states that a more general property exist (e.g. official model model), rdfs:domain states the subjects of a property must be one of a set of classes, rdfs:range states the same for objects of the property. The vocabulary also includes two important properties for describing instances: rdf:type states which class an instance is (e.g. human) and rdfs:label states a human readable name of the resource (e.g. wd:Q20 rdfs:label "Norway").
OWL provides more TBox statements and allows for inference of classes and properties. E.g. all persons with a mother of or father of statement are parents. These two vocabularies form the basis for building class- and property-hierarchies in the ontology.

2.1.3 SPARQL Standard Protocol and RDF query language

SPARQL fills the need for a language to effectively query RDF data. Most open datasets are accessed by using a SPARQL endpoint, a URL where any agent can use SPARQL queries. Queries are written in turtle[2] and contains a set of types, operands, and functions, for a full specification see [3]. Furthermore, a query has a list of statement patterns that are matched, each pattern is composed of unknown variables, prefixed with a "?", and/or known variables which use a prefix (wdt, wd in 2.3) or a full URI (e.g.`http:`

[1] `https://www.w3.org/TR/rdf-schema/`
[2] `https://www.w3.org/TR/turtle/`
[3] `https://www.w3.org/TR/sparql11-overview/`

//dbpedia.org/resource/Ringo_Starr)."?s ?p ?o" is a query which returns all statements in the graph.

2.2 Linked open data (LOD)

As aforementioned, reusing URIs is important, it connects datasets and uniquely identifies entities. Resolvable URIs returning RDF data and accessible dataset are required for a dataset to be part of the LOD cloud. Currently, about 1254[4] datasets exist in the LOD cloud, but despite the size the LOD cloud has not matured yet Schmachtenberg et al. (2014). Issues in most datasets: link to few others, few global vocabularies exist and meta data describing the dataset are missing. In this work, we label big RDF datasets as knowledge graphs (KG).

2.2.1 Knowledge Graph

Knowledge Graph is a recently new term introduced by Google in 2012 Paulheim (2017). It is used as a label for a dataset with a large amount of diverse (not restricted to a specific domain) instance data described by an ontology Färber et al. (2015). DBpedia[5], Wikidata[6], Yago[7], and Google Knowledge Graph[8] are examples of knowledge graphs where Google Knowledge Graph is the only one not publicly accessible and therefore not part of the LOD cloud.

2.3 Information Retrieval

Information retrieval concerns finding objects from a data source relevant to a query, the objects found are then labeled as relevant or irrelevant. If the data source is a KG the objects are most likely URIs and if the data source is a news paper the objects are most news articles.

2.3.1 Vector space model

Vector space model is often used to represent the retrievable objects. Generally, a vector space model (see Schütze, 2008, Chap. 6.3 p 120) use a transformation to map input into a vector space. A text document can be represented by using the english dictionary as a transformation where we label each unique word in the dictionary with a number. By creating a vector with a dimension equal to the dictionary size, and setting all terms found in a text document to one, we have a vector representation.

[4] https://datahub.io/dataset?tags=lod
[5] http://wiki.dbpedia.org/about
[6] https://www.wikidata.org/wiki/Wikidata:Main_Page
[7] http://yago-knowledge.org/
[8] https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html

2.3.2 TF-IDF

TF-IDF is a measure to determine how important a term (word) in a document is. TF stands for term frequency and is the count of the term in a document. IDF is the inverse document frequency, how many documents do not contain a term, measuring how rare a term is (see Schütze, 2008, Chap. 6.2,2 p 119). The TF-IDF approach prioritizes rare and frequent terms for representing documents.

2.3.3 Precision and recall

Precision and recall measure how well the algorithm did at finding relevant objects to the query. Precision is defined as the proportion of retrieved objects that were relevant, and recall is defined as the proportion of all the relevant objects that were retrieved (see Schütze, 2008, Chap. 1 p 42). E.g. An artist have 10 other similar labeled artists, we name them the expected similars. Suppose an algorithm for finding similar entities retrieves 20 entities and 5 of these are among the expected similars, then precision= $\frac{5}{20}$= 25% and recall = $\frac{5}{10}$ = 50 %

Chapter 3

Related work

The four main challenges discussed are: Understanding the information structure, extracting features, ranking features, and presenting the knowledge to a person.

This chapter gives an overview of related work areas within the Semantic Web that are most relevant to this thesis. As this section aims to give an overview of related work and give the reader a starting point for finding literature, it is less detailed than the related work sections in the two papers. It groups important literature into the four challenges presented in the introduction.

1. As a starting point for understanding the semantic web it is worth reading Bizer et al. (2009) except chapter 5 which is outdated. Then the comparison between the four knowledge graphs Wikidata, DBpedia, Freebase, OpenCyc and YAGO Färber et al. (2015) gives a good introduction to aspects of a knowledge graph as well as differences between the KGs. Finally, the next chapter and its references should be valuable if Wikidata and DBpedia are inteded for usage.

2. For extracting features chapter 7 on transformation in Ristoski and Paulheim (2016) is a good starting point. This study is a comprehensive survey which looks at how KGs are being used. If the purpose is only to add data to an existing dataset then Paulheim and Fümkranz (2012) can be a lightweight alternative. Another point of interest is from the interlinking dataset area. This area contains methods for detecting similarities between datasets, ontologies and entities which can enable aggregation of features. It should also be noted that named entity recognition is often necessary to use a KG. However, this is not the focus of this thesis and third party tools exist for the task, e.g. DBpedia Spotlight Daiber et al. (2013) and Entityclassifier Dojchinovski and Kliegr (2013) .

3. Roa-Valverde and Sicilia (2014) gives a great overview of different aspects of ranking statements in addition to the existing state of the art ranking methods. Inspiration can also be gathered from the area of relationship queries (e.g. Yahya et al. (2016) which use some method for prioritizing statements connecting two or more entities.

4. The challenge of presenting the knowledge to a person is most studied in the area of exploratory search. Exploratory search (also know as Discovery search) focus on giving the user a tool for exploring a knowledge graph from one or more entities. Using Ringo Starr as the input entity the tool might link to his songs, albums, related artists, bands, etc. Two examples of such tools are: Discovery Hub Marie et al. (2013) (figure 3.1) and Google Search (figure 3.2, right square). Marie and Gandon (2014) provides a great overview of exploratory search task and existing approaches.

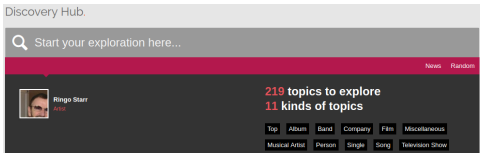

Figure 3.1: Discovery Hub interface

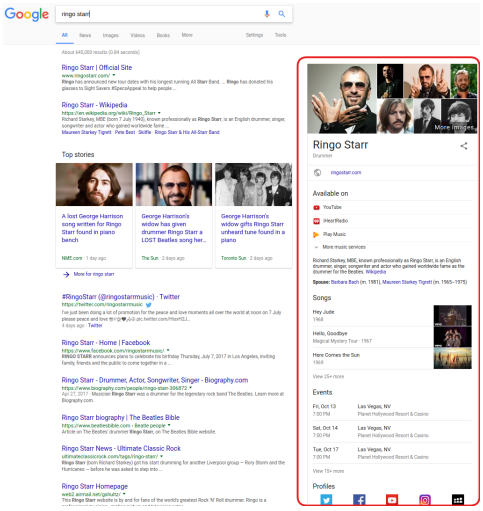

Figure 3.2: Google search interface

Chapter 4

How Wikidata and DBpedia are organized and how to access them

This chapter describes the two knowledge graphs: DBpedia and Wikidata. The goal of the description is to give the reader an understanding of how the KGs: store statements, organize the type- and property-hierarchy, and are accessible. Their main characteristics are summarized in table 4.1.

Characteristic	Wikidata	DBpedia
Statement count	150M	241M
Class count	38533	754
Property count	2398(12926*)	1396
Type-hierarchy depth	15	7
Type-hierarchy has cycles	Yes	No
All types in type-hierarchy	No	Yes
Deepest child example From top of the hierarchy	Too deep	owl:Thing, Place, PopulatedPlace, Region,AdministrativeRegion, GovernmentalAdministrativeRegion, Province, HistoricalProvince

Table 4.1: Characteristics of Wikidata and DBpedia. M = Million. * count with qualifier properties (see sec: 4.2)

4.1 DBpedia

4.1.1 Background and data-model

DBpedia is the most used KG in the literature. Its: long existence (11 years), big size (more than 1 billion statements), and central function as a hub (connecting many datasets) makes it a great choice Lehmann et al. (2015) . The statements in DBpedia are found by collaboratively created bots. These bots are executable scripts that collect data from Wikipedia. The ontology in DBpedia is used to organize the collected data. It was constructed manually and consist of 754 types and 1396 properties. The DBpedia type-ontology is well organized, the 754 types are distributed on 7 levels where every type is connected to the hierarchy. Only 5 types in the hierarchy have two parents and there is no cycles. The distribution of entities per type are more uniform compared to Wikidata (Wikidata having a greater amount of lower counting types). Statements in DBpedia are on the standard RDF form with subject-predicate-object. Although the statements have provenance information[1], specifying where the data was collected (Wikipedia page and location), their reliability relies on the Wikipedia collaborators.

4.1.2 Accessing DBpedia

There is a public SPARQL access point available at: `http://dbpedia.org/sparql` However, using exclusively the online access point is suboptimal because of the extra round trip time; therefore, storing a local version is advisable. The latest full dump versions can be found at `http://wiki.dbpedia.org/downloads-2016-04` (the version used throughout this thesis) . Alternatively, the scripts can be executed manually [2].

The dump is partitioned into different datasets (see [3] for descriptions). This thesis selected 8 of these datasets (shown in table 4.2). The most noteworthy datasets and their role are: Page links that collects all links between wikipedia pages, Mappingbased Literals and Mappingbased Objects that collect data formated in the tables of wikipedia pages, SKOS categories that categorizes entities, and Interlanguage Links that links to Wikidata ids. Adding these datasets to a local RDF triple store are trivial since all the statements are already on the subject-predicate-object form.

4.2 Wikidata

4.2.1 Background and data-model

Wikidata is a collaboratively edited knowledge-base with about 2398 properties and more than 24 million items. It started in October 2012 with the intention of gathering knowledge from different languages in Wikipedia into one repository, making the data reusable

[1] `http://wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets#h434-17`

[2] `https://github.com/dbpedia/extraction-framework`

[3] `http://wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets`

Dataset name	Statement count
SKOS categories	5.6M
Topical concepts	179.8K
Instance types	5.2M
Article categories	22.6M
Page links	175.9M
Mappingbased Literals	16.9M
Mappingbased Objects	18.3M
Interlanguage Link	34.7M

Table 4.2: Selected DBpedia datasets and their statement counts. K = thousands and M = Million

and readable (Erxleben et al. (2014)). Wikidata can be argued to be more reliable than DBpedia since statements come directly from users and the statements themselves often contain references and qualifiers (described below). The datamodel's two types are items and properties.
An item is a resource with an itemID (Starting with Q), a label, aliases, descriptions in different languages, sitelinks (to wikipedia pages), and claims (see Fig. 4.1 [4]). Claims are RDF statements and can have: references (statement sources), qualifiers (further describe the claim), and a rank. The rank determines the claim display order e.g. the rank determines that the current population of Berlin should be displayed at the top of the result and as the selected value in queries.
Properties are similar to items but differs by having an unique property id starting with P and datatypes instead of sitelinks. Differently from DBpedia, Wikidata do not reuse properties from other vocabularies directly. Instead they have their own equivalent properties: e.g. wd:P31 = rdf:type and wd:P279 = rdfs:subClassOf.
The type-hierarchy in Wikidata has 15 levels and represent 70% of all possible types in the knowledge graph. The type-hierarchy contains cycles, which can complicate searching. The number of entities per type varies extensively, with more than 36000 types having a count less than 100 and 23 types having a count greater than 100 000. The deep hierarchy level and diversity of types provide rich information by being specific and connected. Nevertheless, the great variance in entities per type, types outside of hierarchy, and multiple types per entity make it a challenge to conclude similarity between types. Therefore, it might be necessary to pre-process the types and filter out or combine types if one plans to use the type information.

4.2.2 Accessing Wikidata

There is a public SPARQL access point available at: `https://query.wikidata.org/`. As mentioned above for DBpedia, the online access point has its limitations and a local version is advisable. The complexity of the data model in Wikidata makes representing the data in RDF more complicated. This process called reification has been studied by Hernández et al. (2015). Although their methods captures the data, they all introduce an

[4] `https://www.mediawiki.org/wiki/Wikibase/DataModel/Primer`

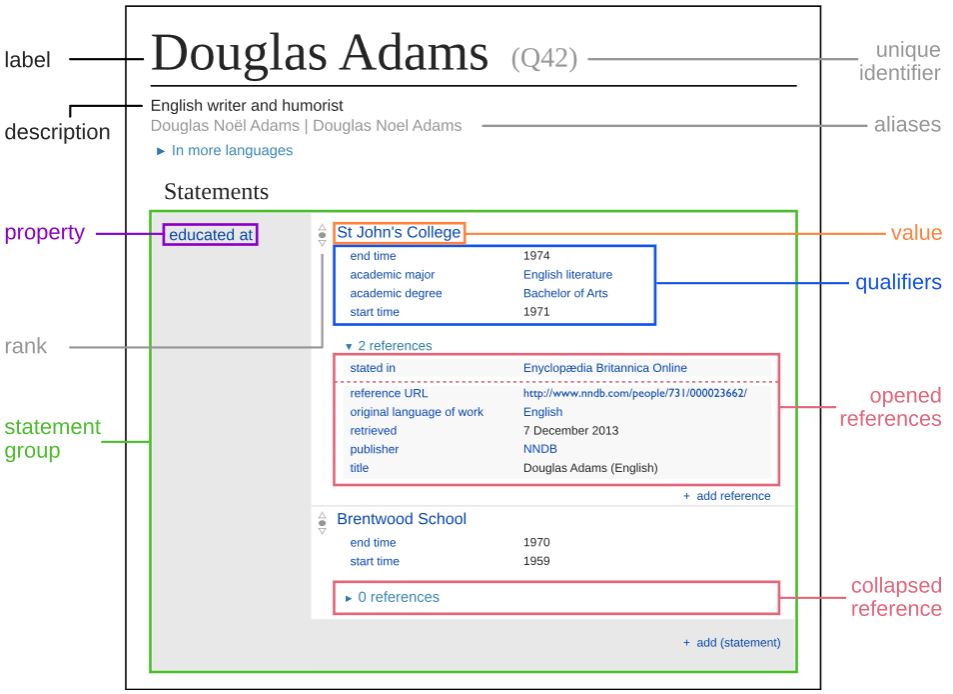

Figure 4.1: Wikidata data model from `https://upload.wikimedia.org/wikipedia/commons/a/ae/Datamodel_in_Wikidata.svg`

intermediate node for qualifier properties that makes the subject not directly connected to the object of the qualifier property. By creating a new property with property-id= concatenating the property-ids and adding a "q", our approach avoids this issue (see Figure 4.2 for details). Therefore, obtaining a local version in the standard RDF format of the Wikidata datset is a bit complicated. A guide for reproducing the dataset can be found here[5].

[5] `https://github.com/EspenAlbert/readKnowledgeGraph`

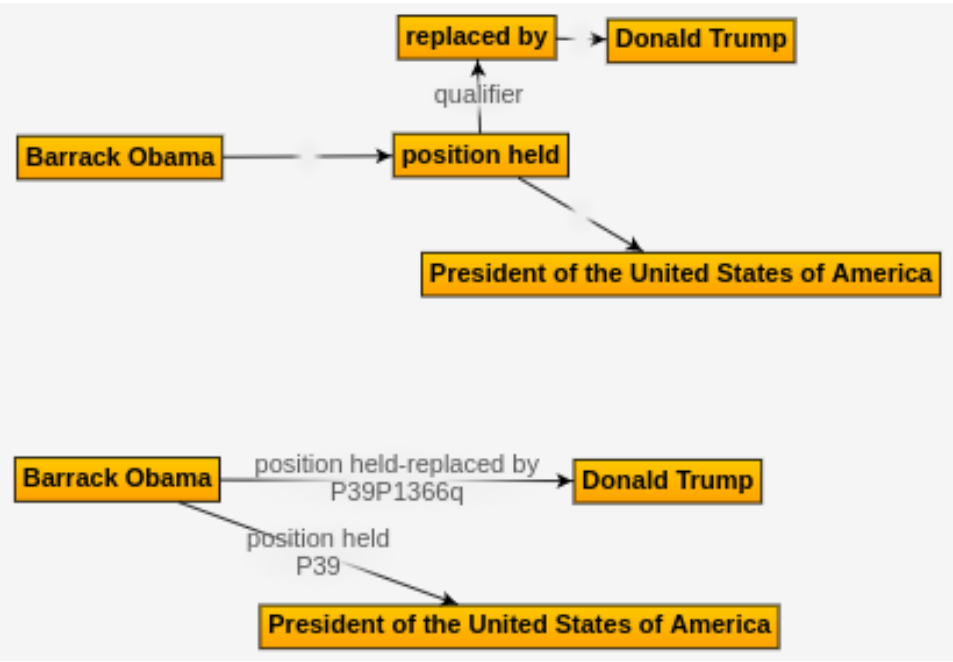

Figure 4.2: Reification of the Wikidata data model from `https://www.wikidata.org/wiki/Q76#P39` (not all facts)

Chapter 5

Paper 1: How to retrieve relevant information from a knowledge graph

How to retrieve relevant information from knowledge graphs using common subsumers: Entity types, similar types, and entity requirements

Espen Albert
Department of Computer and Information Science NTNU
albertespen@gmail.com

Cristina Marco
Department of Computer and Information Science NTNU
cristina.marco@ntnu.no

Jon Atle Gulla
Department of Computer and Information Science NTNU
jag@idi.ntnu.no

ABSTRACT

A challenge when using big knowledge graphs (KGs) such as Wikidata and DBpedia is to select a proper subset of the KG. This paper compares using entity types and property-roles as filters for selecting relevant entities in a KG. Property-roles e.g. performedBy-object (entity is a performer) or genre-subject (entity has a genre) are found by using the most frequent statements for a query entity. The Semantic Artists Similarity dataset contains artists with their top ten most similar artists, by using each artist as a query entity and filtering the KG on the query entity's types, similar types, and property-roles we measure recall (percentage of similar artists found) and precision (percentage of entities in knowledge graph not satisfying the filter). The results show that using property-roles outperforms entity types in Wikidata, recall 99.5% vs 88% and precision 98.4% vs 84.2% (compared to using the query entity's types). Interestingly, there is little difference in using property-roles and similar entity-types in DBpedia: 97.6% vs 98.4% recall and 93.1% vs 88.6% precision, likely because of a more well defined type-hierarchy in DBpedia compared to Wikidata.

1. INTRODUCTION

A knowledge graph (KG) is defined as a large set of statements (subject-property-object, e.g. figure 1) describing and linking entities (also known as resources) in an ontology [7]. Wikidata [6] and DBpedia [2] are two popular KGs with in excess of 19 and 6 million entities where each entity has the role of subject or object in more than 50 statements on average. The big size presents a wealth of information. However, consider the problem of finding similar entities to one or more query entities in the KG. This problem requires selecting and representing relevant entities. The most common solution is to perform a breath first search (BFS) starting from the query entity(s) and traversing the knowledge graph (using statements) adding entities that are connected by paths (chained statements) to the given entity(s) [16][3][9][8]. Although this method are well suited for finding similar entities, there are some challenges: 1. Selecting relevant statements to explore 2. Ranking relevance of a statement to the given entity(s) 3. Knowing when to stop the search 4. Handling multiple entities. The literature contains vast amount of work tackling these challenges. Differently from previous work, this paper isolates the process of finding relevant information by selecting entities based on a common subsumer (CS) [4] instead of performing a BFS. A CS is used to represent the commonalities between entities. By representing each entity as an r-graph [4](a graph starting with the entity as the root. Built by exploring the entity's statements) and replacing statement components (subject, property, object) by blank nodes one can create a CS in polynominal time [4] (see table 2 for examples). The three main advantages of using a CS is: performance (can be computed in polynominal time), testability (can be represented as a SPARQL query), and flexibility (can be combined to fit multiple entities or used together with a search). The contributions of this paper are:

1. Four methods for retrieving relevant information given a single entity. The methods are based on CS, entity types, and rareness of statements
2. Evaluation of the four methods using the Semantic Artists Similarity Dataset on Wikidata and DBpedia. The results demonstrate the methods ability to reduce the size of the knowledge graph while maintaining a high recall of the relevant entities and differences in the type-hierarchy between Wikidata and DBpedia.

The rest of the paper is structured as follows: Section 2 present related work, section 3 presents the methods, section 4 presents the evaluation setup and results. Results and further work are discussed in section 5 while section 6 concludes the paper.

2. RELATED WORK

This section describes related work for selecting or prioritizing relevant information in a knowledge graph. This section is limited to : exploratory search, relationship and similarity finding, and entity type comparison, although more application areas apply. Each approach is described by its solution related to one or more of the four challenges presented in the introduction.

Exploratory search: [8]'s approach finds entities by exploring a fixed number of paths of length 2 starting from the

query node. These paths are selected based on probabilities where higher probability is given to properties with rare edges (low count). Finally, the found entities are clustered based on their properties. [10] starts at the query entity as the previous approach but explores paths of various length both where the entity is subject or object. They use the entity's type to filter the search and a spreading activation algorithm. The algorithm prioritize entities sharing a high degree of statements with the query entity and entities where the neighbors also are similar to the query entity.
Relationship and similarity finding: [1] use the KG to rank relationships (paths) between entities. Their approach use the path's entity types to check for coherence with the schema (expected type in domain/range) and the path's properties rareness (low count) to rank the paths. [14] ranks relationships by finding paths connecting entities with a length less than 5. Each path are ranked based on three factors: property rareness (low count globally), property diversity, and entity proportion of statements with property (e.g. two paths going through the same entity, the path using the most common property for that entity will be ranked higher). [12] measures semantic distance, a measure between two entities calculated by finding paths of length one: either two entities are directly connected (one is subject and the other is object) or the entities share a property and value (subject or object). [13] finds document similarity by comparing entities in the documents. Their approach explores paths of length 3 where the entity has the subject role. Entity similarity is based on the number of paths connecting two entities and the length of these paths. They also use the hierarchy in the KG by giving a higher similarity to entities where their types have a close common ancestor.
Entity type comparison: [15] builds a type-hierarchy across KGs and ranks an entity's type given a context document. Their results show that using the type-hierarchy and prioritizing entity types lower in the hierarchy works better than prioritizing entity types with many entities. [5] goes into depth on how to compare entity types suggesting that entity types sharing an ancestor lower in the hierarchy with compatible children are similar.
The methods presented in this paper use some of the same assumptions as the papers above: 1. A lower global property count for a statement makes it more relevant 2. An entity with a high local count for a property makes the property relevant 3. An entity's type and its position in the type-hierarchy can be used as a filter for relevant entities.
Of the approaches using entity types it has been difficult to see the impact of filtering or finding relevant information based on entity types since they are used in conjunction with some other method or only as a mean to determine the best entity type. By isolating the approaches using entity types we assess the effectiveness of filtering based on entity type. Differently from the entity requirement method (section 3), the mentioned approaches are unable to find relevant information from statements where the object is a literal (e.g. string, integer, date,etc.) since they all use paths where the object is an entity to find entities.

3. METHOD

This section describes four methods for retrieving relevant information from a knowledge graph based on creating Common Subsumers (CS). The CSs presented in this section are distinguished from the original paper [4] in three ways:

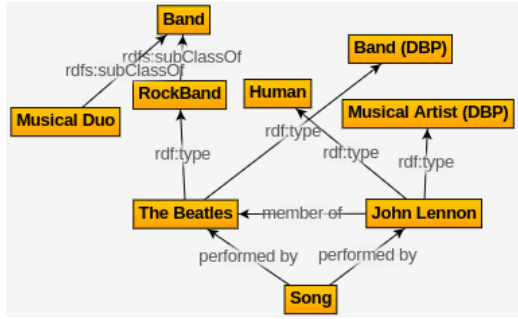

Figure 1: Example RDF graph, Band (DBP) and Muscial Artist (DBP) is the types used in DBpedia. Muscial Duo, Band, RockBand, and Human are the types used in Wikidata. Note, the example is only for illustration and do not accurately represent the KGs

Word	Example
Statement1: Subject-property-object	John Lennon-Member Of-The Beatles
Subject-role	John Lennon (Statement1)
Object-role	The Beatles (Statement1)
Domain count	rdf:type=4, performedBy=1
Range count	performedBy=2
Property-role count for entity	rdf:type-subject=2 (John Lennon)
Property-role count for entity	performedBy-object=1 (John Lennon)
Ancestor	Ancestor(RockBand)=Band
Type-hierarchy	Band, Musical Duo, RockBand, Human (Wikidata)
Type-property-distribution	MemberOf-subject, performedBy-object, and rdf:type-subject (Human)

Table 1: Vocabulary with examples from Figure 1

1. The CS is created from a single enitity instead of multiple entities 2. The CS can include statements where the entity has the object role in addition to the subject role. 3. The CS can contain optional statements.

All methods receive an entity as input and produce a CS, the methods are shown in table 2 with an example of the CSs created for John Lennon.

Baseline. Select entities based on entity type. Identify all the types of the entity (using the rdf:type or an equivalent property). Then select entities having at least one of the entity types.

Hierarchy search. This method assumes that types within a short distance in the type-hierarchy are similar. It is inspired by previous work using the type hierarchy to find a common ancestor [13] [15] [5]. It performs a search of length D starting at the query entity's types and follows the rdfs:subClassOf property (or an equivalent property) in ei-

Method	CS John Lennon Each line wrapped in {} UNION between lines
Baseline	?entity a dbo:MusicalArtist
HierarchySearch D=1	?entity a dbo:MusicalArtist (same prefix as above)Band, Artist,ClassicalMusicArtist,BackScene, MusicDirector,Instrumentalist, Singer
Types matching Property-Roles K=10, T=6	?entity wd:instanceOf** wd:Twin* (same prefix as above)Human, Band,Duo,RockBand,Fictional Human
Entity Requirements K=6	?s wd:performedBy ?entity ?s wd:lyricsBy ?entity ?s wd:composer ?entity ?s wd:castMember ?entity ?entity wd:instrument ?o ?entity wd:occupation ?o

Table 2: The four methods with example SPARQL queries selecting distinct ?entity for John Lennon. The two first methods for DBpedia the last two methods for Wikidata. *Labels are used instead of IDs (QXXX) for types in Wikidata, **instanceOf is equivalent of rdf:type in Wikidata.

ther direction (up or down in the hierarchy) to find similar types. This implies: the distance between two siblings (e.g. RockBand and MusicalDuo figure 1) is 2 and the distance between a child and parent (e.g. RockBand and Band figure 1) is 1. The method require no pre-processing, but it depends on adjusting the distance D of the search which are influenced by the type-hierarchy of the KG. KGs with a deeper hierarchy might require a longer distance, conversely, KGs with multiple parent types might require a shorter distance.

Types matching property-roles Assumes two types are similar based on sharing property-roles. E.g. Humans and RockBands (fig: 1) have the performedBy-object role; therefore, humans and bands are similar. Given a single entity the method finds the top K property-roles. These are selected by prioritizing property-roles with a high count for the query entity. Then T similar types matching the property-roles are determined by giving higher weight to property roles with fewer type matches. Increasing T leads to more information being retrieved while increasing K will impact which types are selected. In order to find type having the property-roles the method performs a pre-processing step storing a type-property distribution for each type in the knowledge graph. A type-property distribution contains property-roles if entities of the type with the property-role exist. The complexity of finding the type-property distributions is $O(|Types| * |Properties| * 2)$. In reality, this memory complexity is much less since most types are not represented for all properties. We decrease the memory complexity by excluding distributions for types with less than 100 entities. We also reduce the risk of noise by requiring each type to have more than 100 or 0.01% of its entities with the property-role.

Entity requirements. Like the previous method, entity requirements selects the top K property-roles of the query entity. However, it filters property-roles with a count greater than a threshold PT. The method finds entities satisfying at least one of the property-roles. A greater K or PT increases the number of entities retrieved. Note that requiring similar entities to satisfy every property-role would make the number of retrieved entities much less and risks ignoring relevant entities. Knowing property counts is the only required pre-processing step. Its complexity is $O(|Properties| * 2)$ (domain and range count per property).

Method	Configuration
Baseline	No configuration
HierarchySearch	DBp: D=2, WD: D=3
Types matching Property-Roles	DBp & WD: K=10, T=10
Entity Requirements	DBp & WD: K=10 PT=250k

Table 3: The four methods with the best configuration parameters for the evaluation

4. EVALUATION AND RESULTS

This section describes the experiment and presents the results of using the Semantic Artist Similarity dataset [11] to test the methods from the previous chapter on Wikidata and DBpedia. The dataset contains 2363 artists where each artist has a list of the ten most similar artists, we will refer to these ten artists as the expected similars. It was constructed by using the Last.fm API and after mapping the ids to Wikidata and DBpedia there are 2315 artists with on average 9.87 expected similars.
The experiment assess the relevance of an entity by using the following premise: An entity is relevant to an artist if it is one of the expected similars (all other entities are irrelevant). In reality, this is a simplification since there are more than 10 artists similar to an artists. But using this simplification the performance is measured in two ways:

1. Recall, percentage of relevant entities satisfying the common subsumer (CS)
2. Precision/compression factor, percentage of entities in the knowledge graph not satisfying the CS.

The experiment is performed on Wikidata and DBpedia (the detailed setup can be found on github[1]) with the configuration of each method shown in table 3. This configuration was found by picking the best performing method from the set of parameter options: D=[1,2,3], K=[1,3,5,10,15,20], T=[5,10,15,20,30], PT=[100k,250k,500k]

Recall levels is shown in figure 2 and precision is shown in figure 3.

5. DISCUSSION OF RESULTS

As shown by the low recall (<88%) in figure 2 selecting only the type based on query entity (baseline) is not enough to retrieve all relevant artists . (Human,Band) and (Band, MusicalArtist) are the most frequent non-recalled type pair

[1] `https://github.com/EspenAlbert/SimilarEntitiesWikidata`

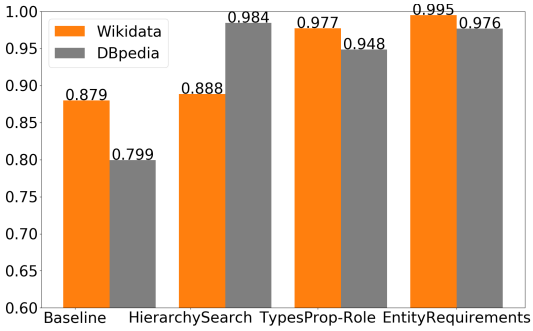

Figure 2: Recall levels

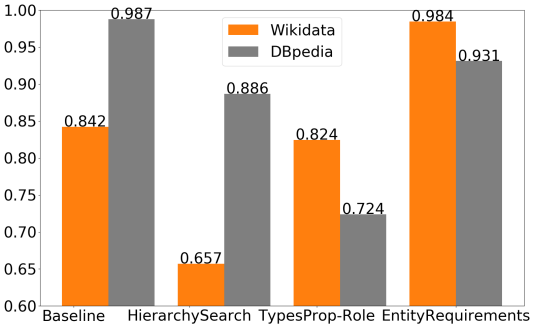

Figure 3: Precision

(query entity type, expected similar type) in Wikidata and DBpedia. The hierarchy search method increases the recall in DBpedia by 18.5 %, by finding all the types within a distance of 2 the method is the best at recalling entities in DBpedia. Conversely, in Wikidata it is increasing recall with less than 1% with the cost of a 19% decrease in precision. This is probably due to the quality of the type-hierarchy: Wikidata has more than 50 times as many types as DBpedia (38533 vs. 754) and the type-hierarchy is more than twice as deep (15 vs. 7 levels). Human is the largest entity-type in Wikidata (3 372 432 entities), while DBpedia have more balanced entity-types (Person entity type has entity count = 502 661). This helps explain why the baseline has the highest precision in DBpedia.

The types selected by using property-roles do not suffer the same way as the hierarchy search and seem to imply that properties is a better way of selecting comparable types in Wikidata, whereas the opposite is true in DBpedia. Finally, using entity requirements proves to work well on both KGs achieving the best precision and recall in wikidata and the 2nd best recall and precision in DBpedia. However, the sparsity of the entity seem to have a great influence on the performance. Of the entities not recalled the average minimum non-common statements count (domain count less than 250 000) of the pair query entity and expected similar was 9.3 in wikidata (average for dataset is 23.7) and 3.1 in DBpedia (average for dataset is 10.7).

Two weaknesses of the proposed approaches are: First, the methods works only for a single KG, having seen the difference in result, the performance could increase by using multiple KGs. Secondly, the dataset do not show the performance on entities with a high degree (many statements). Using the object-role on such entities might cause the selection of the property-roles to be slow. For further work it would be interesting to see the benefit of selecting property-roles based on a machine learning approach such as a decision tree or a support vector machine. Furthermore, testing with a dataset having multiple entities as input would be interesting since CSs are easily combined. Moreover, selecting different patterns such as property and value for the CSs could also increase the performance.

6. CONCLUSION

Freely available knowledge graphs continue to expand, but making use of such KGs is a challenge due to the big size. In this work, we proposed four methods that receives an entity as input and constructs a common subsumer which are used for retrieving relevant information in the KG. These methods can be applied in conjunction with a search or standalone to select data based on query entity(s). The methods were evaluated using the Semantic Artist Similarity dataset on two popular knowledge graphs Wikidata and DBpedia. The result showed that selecting entities based on entity types alone was not enough to retrieve all relevant information. Furthermore, using the type-hierarchy works only if the KG has a well defined type-hierarchy. Moreover, using an entity's property-roles worked well for finding relevant types as well as for finding relevant entities directly, proved by retrieving 97.6% and 99.5% of the relevant entities while filtering 98.4% and 93.1% of the entities in DBpedia and Wikidata.

7. REFERENCES

[1] K. Anyanwu, A. Maduko, and A. Sheth. Semrank: ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on World Wide Web*, pages 117–127. ACM, 2005.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

[3] G. Cheng, Y. Zhang, and Y. Qu. Explass: exploring associations between entities via top-k ontological patterns and facets. In *International Semantic Web Conference*, pages 422–437. Springer, 2014.

[4] S. Colucci, F. Donini, S. Giannini, and E. Di Sciascio. Defining and computing least common subsumers in rdf. *Web Semantics: Science, Services and Agents on the World Wide Web*, 39:62–80, 2016.

[5] C. dâĂŹAmato, S. Staab, and N. Fanizzi. On the influence of description logics ontologies on conceptual similarity. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 48–63. Springer, 2008.

[6] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić. Introducing wikidata to the linked data web. In *International Semantic Web Conference*,

pages 50–65. Springer, 2014.

[7] M. Färber, B. Ell, C. Menne, and A. Rettinger. A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web Journal, July*, 2015.

[8] S. Lam, C. Hayes, N. Deri, and I. Park. Using the structure of dbpedia for exploratory search. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, 2013.

[9] N. Marie, O. Corby, F. Gandon, and M. Ribière. Composite interests' exploration thanks to on-the-fly linked data spreading activation. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 31–40. ACM, 2013.

[10] N. Marie, O. Corby, F. Gandon, and M. Ribière. Composite interests' exploration thanks to on-the-fly linked data spreading activation. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 31–40. ACM, 2013.

[11] S. Oramas, M. Sordo, L. Espinosa-Anke, and X. Serra. A semantic-based approach for artist similarity. In *16th International Society for Music Information Retrieval Conference*, pages 100–106, Málaga, Spain, 26/10/2015 2015.

[12] A. Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *AAAI spring symposium: linked data meets artificial intelligence*, volume 77, page 123, 2010.

[13] C. Paul, A. Rettinger, A. Mogadala, C. A. Knoblock, and P. Szekely. Efficient graph-based document similarity. In *International Semantic Web Conference*, pages 334–349. Springer, 2016.

[14] G. Pirrò. Explaining and suggesting relatedness in knowledge graphs. In *International Semantic Web Conference*, pages 622–639. Springer, 2015.

[15] A. Tonon, M. Catasta, G. Demartini, P. Cudré-Mauroux, and K. Aberer. Trank: Ranking entity types using the web of data. In *International Semantic Web Conference*, pages 640–656. Springer, 2013.

[16] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. Weikum. Relationship queries on extended knowledge graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 605–614. ACM, 2016.

Chapter 6

Paper 2: Feature generation methods for Knowledge Graphs based on Common Subsumers and counts

Feature generation methods for Knowledge Graphs based on Common Subsumers and counts

ABSTRACT

Having millions of statements, knowledge graphs (KGs) are good sources for seeking information about similar entities. However, their big size makes it challenging to efficiently gather relevant information for similarity computation. A breath first search can be too slow and inadequate to find all the similar entities.

This paper proposes five methods for feature generation that use two factors: rareness and context. Rareness is determined by the global feature count where a lower count makes the features faster and more specific. Context refers to the statements of the query entity which are used to create features. To test the feature generation techniques we used the Semantic Artists Similarity Dataset on both Wikidata and DBpedia. The result shows that most features are created within 1 second and that the recall is 15- and 7 % higher than a breadth first search of length 2 for Wikidata and DBpedia.

1. INTRODUCTION

The semantic web and its many knowledge graphs (KGs) describe millions of entities. Consider using a KG exclusively to find similar entities. There are primary three steps: Finding RDF statements, creating features from the statements, and ranking these features. Creating these features are challenging and much effort is put into creating features [11, 4] and ranking features [12].

Differently from previous work this paper seeks to create features that are independently testable without any information source other than a knowledge graph. The assumption that similar entities share the same properties enables the features to be generated without performing a deep search such as a breath first search (BFS) which is a solution widely adopted in the literature [5, 1, 8, 13].

This paper presents 5 feature generation methods that generate features from a query entity. Finding the statements is done by finding all statements where the entity is either subject or object (e.g. S2 and S1 in figure 1). By generalizing the statements: replacing the subject, property, or object the features are created (table 1). Finally, by using the feature counts (table 2) one can rank the features. The contributions of this paper is:

5 feature generation strategies for a KG tested indepedently and compared to a BFS using the Semantic Artists Similarity dataset [7] for speed, recall, and information content on the two knowledge graphs Wikidata and DBpedia. The result show great performance for recall (achieving nearly 100%) and query time (most features below 1s).

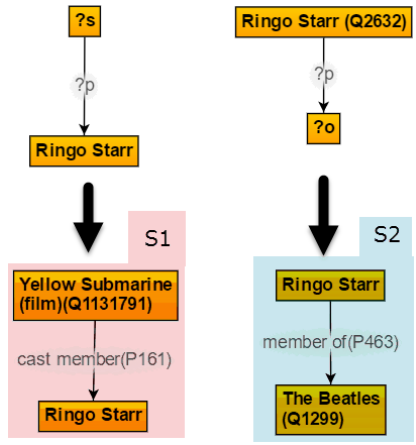

Figure 1: How statements are found from the query entity

The rest of the paper is structured as follows: Section 2 describes common subsumers and related work for creating features from a knowledge graph. Section 3 describe the feature generation methods in detail. Section 4 present the setup of the two experiments, dataset, and the results. Section 5 discuss the results while section 6 concludes the paper

2. RELATED WORK

This section relates the feature generation methods to the work with common subsumers and presents existing literature which use similar or identical features.

A common subsumer (CS) is a recent method for finding commonalities between entities which can be computed in

Feature	Representation (S1 & S2)
Statement-Match	Q1131791 P161 Q2632 Q2632 P463 Q1299
ValueProperty-Match	Q1131791 P161 ?o ?s P463 Q1299
Value-Match	Q1131791 ?p ?o ?s ?p Q1299
Type-Match	[a Film] P161 ?o ?s P463 [a RockBand]
Property-RoleMatch	?any P161 ?o ?s P463 ?any

Table 1: Representations of S1 & S2 for the five feature methods . The id variables (QXX, PXXX) in the representations should be prefixed with wd: `http://www.wikidata.org/entity/`

Name	Count S1	Count S2
StatementMatch	2	2
ValuePropertyMatch	10	5
ValueMatch	36	360
TypeMatch	97058	809
PropertyRoleMatch	113258	134277

Table 2: The number of entities satisfying the five features created from S1 & S2.

polynominal time [3]. It consists of RDF paths which are RDF statements chained together where each subject, property or object can be a blank node. Therefore, the features in this paper can be seen as RDF paths of length 1. However, the feature generation methods differ in three ways: First, creating features from a single entity instead of two entities. Secondly, creating multiple features per statement. Thirdly, finding entities from each feature instead of using the whole CS. (1) allows more flexible input, (2) is used to find as many relationships as possible, and (3) is used to rank each feature.

A challenge when creating features is to handle entities with many connections. One solution limits the generation of features to statements where the entity has the subject role [5, 1, 8]. Another solution use stop URIs to filter such entities [4, 13] .

Finding statements to create features from often involve using counts. [5, 9, 13, 2] prefer statements with a lower global count when selecting and ranking statements. Conversely, most prefer statements with a higher local count.

[1, 8, 6] use the hierarchy structure of a knowledge graph to compute relevance based on closeness in the hierarchy and/or the schema cohesion. This implies prioritizing entities with a short distance in the hierarchy and prioritizing entities based on the expectancy of a certain entity type for a property. Also, most of the work above use the distance from the query entity to rank statements (preferring a shorter distance)

Gathering information from literal values is rarely utilized, except in the area of interlinking datasets. [15] offers a set of methods for different datatypes but require a user to specify the applicability. [14, 10] interlinks data by using string edit distance, the first paper also use numeric values.

The feature generation methods in this papers differs mainly from previous work in three ways: use a single entity to create features, do not perform a nested search, and require no human intervention.

3. METHOD

This section describes the feature generation process and how it can be used to find similar entities. The feature generation process consists of three major stages: Finding statements from a query entity (fig. 1) , creating features from the statements (tbl. 1), and ranking the features.

Selecting statements from an entity is straightforward: use the query entity as the subject and object and find all property-value pairs (fig. 1). S2 is an example statement where Ringo Starr is the subject and the property-value pair is member of-The Beatles. S1 exemplifies the second case where Ringo Starr is the object and the property-value pair is cast member-Yellow Submarine.

StatementMatch finds all entities directly connected to the query entity by using the statement as it is. The rest of the methods replace some part of the statement with a blank node. ValuePropertyMatch replaces the query entity role. It finds all entities connected to the same value by the same property. For S2 (fig. 1) the feature will find other members of the Beatles. Conversly, ValueMatch finds all entities somehow connected to the Beatles by replacing both the property and the query entity role (subject).

$$
\begin{aligned}
SharedValueRatio(property) &= \frac{Domaincount(property)}{Rangecount(property)} \\
Domaincount(property) &= unique\ subjects\ of\ property \\
Rangecount(property) &= unique\ objects\ of\ property
\end{aligned}
\tag{1}
$$

TypeMatch finds the type of the property-value and replace the value with its type (e.g. The Beatles is replaced by rock band) creating features for any entity that is connect with the same property to the same type. Again, for S2 (fig. 1) this is all entities who are member of a rock band. However, the feature is only applicable if two conditions are met. First, shared value ratio (eq. 1) is less than 10. Secondly, there is not any single object-value with more than 1000 entities. These two conditions are necessary to avoid using properties where an object-value has many subjects, such as gender and country of citizenship.

Lastly, PropertyRoleMatch replaces both the query entity role and the value of the property-value pair. This creates features for all entities sharing the same role as the query entity for the same property, e.g. entities that are member of something for S2 and entities that are cast member of something for S1 (fig. 1). Table 2 shows the number of entities found for the two example statements S1 and S2. StatementMatch is the most specific and PropertyRoleMatch is the least specific. Finding these counts is trivial since all features are represented as SPARQL statements. Depending on the application needs this can be performed at query time or as a pre-processing step.

Finding similar entities from a query entity can be done by first creating features for all the statements of the query entity and then executing the SPARQL representation of each feature. Optionally, the counts can be used to only execute features with a low count. Aggregating the features for each entity found and weighing the features by a formula based on the count produce a ranking of similar entities.

4. EVALUATION

This section describes the experiments used to test the feature generation strategies from the last chapter and the dataset and knowledge graphs used throughout the experiments.

4.1 Experiments

The first experiment use the feature types to find similar entities to a query entity. The performance of each feature type is measured by recall, query time, and the average number of features per entity found (see below). There are two goals of the experiment: assess the performance of each individual feature type and assess the importance of high counting features. The importance of high counting features are found by varying a threshold count: t, only executing features with a count less than t.

The second experiment combines the features to find similar entities to a query entity. The performance is compared to a breath first search (BFS) of length 2 which has a 5 second query timeout exploring all statements where the entity is subject or object for Wikidata and DBpedia, except for the wikipageWikiLink property (DBpedia) where only the statements where the entity has the subject role are explored. The goal of the evaluation is to analyze the recall compared to the BFS and see the impact of not using statements where the entity is object.

4.2 Dataset

The Semantic Artist Similarity dataset [7] is used throughout the experiments with Wikidata and DBpedia as the KGs. The dataset contains 2363 artists where each artist has a list of the ten most similar artists, we will refer to these ten artists as the expected similars. It was con- structed by using the Last.fm API and after mapping the ids to Wikidata and DBpedia there are 2315 artists with on average 9.87 expected similars. The experiments assess the relevance of an entity by using the following premise: An entity is relevant to an artist if it is one of the expected similars (all other entities are ir- relevant). In reality, this is a simplification since there are more than 10 artists similar to an artists. But using this simplification the performance is measured in three ways:

1. Speed (query time)
2. Recall (proportion of expected similars found)
3. Relevant information (the number of relationships found for expected similars)

The experiment is performed on Wikidata and DBpedia (see [1] for details on their data).

4.3 Results

The recall, query-time, and relationship count per entity of experiment 1 for Wikidata are shown in figure: 3, 4, and 5. Furthermore, the result of experiment 2 on DBpedia are shown in figure: 6, 7, and 8. The recalls from experiment 2 are shown in figure 2.

[1] `https://github.com/EspenAlbert/readKnowledgeGraph`

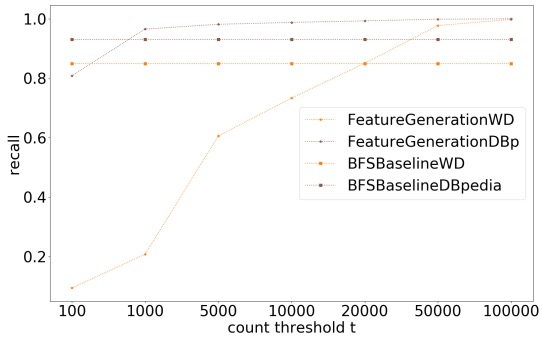

Figure 2: Experiment 2: Recall feature types combined vs. BFS baseline of L=2. (Baselines are independent of threshold t)

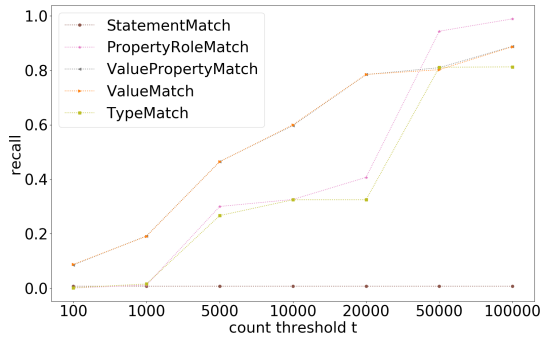

Figure 3: Feature types recall Wikidata

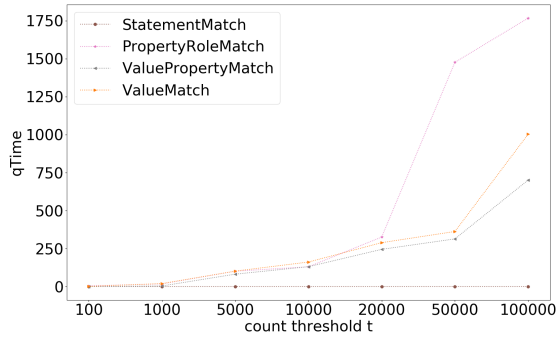

Figure 4: Feature types query time (in ms) for Wikidata. TypeMatch left out since it was significantly slower than the other techniques. (1-140s)

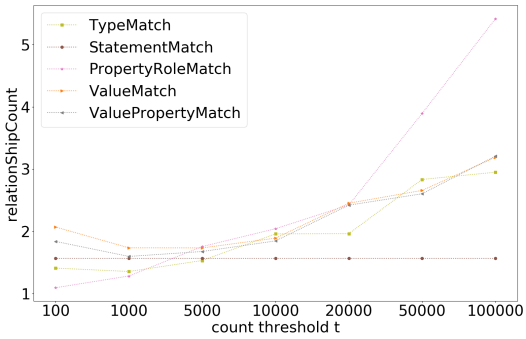

Figure 5: Feature types relationship count Wikidata

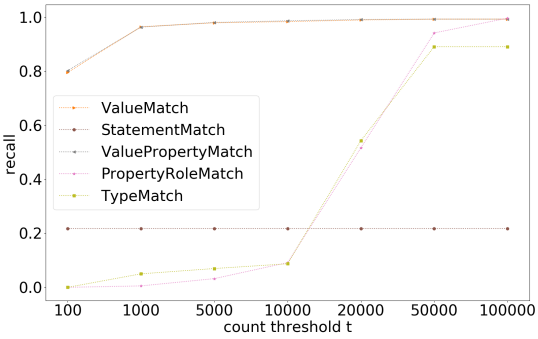

Figure 6: Feature types recall DBpedia

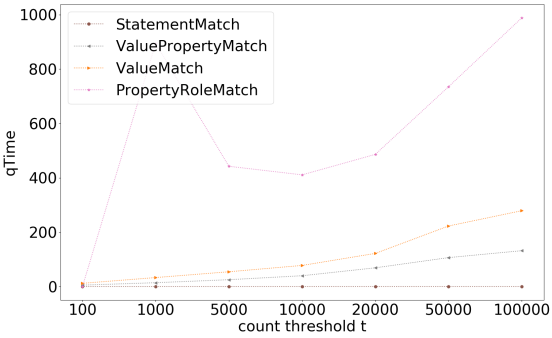

Figure 7: Feature types query time (in ms) for DBpedia. TypeMatch left out since it was significantly slower than the other techniques. (14-325s)

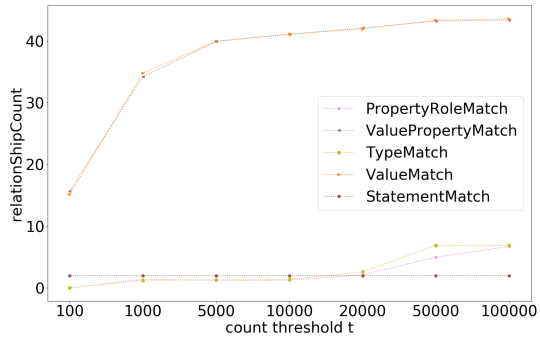

Figure 8: Feature types relationship count DBpedia

5. DISCUSSION OF RESULTS

5.1 Experiment 1

The only feature type independent of the threshold count t is the StatementMatch. Looking at the recall levels it has only features for 0.8% of the entities in Wikidata and 21.7% in DBpedia. This shows that few of the similar entities are directly connected, especially in Wikidata.

The performance of ValueMatch (VM) and ValueMatch-Property (VMP) are almost identical, VMP has slightly lower query time in both KGs. As VMP produce features for all statements where VMs produce features this was expected.

The difference in feature count per entity between the KGs are major. This is due to DBpedia having the "wikiPageWikiLink" property which connects many entities, reflected by the high feature count for VM and VMP with more than 132 features per entity compared to 3 in Wikidata. Conversly, the difference is almost none when looking at the feature count from the PropertyRoleMatch features (6.7 vs. 5.4).

Moreover, the result show that a high feature count increase the query time indepedent of feature type. Finally, the slow execution time of the TypeMatch features shows that it is unsuitable when query time is of importance, using on average more than 2 minutes when features with less than 100,000 entities is allowed. As TypeMatch is the most advanced feature (use a blank node with a type instead of only a blank node) this could indicate that more complex features will be slower.

5.2 Experiment 2

The average query times in Wikidata and DBpedia of 38 and 68 seconds (not shown in figure) shows that using a normal BFS to find similar entities can be slow. Although most of the features except the TypeMatch have a query time less than 1s, it is hard to tell exactly how much they will reduce the query time.

The recall level of 85% and 93% also reveals that the BFS is unable to find all the similar entities. The high recall levels for low counts in DBpedia show a potential for both low query times and a high recall. Differently, Wikidata needs a threshold above 20,000 to beat the recall of the baseline. Note that the high potential for recall improvement in DBpedia is partly due to setup of the experiment: not exploring statements with the wikipageWikiLink property where the entity is object in DBpedia. This indicates that exploring

exclusively properties where the entity is subject risks loosing information.

6. CONCLUSION

Creating features from a knowledge graph can be useful for many applications. However, the feature generation process can be challenging due to the big size of the KG. This paper investigated 5 feature generation methods based on common subsumers [3] and counts. With the task of finding similar artists using the Semantic Artist Similarity dataset [7] and measuring recall, query time, and feature count of the recalled entities each method was assessed independently.

The assessment showed that features created from shared values were the only features able to recall more than 50% (78.5% and 99% in Wikidata and DBpedia) of the entities when each feature had a limit of 20,000 entities. Furthermore, the result showed the importance of the KG and its properties where a single property in DBpedia (wikiPageWikiLink) made a huge impact on the feature count.

The second experiment showed how a normal breath first search is slow and unable to recall all similar entities. Moreover, how selecting only a properties where the entity is subject can be detrimental to performance. Lastly, the exeriment showed the potential of the feature generation methods to both increase recall from 85% and 93% to almost 100% and decrease query time by a factor greater than 10 (3s vs. 38 and 68s) in Wikidata and DBpedia respectively.

7. REFERENCES

[1] K. Anyanwu, A. Maduko, and A. Sheth. Semrank: ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on World Wide Web*, pages 117–127. ACM, 2005.

[2] G. Cheng, Y. Zhang, and Y. Qu. Explass: exploring associations between entities via top-k ontological patterns and facets. In *International Semantic Web Conference*, pages 422–437. Springer, 2014.

[3] S. Colucci, F. Donini, S. Giannini, and E. Di Sciascio. Defining and computing least common subsumers in rdf. *Web Semantics: Science, Services and Agents on the World Wide Web*, 39:62–80, 2016.

[4] I. Hulpus, C. Hayes, M. Karnstedt, and D. Greene. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 465–474. ACM, 2013.

[5] S. Lam, C. Hayes, N. Deri, and I. Park. Using the structure of dbpedia for exploratory search. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, 2013.

[6] N. Marie, O. Corby, F. Gandon, and M. Ribière. Composite interests' exploration thanks to on-the-fly linked data spreading activation. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 31–40. ACM, 2013.

[7] S. Oramas, M. Sordo, L. Espinosa-Anke, and X. Serra. A semantic-based approach for artist similarity. In *16th International Society for Music Information Retrieval Conference*, pages 100–106, Málaga, Spain, 26/10/2015 2015.

[8] C. Paul, A. Rettinger, A. Mogadala, C. A. Knoblock, and P. Szekely. Efficient graph-based document similarity. In *International Semantic Web Conference*, pages 334–349. Springer, 2016.

[9] G. Pirrò. Explaining and suggesting relatedness in knowledge graphs. In *International Semantic Web Conference*, pages 622–639. Springer, 2015.

[10] Y. Raimond, C. Sutton, and M. B. Sandler. Automatic interlinking of music datasets on the semantic web. *LDOW*, 369, 2008.

[11] P. Ristoski and H. Paulheim. Semantic web in data mining and knowledge discovery: A comprehensive survey. *Web semantics: science, services and agents on the World Wide Web*, 36:1–22, 2016.

[12] A. J. Roa-Valverde and M.-A. Sicilia. A survey of approaches for ranking on the web of data. *Inf. Retr.*, 17(4):295–325, Aug. 2014.

[13] M. Schuhmacher and S. P. Ponzetto. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 543–552. ACM, 2014.

[14] B. Spahiu, C. Xie, A. Rula, A. Maurino, and H. Cai. Profiling similarity links in linked open data. In *Data Engineering Workshops (ICDEW), 2016 IEEE 32nd International Conference on*, pages 103–108. IEEE, 2016.

[15] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk-a link discovery framework for the web of data. *LDOW*, 538, 2009.

Chapter 7

Conclusion

This chapter attempts to summarize the main findings in chapter 4, 5, and 6. And suggest directions for further work

7.1 Summary of Contributions

Chapter 4 bridge the graph left by the theoretical background (chpt. 2) by showing how the building blocks of the Semantic Web and the principles of linked open data are applied in DBpedia and Wikidata. DBpedia has a well defined type-hierarchy and its reuse of vocabularies makes it maybe the most important public knowledge graph in the Semantic Web. However, since its data depends on bots, the data provenance is questionable and the user skill for contributing is high.
Differently, Wikidata has a simpler user interface which make contribution easier. Moreover, the data often contain provenance information which makes it reliable. But the reliability has the cost of a more complex data model. Furthermore, Wikidata is at an early stage shown by the immature type-hierarchy which has cycles and more than 33,000 types.

The lack of clear results in the literature whether type or some other information can be used as a filter for relevant information motivates the paper from chapter 5. The paper use a state of the art dataset of similar artists to measure if relevant information can be filtered. The paper finds that using the type itself is insufficient as it risks filtering away similar artists. But filtering on the type and its similar types can work if the types are found with correspondence to the knowledge graphs. Using the type-hierarchy proved to work well in DBpedia (keeping 98.7% of the similar artists while reducing the information by 88.6%). Conversely, this did not work well for Wikidata (88.8- and 65.7%) which had a much better result by using the property-roles of the artist to select types (97.7- and 82.4%).
The innovative method of using property-roles instead of types performed decently in both KGs: Keeping 97.6- and 99.5% of the similar artists while reducing the information by 93.1 and 98.4 % in DBpedia and Wikidata respectively.

The paper in chapter 6 continues the approach from chapter 5 of testing methods exclusively dependent on the KG (no outside features, e.g. social media, search engines, etc.) in isolation. This time the paper seek to answer which feature types are able to create relevant information, how fast are they found, and how much information do they gather. Again, the similar artist dataset is being used.
The findings show that a breath first search of length 2 with a query timeout of 5s can be inadequate of finding paths between similar artists as well as being slow. It finds paths for 93- and 85% of the similar artists in 68- and 38s on average for DBpedia (only paths were entity is subject for the wikipageWikiLink property) and Wikidata. Alternatively, the feature generation methods found by generalizing the statements of the queried artist led to almost 100% of the similar artists sharing features. This seem to imply that generalizing statements based on a query entity are highly applicable for similarity computation.
Moreover, the paper finds that more complex features comes with a high cost in query time (e.g. TypeMatch; creating features by finding entities which has the same type in the value of the statement). The paper also demonstrated how query time increase with the feature count.
Furthermore, the DBpedia property wikipageWikiLink dominated the proportion of features created in DBpedia. It makes finding similar entities from only low counting features a possibility in DBpedia, a possibility not present in Wikidata (96.5- vs. 20.8% of similar artists sharing features with less than 1000 entities).
Moreover, the PropertyRoleMatch features showed that artists in DBpedia and Wikipedia share a similar amount of properties (6.7 and 5.4) with less than 100,000 entities.
Lastly, the features from shared entities (ValuePropertyMatch and ValueMatch) were the prevailing features with low counts (less than 20,000) (99.1% and 78.5% recall), outperforming features from directly linked entities, StatementMatch (21.7% and 0.8% recall).

7.2 Further Work

This thesis has hopefully provided some guidance for using knowledge graphs. The further work proposed in this section attempts to connect the work in this thesis to the research questions defined in section 1.3 and propose further work. The questions are reiterated below to enhance readability.

RQ1: Which features are the most important to compute similarity and/or relatedness between two or more entities in Wikidata and DBpedia?

RQ2: How can the relationship between two or more given entities be described or labeled?

RQ3: How can similar/related entities be found in Wikidata and DBpedia?

RQ4: How can the context from news articles (containing the entity) be used in order to find similar/related entities?

RQ5: How can the informativeness of the properties of the entities in Wikidata and DBpedia be measured?

Continuing the work of chapter 6 to better answer RQ1 and RQ5 it would be valuable to count the features created per dataset. Then comparing these feature counts with the global counts one could find the most important features (RQ1). Furthermore, grouping by the properties of these features would answer help RQ5.
For describing the relationship between entities (RQ2), aggregating the features (found by the method in chpt. 6) shared by two or more entities by feature type might prove valuable in describing their relationship. As an alternative, the types or property-roles found by the methods in chapter 5 can be used.
Improving the method for finding similar and related entities by also incorporating the context of a news article is an exciting area for future work (RQ3 and RQ4). By accessing a dataset of news articles and assuming that the entities in an article are related, one could use machine learning to learn a connection between words in the article and the features shared by the entities. This connection between words and features could then be used to prioritize features that have connecting words present in a query context to find similar or related entities.

Bibliography

Bi, B., Ma, H., Hsu, B.-J. P., Chu, W., Wang, K., Cho, J., 2015. Learning to recommend related entities to search users. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. ACM, pp. 139–148.

Bizer, C., Heath, T., Berners-Lee, T., 2009. Linked data-the story so far. Semantic Services, Interoperability and Web Applications: Emerging Concepts, 205–227.

Cheng, G., Zhang, Y., Qu, Y., 2014. Explass: exploring associations between entities via top-k ontological patterns and facets. In: International Semantic Web Conference. Springer, pp. 422–437.

Daiber, J., Jakob, M., Hokamp, C., Mendes, P. N., 2013. Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics).

Di Noia, T., Mirizzi, R., Ostuni, V. C., Romito, D., Zanker, M., 2012. Linked open data to support content-based recommender systems. In: Proceedings of the 8th International Conference on Semantic Systems. ACM, pp. 1–8.

Dojchinovski, M., Kliegr, T., 2013. Entityclassifier.eu: Real-time classification of entities in text with wikipedia. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. ECMLP-KDD'13. Springer-Verlag, pp. 1–1.

Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., Vrandečić, D., 2014. Introducing wikidata to the linked data web. In: International Semantic Web Conference. Springer, pp. 50–65.

Färber, M., Ell, B., Menne, C., Rettinger, A., 2015. A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web Journal, July.

Hendler, J., Berners-Lee, T., 2010. From the semantic web to social machines: A research challenge for ai on the world wide web. Artificial Intelligence 174 (2), 156–161.

Hernández, D., Hogan, A., Krötzsch, M., 2015. Reifying rdf: What works well with wikidata. In: Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems. Vol. 1457. pp. 32–47.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al., 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6 (2), 167–195.

Marie, N., Corby, O., Gandon, F., Ribière, M., 2013. Composite interests' exploration thanks to on-the-fly linked data spreading activation. In: Proceedings of the 24th ACM Conference on Hypertext and Social Media. ACM, pp. 31–40.

Marie, N., Gandon, F., 2014. Survey of linked data based exploration systems. In: Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data-Volume 1279. CEUR-WS. org, pp. 66–77.

Oramas, S., Sordo, M., Espinosa-Anke, L., Serra, X., 26/10/2015 2015. A semantic-based approach for artist similarity. In: 16th International Society for Music Information Retrieval Conference. Málaga, Spain, pp. 100–106.
URL http://dblp.org/rec/conf/ismir/OramasSAS15

Paulheim, H., 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web 8 (3), 489–508.

Paulheim, H., Fümkranz, J., 2012. Unsupervised generation of data mining features from linked open data. In: Proceedings of the 2nd international conference on web intelligence, mining and semantics. ACM, p. 31.

Pirrò, G., 2015. Explaining and suggesting relatedness in knowledge graphs. In: International Semantic Web Conference. Springer, pp. 622–639.

Ristoski, P., Paulheim, H., 2016. Semantic web in data mining and knowledge discovery: A comprehensive survey. Web semantics: science, services and agents on the World Wide Web 36, 1–22.

Roa-Valverde, A. J., Sicilia, M.-A., Aug. 2014. A survey of approaches for ranking on the web of data. Inf. Retr. 17 (4), 295–325.
URL http://dx.doi.org/10.1007/s10791-014-9240-0

Sazonau, V., Sattler, U., Brown, G., 2015. General terminology induction in owl. In: International Semantic Web Conference. Springer, pp. 533–550.

Schmachtenberg, M., Bizer, C., Paulheim, H., 2014. State of the lod cloud 2014. University of Mannheim, Data and Web Science Group [en ligne] 30.

Schuhmacher, M., Ponzetto, S. P., 2014. Ranking entities in a large semantic network. In: European Semantic Web Conference. Springer, pp. 254–258.

Schütze, H., 2008. Introduction to information retrieval. In: Proceedings of the international communication of association for computing machinery conference.

Shannon, V., 2016. A 'more revolutionary' web.
URL http://www.nytimes.com/2006/05/23/technology/23iht-web.html

Yahya, M., Barbosa, D., Berberich, K., Wang, Q., Weikum, G., 2016. Relationship queries on extended knowledge graphs. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, pp. 605–614.

Yu, L., 2014. A developer's guide to the semantic web, 2nd Edition. Springer.

Zhang, K., Zhu, K., Hwang, S.-w., 2015. An association network for computing semantic relatedness. In: AAAI. pp. 593–600.