Semantics-based Summarization of Entities in Knowledge Graphs

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

Kalpa Gunaratna
B.Sc., University of Colombo, 2008

2017
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

April 19, 2017

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Kalpa Gunaratna ENTITLED Semantics-based Summarization of Entities in Knowledge Graphs BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Amit Sheth, Ph.D.
Dissertation Co-Director

Krishnaprasad Thirunarayan, Ph.D.
Dissertation Co-Director

Michael Raymer, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Robert E.W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School

Committee on Final Examination

Amit Sheth, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Keke Chen, Ph.D.

Gong Cheng, Ph.D.

Edward Curry, Ph.D.

Hamid R. Motahari Nezhad, Ph.D.

ABSTRACT

Gunaratna, Kalpa. PhD, Department of Computer Science and Engineering, Wright State University, 2017. Semantics-based Summarization of Entities in Knowledge Graphs.

The processing of structured and semi-structured content on the Web has been gaining attention with the rapid progress in the Linking Open Data project and the development of commercial knowledge graphs. Knowledge graphs capture domain-specific or encyclopedic knowledge in the form of a data layer and add rich and explicit semantics on top of the data layer to infer additional knowledge. The data layer of a knowledge graph represents entities and their descriptions. The semantic layer on top of the data layer is called the schema (ontology), where relationships of the entity descriptions, their classes, and the hierarchy of the relationships and classes are defined. Today, there exist large knowledge graphs in the research community (e.g., encyclopedic datasets like DBpedia and Yago) and corporate world (e.g., Google knowledge graph) that encapsulate a large amount of knowledge for human and machine consumption. Typically, they consist of millions of entities and billions of facts describing these entities. While it is good to have this much knowledge available on the Web for consumption, it leads to information overload, and hence proper summarization (and presentation) techniques need to be explored.

In this dissertation, we focus on creating both *comprehensive* and *concise* entity summaries at: (i) the single entity level and (ii) the multiple entity level. To summarize a single entity, we propose a novel approach called FACeted Entity Summarization (FACES) that considers importance, which is computed by combining popularity and uniqueness, and diversity of facts getting selected for the summary. We first conceptually group facts using semantic expansion and hierarchical incremental clustering techniques and form facets (i.e., groupings) that go beyond syntactic similarity. Then we rank both the facts and facets using Information Retrieval (IR) ranking techniques to pick the

highest ranked facts from these facets for the summary. The important and unique contribution of this approach is that because of its generation of facets, it adds diversity into entity summaries, making them comprehensive. For creating multiple entity summaries, we propose RElatedness-based Multi-Entity Summarization (REMES) approach that simultaneously processes facts belonging to the given entities using combinatorial optimization techniques. In this process, we maximize diversity and importance of facts within each entity summary and relatedness of facts between the entity summaries. The proposed approach uniquely combines semantic expansion, graph-based relatedness, and combinatorial optimization techniques to generate relatedness-based multi-entity summaries.

Complementing the entity summarization approaches, we introduce a novel approach using light Natural Language Processing (NLP) techniques to enrich knowledge graphs by adding type semantics to literals. This makes datatype properties semantically rich compared to having only implementation types. As a result of the enrichment process, we could use both object and datatype properties in the entity summaries, which improves coverage. Moreover, the added type semantics can be useful in other applications like dataset profiling and data integration. We evaluate the proposed approaches against the state-of-the-art methods and highlight their capabilities for single and multiple entity summarization.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to acknowledge and thank my advisers Prof. Amit Sheth and Prof. Krishnaprasad Thirunarayan. I remember the day I first contacted Prof. Sheth with my ambition to join the Ohio Center of Excellence in Knowledge-enabled Computing (Kno.e.sis) at Wright State University. From the very first day, Prof. Sheth demanded high quality outcomes which led to the successful completion of my PhD. He often reminded me the importance of taking the initiative and participating in activities and professional services that go beyond my focused research (to overcome the unfair advantage of the others) to compete with the best in the world. Organizing SumPre workshop series at ESWC conference, doing high quality internships, participating in hackathons, and serving in program committees are some of them. In fact, I decided to join Kno.e.sis for my graduate studies mainly because of Prof. Sheth's experience and excellence in the field and world class achievements of Kno.e.sis students. Prof. Sheth was patient with me during the early days of my PhD and supported me by providing the necessary resources, a nurturing ecosystem, and with his vast knowledge, experience, and time. With Prof. Sheth's initiation, I started working closely with Prof. Krishnaprasad Thirunarayan (a.k.a Prof. T.K. Prasad) after joining Kno.e.sis who later became my co-adviser. I must say that I thoroughly enjoyed exchanging ideas with Prof. Prasad and am thankful for all the hard work he put into improving my technical and soft skills. Prof. Prasad was always there to discuss research ideas and technical details related to my research problems. I have fond memories of spending hours with him discussing last minute edits when we submitted manuscripts for publication. His guidance in technical writing and algorithmic analysis greatly helped me to develop myself over the years.

I would like to thank Prof. Gong Cheng, Dr. Edward Curry, Dr. Hamid Motahari, and Dr. Olivier Bodenreider for their sincere support. Prof. Gong Cheng is an expert in the area of Semantic

Web and summarization and I learned a lot collaborating with him. He possesses excellent technical and theoretical knowledge and guided me in getting high quality research publications. Dr. Edward Curry was my mentor during my first internship. I enjoyed working with him and I believe he influenced and motivated me in my early days in the PhD program to work hard and to never give up. Dr. Hamid Motahari was my mentor during my last internship and I am thankful to him for giving me the opportunity to work on a real-world enterprise research problem. He helped me to improve my self-confidence in doing research. Dr. Olivier Bodenreider was my mentor during my second internship and I am glad that I had the opportunity to work with him. He taught me how to think about a research problem and gave me the freedom to do research in line with my interests, which helped me to start growing as an independent researcher.

I am grateful for all the love and encouragement from my wife Ronali Gunaratna and daughter Katelyn Gunaratna, my parents Susil Gunaratna and Shirani Gunaratna, my sister Nishadi Fernando and her family, my wife's parents Ravi Perera, Belinda Perera, and brother in-law Ryan Perera, and all the cousins. They helped me overcome many tough situations over the years.

I would also like to acknowledge several influential people who helped, guided, and advised me along this journey towards achieving a PhD. I thank Dr. Gamini Palihawadana for motivating, encouraging, and supporting me in many ways over the years. Special thanks to Prasantha Warnakula, Palitha Perera, Anil Jayawardena, Kithsiri Perera, Rose Gunaratna, Gamini Ratnayake, Rohan Wijesinghe, Amaradasa Kodikara, and Leslie Silva.

I enjoyed exchanging research ideas, life experiences, and jokes with my colleagues at Kno.e.sis over the years. The list is too long to mention each and every one of them. I want to thank Tonya Davis, administrative assistant at Kno.e.sis for the support over the years. My special thanks to Dr. Ajith Ranabahu for helping me out when I first joined Kno.e.sis in many ways, from providing lodging to being a close friend and looking after me. He was also my first mentor in my PhD program. I would also like to mention Dr. Thilina Gunarathne who helped me in finding a good graduate program. I also thank the Sri Lankan community in Dayton Ohio for providing me a home

away from home and Dr. Shiral Fernando, Dr. Anil Fernando, and their families for always being there for us.

This dissertation is based upon work supported by the National Science Foundation (NSF) and National Institute of Health (NIH). NSF provided support through Grant No. 1143717 III: EAGER Expressive Scalable Querying over Linked Open Data, and Grant No. EAR 1520870: Hazard-SEES:Social and Physical Sensing Enabled Decision Support for Disaster Management and Response. NIMH of the NIH provided support under award number R01MH105384-01A1. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or NIH.

Dedicated to

my parents, Susil and Shirani Gunaratna for their love and sacrifices

my lovely wife Ronali Gunaratna for her endless support and our little daughter Katelyn Gunaratna for bringing us endless joy in life!

1

Introduction

Publishing structured machine readable data on the Web for wide-scale use has been gaining attention in the recent past mainly due to the introduction of the Linking Open Data initiative and set of principles called Linked Data Principles [Berners-Lee 2006] that encouraged data publication and re-use. Following these guidelines, people and communities have created the Linked Open Data (LOD) cloud[1]. The LOD cloud consists of datasets that are inter-linked and have Uniform Resource Identifiers (URIs) to represent resources that can be dereferenced. It has grown from 295 datasets in 2011 to 1146 datasets in 2017 (as of January), which is also accessible online. Some of these datasets are encyclopedic knowledge graphs, like DBpedia [Lehmann et al. 2015], and domain specific, like LinkedMDB [Hassanzadeh and Consens 2009] for movies and LinkedGeoData [Auer et al. 2009] for geographic locations. We refer to a dataset as a knowledge graph in this dissertation when its representation is rich with semantics and additional knowledge. Semantics are introduced by adding a rich schema on top of the data level of the dataset. Further, a good schema is important in modeling concepts (classes) and relationships between them, which in turn facilitates making inferences from the data level of the knowledge graphs. Therefore, knowledge graphs can help machines understand and process data intelligently. In addition to the growth in the number of datasets in LOD, the content in some of the datasets has also evolved significantly over time. For example, in 2007 DB-

[1] `http://lod-cloud.net/`, accessed 04/10/2017

pedia English version 2.0 describes 1.95 million entities, whereas in 2016 DBpedia English version 2016-04 describes 6 million entities[2]. Further, the number of facts describing each resource has also increased, providing more information to data consumers.

In the enterprise environment, knowledge graphs are becoming increasingly popular in modeling and representing world knowledge that in turn can make commercial applications more intelligent. The earliest efforts in this direction came with the rise of the Semantic Web technologies. The concept of creating a rich dataset (what we refer to as knowledge graph now) was proposed in the implementation efforts of Taalee system [Sheth et al. 2001]. They built a knowledge graph combining "WorldModel" (definitional component as the ontology) and knowledge base (assertional component as the data layer) to capture knowledge on the Web to support semantic search and faceted information presentation to the users. This was termed as "Rich Media Reference" [Sheth et al. 2002][3]. Since the early inception of these ideas, Google Knowledge Graph [Singhal 2012] and Bing Satori [Qian 2013] have emerged as two of the prominent commercial knowledge graphs to facilitate Google search[4] and Bing search[5] respectively on a massive scale. When Google Knowledge Graph was in the initial stages, it had around 570 million entities and 18 billion facts[6]. Knowledge graph creation is supervised by human experts and involves much manual effort to ensure high quality but recently there have been efforts like Google Knowledge Vault [Dong et al. 2014] that automate this process at commercial scale but its success is doubtful as of today. There are also different kinds of knowledge graphs that serve specific purposes. For example, Amazon's product graph facilitates product search on Amazon, and Facebook's social graph captures social network details to improve interactions among its users. In this dissertation, we focus on processing data on the knowledge graphs (like DBpedia and Google knowledge graphs) that contain facts describing

[2] `http://wiki.dbpedia.org/dbpedia-version-2016-04`, accessed 04/10/2017

[3] This idea is similar to Google search summary box information presentation.

[4] `https://www.google.com/`, accessed 04/10/2017

[5] `http://www.bing.com/`, accessed 04/10/2017

[6] `http://insidesearch.blogspot.com/2012/12/get-smarter-answers-from-knowledge_4.html`, accessed 04/10/2017

entities for general purpose consumption. Note that the techniques we discuss here are not restricted to general purpose or encyclopedic knowledge graphs.

Human consumption of structured data published on the Web (e.g., DBpedia) or stored in proprietary storage facilities (e.g., Bing Satori) as knowledge graphs is hard due to their large volume (i.e., information overload). For example, on average, DBpedia 2016-04 version has over 200 facts describing each entity. Processing this much information for quick understanding of a entity for a human user is very inefficient and impractical. Therefore, a better representation mechanism is required. Generating summaries is a method that has been used for many years by the linguists to address this issue. Merriam-Webster dictionary[7] defines a summary as *"using few words to give the most important information about something"*. Summarization is the process of producing summaries. Text summarization [8] has been widely used in the digital context (Web documents) as well as printed formats to provide quick insights of the documents by retrieving popular and important topics [Jones 2007; Gambhir and Gupta 2017]. Similarly, summaries for entities in knowledge graphs can facilitate quick understanding of them without the need to look at their complete descriptions.

Summaries are short (i.e., concise) but they may not always be comprehensive. When facts selected in a summary are diverse, comprehensiveness of length-limited summaries is improved. We introduce two diversity-aware summarization approaches in two different settings. The first applies to the processing of individual entities and the second to the processing of collections of entities.

1.1 Concepts of Entities, Knowledge Graphs, and Entity Summaries

We introduce the terms entity, knowledge graph, and entity summary in the following sections.

[7] `http://www.merriam-webster.com/dictionary/summary`, accessed 04/10/2017

[8] Also known as document summarization

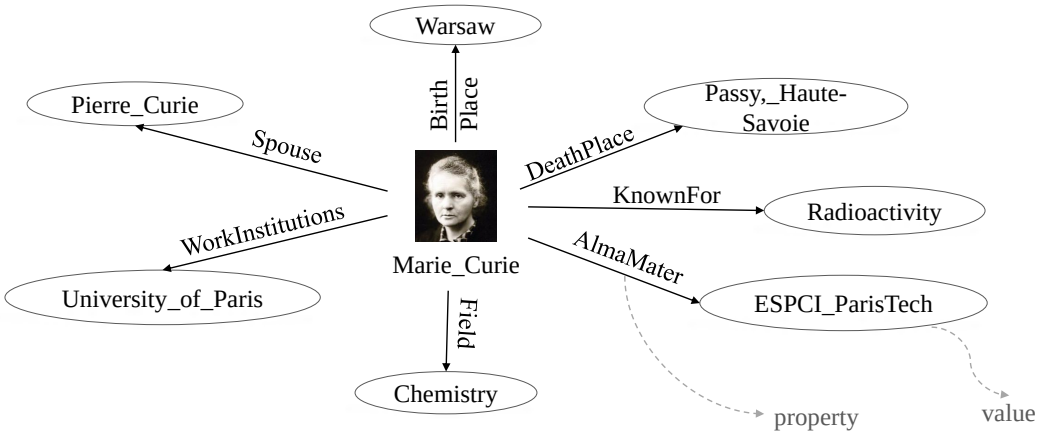


Figure 1.1: Sample entity description of the entity Marie Curie.

1.1.1 Entities and Knowledge Graphs

Entities refer to real-world things and they can be described using facts. Hence, capturing entity descriptions in a machine readable format enables machines to process and understand the world. See Figure 1.1 for an example that shows a subset of facts of the entity description for the entity Marie Curie taken from DBpedia. An entity description comprises facts (referred to as *features*) describing the entity, and each fact is made up of a property-value pair. A property is the relationship that describes the connection of the entity to the value. For simplicity, all the values shown in Figure 1.1 are entities, whereas they can also be literals. Entities are useful in many applications of today's data-rich world like answering search queries and understanding documents by analyzing entities and their relationships. Specifically, complex questions can be answered by analyzing the graph structure of the entities in the knowledge graph and inferring new knowledge. Google expressed this capability by stating "*Things but not Strings*" [Singhal 2012]. By mapping strings (literals) into entities when appropriate, machines can better understand and relate to the content in an application since each entity has its own description to compare against others[9]. Along this line, Schema.org project was initiated by major search engine companies (including Google, Microsoft,

[9] `http://searchengineland.com/future-seo-understanding-entity-search-172997`, accessed 04/10/2017

Knowledge Graph	# Entity Types	# Entity Instances	# Relation Types	# Facts
Google Knowledge Graph	1,500	570M	35,000	18,000M
Freebase	1,500	40M	35,000	637M
YAGO2	350,000	9.8M	100	4M
NELL	271	5.19M	306	0.435M
DBpedia (2016-04-English)	754	6M	2,711	1,300M

Table 1.1: Statistics of some known knowledge graphs

taken from [Dong et al. 2014] and DBpedia statistics [10].

and Yahoo) to improve visibility of Web document content by annotating entities with their types using Schema.org taxonomy.

Knowledge graphs consist of large number of entities, their descriptions, and the relationships among them at the data level. There is a schema level on top of the data level to add semantics for the knowledge graphs where concepts (classes that the entities belong) and relationships (to connect entities with information) are defined. The number of entities and facts in a knowledge graph can be in the millions and billions, respectively, and Table 1.1 presents statistics of some of the popular knowledge graphs on the Web. The statistics, except for DBpedia, are taken from [Dong et al. 2014]. In today's digital and knowledge-empowered world, knowledge graphs play a significant role in making computers smart and intelligent. For example, the search engines Google and Bing use their respective knowledge graphs to answer entity-related user queries and interact with the users. Moreover, comprehensive knowledge graphs act as connecting hubs for online data publishing. For example, DBpedia is an encyclopedic dataset and a knowledge graph created by extracting facts from Wikipedia[11] that acts as a central hub in the LOD cloud. Many datasets have links going into and coming out of DBpedia that make it a search entry point for many datasets on LOD.

[11] `https://www.wikipedia.org/`, accessed 04/10/2017

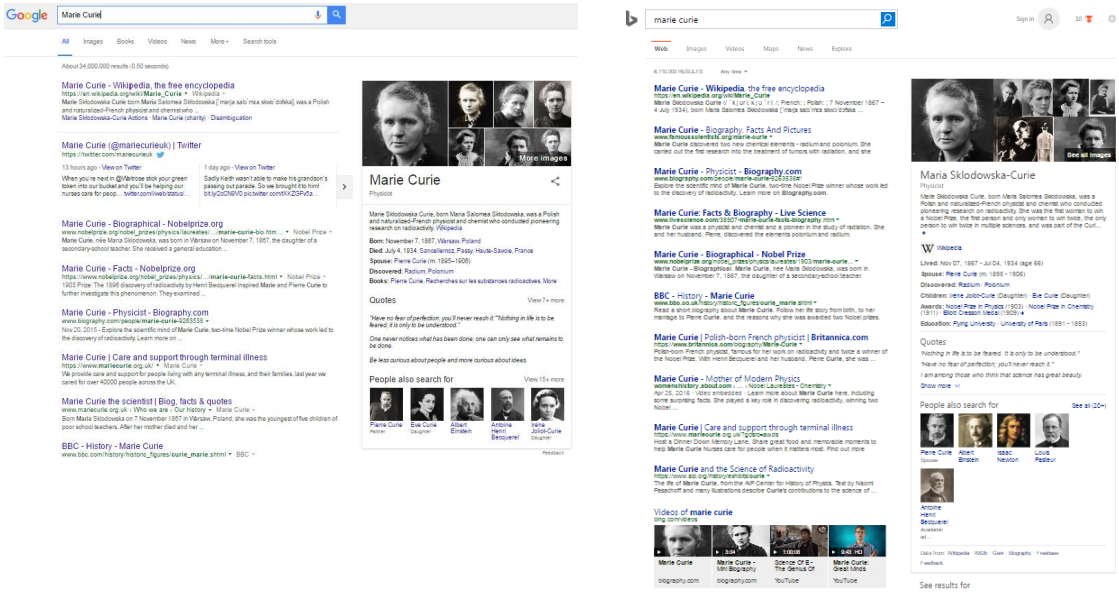


Figure 1.2: Entity summaries shown in (a) Google Search and (b) Bing Search, at the top right-hand corner of the search page.

1.1.2 Entity Summaries

[Cheng et al. 2011] defined an entity summary as a selected subset of features of the original entity description. Creating entity summaries out of their descriptions in the knowledge graphs is a well motivated task. For example, Google identified the importance and significance of entity summaries and considers it high priority in their knowledge graph building effort [Singhal 2012].

Entity summaries are important for quick understanding and analysis of content. The modern Web search task is one such obvious example use case. For example, consider Figure 1.2 which shows screenshots taken from Google Search and Bing Search when searching for the entity Marie Curie. We can see the summary generated for Marie Curie by Google in Figure 1.2 (a) and Bing in Figure 1.2 (b), where the purpose is to help the user understand and verify the information need, without going through a lengthy entity description. Further, entity summaries can be helpful in facilitating certain technical tasks like entity linking, where summaries assist human users in

selecting the correct entity from a knowledge graph to link to an entity label in document content by analyzing only the summary [Cheng et al. 2015b].

1.2 Challenges in Automatic Entity Summarization

Automatically generating entity summaries from the entity descriptions has its own challenges. The most challenging problem is the selection of important features for a given summary length constraint. This can also be positioned as a personalization problem, but in this dissertation our main focus is on general purpose entity summary creation. That is, how to create entity summaries for the entire population of users. In fact, personalization can be applied as an add-on to the summarization approach by modifying the ranking algorithms.

The problem of selecting a subset of features as the entity summary can be positioned as a ranking problem where top-k features can be selected as the summary. But, in practice, a pure ranking algorithm can be less effective because ranking alone cannot determine what kinds of features are selected for the summary. For example, consider a two feature summary for an entity listing *birthplace* and *deathplace* simply because the ranking algorithm ranked them higher than the others (e.g., they are more popular). In this case, we refer to this entity summary as less diverse. We could argue that diversity introduces comprehensiveness to the summary as it tries to provide a complete overview of the entity given the length constraint of the summary.

Adding diversity to entity summarization is challenging as we are referring to something that goes beyond analyzing syntactic dissimilarities. To achieve this kind of intelligence, we can look for semantics present in the knowledge graphs (schema level) and external knowledge sources. For example, entities are normally associated with type semantics. That is, an entity can be assigned a concept (an ontology class) as its type (e.g., the entity Barack Obama is typed as a Politician in DBpedia). But sometimes, a value in the entity description can be a literal (e.g., string). In this case, there is no explicit semantics associated with the value, and hence it is difficult to process them

intelligently. To bridge this gap, knowledge enrichment can be performed by computing semantic types (ontology classes) of the literals where applicable.

On the other hand, processing multiple entities to create entity summaries is non-trivial because: (i) there is more than one entity to consider in selecting features for the summaries and (ii) the objectives we want to achieve (diversity, relatedness, uniqueness, etc.) need to be satisfied for more than one entity at the same time (multiple constraints). For example, we would like to have diversity within an entity summary but relatedness among the summaries for the entire collection of entities. To satisfy these somewhat competing objectives, we need to employ techniques that can handle more than one entity at a time at different granularity (intra-entity vs. inter-entity).

In this dissertation, we present an intelligent framework to generate diversified entity summaries that have the qualities of *conciseness* and *comprehensiveness* considering individual entities and collections of entities.

1.3 Why? and What?

We discuss the importance of entity summarization and what we propose in this context briefly.

Why?

- Creating summaries has been the go-to method in conveying important information in less time and has practical value. For example, document summaries provide a quick glance of the content before a user can explore them. Entity descriptions, similar to web documents or text, are lengthy and hence require concise presentation.

- Entity summarization problem cannot be fulfilled by just "reusing" techniques proposed in Information Retrieval (IR) or Artificial Intelligence (AI) to handle related problems like document summarization and ranking. Entities are similar and also different in many ways to Web documents or text. Hence, challenges in (i) processing, (ii) analyzing, and (iii) computing them need

to be addressed in new or complementary ways.

- Entities and their descriptions (can) capture world knowledge and are important for modeling world and human knowledge. Summarizing them to help specific tasks (e.g., resource linking) or general purpose content understanding (e.g., web search assistance) is necessary in the modern world.

What?

- We propose methods to create entity summaries that preserve "conciseness" and "comprehensiveness" (through diversity). Being concise provides quick processing of features and comprehensiveness provides better coverage of features.
- We generate entity summaries for single and multiple (collection of) entities.

1.4 Thesis Statement

Entity related structured data on the Web can be concisely and comprehensively summarized for efficient and convenient information presentation. This can be achieved through synergistic use of: (i) Unsupervised knowledge-based methods to conceptually group, (ii) Information Retrieval-based techniques to intuitively rank, (iii) Natural Language Processing techniques to semantically enrich structured data, and (iv) Combinatorial optimization techniques to handle relatedness of multiple entities.

1.5 Entity Summarization Framework (How?) and Dissertation Focus

We propose a diversity-aware entity summarization framework for individual entities and a collection of entities. In the individual entity summarization process, entities are processed in isolation to create

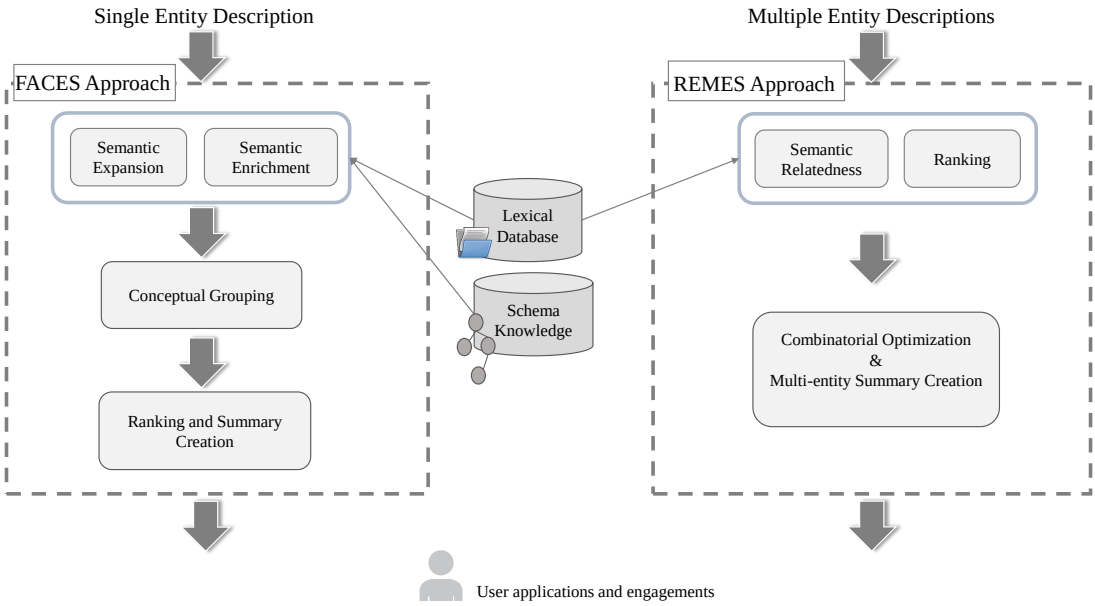


Figure 1.3: Diversity-aware entity summarization framework for individual and multiple entities.

summaries by picking diverse and important features. In contrast, the multiple entity summarization process considers a collection of entities as a whole to create entity summaries that prefers diversity within each entity summary and related features between the entity summaries. Figure 1.3 shows an overview of the two process flows. These two approaches are briefly described below.

1.5.1 Single Entity Summarization

We propose FACeted Entity Summarization (FACES) approach to create single entity summaries (entities are processed in isolation). First, we apply semantic expansion and enrichment processing steps to property-value pairs, and then partition them into conceptually similar groups. Finally, a faceted entity summary is created using the groups. Semantic expansion is the process that adds additional data semantics, and semantic enrichment process adds missing semantics to the data to make them usable. For semantic expansion and semantic enrichment, we make use of machine readable knowledge.

Machine Readable Knowledge

Knowledge is modeled in various ways that is processable by machines. Ontologies and lexical databases are two such resources that we use in our framework. An ontology captures general or domain-specific knowledge and represents them in a machine readable and community agreed form. Gruber et al. defined an ontology as "*an explicit specification of a conceptualization*" [Gruber et al. 1993] and later evolved with the term "ontological commitment" to refer to community agreement. Modeling ontologies and knowledge need special attention and there exist formal languages for this purpose. Resource Description Framework (RDF) [Manola et al. 2004], RDF-Schema (RDFS) [Brickley and Guha 2004], and Web Ontology Language (OWL) [Dean et al. 2004] are such knowledge representation languages listed in increasing order of their expressiveness.

Apart from ontologies, Natural Language Processing (NLP) communities have created lexical databases that capture the semantics of words such as, synonyms, antonyms, hypenyms, and hyponyms. These resources are helpful in enabling machines to understand document content by analyzing word semantics. For example, machines can identify two words that have the same meaning by determining whether they are synonyms or have similar hypernyms. WordNet [Miller et al. 1990] is a widely used lexical database that captures the semantics of words. We use WordNet to get abstract terms (hypernyms) for words in our framework.

1.5.1.1 Semantic Expansion

We employ a partitioning algorithm to understand conceptually similar groups in our framework, and for the algorithm to achieve good results, features need to be semantically expanded for improved agreement among them. The expansion can be performed in many different ways, and for our problem space, we need the features in each part (group) of the partition to agree at an abstract level. For example, entities related to cities, countries, and regions should be grouped together because they all are spatially related. To achieve this kind of similarity among items within a particular part in the partition, we follow the following two steps: (1) extract the type information

of the entities from schema knowledge in the knowledge graph and (2) expand tokens for each feature using hypernyms extracted from the WordNet lexical database.

Type information provide some level of semantic agreement between features (from entities mentioned in the features), but are not sufficient to achieve our objective. Type information are the ontology classes assigned to entities as their semantic type. For example, Marie Curie can be typed as a "Scientist" where Scientist is a class (concept) in the ontology (schema knowledge). Expanding these terms further, by extracting super classes from the ontology and using hypernyms retrieved from a lexical database like WordNet, we can improve the quality of the partition. In addition to taking types of the entities (value of the feature), we take tokens of the property of the feature and expand them using WordNet to get input from the relationship part of the feature. Recall that a feature is comprised of a property and a value (see Figure 1.1).

1.5.1.2 Semantic Enrichment

The values of entity descriptions can be entities or literals. When they are entities, their semantics can be represented using type information and machines can process this knowledge and get to know what ontology class they belong to. This leads to machine understanding of the data. But for literals, there is no such knowledge, except the primary implementation types assigned to them like "string". "Incomplete" semantics lead to more syntactic treatment. Therefore, we propose a method to compute types for literals in knowledge graphs where possible, to enable richer semantic processing capabilities. To achieve this goal, we utilize NLP techniques and knowledge available at the schema level (ontology).

1.5.1.3 Conceptual Grouping

After features are semantically expanded and enriched, this step tries to create groups that go beyond syntactic similarity. One challenge in this process is that the number of groups that features belong to an entity cannot be determined a priori. Therefore, unsupervised techniques in finding

these groups have been explored and hierarchical clustering algorithms work best in these scenarios. Further, in grouping features, we would like the features in a group to be more semantically similar.

Cobweb [Fisher 1987] is an incremental and hierarchical clustering algorithm that works with probability distributions. That is, if more similar things are grouped together, other similar items to come tend to be grouped into the same group. This behavior is explained as "conceptual clustering" and Gennari et al. [Gennari et al. 1989] argue that this is similar to human judgment in grouping items. Further, Cobweb provides the additional benefit of being incremental. The hierarchical clustering approaches generate dendrograms (tree structures) that we can get the partitions from by pruning at desired levels. Since the Cobweb algorithm works on a probability function, having a good distribution of attribute-value pairs for each item to be partitioned helps it to have good quality partitions. In our problem, we make use of the semantic expansion process to compensate for this. We adapt the Cobweb algorithm to conceptually group features for each entity with the use of semantic expansion and enrichment.

1.5.1.4 Ranking and Faceted Entity Summary Creation

The conceptual grouping component creates a partition for all the features for a given entity. Each part in this partition represents semantically different features. Finally, creating faceted (i.e., diversified) entity summaries consist of two steps: (1) ranking features and groups and (2) picking features from different groups to create the entity summary.

Instead of employing just a ranking mechanism, our approach creates a partition for the features of an entity and then applies a ranking mechanism to select features from the partition. We rank features within each group and based on the ranking of the features, we also rank groups. The primary building block of the ranking process is the ranking measure for each feature. For this purpose, we combine informativeness and popularity measures inspired by tf-idf measure[12]. Informativeness is similar to idf (inverse document frequency) and popularity is similar to tf (term frequency). Specif-

[12] `https://en.wikipedia.org/wiki/Tf-idf`, accessed 04/10/2017

ically, we would like each feature (property and value together) to be informative and the value to be popular.

Features are ranked within each group based on this tf-idf based score and groups are ranked accordingly based on scores of the features. Finally, faceted entity summary creation is the process of picking top ranked features from each group.

1.5.2 Multi-Entity Summarization

The main focus of the FACES approach is to generate "concise" and "comprehensive" entity summaries and it processes one entity at a time (in isolation). While FACES shows superior entity summary quality compared to the state-of-the-art systems, it cannot process a collection of entities maximizing relatedness of inter-entity features in entity summaries. An entity summary created by analyzing the other entities of interest is important to capture commonality among the entities. This can help the user to get a better understanding of the entire collection of entities. We proposed RElatedness-based Multi-Entity Summarization (REMES) approach to address this problem. A brief overview of REMES process is discussed below.

1.5.2.1 Relatedness and Ranking

In REMES, we try to maximize intra-entity diversity and inter-entity relatedness, in addition to trying to find important features for the summaries. To achieve these objectives, we measure pairwise feature relatedness and individual importance of the features.

For relatedness measures between feature pairs, we utilize a combination of graph-based relatedness measure called RDF2Vec [Ristoski and Paulheim 2016] applied on values and a Jaccard co-efficient computed on properties after applying the semantic expansion described in the earlier section. Further, the importance of a feature is measured using the tf-idf ranking score described earlier.

1.5.2.2 Optimization and Multi-Entity Summary Creation

Compared to single entity summary creation, multi-entity summary creation is computationally expensive because the algorithm has to process multiple entities at the same time to pick related features between entities. We map the Quadratic Knapsack Problem (QKP) [Gallo et al. 1980] with multiple constraints (one knapsack per entity) to generate these summaries. To compute a solution for this problem, we utilize a greedy optimization algorithm called the Greedy Randomized Adaptive Search Procedure (GRASP) [Yang et al. 2013]. More details on this optimization process are discussed later in the dissertation. Then, the computation of entity summaries for the entity collection is achieved by satisfying the knapsacks allocated to each entity in the algorithm. The size of each knapsack is the desired summary length for the entity.

1.5.3 Dissertation Focus

In this dissertation, our primary focus is on how to create concise and comprehensive entity summaries for single and multiple entities. Comprehensiveness is achieved by having diversity in the features that get selected for the summaries. In the multi-entity summarization approach, in addition to selecting diverse and important features for each summary, the relatedness among different summaries is also valued.

Main Contributions

Our main technical contributions with regards to entity summarization are as follows.

1. We present an algorithm to conceptually group features in individual entity descriptions in order to introduce diversity in selecting features for the summary. Each group in the entity description represents a unique latent dimension.

2. We enhance the grouping capabilities of features by semantically expanding them using a lexical database and schema knowledge. This makes the features in each group not only syntactically

similar but also semantically related.

3. We introduce a method to enrich knowledge graphs by adding type semantics to literals and demonstrate that such enrichment is useful in practice by offering entity summarization as a use case.

4. We propose a simple and effective ranking mechanism similar to tf-idf to rank features in entity descriptions.

5. We propose a diversity-aware entity summarization approach for individual entities that combines semantic expansion and enrichment, grouping, and ranking mechanisms.

6. We measure the relatedness of two features by computing their property similarity using semantic expansion and value relatedness using graph-based co-appearance measures.

7. We propose a relatedness-based multi-entity summarizataion approach that maximizes the relatedness of features between different entity summaries and diversity of features within each entity summary. To efficiently solve this, we adapt a greedy solution for the Quadratic Multidimensional Knapsack Problem (QMKP).

Other Contributions

In this dissertation, we explore two additional applications related to entities, entity summarization and knowledge graphs. The first is on how to align equivalent relationships between linked datasets, which is helpful in integrating and summarizing heterogeneous entities across datasets. The second is demonstrating the value of using extracted entities and their descriptions (knowledge from text) in performing a computing task of measuring similarity between the original sources where the entities are extracted.

- Grouping equivalence relations: We propose a statistics-based method to identify equivalence relationships between linked datasets. Equivalence membership is a strict grouping criteria that we explore in this use case.

- Ranking documents based on similarity: We propose a method to use knowledge (entities and their descriptions) extracted from documents (in the form of knowledge graphs) to measure inter-document similarity to rank documents for a given document.

1.6 Dissertation Organization

The rest of the dissertation is organized into chapters.

Chapter 2, presents background on World Wide Web (WWW), knowledge representation and Semantic Web, and discuss related literature for entity summarization.

Chapter 3, discusses the FACES approach for generating both *concise* and *comprehensive* entity summaries. It introduces the concepts of semantic expansion, conceptual grouping, ranking, and faceted entity summary creation. Entity descriptions can contain both object and datatype properties and, in this chapter, we focus on object property-based entity summarization. We present an evaluation of the system against the state-of-the-art approaches, and discuss the results, focusing on diversity-aware entity summarization (FACES system).

Chapter 4, introduces our semantic enrichment approach that can assign an ontology class as the type (referred to as typing) to a literal in an RDF triple. This process is directly related to the semantic enrichment component in the FACES framework. We device an approach that analyzes the focus of the entire literal and entities present in picking the correct type for the literal. We show the usefulness of such an enrichment process by taking up the entity summarization use case.

Chapter 5, presents the REMES approach for computing intra-entity diversity and inter-entity relatedness-based multi-entity summarization. We present methods for combing graph-based and semantic expansion-based relatedness measures. We adapt the Quadratic Multidimensional Knapsack Problem to solve the problem and present an evaluation of the approach against the state-of-the-art standalone entity summarization techniques.

Chapter 6, discusses two application use cases of grouping and ranking mechanisms using struc-

tured knowledge. We propose an approach to find equivalent relationships between datasets and an approach to compute (and rank) document similarity utilizing structured knowledge.

Chapter 7, summarizes the key insights of the contributions presented in the dissertation and discusses future directions.

2

Background and Related Work

2.1 WWW, Semantic Web, and Knowledge Representation

The World Wide Web (WWW or Web for short) was initiated more than twenty five years ago in 1989 by Sir Tim Berners-Lee with his innovative idea to connect remote computers over the Internet to access documents scattered over different machines [Berners-Lee 1989]. Over the years, the Web has made tremendous progress in serving that need and currently has around 50 billion documents (Web pages) indexed by popular Web search engines [Web-Statistics 2017]. Tim Berners-Lee's original idea to combine hypertext with the Internet used three component technologies:

- Uniform Document Identifier (UDI) to uniquely identify a document which later became as Uniform Resource Identifier (URI).
  - Subsequent extension to this is called Internationalized Resource Identifier (IRI) which can include unicode characters whereas URI can only contain ASCII characters[1].
- Web publishing markup language called Hypertext Markup Language (HTML).
- Hypertext Transfer Protocol (HTTP) for communication between machines.

[1] https://www.w3.org/2004/11/uri-iri-pressrelease, Accessed 04/10/2017

While WWW has been a real success, it lacked the ability to represent semantics of the content explicitly to enable the machines to understand the content.

The Semantic Web and Knowledge Representation

The Semantic Web [Berners-Lee et al. 1999; Sheth 2000; Berners-Lee et al. 2001; Antoniou and Van Harmelen 2004; Hitzler et al. 2009; Sheth and Thirunarayan 2012] is an extension of the WWW to allow more meaningful sharing and exchange of data using knowledge representation languages like RDF. The World Wide Web Consortium (W3C), the organization that defines the standards for the Web, envisions Semantic Web [Semantic-Web 2017] as "the Web of linked data and Semantic Web technologies as enabling people to create data stores on the Web, build vocabularies, and write rules for handling data." In 2001, Berners-Lee et al. [Berners-Lee et al. 2001] expressed the idea of evolving Web into Semantic Web by highlighting the term "ontology" [2]. Even before the term "Semantic Web" became popular with Tim Berners-Lee's 2001 publication, there are early efforts that made use of ontology-based Web data integration, search, and querying in the mid-to-late 1990s such as Observer [Mena et al. 1996], InfoHarness [Shah and Sheth 1999], VideoAnywhere [Sheth et al. 1999], and MediaAnywhere search [Sheth et al. 2001]. Before the wide-scale use of entity summaries in regular Web search engines as in Google and Bing (showed in Figure 1.2), the "Rich Media Reference" system [Sheth et al. 2002] in early 2000s demonstrated the idea of providing a snapshot (i.e., summary) of the entity being searched. This system used an underlying ontology (also called WorldModel^{TM}) and extracted data from Web documents (termed as creating a knowledgebase at that time) and these two components collectively formed a knowledge graph. Figure 2.1 shows an output of the system at that time.

Capturing and representing knowledge in terms of ontologies, taxonomies, vocabularies, and datasets can be achieved using formal knowledge representation languages such as first order logic, logic programming languages, and more tractable fragments such as RDF [Manola et al. 2004],

[2]Note that this is not necessarily the first article to talk about Ontology-based data processing.

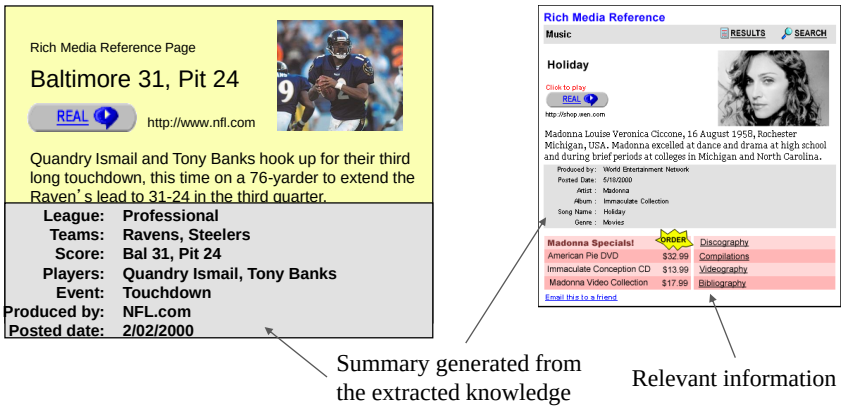


Figure 2.1: Rich Media Reference system in early 2000s for different domains.

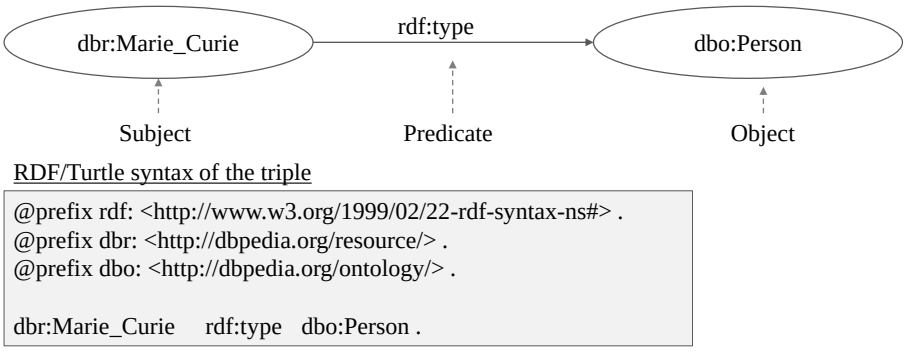


Figure 2.2: Example triple and its RDF/Turtle representation.

RDFS [Brickley and Guha 2004], and OWL [Dean et al. 2004]. The formal knowledge representation languages such as RDF and RDFS represent sentences in the form of "triples". A triple consists of a "Subject", a "Predicate", and an "Object". Figure 2.2 shows an example triple and its RDF/Turtle representation. Turtle is the simplest syntax for RDF (and easier to understand) and RDF/XML is the XML embodiment (that is more suitable for machine processing).

Linked Data

The focus of Semantic Web now is to create a Web of Linked Data where anybody can publish data on the Web linked to other data sources [Semantic-Web 2017]. The inter-linking of datasets helps

discover new knowledge/information and complement datasets. Tim Berners-Lee introduced the concept of Linked Data in 2006 to realize this vision [Berners-Lee 2006] highlighting a set of best practice rules. The rules are as follows.

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

Following these best practice rules, people started publishing datasets on the Web linking to other datasets mainly using *rdf:sameAs* at the data level (i.e., connecting entities). This resulted in the LOD cloud shown in Figure 2.3. The LOD cloud consists of interconnected domain specific and domain independent as well as encyclopedic datasets creating a huge knowledge repository, which is impossible to create by a single governing authority. These huge number of interconnected datasets on the Web are used to query [Joshi et al. 2012; Freitas et al. 2012; Freitas and Curry 2014; Freitas et al. 2015] and align [Gunaratna et al. 2014] to support user applications. Data compression techniques have also been explored to handle volume of the data [Joshi et al. 2013] and graph summarization approaches have been explored to provide a quick snapshot of the entire graph [Cheng et al. 2016].

The datasets on LOD are open to changes and continue to grow in size [Auer et al. 2013]. The number of entities and also features per entity have increased over time and become lengthy for human processing (and also for machine in some tasks). For example, DBpedia 2016-04 English version has about 6 million entities described in 1.3 billion triples. On average, it lists around 200 features per entity and there arises the necessity to summarize these features depending on the application. For example, a selected subset of the features of an entity can be shown to a user in answering a Web search query (see Figure 1.2).

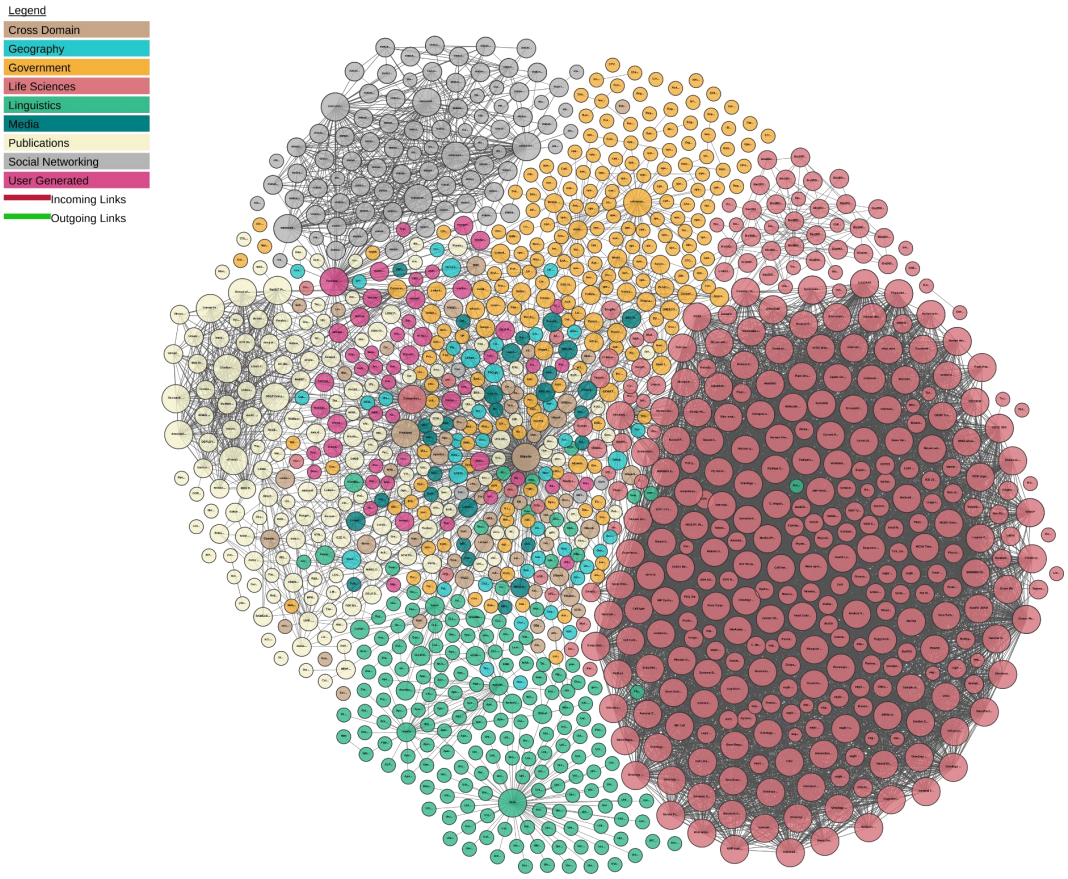


Figure 2.3: LOD cloud diagram in 2017 (as of April). See http://lod-cloud.net/ for details.

2.2 Entity Summarization

Entity summarization is the task of selecting/formulating a subset of features from an entity description to describe the entity. Summarization task can be categorized as *extractive* or *non-extractive*. Extractive summaries choose a subset of the features for the summary whereas non-extractive summaries include reformulation of the extracted facts. In this dissertation, we focus on extractive methods for entity summarization. A different categorization to this is summarizing entity descriptions for a specific task or for general consumption. For example, entity summaries can help perform a specific task like Entity Linking (EL) and Entity Resolution (ER). Entity Linking [Hachey et al. 2013] is the task of assigning an entity taken from a knowledge graph to an entity mention in the text whereas entity resolution is to determine whether two or more entity descriptions refer to the same real world entity. [Cheng et al. 2015b] and [Cheng et al. 2015a] handles the tasks of entity linking and entity resolution using entity summaries, respectively. We present methods in creating entity summaries for general purpose consumption as we encounter entity summaries in search (e.g., Google search and Bing search). We further breakdown our methods in two different granularity levels: (i) single entity summary generation and (ii) multiple entity summary generation. Chapter 3 and Chapter 4 focus on single entity summarization and Chapter 5 presents an approach for multi-entity summarization.

2.2.1 Single Entity Summarization

In this section, we focus on systems and approaches that generate single entity summaries.

Entity Summarization Methods and Ranking Algorithms

[Cheng et al. 2011] introduced and defined the problem of entity summarization in RDF graphs and showed its usefulness in quick identification of an entity. Their system, called RELIN, generalizes the PageRank algorithm to select both related and informative features. The problem with RELIN's

random surfer, which selects both related and informative features, is that it tends to emphasize central themes and similar and correlated features of an entity because of the centrality based ranking mechanism of the PageRank algorithm. SUMMARUM [Thalhammer and Rettinger 2014] is an entity summarization system for DBpedia that is also based on PageRank and utilizes the global popularity of resources gleaned with the help of information from the corresponding Wikipedia pages. LinkSum [Thalhammer et al. 2016] is an approach that makes use of PageRank for ranking and also link structure of entities in Wikipedia to select facts for entity summaries. [Thalhammer et al. 2012] proposed an approach that utilizes usage data for creating entity summaries and evaluated it in the movie domain using user ratings for movies. Such an approach is hard to generalize because usage data may not be readily available for entities. [Yan et al. 2016] pursued an effort to incorporate context in generating entity summaries by mining query logs. In the recent past, workshops (SumPre at ESWC) [3] and challenges [Gunaratna et al. 2016] have been organized to support the efforts in the area.

Ranking in the Semantic Web is closely related to entity summarization as the latter task can be perceived as selecting the top k features from an RDF graph. Various ranking algorithms exist in the literature for RDF graphs including TripleRank [Franz et al. 2009] that ranks triples, SemRank [Anyanwu et al. 2005] that ranks associations or paths, [Ding et al. 2005] rank documents, and TRank [Tonon et al. 2013] that ranks concepts. These approaches incorporate ranking algorithms for different reasons. For example, TripleRank [Franz et al. 2009] groups triples using tensors and link analysis. TripleRank's goal is to rank and identify authoritative sources for a given entity.

Proposed Approach

We propose FACES [Gunaratna et al. 2015], which conceptually groups features of an entity (to add diversity in the summary) and then ranks features based on tf-idf based ranking scheme to pick features for the entity summary. The facet or latent grouping concept that we introduce in

[3] `http://ceur-ws.org/Vol-1556/` and `http://ceur-ws.org/Vol-1605/`

FACES is different from facets in TripleRank as TripleRank's grouping is based on authority whereas FACES's is based on semantic overlap of the expanded terms of the features. Also, [Cheng et al. 2011] pointed out that it is hard to align the TripleRank approach to the entity summarization problem but its authoritative ranking is similar to RELIN where centrality dominates. Grouping in RDF datasets has been investigated in upper level ontology creation [Zhao and Ichise 2012] and property alignment [Gunaratna et al. 2013], but these groupings are different from what we explore in FACES. We find *conceptually similar* groups (explained later) that are not just related (i.e., object value overlap of RDF triples) [Zhao and Ichise 2012] or equivalent [Gunaratna et al. 2013] groups.

Instead, diversity has been shown to be useful in creating graphical entity summarization [Sydow et al. 2013], which is different from entity summarization as it produces a graph (includes neighboring entities) rather than a set of features. They pick 'lexicographically different' property names to achieve diversity in the summary using syntactic measures (e.g., *birthPlace* and *deathPlace* are different in their context) whereas FACES groups them together (i.e., going beyond string similarity). Further, FACES differs from the existing entity summarization and ranking systems in that it creates both concise and comprehensive summaries. We hypothesize that diversity provides a more complete picture of the entity (i.e., comprehensiveness) when subjected to length constraint (i.e., conciseness).

Improving Type Semantics and Entity Summarization

Type assignment to text fragments is known as Named Entity Recognition (NER) [Nadeau and Sekine 2007]. NER consists of two subtasks: (i) segmenting and (ii) classifying segmented text blocks into pre-defined categories (types). NER produces types for segments of the input text whereas EL identifies entities from which types can be inferred. Finding missing types for entities [Paulheim and Bizer 2013; Sleeman and Finin 2013] is important for the reliability of datasets and reasoning. [Paulheim and Bizer 2013] infer types for entities in DBpedia and [Sleeman and Finin 2013] predict types of entities for efficient co-reference resolution. TRank [Tonon et al. 2013] ranks entity types based on the context in which they appear (disambiguation). [Fang et al. 2010] use type

information for search, and [Tylenda et al. 2011] generate summaries by analyzing type graphs. The diversity-aware FACES approach proposed for single entity summarization requires type semantics for the values of the features to conceptually group them. Hence, it is applicable only to object property-based features.

Proposed Approach

We propose FACES-E [Gunaratna et al. 2016], which includes a typing module to compute semantic types for literals in datatype-based property values. The type computation tries to carefully select the proper type for the literal based on special processing that identifies the focus of the term. Then FACES-E utilizes both object and datatype property-based features to create comprehensive and concise entity summaries. Further, FACES-E extends FACES approach by ranking facets in selecting the entity summaries. NER and EL, however, differ from our problem in that they do not try to suggest a type based on the focus of the text but rather try to determine the types of all the entities present, causing ambiguity from the perspective of our problem (e.g., as illustrated using our evaluation with DBpedia Spotlight [Mendes et al. 2011]). Also, the approaches mentioned above work on entities and/or object properties or infer types for entities, whereas we focus on typing datatype properties where no semantic types are readily available.

2.2.2 Multi-Entity Summarization

Entity summaries have been shown to be effective in performing specific tasks (supporting human effort) like Entity Linking [Cheng et al. 2015b] and Entity Resolution [Cheng et al. 2015a; Xu et al. 2014]. Unlike previously mentioned entity summarization approaches above, these consider more than one entity simultaneously in creating summaries. Multi-entity summarization is a fairly new topic in this area and hence it is at the early stage of making progress. These approaches generate multi-entity summaries for specific tasks and we propose an approach to generate general purpose multi-entity summarization approach based on relatedness.

Proposed Approach

We propose REMES [Gunaratna et al. 2017] to generate muti-entity summaries by capturing intra-entity diversity and importance like some of the single entity summarization methods (e.g., FACES and FACES-E) and inter-entity relatedness of features. We enforce selecting diverse features for each entity and related features among entities.

3

Diversity-Aware Single Entity Summarization

The entity descriptions in the datasets published online as LOD evolve over time [Auer et al. 2013] and grow in size. For example, DBpedia [Lehmann et al. 2015] is a very large and central dataset in the LOD cloud extracted from Wikipedia. The DBpedia English version 3.9 has 4 million entities described in 470 million RDF triples (facts) and current English version 2016-04 has 6 million entities described in 1.3 billion RDF triples, averaging over 100 and 200 triples per entity, respectively. This amount of information is too much for the quick assimilation of essential information about an entity for a user. Therefore, selecting a small subset of the original triples associated with an entity as a summary is necessary for quick/convenient identification/access of entity-related information. This problem has been called *Entity Summarization* [Cheng et al. 2011] in the literature.

Entity summarization is a challenging task, like the well-established field of document summarization. Document summarization [Nenkova and McKeown 2012; Mani 2001] has been a topic of interest for data mining and information retrieval communities. Though entity summarization and document summarization are conceptually similar tasks, they differ in the nature of features, the techniques to use, and the structure of the content. Documents are unstructured with significant

textual content, so word frequency can be used to glean the essence and used in summarizing their content. For example, frequent words are good candidates for inclusion in a summary of a document. In contrast, entities are structured and do not have frequent word appearances (no duplicates most of the time). Therefore, techniques adopted for document summarization do not always carry over to entity summarization.

Many entity summarization approaches focus more or solely on centrality measures (including popularity) and ranking measures to generate entity summaries. For example, [Cheng et al. 2011; Thalhammer and Rettinger 2014] use a variation of PageRank algorithm to rank features and then pick the top k features for the summary. But we hypothesize that centrality measures and ranking mechanisms alone are not sufficient to improve the quality of entity summaries. Rather, the added use of orthogonal semantic groups of features to diversify the summaries can be effective. To investigate our hypothesis, we propose the **FAC**eted **E**ntity **S**ummarization (FACES) approach [Gunaratna et al. 2015]. Our contributions in FACES are two fold:

1. We identify conceptually similar groups of features of an RDF entity by adapting and modifying an incremental hierarchical conceptual clustering algorithm called Cobweb [Fisher 1987] and introduce an algorithm to rank features within a group.

2. We combine three dimensions: diversity, popularity, and uniqueness, to create human friendly entity summaries in a time efficient manner.

Moreover, FACES has other distinct characteristics compared to the existing state-of-the-art entity summarization tools. They are as follows.

- It selects features considering diversity which eliminates redundancy (by filtering similar features).

- It is dynamic as it is less sensitive to the order of input features and is robust with regards to evolving features (thus applicable in streaming contexts).

- It is relatively fast due to its hierarchical and incremental processing structure. FACES groups conceptually similar features in order to select the highest ranked feature (based on uniqueness and popularity) from each group to form a faceted (diversified) entity summary.

3.1 Preliminaries

Entity summarization has been a topic of interest in the Semantic Web community in the recent years and we present notions related to entity summarization adapted from [Cheng et al. 2011] below.

A data graph is a graph based data model, which describes entities using properties and their values. It consists of entities (E), literals (L), and properties (P). An entity e ($e \in E$) is described in a data graph using property-value pairs $(a, v) \in P$ x $(E \cup L)$.

Definition 1 ***(data graph)*** *A data graph is a digraph $G = \langle V, A, Lbl_V, Lbl_A \rangle$, where (i) V is a finite set of nodes, (ii) A is a finite set of directed edges where each $a \in A$ has a source node $Src(a) \in V$, and a target node $Tgt(a) \in V$, (iii) node labels $Lbl_V : V \mapsto E \cup L$ and (iv) edge labels $Lbl_A : A \mapsto P$. Lbl_V and Lbl_A are labeling functions that map nodes to entities or literals and edges to properties, respectively.*

Entities are described using property-value pairs that we call as features. The *feature* and *feature set* are defined as follows.

Definition 2 ***(feature)*** *A* feature *f is a property-value pair where $Prop(f) \in P$ and $Val(f) \in E \cup L$ denote the property and the value of the feature f, respectively. An entity e has a feature f in a data graph $G = \langle V, A, Lbl_V, Lbl_A \rangle$ if there exists $a \in A$ such that $Lbl_A(a) = Prop(f)$, $Lbl_V(Src(a)) = e$ and $Lbl_V(Tgt(a)) = Val(f)$.*

Definition 3 ***(feature set)*** *Given a data graph G, the* feature set *of an entity e, denoted by $FS(e)$, is the set of all features of e that can be found in G.*

An entity summary is a subset of all the features associated with the entity.

Definition 4 ***(entity summary)*** *Given an entity e and a positive integer $k < |FS(e)|$,* summary *of entity e is $Summ(e,k) \subset FS(e)$ such that $|Summ(e,k)| = k$.*

3.2 Faceted Entity Summarization

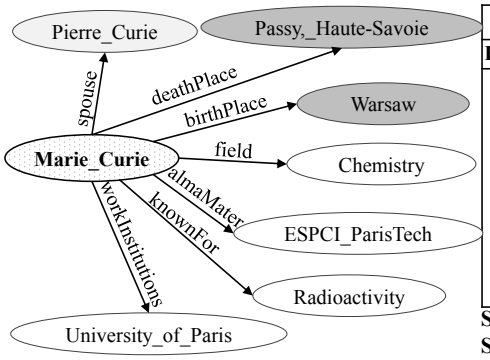
The entity summarization is application and task dependent and the general definition specifies the selection of a subset of features of an entity. In this work, we want to add diversity into the selection process to have improved coverage for the entity summary whereas in the general case, the entity summary can have similar features to each other simply because they are ranked higher by the ranking algorithm. Our objective in this regard can be informally outlined as follows:

An entity is usually described using conceptually diverse set of features to have wider coverage of information related to the entity. We want to select a 'representative' *subset of this set to uniquely identify the entity.*

Specifically, we identify semantically similar/dissimilar features and use them in ranking and selecting features for the summary. For this purpose, we define the *facet* of a feature set as follows.

Definition 5 ***(facet).*** *Given an entity e, a set of facets $F(e)$ of e is a* partition *of the feature set $FS(e)$. That is, $F(e) = \{A_1, A_2, ...A_n\}$ such that $F(e)$ satisfies the following criteria: (i) Non-empty: $\emptyset \notin F(e)$. (ii) Collectively exhaustive: $\bigcup_{A \in F(e)} A = FS(e)$. (iii) Mutually (pairwise) disjoint: if A_i,$A_j \in F(e)$ and $A_i \neq A_j$ then $A_i \cap A_j = \emptyset$. Each A_i is called a* facet *of e.*

According to the Definition 5, a facet can contain features that are similar, dissimilar, or randomly picked. In our faceted entity summarization approach, we want to identify "conceptually" similar features so as to be able to diversify the summary. Our hypothesis is that the feature set of an entity can be divided into conceptually orthogonal groups, approximated by the facets, using a partitioning (clustering) algorithm. Furthermore, a facet can be viewed as a *hidden variable*. See Figure 3.1. {F1, F2, F3} is a partition of the feature set FS. Note that the features within each facet are similar



Entity - Marie Curie				
Feature Set	**Facets**	**Features**	**Property**	**Value**
FS	F1	f1	spouse	Pierre_Curie
	F2	f2	birthPlace	Warsaw
		f3	deathPlace	Passy,_Haute-Savoie
	F3	f4	almaMater	ESPCI_ParisTech
		f5	workInstitutions	University_of_Paris
		f6	knownFor	Radioactivity
		f7	field	Chemistry

Summary 1 (normal) : {f2, f3, f4}
Summary 2 (faceted) : {f1, f2, f6}

Figure 3.1: Facets of entity - Marie Curie. Values for conceptually similar features are in the same color pattern.

than those between facets. That is, the features that are expressed through *KnownFor* and *Field* properties are *conceptually similar* because they both represent the entity's professional life. Also note that they are syntactically dissimilar. But features having *birthPlace* and *knownFor* properties are conceptually dissimilar (and also syntactically dissimilar) as they represent completely different information to each other. Next, we define *Faceted Entity Summary* for an entity based on facets that we described here.

Definition 6 ***(faceted entity summary).*** *Given an entity e and a positive integer* $k < |FS(e)|$, faceted entity summary *of e of size k, FSumm(e, k), is a collection of features such that FSumm(e, k)* $\subset$ *FS(e),* $|FSumm(e,k)| = k$. *Further, either (i)* $k > |F(e)|$ *and* $\forall X \in F(e)$, $X \cap FSumm(e,k) \neq \emptyset$ *or (ii)* $k \leq |F(e)|$ *and* $\forall X \in F(e)$, $|X \cap FSumm(e,k)| \leq 1$ *holds, where* $F(e)$ *is a set of facets of* $FS(e)$.

Informally, if the number of facets is n and the size of the summary is k, *at least one feature* from each facet is included in the summary when $k > n$. If $k \leq n$, then *at most one feature* from each facet is included in the summary. For example, a faceted summary of length 3 for the entity Marie Curie can be {f1, f2, f6} as shown in Figure 3.1. Next, we discuss the partition algorithm that is used to create facets.

3.2.1 Partitioning Algorithm - Cobweb

Facets are obtained from running a suitable partitioning algorithm for our problem. There are many partitioning algorithms available in the literature, mainly as supervised and unsupervised. Since, we do not know how many partitions are there for a feature set, we have to use an unsupervised partitioning algorithm, which is also known as clustering. We adapted a clustering algorithm called Cobweb [Fisher 1987].

3.2.1.1 Cobweb

Cobweb is an incremental, hierarchical, and conceptual clustering algorithm.

Incremental: The algorithm is incremental in the sense that it can start, stop and then resume from where it stopped last time. That is, Cobweb does not require all the items present at the start of execution. This is enabled by the four operations that Cobweb has, namely: insert, create, merge, and split. These operations are explained later.

Hierarchical: The algorithm is hierarchical as it organizes facts in a dendrogram (tree-like structure) where clusters are determined by the level the tree is cut.

Conceptual: The algorithm belongs to the class of conceptual clustering, which is a machine learning task defined by [Michalski 1980]. It basically accepts a set of object descriptions (observations, facts) and produces a classification over the observations. Thus conceptual clustering paradigm is *learning by observations* as opposed to *learning from examples.*

Cobweb algorithm has two major components. First is its *Category Utility* function that decides how to navigate the hierarchy when inserting a new item. Second is the four operators: insert, create, merge, and split.

Category Utility

An item has attribute-value pairs. Category utility function guides search within Cobweb by maximizing similarity among items in the same class (intra-class) and dissimilarity among items between classes (inter-class). Intra-class similarity is measured by the conditional probability of $P(A_i = V_{ij}|C_k)$, where $A_i = V_{ij}$ is an attribute-value pair and C_k is a class. When this probability is higher, many items with similar values will be in the same class (*predictable*). That is, the value is predictable with the class. On the other hand, inter-class similarity is measured by the probability of $P(C_k|A_i = V_{ij})$. When this probability is higher, few items in contrasting classes share the same value (*predictive*). That is, the value is more predictive of the class. The objective of the algorithm is to maximize both predictability and predictiveness. Probabilities of individual values $P(A_i = V_{ij})$ are not directly related to predictability and predictiveness, but all three can be combined as follows to give a measure of partition quality, for all classes (k), attributes (i), and values (j). Also note that it is important to increase predictability and predictiveness than individual values (this becomes low for infrequent values).

$$\sum_{k=1}^{n} \sum_{i} \sum_{j} P(A_i = V_{ij}) P(C_k|A_i = V_{ij}) P(A_i = V_{ij}|C_k)$$

From the Bayes rule, we get, $P(A_i = V_{ij})P(C_k|A_i = V_{ij}) = P(C_k)P(A_i = V_{ij}|C_k)$. By substituting this in the above, we get the following.

$$\sum_{k=1}^{n} P(C_k) \sum_{i} \sum_{j} P(A_i = V_{ij}|C_k)^2$$

Fisher explains that $\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$ is the expected number of attribute values that can be correctly guessed for class C_k [Fisher 1987]. Finally, the Category Utility (CU) is defined as the increase in the number of attribute values that can be guessed ($\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$) given a partition $C_1..., C_n$ over the number of correct guesses with no such knowledge ($\sum_i \sum_j P(A_i = V_{ij})^2$). Then the Category Utility CU for the partition $C_1, C_2, ..., C_n$ is as in Equation 3.1.

$$CU = \frac{\sum_{k=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2)]}{n} \quad (3.1)$$

The denominator n is the number of classes in the partition. The next important part of the Cobweb algorithm is its four operators.

Insert Operator

Cobweb places an item in each class and checks how successfully it fits that class. The partition resulting from including the item in a given node (which represents a class) is evaluated using the CU function given in Equation 3.1. The node that contributes to the best partition is identified as the best host for the item.

Create Operator

In addition to placing items in existing nodes in the hierarchy, Cobweb can create new nodes to represent new classes in the hierarchy. The quality of the partition is checked using CU for placing the item in an existing node or a newly created node. This operator allows Cobweb to automatically increase the number of classes.

Merge and Split Operators

Merge and split operator are important compared to insert and create operators because these two operators help the algorithm to be insensitive to the order of the initial input. This effectively makes it incremental, which is useful in dynamic and stream processing environments where the initial input cannot be finalized apriori. The merge operator takes two nodes at level n and combines them to have a partition of n - 1 nodes hoping for a better quality partition than before merging. The operation creates a new node and adds up attribute-value counts of the two nodes being merged and the two nodes become children of the new node. This is illustrated in Figure 3.2. Merging could be attempted on all nodes in the hierarchy but it is only performed on the best two hosts (determined

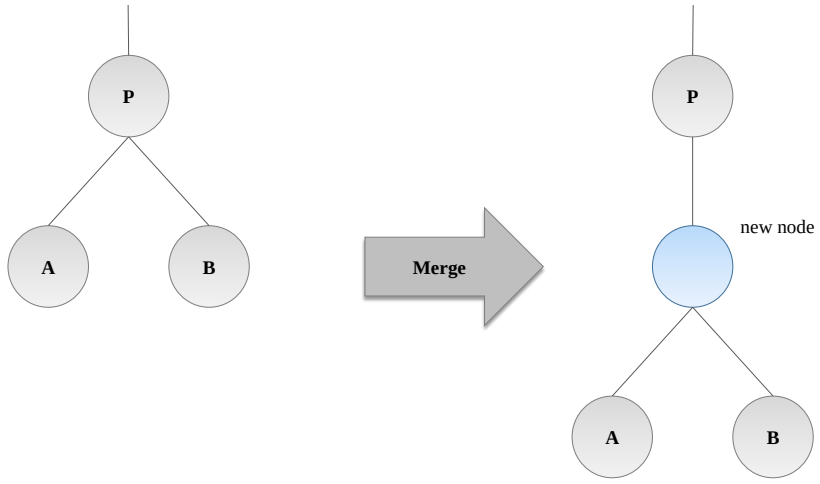


Figure 3.2: Merging in Cobweb.

through category utility) to minimize computational cost.

The split operator deletes a node and promotes the deleted node's children up in the hierarchy. A partition of n nodes results in a partition of $n + m$ - 1 nodes, where the deleted node had m children (at the current level in the hierarchy). This is illustrated in Figure 3.3. Splitting is performed only on the node that has the highest CU score. Merging and splitting are two inverses of each other and they allow Cobweb to have flexibility in creating the best possible partition from many possible ones. In other words, merging can be used to undo a previous splitting and vise-versa. Merging is performed to reduce highly similar nodes in the hierarchy and splitting is performed when a node is too compressed or general than it should be.

Table 3.2.1.1 outlines the four operators that we described above (in pseudo code format). Then Algorithm 1 presents the Cobweb algorithm using these four operators. Table 3.2.1.1 and Algorithm 1 are reproduced from [Gennari et al. 1989] to provide the process flow of Cobweb to better understand how it works. First, Cobweb method is called and if it for the first time, it creates the root node and stops. Then for each new item, it calls Cobweb starting from the root node, inserting content of the item into the current processing node. Then, category utility score is computed for each child of the node simulating inserting item into each child node. Only the nodes with the highest and the

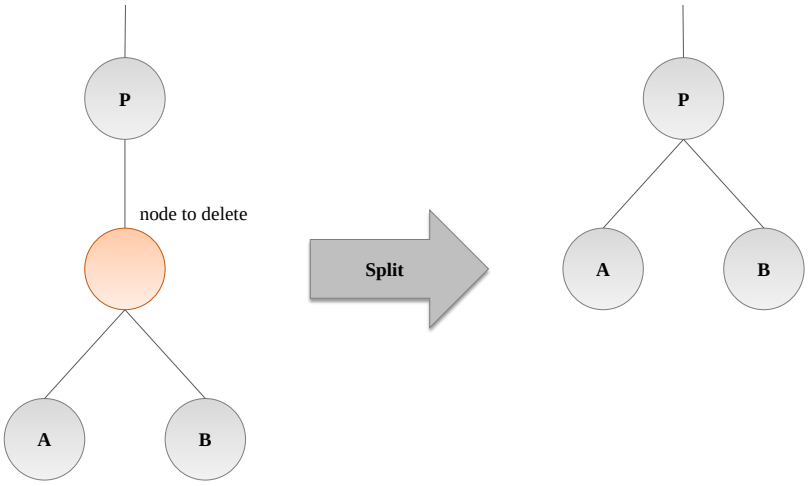


Figure 3.3: Splitting in Cobweb.

second highest category utility scores are selected to check which of the four operations to choose in going down in the hierarchy. The algorithm also uses a cut-off threshold to stop processing further down the hierarchy if the category utility value is less than the cut-off threshold.

3.2.2 Faceted Entity Summary Approach

The faceted entity summary creation consists of three major steps. First, the feature set is partitioned into facets. Second, the features are ranked using some ranking measures and finally, a subset of the feature set is selected from facets to create the faceted entity summary.

3.2.2.1 Partitioning the Feature Set - Creating Facets

We adapt Cobweb algorithm to cluster a feature set into facets. The insights of the general algorithm are discussed in Section 3.2.1. There are several challenges associated with features that we describe in our problem compared to general setting of the Cobweb algorithm. First, Cobweb is designed to work with attribute-value pairs. Higher the number of attribute-value pairs, the algorithm can better discriminate them in creating disjoint clusters (in terms of similarity). In our feature space, each feature f has only two attributes: property and value. These two attributes have values $Prop(f)$ and

Variables: N, O, P, and R are nodes in the hierarchy.

I is an unclassified instance, A is a nominal attribute, V is a value of an attribute.

Insert(N, I)

Update the probability of category N

for each attribute A in instance I **do**

  for each value V of A **do**

    Update the probability of V given category N

  end for

end for

Create(N, I)

  Create a new child M of node N

  Initialize M's probabilities to those for N

  Create a new child O of node N

  Initialize O's probabilities using I's values

Merge(P, R, N)

  Make O a new child of N

  Set O's probabilities to be P and R's average

  Remove P and R as children of node N

  Add P and R as children of node O

  Return O

Split(P, N)

  Remove the child P of node N

  Promote the children of P to be children of N

Table 3.1: Cobweb operations

Algorithm 1 Cobweb algorithm

```
1: Input: The current node N of the concept hierarchy. An unclassified (attribute-vallue) instance I.
2: Results: A concept hierarchy that classifies the instance. Top-level call: Cobweb(Top-node, I).
3: Variables: C, P, Q, and R are nodes in the hierarchy. U, V, W, and X are clustering (partition) scores.
4: Cobweb(N, I)
5: if N is a terminal node then
6:     Create(N, I)
7:     Insert (N, I)
8: else
9:     Insert(N, I)
10:    for each child C of node N do
11:        Compute the score for placing I in C
12:    end for
13:    Let P be the node with the highest score W
14:    Let R be the node with the second highest score
15:    Let X be the score for placing I in a new node Q
16:    Let Y be the score for merging P and R into one node
17:    Let Z be the score for splitting P into its children
18:    if W is the best score then
19:        Cobweb(P, I) {place I in node P}
20:    else if X is the best score then
21:        initialize Q's probabilities using I's values {place I by itself in the new category Q}
22:    else if Y is the best score then
23:        let O be Merge(P, R, N)
24:        Cobweb(O, I)
25:    else if Z is the best score then
26:        Split(P, N)
27:        Cobweb(N, I)
28:    end if
29: end if
```

$Val(f)$, respectively. Since two attribute-value pairs are too low for the algorithm to meaningfully cluster conceptually similar features, we expand the two attributes and create a word set (WS) to expose their semantics to work with the algorithm. Our expansion process captures the abstract (i.e., higher level) meaning of the words being expanded. Then each word in the word set (WS) becomes an attribute and its value is either 0 or 1, meaning it is either present or absent.

Each feature f (property-value pair) is expanded using the techniques described as follows. We expand property ($Prop(f)$) and value ($Val(f)$) of the feature using WordNet [1] and typing information of the values, respectively. WordNet is a lexical database available online and typing information is available for object URIs representing values in the knowledge graph. Recall that entity is described using property-value pairs, and in this approach, we consider only object properties where property values are URIs. In this case, both properties and values are represented using URIs in the knowledge graph. We take labels of the URIs for processing, and if labels are not available, the local names (last part) of the URIs are utilized. For property expansion, first, we tokenize and remove stop words. Then we retrieve the higher level abstract terms for the tokens. The higher level abstract terms are determined by the hypernyms retrieved from the WordNet. For the value expansion, we first retrieve typing information (ontology classes assigned) and then, we tokenize them, remove stop words, and get higher level abstract terms as before using WordNet. Tokenization includes processing camel case, spaces, punctuation, underscores, etc. and we consider only nouns to be valid tokens. The expanded terms include all the tokens resulting from the tokenization step, the original term, and the added expanded terms of the tokens. Finally, we union all the expanded terms of the property and value to get the word set $WS(f)$ of feature f. Figure 3.4 shows examples of word set.

Our term expansion process enables the algorithm to have a rich set of words capturing the semantics for reliable semantic clustering. But note that an attribute has only boolean values unlike original Cobweb algorithm and hence, the category utility function is refined as follows. Let C_p be a node in the hierarchy with a set of features that has child clusters (partition) $C_1, C_2, .., C_n$. Then

[1] `https://wordnet.princeton.edu/`, accessed 04/10/2017

Feature (f)	Property expansion	Value expansion	Word set (WS)
region:Illinois	{region, location, domain}	{place, PopulatedPlace, populated, place, point, area, locality}	{region, location, domain, PopulatedPlace, populated, place, point, area, locality}
birthPlace:Honolulu	{birthPlace, birth, place, beginning, point, area, locality}	{place, PopulatedPlace, populated, point, area, locality}	{birthPlace, birth, place, beginning, point, area, locality,PopulatedPlace, populated, point, area, locality}
vicePresident:Joe_Biden	{vicePresident, vice, president, corporate executive, head of state}	{person, OfficeHolder, office, holder, organism, flesh, human body, occupation, job, staff, possessor, owner}	{vicePresident, vice, president, corporate executive, head of state, person, OfficeHolder, office, holder, organism, flesh, human body, occupation, job, staff, possessor, owner}
predecessor:George_W._Bush	{predecessor, forerunner, precursor}	{person, officeholder, office, holder, organism, flesh, human body, occupation, job, staff, possessor, owner}	{predecessor, forerunner, precursor, person, officeholder, office, holder, organism, flesh, human body, occupation, job, staff, possessor, owner}

Figure 3.4: Feature expansion using WordNet and type information.

CU of partition $\{C_1, C_2, .., C_n\}$ of C_p can be computed as in Equation 3.2. $P(C_x)$ is the probability of a random feature belonging to child cluster C_x and (A_i, V_i) is the i^{th} attribute-value pair of the new feature being clustered.

$$CU(C_p) = \frac{\sum_{x=1}^{n} P(C_x) \sum_{i=1}^{2} [P(A_i, V_i|C_x)^2 - P(A_i, V_i|C_p)^2]}{n} \tag{3.2}$$

Since our attributes are simply boolean valued (values are either 0 or 1 signifying presence or absence for a word), we take the probability of word appearances instead of attribute-value pair notation. The final category utility function for our process is as shown in Equation 3.3. W_i is a word appearing in $WS(f)$. Equation 3.3 is the function that we use to compute category utility score for each node as we go down in the hierarchy according to the Algorithm 1 and decide on what operation to perform out of create, insert, merge, and split. The operation that receives the highest score is performed if it satisfies the cut-off threshold. If not, we stop adjusting the hierarchy at the node where the cut-off threshold is not satisfied.

$$CU(C_p) = \frac{\sum_{x=1}^{n} P(C_x) \sum_{i} [P(W_i|C_x)^2 - P(W_i|C_p)^2]}{n} \tag{3.3}$$

Next, we discuss how to rank a feature for selecting features to form the faceted entity summary from each facet.

3.2.2.2 Ranking Features

Our approach ranks features in facets and hence it does not suffer from ranking similar features to be higher globally. In contrast, PageRank-based algorithms rank all the features together, biasing similar features taking top spots. This adversely affects the ability to have diverse features (hence less comprehensive) in the summary as the summary can have only limited number of features (because conciseness). Our introduction of facets and then picking features ranked in each facet eliminates this limitation, helping the faceted entity summaries to be both concise and comprehensive at the same time.

We employ a special ranking algorithm influenced by the tf-idf technique to get the top ranked features from each facet to form the faceted entity summary. For a feature f and the value v of f, the ranking takes into consideration the informativeness/uniqueness (reflected using idf) of f ($Inf(f)$), and popularity (reflected using tf) of v ($Po(v)$). $Inf(f)$ is defined in Equation 3.4. The numerator N is the total number of entities in the knowledge graph G while the denominator is the number of entities that have the feature f. $Po(v)$ is the number of distinct triples in the knowledge graph G that has the matching value v. Following the IR tradition we take the log of this value as expressed in Equation 3.5.

The ranking of features within a facet is then the product of the informativeness of the feature and popularity of the value as defined in Equation 3.6. The intuition is that a feature (property-value pair) should be relatively rare (i.e., informative) to be interesting and not the property alone. For example, consider the example "residence" as the property where many person type entities have this property defined for them. For a summary, it is not important how unique or popular a

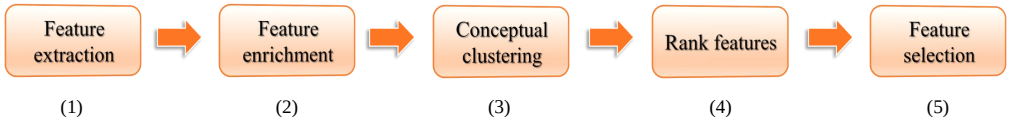


Figure 3.5: Flow of steps in creating faceted entity summaries.

property name is, but how distinct is the property-value combination. Property "residence" may be popular in a dataset but when it says residence is "White House", it is important to see how often other entities have both this property name and the value. This intuition helps to uniquely identify and distinguish the entity from others. In this case, informativeness is higher. Also when the value is popular in the dataset, it tends to help form a more human readable summary (because they are well known). Because we like to maximize both uniqueness and popularity (note that it is uniqueness of features and popularity of values), the ranking score is composed of product of the two scores.

$$Inf(f) = log(\frac{N}{|\{e|f \in FS(e)\}|}) \tag{3.4}$$

$$Po(v) = log|\{triple\ t \in D|\exists\ e, f : t \equiv (e\ Prop(f)\ v)\}| \tag{3.5}$$

$$Rank(f) = Inf(f)\ *\ Po(Val(f)) \tag{3.6}$$

3.2.2.3 Faceted Entity Summary Creation

The sequence of steps for faceted entity summary creations is shown in Figure 3.5. For a given entity e and a positive integer k (which is $< |FS(e)|$), the steps are as follows.

1. Get the feature set $FS(e)$ for the entity e.

2. Each feature f in $FS(e)$ is enriched to have a wordset $WS(f)$.

3. The enriched feature set $FS(e)$ is input to the partitioning algorithm to create facets. The algorithm yields a dendrogram (hierarchical tree) for $FS(e)$ that is cut at a desired level to get the facet set $F(e)$ of $FS(e)$.

4. The features in each facet are ranked using the ranking algorithm.

5. The top ranked features from each facet are picked according to Definition 6 (typically in a round robin fashion through the facets) to form the faceted entity summary of length k. In this implementation, we avoid picking features that have the same property name from each facet when $k > |F(e)|$.

Note that in this implementation of FACES, we do not rank facets but rank only features in each facet. Therefore, when $k < |F(e)|$, selecting a facet (to pick the highest ranked feature in the facet) is random. We will discuss ranking facets and other improvements over this in Chapter 4.

3.3 Evaluation

The evaluation of our novel approach to entity summarization is conducted in two ways, both comparing ours to the state-of-the-art systems: (1) Using a manually created gold standard and (2) Analyzing user preference. The state-of-the-art summarization approaches we considered in our evaluation are RELIN [Cheng et al. 2011] and SUMMARUM [Thalhammer and Rettinger 2014]. We picked these two because, RELIN outperformed earlier entity summarization tools in the literature and SUMMARUM is DBpedia specific. RELIN works on both object and datatype properties whereas SUMMARUM works only on object properties. We did not choose domain specific summarization tools like [Thalhammer et al. 2012] as they are not applicable in general and require additional domain specific usage data. We do not consider the graphical entity summarization [Sydow et al. 2013] approach as it is different from an entity summarization captured by RELIN and FACES.

We selected the DBpedia dataset for our evaluation as it was the benchmark dataset selected in [Cheng et al. 2011] and contains entities that belong to different domains. We created a gold standard for the evaluation due to unavailability of the evaluation data of RELIN (as confirmed by the authors of RELIN). We randomly selected 50 entities [2] from DBpedia (English version 3.9) that have at least 17 distinct properties per entity. The average number of distinct features per entity is 44. Further, RELIN's results and user agreement for ideal summaries are not the same as those reported in [Cheng et al. 2011] because of the differences in the test set. We filtered out schema information and dataset dependent details such as *dcterms:subject*, *rdf:type*, *owl:sameAs*, *wordnet_type* and *Wikipedia related links* to ease manual evaluation and further they do not express facts about the entities. We extracted object-type properties for this dataset as that is our focus now. Further, use of object-type properties is consistent with SUMMARUM, and hence enables a meaningful comparison. We asked 15 human judges with background in Semantic Web to select 5 and 10 feature length summaries for each of the entities. These are referred to from now on as the *ideal summaries*. The judges were not given specific information about any system and asked to select a summary that can best represent the entity (facilitate quick identification). We provided them the Wikipedia page link of each entity in case they needed additional information about an unfamiliar entity. Each entity has at least 7 ideal summaries from 7 different judges and this comprises the gold standard for the evaluation. All experiments were performed using a Core i7 3.4 GHz Desktop machine with 12 GB of RAM. We replicated DBpedia dataset locally and used caches for RELIN as mentioned in [Cheng et al. 2011]. More details about the approach and gold standard dataset can be found at the web page[3].

[2] Selected entities are from the domains of politician, actor, scientist, song, film, country, city, river, company, game, etc

[3] `http://wiki.knoesis.org/index.php/FACES`, accessed 04/10/2017

3.3.1 Evaluating with the Gold Standard

Our objective is to show that faceted entity summaries produce results that are closer to human judgment. We configured FACES that produces the best possible results and RELIN according to its recorded optimal configuration. We empirically determined that cutting FACES cluster hierarchies at level 3 gave good results. We also set the cut-off threshold of Cobweb to 5, which gave the optimal results. For RELIN, we set the jump probability and number of iterations to 0.85 and 10, respectively. Authors of RELIN provided the source code of RELIN in absence of the evaluation data to replicate the environment. We replaced Google search service with Sindice API[4] as Google search API was no longer free of charge (for Point-wise Mutual Information (PMI) computation in RELIN). Sindice indexes the LOD data and is adequate for this purpose. Further, we collected Google API and Sindice API search hits for a small random sample (5 entities) and applied RELIN to both API search hits. The results confirmed that the difference is negligible.

$$Agreement = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} |Summ_i^I(e,k) \cap Summ_j^I(e,k)| \tag{3.7}$$

$$Quality(Summ(e,k)) = \frac{1}{n} \sum_{i=1}^{n} |Summ(e,k) \ \cap \ Summ_i^I(e,k)| \tag{3.8}$$

We use the evaluation metrics used in [Cheng et al. 2011]. When there are n ideal summaries for summary length k denoted by $Summ_i^I(e,k)$ for $i = 1,..,n$ and an automatically generated summary denoted by $Summ(e,k)$ for entity e, the agreement among ideal summaries is measured by Equation 3.7 and the quality of the automatically generated summary is measured by Equation 3.8. In other words, quality of an entity summary is its average overlap with the ideal summaries for the entity. Our evaluation results and statistics are presented in Table 3.2. We modified the original RELIN algorithm to discard duplicate properties and named it as RELINM in Table 3.2. The modification did not yield much improvement in the results and this was to test whether re-ranking

[4]http://sindice.com/developers/searchapiv3, service was accessible until May 2014

System	Evaluation 1 - Gold standard					Evaluation 2 - User preference	
	k = 5			k = 10			
	Avg. Quality	FACES % ↑	Time/Entity	Avg. Quality	FACES % ↑	Study 1	Study 2
FACES	**1.4314**	NA	**0.76 sec.**	**4.3350**	NA	**84%**	**54%**
RELIN	0.4981	187 %	10.96 sec.	2.5188	72 %	NA	NA
RELINM	0.6008	138 %	11.08 sec.	3.0906	40 %	16%	16%
SUMMARUM	1.2249	17 %	NA	3.4207	27 %	NA	30%
Avg. Agreement	1.9168			4.6415			

Table 3.2: Evaluation of the summary quality and FACES %↑ = 100 * (FACES avg. quality - Other system's avg. quality) / (Other system's avg. quality) for k=5 and k=10, respectively, and average time taken per entity for k=5 for Evaluation 1. Evaluation 2 measures user preference % for each system. (NA stands for Not Applicable)

mechanisms can improve RELIN results. We ran each algorithm for all the 50 entities 5 times and recorded the average time taken per entity in seconds. According to the results, FACES achieved 138% and 40% increase in quality against RELINM and 187% and 72% increase in quality against RELIN, and 17% and 27% increase in quality against SUMMARUM for k=5 and k=10, respectively. Even though FACES achieves superior results compared to RELIN, it does not compromise its efficiency. FACES is much more efficient (14 times faster) than RELIN as shown in Table 3.2 in summary computation (i.e., running time). Time for SUMMARUM has been marked as not applicable (NA) as we got results from a black box web service but can be assumed to be similar to RELIN. FACES can be further improved to compute summaries on the fly.

We conducted the *paired t-test* to confirm the significance of the FACES's mean summary quality improvements over others. For $k = 5$ and $k = 10$, p-values for FACES against RELINM are 2.04E-10 and 1.92E-14 and for FACES against SUMMARUM are 0.029 and 3.71E-7. When p-values are less than 0.05, the results are statistically significant.

3.3.2 Evaluating with User Preference

We carried out two blind user evaluations (users didn't know which system produced which summaries) to see how users rated summaries created by FACES, RELINM, and SUMMARUM. We randomly selected 10 instances from our evaluation sample and created $k = 5$ length summaries and had 69 users participate in the evaluation. The results are shown in Table 3.2. For the first user evaluation, we showed them summaries of FACES and RELINM side by side and asked them to select the best summary that helped them to identify the entity. The users average preference was 84% for FACES and 16% for RELINM. This shows that unique properties alone are not desirable to humans. In the second user evaluation, we showed users all three system summaries and their preferences were 54%, 16%, and 30% for FACES, RELINM, and SUMMARUM, respectively. This shows that users sometimes prefer popularity as in SUMMARUM but not in all the cases. RELINM got almost the same percentage in both experiments reflecting that users preferred unique features for some entities. Moreover, results suggest that users like a balanced (unique and popular) and diversified approach like in FACES and confirm our claim that the diversity makes summaries more human friendly.

3.4 Discussion

In summary, our evaluation shows that FACES performs better in all cases. It was able to achieve 138% and 40% increase over RELINM and 17% and 27% increase over SUMMARUM in summary quality for k=5 and k=10, respectively. The results of the paired t-test confirms that FACES results are not random and consistently outperforms the other two systems. FACES achieves this by its non-trivial facet identification process and persistence in selecting a diverse summary based on the facets. Note that agreements between ideal summaries are not very high in the sample dataset due to the large number of distinct features present in each entity (see Table 3.2). The second evaluation shows that the faceted summaries are desired by human users.

FACES	RELINM	SUMMARUM
<birthPlace, Warsaw>	<isPrimaryTopicOf, Marie_Curie>	<birthPlace, Poland>
<workInstitutions, University of Paris>	<wasDerivedFrom, oldid=547107936>	<birthPlace, Warsaw>
<field, Physics>	<knownFor, Polonium>	<birthPlace, Russian_Empire>
<spouse, Pierre Curie>	<almaMater, ESPCI>	<field, Physics>
<deathPlace, Passy, Haute-Savoie>	<deathPlace, Passy, Haute-Savoie>	<field, Chemistry>

Figure 3.6: Entity summaries for the entity Marie Curie by each system. $k = 5$ and the size of feature set is 39.

FACES behaves similar to an ordinary summarization algorithm such as RELIN and SUMMARUM when there are few facets available. It behaves as if it is ranking a flat list of features. A key reason why RELIN (including RELINM), which is based on PageRank that exploits both informativeness and relatedness measures, underperforms is that the summary can include redundant and correlated features. This also affects SUMMARUM. The redundancy comes at the expense of reduced coverage. We address this limitation by emphasizing diversity to suppress redundant features[5] and improve coverage by identifying facets to pick representative features. Figure 3.6 shows summary examples generated by the three approaches for the entity Marie Curie.

We observed that sometimes FACES clusters features differently from what we expect. For example, *spouse* property of Barack Obama is clustered into a facet that contains *vicePresident* property. This happens because sometimes typing information of values is too specific[6] and affects the clustering process. Moreover, FACES performs better in almost all cases according to the second evaluation except for some specific entities. E.g., *Usain Bolt* is an Olympic athlete with many records. Some users preferred facts about his 100 meter records over his 200 meter records, while both information were present in a facet. FACES can generate either of the two facts in a summary (based on a subjective ranking within a facet) and the tie can be broken. We can further investigate how to effectively combine diversity, uniqueness, and popularity of features as the user preference varies. Furthermore, clustering phase of FACES can be tuned for fine grained grouping by

[5] May not be syntactically the same but conceptually similar

[6] *Michelle Obama*'s typing information is similar to a politician

modifying the enrichment process. E.g., adding hyponyms to the word set makes features containing *birthPlace* and *stateOfOrigin* properties fall into different facets.

Limitations

FACES identifies different groups that are conceptually similar using type information of the values of the features and hypernyms retrieved for properties of the features. FACES can successfully create faceted entity summaries for object properties where object properties have types associated with values. Further, we use WordNet lexical database to retrieve hypernyms and all available hypernyms (for all senses) are used. Even though Cobweb algorithm is claimed to be "less sensitive" for the order of input features, it is not guaranteed to produce the same groupings all the time.

3.5 Conclusion

We have investigated how to create entity summaries that are *concise* and *comprehensive* for the purpose of quick identification of an entity. We adapted a well known incremental hierarchical conceptual clustering algorithm for entities (in RDF format) to identify facets (that addressed diversity) and developed intra-cluster ranking algorithm for features (that addressed uniqueness and popularity) to create *faceted entity summaries*. We showed that faceted entity summaries created by combining diversity, uniqueness, and popularity are better representatives for an entity and closer to ideal ones. Our approach shows superior results (improvement in summary quality in the range 17% - 187%) and does not require pre-computation of values like the existing systems.

In the next chapter, we discuss ways of incorporating datatype properties into faceted entity summaries by typing them and improving the algorithm by ranking facets. Further, we introduce new ranking equations specifically tailored for datatype property based features.

4

Typing Literals in RDF Triples for Improving Coverage of Features in Entity Summarization

The importance of entity summarization has been discussed in Chapter 3 in the context of volume of structured information becoming available on the Web. We introduced a novel approach called FACES for creating *diversity-aware* entity summaries for object property-based features. In this chapter, we investigate how to incorporate datatype properties into entity summaries which can improve the coverage of information in an entity summary.

The RDF has been used extensively to encode information and publish as Semantic Web datasets and knowledge graphs. The direct consumers of these datasets, most of the time, are machines or software such as search and rank services. Therefore, it is imperative to enrich the datasets with additional information so that machines can interpret them properly. Assigning ontology classes as types to resources (typing) in RDF via the `rdf:type` property (which we refer to as semantic types) is one example of a data enrichment process. This can help machines to identify similar or related

resources by analyzing their types. Such enrichment can be exploited to improve the quality and reliability of datasets and facilitate analytics. Typing is performed on URI resources and, hence, only applies to object properties. On the other hand, datatype property values are literals, and are usually associated with types by virtue of their syntactic data representation (which we refer to as syntactic types). For example, one assigns `datatypes` such as `xsd:string, xsd:integer,` and `xsd:date` to literals.

Syntactic types do not make explicit information that can be exploited for data analysis. However, the amount of information that datatype properties represent compared to object properties is significant in some real-world datasets. For instance, DBpedia, which is one of the largest and most comprehensive encyclopedic datasets on the Web, has 1,608 datatype properties compared to 1,103 object properties in its 2016-04 English version. Many of the literal values (other than noise) can be associated with types selected from a set of ontology classes that can promote proper semantic interpretation and use. For example, the property `http://dbpedia.org/property/location` has about 1,05,047 unique and simple literals that can be mapped to entities to infer the types (e.g., "California", "United States").

The importance of type information has been demonstrated by researchers in the Semantic Web community, including for inferring missing types for entities [Paulheim and Bizer 2013], ranking types for entities [Tonon et al. 2013], and generating summaries using type graphs [Tylenda et al. 2011]. All these approaches make use of existing type information of "entities" or infer additional/missing types from them. There has not been any work related to inferring or computing types for literals in RDF/S datasets. In this work, we propose to address the issue of computing "semantic types" whenever possible for literal values of datatype properties. When types are available for datatype properties, they can be used in many interesting applications including but not limited to data integration, property alignment, and entity summarization. For example, in property alignment [Gunaratna et al. 2013; Gunaratna et al. 2014], we can utilize types to prune the candidates for alignment. Further, type prediction on datatype properties provides benefits similar to the works on type prediction

for entities as in SDType [Paulheim and Bizer 2013]. We demonstrate the application of generated semantic types by extending the FACES entity summarization algorithm, which we described in Chapter 3 and show how to group and rank features based on datatype properties [Gunaratna et al. 2016]. Our contributions in this work are twofold:

1. We analyze the object value of datatype properties and select a suitable class as the type from a given set of ontology classes.

2. We extend FACES and implement FACES-E [Gunaratna et al. 2016] to group and rank both object and datatype properties to create entity summaries and demonstrate the usefulness of types to generate comprehensive entity summaries.

4.1 Typing Literals in RDF Triples

In this section, we describe the problem and present the approach to generate types for literals in knowledge graphs.

4.1.1 Object Property and Datatype Property

OWL defines two types of properties: (i) object properties that connect individuals to individuals and (ii) datatype properties that connect individuals to data values (literals) [1]. The object value of an object property is a URI that can be assigned an ontology class as its type via `rdf:type` property. But object values of datatype properties do not have ontology classes assigned as types and the only types available for them are the syntactic types referring to primitive, low-level implementation types. Figure 4.1 illustrates the difference of the two properties.

[1] `http://www.w3.org/TR/owl-ref/#Property`, accessed 04/10/2017

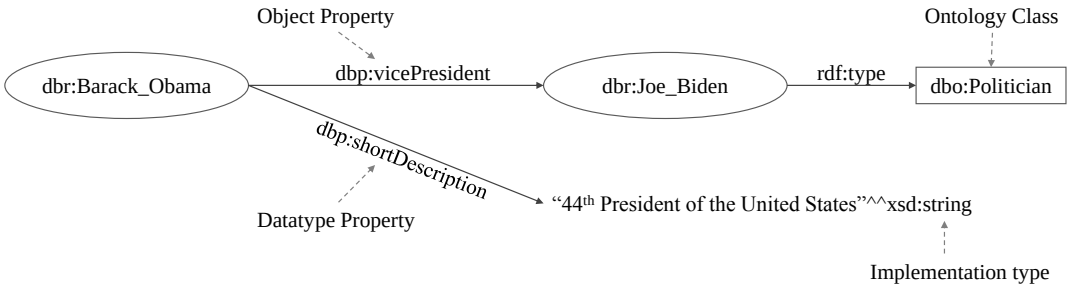


Figure 4.1: An object and datatype property instances for the entity Barack Obama in DBpedia. dbo, dbp, and dbr represent http://dbpedia.org/ontology, http://dbpedia.org/property, and http://dbpedia.org/resource namespaces, respectively.

4.1.2 Problem Analysis

Datatype properties do not have ontology classes assigned as types for their values but have only implementation types assigned. In this work, we try to suggest a class from a given set of classes as the type of the literal of the datatype property. See Figure 4.2 which shows two triples ((2) and (3)) having datatype properties of the entities `dbr:Barack_Obama` and `dbr:Calvin_Coolidge` taken from DBpedia. The dotted boxes show the types for the two literals that we intend to compute, supplementing the syntactic type `xsd:string` which is already available. Whenever a semantic type can be computed for a literal, it can be used in practical applications for inferencing, grouping, and matching. For example, both computed types for the values of the two datatype properties, that is `dbo:President` and `dbo:Governor`, are `rdfs:subClassOf dbo:Politician` class in the DBpedia ontology and hence can be utilized for grouping them together based on the type (considering the fact that they both represent politicians).

Datatype properties may have been provided instead of object properties for entities in datasets for various reasons, such as (i) the creator was unable to find a suitable entity URI for the object value, and hence chose to use a literal instead, (ii) the creator of the triple did not want to attach more details to the value and hence represented it in plain text, (iii) the value contains only basic

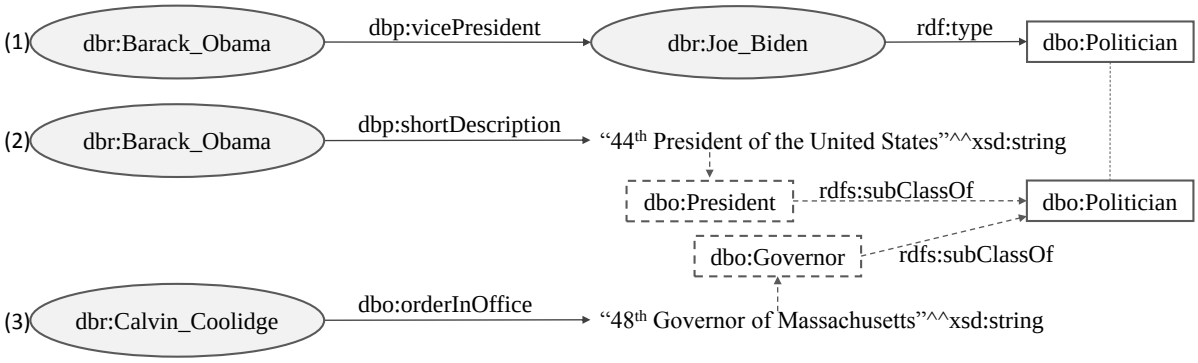


Figure 4.2: Two triples corresponding to datatype properties and one triple corresponding to an object property taken from DBpedia. Computed types are shown in dashed boxes. dbo, dbp, and dbr represent http://dbpedia.org/ontology, http://dbpedia.org/property, and http://dbpedia.org/resource namespaces, respectively.

implementation types like integer, boolean, and date, and hence not meaningful to create an entity, or (iv) the value has a lengthy description spanning several sentences (e.g., `dbo:abstract` property in DBpedia) that covers a diverse set of entities and features. We attempt to assign a semantic type by analyzing cases (i) and (ii) for text (up to a sentence long, delimited by a period) and avoid assigning a "type" to lengthy literal values as mentioned in (iv) because its focus may not be clear (and multiple conflicting types can result).

4.1.2.1 Problem Statement

Let S P O be an RDF triple specifying subject (S), property (P), and object (O), and C be all classes (in the schema) for the set of triples D. If P is an object property, then O is an entity (i.e., individual). The type of O is a class assignment to O via the RDF triple, O `rdf:type` c, where $c \in C$. We refer to this as "semantic typing" in this work. Since the value of a datatype property instance is a literal, no semantic typing can be readily found for the value. We want to suggest a class $\bar{c} \in C$ for the value of a datatype property in addition to the syntactic type. We focus on text

that is up to one sentence long. This realistic restriction has been imposed to ensure entity and type coherence. That is, we avoid deriving a unique type for longer texts, say a paragraph. Figure 4.2 illustrates types suggested for the object values of the triples (2) and (3).

For clarity of presentation, we define Type Set ($TS(v)$) for property value v as the set of classes that are assigned (via `rdf:type`) or inferred (via `rdfs:subClass Of`) from the class set C. If p is an object property, then $|TS(v))| > 0$, otherwise $|TS(v)| = 0$.

4.1.3 Type Generation for Literals

Recall that we are interested in typing literals in RDF triples. These literals appear in object position where the property of the triples is of type datatype property. Determining the relevant type for a datatype property value is challenging due to several reasons. First, picking some term used in the literal value to determine the entity or class for the datatype property value does not work. For example, in triple (3) in Figure 4.2, if we select the term "Massachusetts" as the entity to represent the entire value and use its type `dbo:PopulatedPlace` as the type, we obtain an incorrect interpretation. The main focus of the text is the term "Governor" and not "Massachusetts", as it intends to conveys information about the governor. Therefore, we propose identifying this term, which we call the *focus term*, by analyzing the grammatical structure of the text. Then, we match the identified focus term to a suitable entity or a class in deriving the type for the value.

We utilize the Collins head word detection technique [Collins 2003] to identify the focus terms. The Stanford CoreNLP [2] API offers an implementation of this technique using parse trees. We use the UMBC semantic similarity service [Han et al. 2013] to compare the suggested class and the set of given classes. It facilitates the computation of phrase similarity, which generalizes and improves upon WordNet-based similarity. The algorithm for generating a type set ($TS(v)$) for a datatype property value v is presented in Algorithm 2.

Algorithm 2 shows details of the method *getTypesForText*, which generates types for the input

[2] `http://nlp.stanford.edu/software/corenlp.shtml`, accessed 04/10/2017

Algorithm 2 getTypesForText($Text\ v$)

```
1: initialize Set types to {} and pre-determined Integer n
2: Set X ← getPhrases(v)
3: for each Phrase x ∈ X do
4:   if isNumeric(x) then
5:     Set cls ← predefined date/numeric type
6:   else
7:     Set ngrams ← getNGrams(x, n)
8:     Text focusTerm ← parseHeadWord(x) {head word identifier}
9:     Set cls ← getTypeFromLabel(focusTerm)
10:    if isEmpty(cls) then
11:      cls ← getTypesFromNGrams(focusTerm, ngrams)
12:    end if
13:    if isEmpty(cls) then
14:      cls ← getMatchedType(focusTerm) {semantic matching}
15:    end if
16:  end if
17:  types ← cls
18: end for
19: return types
```

text (datatype property value). We avoid processing if the input text is more than one sentence long (segmented by "period"). If the identified sentence has phrases delimited by comma, we segment them at comma (lines 2-3) and generate types for them. This is because these segments normally align for the same abstract meaning (e.g., "Austrian-American bodybuilder, actor", "Denison, Texas"). We identify numeric or date values using simple regular expressions (lines 4-5). If the value is not numeric, we start the type computation process for the phrase by identifying n-grams associated with the phrase up to the maximum token length of n (line 7). Then, we retrieve the focus term by parsing the phrase using the head word identifier (line 8). Next, we check whether there is an exact match of the focus term and any of the types (via `rdfs:label` of classes) in the dataset. If a match is found, we take the class as the type of the phrase (line 9). Otherwise, we further analyze all the generated n-grams with the focus term to infer a type in the *getTypesFromNGrams* method (line 11). If there is still no match, we compute the similarity scores of the focus term against all the types in the dataset (via `rdfs:label` of classes) and get the highest match (> 0) as the type of the phrase (line 14). Finally, we aggregate types generated for each phrase to obtain the set of types for the input text.

The method *getTypesFromNGrams* processes the n-grams set to allow for a maximal match of entity labels. It processes n-grams to extract types only if they contain the focus term. For each of those n-grams that contain the focus term, we check to see whether there is an exact match of the n-gram to a type. If no match is found, we spot entities for the n-gram and then get the types of those entities. We spot entities by exact matching of their labels (`rdf:label`) to n-grams. Looking for n-grams that contain the focus term (in descending order of n-gram token lengths from n to 1) can improve the quality of the identified types. For example, consider "Harvard Law School" as a datatype property value in DBpedia. The identified focus term for this phrase is "School." When we start processing n-grams in descending order of n, we encounter "Harvard Law School" as the first candidate for typing. This matches the entity `dbr:Harvard_Law_School` whose type `dbo:Educational_Institution` is then taken as the type of the phrase. We do not generate types

Property Value	Generated Types
Team Knight Rider	Television Show, Work
American politician, 44th President of the United States	Agent, Politician, Person, President
List of The Cosby Show episodes	List
English	Language
Lung cancer	Disease

Table 4.1: Types generated for a sample of values.

for long text[3] (e.g., paragraphs) because they cannot be unambiguously typed as they can represent many different entities (with contrasting descriptions) and need further analysis to pick the correct type. Table 4.1 shows examples for the computed types for a sample of literal values taken from DBpedia entities.

4.1.4 Evaluating Type Generation for Datatype Property Values

Generating types for all the available datatype property values is not meaningful because there are labeling properties that simply represent human readable names for entities. The RDFS standard defines the `rdfs:label` property to provide such information, but in practice, there exist many such labeling properties (e.g., foaf:name). Ell et al. [Ell et al. 2011] studied the characteristics of these properties by manually inspecting the properties and their instance data. Similarly, we created a list of labeling properties for our data sample and filtered them out. We extracted a sample of unique datatype property-value pairs from DBpedia (version 3.9 and 2015-04). *Precision* for the identified types of a property value v is defined in terms of $TS(v)$ in Equation 4.1. Then, we define the *Mean Precision* (MP) of property values as in Equation 4.2, where n is the number of property values in the sample that have $|TS(v)| > 0$.

[3]Note that we can still run the algorithm for each sentence to generate types.

$$Precision(TS(v)) = \frac{\#correct\ types\ in\ TS(v)}{|TS(v)|} \tag{4.1}$$

$$MeanPrecision = \frac{\sum_{i=1}^{n} Precision(TS(val(f_i)))}{n} \tag{4.2}$$

Mean Precision is the average of precision over the property values in the feature sample. When the MP value is higher, the algorithm generates many correct types over different property values. It is important to know how often the algorithm can generate at least one correct type. Therefore, we define *Any Mean Precision* (AMP) as in Equation 4.3, where n is the number of property values in the sample that have $|TS(v)| > 0$. It computes the average of all the ceiling values of Precision($TS(v)$). If the algorithm generates at least one correct type for a value, it counts the precision as 1 in averaging. When AMP is higher, the algorithm generates at least one correct type often. These metrics are useful for comparing different algorithms.

$$AnyMeanPrecision = \frac{\sum_{i=1}^{n} \lceil Precision(TS(val(f_i))) \rceil}{n} \tag{4.3}$$

	Mean Precision (MP)	Any Mean Precision (AMP)	Coverage
Our approach	0.8290	0.8829	0.8529
Baseline	0.4867	0.5825	0.5533

Table 4.2: Type generation evaluation. DBpedia Spotlight is used as the baseline system.

Table 4.2 shows the evaluation results performed by one evaluator. We constructed a baseline using a state-of-the-art tool to identify entities in the values and retrieved their types and super types (except `owl:Thing`). Specifically, we used DBpedia Spotlight [Mendes et al. 2011] for this purpose and configured it with default parameters including the confidence of 0.5. We had a total of 1,117 unique property-value pairs after filtering out 118 labeling and noisy pairs. Coverage is the fraction of features that had a type generated. Our approach performed better compared to the baseline because we identify types using a combination of focus terms and matching entities and

types. We did not measure recall because it is hard to produce an exhaustive list of all correct types for each value.

4.2 Incorporating Datatype Properties into Faceted Entity Summaries

Ranking and grouping (clustering) object property-based features is discussed in Section 3.2.2 of Chapter 3. But these cannot be directly applied to datatype property-based features for many reasons like object value is not a URI and hence type information is not available and searching literals for frequency is not desirable. Therefore, we adapt them and generate faceted entity summaries for both types of properties.

4.2.1 Problem Statement

Ranking and grouping features of object properties can be achieved using the entities that their values represent. For ranking, we can utilize Equations 3.4 to 3.6, and for grouping semantically related features, we could use their types as discussed in Chapter 3. Ranking features belonging to datatype properties cannot be done similarly because their literal values do not have a unique representation across the dataset (which URIs do provide for object properties) as the same entity may be referred to using minor variants of a literal. Therefore, we need to reflect related entities in ranking datatype properties by modifying Equations 3.4 to 3.6. For example, consider the third triple of Figure 4.2 where we can spot: `dbr:Governor` and `dbr:Massachusetts`; here, we can use their frequency as opposed to checking the frequency of the entire literal value of the property.

Grouping (conceptually similar) datatype properties is non-trivial compared to grouping object properties where types are available. Note that multiple entities and/or classes spotted in a datatype property value can confuse the groupings. For example, the second triple in Figure 4.2 has the entity `dbr:United States` having the type `dbo:Country` but eventually results in the class `dbo:`

`President` as the type. In general, this requires recognizing multiple types (e.g., country and president) and then resolving them suitably (e.g., to president) to enable the grouping of similar triples (among both object and datatype properties). See triples (1) and (2) in Figure 4.2, where the first represents an object property and the second represents a datatype property. In fact, both values convey information about a person, while for the datatype property value, it is not explicit. The object property clearly has a type assigned to its value and if we compute the type for the datatype property as `dbo:President`, then we can abstract their values to type `dbo:Politician` which can be inferred for the datatype property value using `rdfs:subClassOf`.

4.2.2 Grouping Datatype Property Features

Grouping of features can be done at two levels: exact/syntactic similarity and semantic/abstract similarity. Exact/ syntactic similarities can result in very fine-grained groups, while we are interested in groups based on their abstractions as in FACES. For example, triples (2) and (3) in Figure 4.2 present two literal values that do not share any common token (no/less syntactic similarity). However, when we compute a type for each, they are sub-types of the class `dbo:Politician`. The clustering algorithm that uses such type information of the values as in FACES can group them together. Further, it can also group features of both object and datatype properties which was hitherto not possible. For example, it can group features represented by triples (1) and (2) in Figure 4.2 because the values are indirect instances of the type `dbo:Politician`. Figure 4.3 illustrates grouping of similar features with same color using the clustering algorithm presented in Chapter 3.

4.2.3 Ranking Datatype Property Features

We discuss ranking measures for datatype properties usable in the context of entity summarization. Recall that Equations 3.4 to 3.6 can be used only to rank object properties. If we compute $Inf(f)$ as in Equation 3.4 for datatype properties, it will have an artificially high value because the exact literal denoting an entity appears infrequently compared to URI references of the entities for object

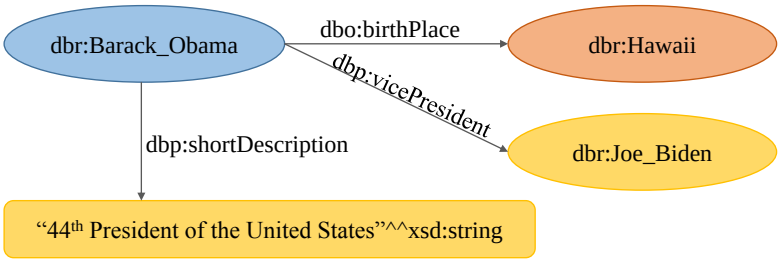


Figure 4.3: Grouping both property features. *dbo:birthPlace* and *dbp:vicePresident* are object properties and *dbp:shortDescription* is a datatype property. dbo, dbp, and dbr represent http://dbpedia.org/ontology, http://dbpedia.org/property, and http://dbpedia.org/resource namespaces, respectively.

properties. As a consequence, every datatype property will have a high ranking score. To fix this discrepancy, we spot entities in datatype property values and get the frequency of entity URIs as a measure of informativeness and popularity. We first spot entities in the datatype property values by analyzing the n-grams generated in Algorithm 2. Let $ES(v)$ be the set of all entities that can be spotted for value $v = Val(f)$. We process all the n-grams generated for v (for a pre-defined n-gram token length n) and match them against the entity labels (`rdfs:label`) in the dataset to obtain $ES(v)$. Then, we choose the most popular (frequent) entity in the dataset from this set as the representative entity for v. The intuition is that humans spot and identify popular entities in text phrases and, hence, they can provide identifiable facts in the summary for datatype properties. Let *max(ES(v))* be a function that returns the most popular entity e_{max} from the set $ES(v)$ based on the frequency of appearance of each entity (using Equation 3.5). For example, for triple (2) in Figure 4.2, function *max* identifies `dbr:President` and `dbr:United_States` as the entities and picks the latter to be the most popular entity. Hence, it is used to calculate the informativeness of the feature and popularity of the phrase.

We compute the informativeness of a feature f of a datatype property, $Inf(f)'$, using Equation 4.4. We check for occurrences of features in entities "similar" to f (as opposed to checking

the same feature as in object properties) by matching datatype property names and values that contain the most popular entity e_{max}. Then, we count those entities to compute informativeness of the feature. That is, informativeness is inversely proportional to the number of entities that are associated with overlapping values containing e_{max}. N is the total number of entities.

$$Inf(f)' = log\Big(\frac{N}{|\{\begin{array}{r} e \in E | \exists\ f' \in FS(e) : Prop(f) = Prop(f')\ and \\ max(ES(Val(f))) \in ES(Val(f')) \end{array}\}|}\Big) \tag{4.4}$$

Similarly, for measuring the popularity $Po(v)'$ of a datatype property value v, we take the frequency of the most popular entity $e_{max} = max(ES(v))$ in v, as specified in Equation 4.5. Then, the ranking score of feature f that belongs to datatype properties, $Rank(f)'$, is calculated using Equation 4.6. When $ES(v) = \emptyset$, we take the denominator as the number of property instances in Equation 4.4 and $Po(v)' = 1$ in Equation 4.5, effectively ranking them low.

$$Po(v)' = log|\{\begin{array}{r} triple\ t \in D | \exists\ e, f : t \equiv (e\ Prop(f)\ e_{max}) and \\ e_{max} = max(ES(v)) \end{array}\}| \tag{4.5}$$

$$Rank(f)' = Inf(f)' \ * \ Po(Val(f))' \tag{4.6}$$

4.2.4 Faceted Entity Summaries using Object and Datatype Properties

Equations 3.4 - 3.6 (which are mentioned in Chapter 3) and Equations 4.4 - 4.6 are used to rank features within each facet (cluster partition). We further extend the FACES approach by ranking facets using the average of feature ranking scores ($FacetRank(F(e))$) generated by Equations 3.6 and 4.6, as shown in Equation 4.7 for a facet $F(e)$. $R(f)$ is a function that selects the proper ranking method depending on whether $Prop(f)$ is an object or datatype property. Then, facets are ordered from the highest to the lowest FacetRank score and we iterate over them in that order to pick individual features for the summary.

$$R(f) = \begin{cases} Rank(f), & \text{if } Prop(f) \textit{ is an } \text{object property} \\ Rank(f)', & \text{otherwise} \end{cases}$$

$$FacetRank(F(e)) = \frac{\sum_{f \in F(e)} R(f)}{n}, \textit{ where } n = |F(e)| \tag{4.7}$$

Given the feature set $FS(e)$ of an entity e and a positive integer $k < |FS(e)|$, the adapted process for the faceted entity summary creation is as follows.

1. The feature set $FS(e)$ is partitioned into facets. The algorithm yields a dendrogram (hierarchical tree) for $FS(e)$ and it is cut at an empirically determined level to get the facet set $F(e)$ of $FS(e)$.

2. Features in each facet are ranked using the ranking algorithms (Equations 3.6 and 4.6).

3. Then the feature ranking scores of features in each facet are aggregated and averaged to get the facet ranking score (Equation 4.7).

4. The top ranked features, from highest to lowest ranked facet, are picked (Definition 6) in a round robin fashion to form the faceted entity summary of length k.

4.2.5 Evaluating Faceted Entity Summaries with Both Types of Properties

We evaluated the proposed extended faceted entity summarization approach FACES-E against another state-of-the-art algorithm called RELIN [Cheng et al. 2011]. It has been shown before in Chapter 3 that FACES outperformed RELIN for object properties. RELIN has been the only tool to generate entity summaries for both datatype and object properties. We evaluate FACES-E against RELIN for the full range of features and show the benefits of the datatype property typing which enabled FACES-E to group features belonging to object and datatype properties in the partition algorithm. We randomly selected 20 entities from the initial FACES evaluation (using DBpedia 3.9) in Chapter 3 and another random sample of 60 entities from DBpedia version 2015-04 for a total of

80 unique entities. We retrieved object properties as mentioned in Chapter 3 and added datatype properties to each entity, filtering labeling properties (including date and numeric). We created a new gold standard for the entity samples by asking 17 human users to create summaries of length 5 and 10 for each of the 80 entities as the "ideal summaries" (total of 900 user-generated ideal summaries for both summary lengths). Each entity received at least 4 different ideal summaries and this comprises the gold standard. The evaluation metrics are the same as mentioned in Chapter 3. That is, the summary quality is measured by Equation 3.8 and the agreement of user generated summaries are computed using Equation 3.7.

System	k = 5		k = 10	
	Avg. Quality	%↑	Avg. Quality	%↑
FACES-E	**1.5308**	–	**4.5320**	–
RELIN	0.9611	59 %	3.0988	46 %
RELINM	1.0251	49 %	3.6514	24 %
Avg. Agreement	2.1168		5.4363	

Table 4.3: Evaluation of the summary quality (average for 80 entities) and %↑ = 100 * (FACES-E avg. quality - Other system avg. quality) / (Other system avg. quality) for k=5 and k=10, where k is the summary length.

We used previously determined thresholds for both FACES-E and RELIN. RELINM is the modified version of RELIN where it discards duplicate properties in the summary. For FACES-E, we cut cluster hierarchies at level 3 and set the cut-off threshold of the clustering algorithm (Cobweb) to 5. For RELIN and RELINM, we set the jump probability to 0.85 and the number of iterations to 10. The evaluation results of entity summarization are presented in Table 4.3. Note that the *summary quality* is better when it is closer to the *agreement* value. Agreement is low for this evaluation, as was the case with previous evaluations in Chapter 3, because the number of features per entity was relatively high (on average 44 features per entity). This evaluation confirmed several of our claims about the approach in creating high summary quality:

1. Our intuition for ranking datatype property features by giving precedence to popular entity mentions in the value instead of getting any other entity mention or the whole literal of the value is effective.

2. Grouping features based on types, including the generated semantic types for datatype property values is appropriate.

3. Ranking facets in selecting faceted entity summaries is effective.

We further conducted a *paired t-test* to confirm the significance of the FACES-E's mean summary quality improvements over RELINM. For $k = 5$ and $k = 10$, p_values for FACES against RELINM are 8.24E-11 and 9.42E-12. When p_values are less than 0.05, the results are statistically significant. According to results shown in Table 4.3 and the paired t-test, FACES-E performed better and benefited from datatype property typing.

4.3 Discussion

Our typing algorithm enabled FACES-E (to overcome a limitation of FACES) to process both types of properties, thereby improving coverage. Likewise, typing for datatype properties can facilitate a wide range of data processing applications. Property alignment [Gunaratna et al. 2013] is a use case where similarly typed property values can be used to limit the properties analyzed for equivalence and relatedness.

In the first evaluation, our algorithm showed MP and AMP values of 0.82 and 0.88 compared to 0.48 and 0.58, respectively of the baseline. Furthermore, our algorithm showed good coverage. Getting both MP and AMP values to a relatively high level is desirable for faceted entity summaries. This is because when the algorithm can generate a correct type most of the time, also with high precision, it helps to group semantically similar features together. This, in turn, facilitates the creation of high quality, "diversified" entity summaries evidenced by the second evaluation in Section 4.2.5.

However, we also note that datatype properties in our entity sample have labeling and noisy properties (due to incorrect or missing details) which is common occurrence in real-world datasets on the Web. We manually filtered such labeling properties; however, this can be a challenging and important problem to solve in the future.

It is possible to use the meaning of the property names and word sequence relationships of values for type generation. For this, a machine learning model similar to Conditional Random Fields (CRF) could be utilized whereas now, focus term detection drives the type computation. This can facilitate predicting whether a value can be typed or not and filtering out noisy and labeling property values. The absence of matching entities in DBpedia can stymie the generation of type information. These are common dataset quality and completeness issues orthogonal to our problem. For missing information, we can: (1) use type inferencing approaches [Paulheim and Bizer 2013] to generate missing types and (2) use a comprehensive set of ontology classes and entities (e.g., from LOD) in our approach.

Our approach does not generate "semantic types" for numeric and date value properties, and challenges exist for measuring their popularity in the dataset for ranking. These properties will be investigated in the future for grouping and ranking in faceted entity summary generation. Furthermore, a formal model/approach needs to be adapted for RDF semantics to encapsulate type generations for datatype properties and then they can be encoded in the datasets similar to object properties whereas now, we keep the computed types for literals as additional information outside the dataset.

Limitations

The type computation for literals is limited for properly formatted short string values. It cannot identify types for values containing dates and numerical values. This is mainly due to the approach's focus on processing values and not taking insights from processing properties. It cannot identify a focus term for numerical values or dates. Further, the type computation depends on the focus term

identification and semantic similarity computation between ontology classes and the candidate type. FACES-E extended FACES approach by ranking facets to create diversified summaries and selection of features depends on this ranking. Further, FACES-E treats object and datatype properties without any difference and hence noise in datatype properties directly transformed into the entity summaries.

4.4 Conclusion

In this Chapter, we discussed an approach to adapt FACES approach that we introduced in Chapter 3 to improve the coverage of summaries. That is, FACES, that handles features based on object properties because datatype property values do not have ontology classes assigned to them as types, and hence, could not be grouped into conceptually similar features. We extended FACES by proposing a method to compute types for literals in RDF triples to deal with datatype properties. In other words, we have investigated the problem of computing types for datatype property values. We generate types for a property value by: (1) exact and semantic matching of the focus term to class labels, and (2) spotting entities related to the focus term and retrieving their types.

Our contributions in this work span over two significant problems: (1) enhancing datatype property values with type metadata, and (2) proposing FACES-E, which extends FACES to generate more comprehensive entity summaries using both types of properties. We evaluated both type generation and the extended entity summarization approach using the DBpedia encyclopedic dataset on the Web and showed improvement over the state-of-the-art. Our novel typing algorithm for datatype property values enhances data with additional semantics and, hence, is useful in applications beyond entity summarization, such as property alignment, data integration, and dataset profiling.

We have discussed ways of creating entity summaries for single entities and in the next chapter, we present an approach to compute summaries for multiple entities considering their relatedness.

5

Relatedness-based Multi-Entity Summarization

The earlier chapters of this dissertation have discussed approaches for creating comprehensive single entity summaries. These focus on summarizing an entity by giving precedence selecting the most important features for distinctly identifying the entity. But summarizing a collection of entities by showing related features (retrieved from a knowledge graph) for quick understanding of the entity collection as a whole compared to individual entities in isolation is an important issue that is yet to be resolved. Such a system can help users to: (i) understand documents when browsing by presenting related features for entities and (ii) interact with related features and entities when searching and browsing on the Web (e.g., Google search shows related entity collections). A solution to this problem should maximize the similarity or relatedness of features about the entities as it presents information about the entity collection in a cohesive manner. For example, Figure 5.1 shows an example of such a summary creation for "Apple Computer" and "Steve Jobs". For the entity Steve Jobs, it shows more facts about computers than other topics because the majority of the entities are talking about computers or entities related to computers. Further, it shows facts related to the entire entity collection (e.g., selection of "California" for Steve Jobs). In other words, the summary

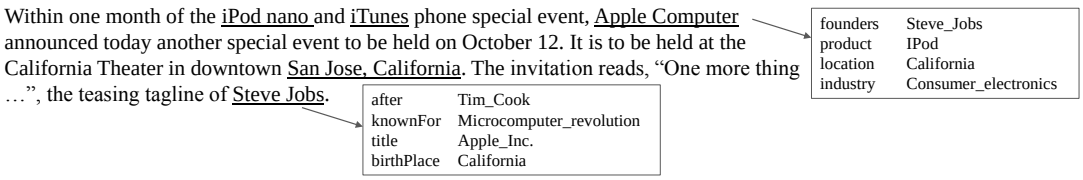


Figure 5.1: Entity summaries maximizing relatedness between them for a news item from Wikinews corpus.

generated for the entity Steve Jobs can vary from document to document depending on the other entities that appear with it. Hence, this kind of a summary is dynamic and context dependent, compared to single entity summaries which are context independent and static.

Diversity is an important characteristic that makes entity summaries comprehensive, subject to the length constraints. Therefore, we should try to maximize the diversity of the features selected for each entity summary; otherwise, they may contain redundant features at the expense of more information. We propose **RE**latedness-based **M**ulti-**E**ntity **S**ummarization (REMES) approach [Gunaratna et al. 2017] that facilitates the above-mentioned characteristics in creating entity summaries for an entity collection. For this purpose, we adapt and map the Quadratic Multidimensional Knapsack Problem (QMKP), which is an extension of the Quadratic Knapsack Problem (QKP) [Gallo et al. 1980] and utilize graph-based relatedness and semantic similarity measures. Specifically, we:

1. Generate entity summaries for a collection of entities by: (i) maximizing inter-entity related features, (ii) maximizing intra-entity importance of features, and (iii) minimizing intra-entity related features, by adapting QMKP. We modify a version of the Greedy Randomized Adaptive Search Procedures (GRASP) algorithm to compute entity summaries efficiently through approximation.

2. Utilize graph-based and semantics-based relatedness measures to create entity summaries.

5.1 Multi-Entity Summarization

We first discuss the problem statement below. The formal definitions for entities, entity summaries, and features are as described in earlier chapters.

5.1.1 Problem Statement and Description

Problem Statement: Given a collection of entities, we select features belonging to these entities maximizing inter-entity relatedness and intra-entity importance, and minimizing intra-entity relatedness of features.

In this problem, we consider generating entity summaries for a collection of entities together by selecting features to show the relatedness among the entities, and importance and diversity within entities. That is, for a given entity collection $\{e_1, e_2, ...e_n\} \subseteq E$, and summary length constraints $k_1, k_2, ..k_n$, we want to generate corresponding entity summaries $Summ(e_1, k_1), Summ(e_2, k_2), ...$ $Summ(e_n, k_n)$ that has the maximum score according to the following objectives:

$$
\begin{aligned}
(Summ(e_1,k_1),..,Summ(e_n,k_n)) = & \underset{(Se_1 \subseteq FS(e_1),..,Se_n \subseteq FS(e_n))}{argmax} \\
& \Big(\alpha * (\Sigma_{x=1}^{n} \Sigma_{f_i \in Se_x}\ rank(f_i)) \\
& - \beta * (\Sigma_{x=1}^{n} \Sigma_{f_i,f_j \in Se_x}\ r(f_i,f_j)) \\
& + \gamma * (\Sigma_{i=1}^{n} \Sigma_{j=i+1}^{n} \Sigma_{f_i \in Se_i} \Sigma_{f_j \in Se_j}\ r(f_i,f_j))\Big) \\
& where\ |Se_x| \leq k_x,\ k_x \in \mathbb{Z}^+
\end{aligned}
\tag{5.1}
$$

The function r and $rank$ compute relatedness and importance scores in the range [0,1], as discussed in Section 5.1.2.3. $\alpha, \beta, \gamma \in \mathbb{R}^+$ are the weights (of the objectives) to be tuned. By maximizing the similarity of features selected in different entity summaries, we provide connections between the entities in their summary descriptions for the coherency of the content. We maximize the selection of important features as well as related ones to make good quality summaries. Further, we avoid selecting similar features for an entity to improve diversity and coverage of features given the summary

length constraints.

5.1.2 Approach

The problem described in Section 5.1.1 requires maximizing relatedness, importance, and diversity of features (property-value pairs), controlled by the length of each entity summary for the entity collection. First, let's consider selecting features for an entity e from its feature set $FS(e)$. Then we discuss how to extend it to process an entity collection.

5.1.2.1 Selecting Features for an Entity

The features $f \in FS(e)$ are numbered from 1 to $|FS(e)|$. First, the important features need to be selected for the summary. For this, we utilize a tf-idf based ranking score for each feature f. Second, the selection of similar features in the summary for an entity should be discouraged to improve diversity (and hence improves coverage given the limit on summary length). To demote the selection of features that are similar to the already selected ones for the summary from the entity, we represent the relatedness between the features with the negation of the similarity value.

By defining a pairwise profit function for the features, we can map this problem as an instance of the QKP [Gallo et al. 1980]. QKP is a generalization of the classical 0-1 knapsack problem where it maximizes a quadratic objective function subject to linear constraints [Gallo et al. 1980; Yang et al. 2013]. We define the profit p_{f_i,f_j} as in Equation 5.2 for selecting the feature pair f_i and f_j for the summary, where α, $\beta \in \mathbb{R}^+$. The function $rank(f_i)$ calculates the importance of the feature f_i and the function $r(f_i, f_j)$ computes relatedness of the two features f_i and f_j. The intuition behind giving a negative value for the relatedness score when the two features belong to the same entity is to make their overall profit lower if they are highly related to each other. This discourages selection of new features which are more related to the already selected features for the entity summary (that leads to increased diversity).

$$p_{f_i,f_j} = \begin{cases} \alpha * rank(f_i), & \text{if } i = j \\ -\beta * r(f_i, f_j), & \text{if } i \neq j \end{cases} \tag{5.2}$$

By introducing a series of binary variables x_a for $a = 1, 2, ..., |FS(e)|$ that indicate whether or not the feature f_a is selected in the optimal summary, the selection of $Summ(e, k)$ for summary length k maximizing the objectives outlined above can be defined as follows in terms of QKP formulation. $w(f)$ defines the weight of the feature f.

$$\begin{aligned} &maximize \ \ \Sigma_{a=1}^{|FS(e)|}\Sigma_{b=a}^{|FS(e)|} \ p_{f_a,f_b} * x_a * x_b \\ &where, \ \ \Sigma_{a=1}^{|FS(e)|} w(f_a) * x_a \leq k, \ x_a \in \{0, 1\} \end{aligned} \tag{5.3}$$

In the QKP, the algorithm optimizes selecting items that maximizes profit computed between items. In other words, it can be used to select features to the entity summary to get maximum profit by analyzing pairwise profit of the selected features. When using both positive and negative weights as shown in Equation 5.2, QKP is NP-Hard, that is, it does not have a polynomial-time algorithm to generate solutions unless P = NP [Gallo et al. 1980]. Therefore, an approximation algorithm like GRASP can be used to compute a solution.

5.1.2.2 Selecting Features for Multiple Entities

The mapping of QKP above refers to creating entity summaries for individual entities. An extension of this to handle multiple entities with the addition of maximizing inter-entity relatedness of features is what we propose in our problem. To achieve this objective, we consider mapping this problem to an instance of QKP with multiple constraints, namely,s Quadratic Multidimensional Knapsack Problem (QMKP). Given a collection of entities $e_1, e_2, .., e_n$, features numbered $f_{i,1}$ to $f_{i,|FS(e_i)|}$ $\in FS(e_i)$, and random variables $x_{i,a}$ for $i = 1, 2, ..., n$ and $a = 1, 2, ..., |FS(e_i)|$ to denote whether or not a feature $f_{i,a}$ is selected for the best possible summary, the optimization goals can be formalized as follows.

$$maximize \ \ \Sigma_{i=1}^{n}\Sigma_{j=i}^{n}\Sigma_{a=1}^{|FS(e_i)|}\Sigma_{b=1}^{|FS(e_j)|} \ p_{f_{i,a},f_{j,b}} * x_{i,a} * x_{j,b}$$
$$where, \ \ \Sigma_{a=1}^{|FS(e_i)|} w(f_{i,a}) * x_{i,a} \leq k_i, \ x_{i,a} \in \{0,1\} \tag{5.4}$$

k_i is the capacity of knapsack belonging to entity e_i. $w(f_{i,a})$ is the weight of the a^{th} feature of e_i. Note that we have n constraints to satisfy (a knapsack for each entity).

In extending QKP, we adapted a memory-based GRASP [Yang et al. 2013] approach, to simply run with multiple constraints. The algorithm runs through several iterations, and in each iteration, it generates a random solution first based on a greedy ranking function and sampling from the candidate item set. The original GRASP algorithm proposes a greedy ranking function [Yang et al. 2013] and we modify it to bias the selection of features to also consider future candidate selection. Given the already selected feature set S and candidate feature f, the modified greedy ranking function $Gr(S, f)$ in the construction phase of the GRASP algorithm is as shown in Equation 5.5. The function w gives weight of each feature and $\tau, \phi \in$ [0,1]. The component related to τ considers the current feature against already selected items, and the component related to ϕ makes the algorithm to consider unselected items in scoring the current feature, making the initial selection of features in the algorithm less random.

$$Gr(S, f) \ = \frac{\Sigma_{i\in S}\Sigma_{j\in S, j\leq i} \ p_{i,j} + \tau\Sigma_{x\in S} \ p_{x,f} + \phi\Sigma_{x\notin S, x\neq j} \ p_{x,f} + p_{f,f}}{\Sigma_{y\in S\cup\{f\}} w(y)} \tag{5.5}$$

Then, in the local search phase, the memory-based GRASP algorithm tries to improve the solution by further maximizing the total profit by swapping selected items with items from the unselected item list. The total profit of the selected items in the summary is calculated by $\Sigma_{i\in S}\Sigma_{j\in S, j\leq i} \ p_{i,j}$.

Since we have more than one entity to consider in the optimization approach, the profit computation is updated to reflect this need as shown in Equation 5.6 below. In the equation, $\alpha, \beta, \gamma > 0$ and are chosen empirically (tuned). The diagonal of the profit matrix contains the ranking scores (signifying the importance of each feature) and non-diagonal entries contain pairwise relatedness of

features. We make profits negative for feature pairs belonging to the same entity so that highly similar feature pairs will not be selected for the same entity.

$$p_{f_{i,a},f_{j,b}} = \begin{cases} \alpha * rank(f_{i,a}), & \text{if } i = j \text{ and } a = b \\ -\beta * r(f_{i,a}, f_{j,b}), & \text{if } i = j \text{ and } a \neq b \\ \gamma * r(f_{i,a}, f_{j,b}), & \text{if } i \neq j \end{cases} \tag{5.6}$$

5.1.2.3 Importance, Relatedness and Diversity

Note that we want diverse features to be selected in each entity summary and related features among entities. Further, we do not want arbitrary features to be selected for the summaries but be influenced by their importance. We try to combine these characteristics as shown in Equation 5.6.

Importance of a Feature

The diagonal of the profit matrix has the importance score for each feature f calculated by $rank(f)$ as shown in Equation 5.9. We rank features based on how informative the property-value pairs are and how frequent the values are [Cheng et al. 2011; Gunaratna et al. 2015]. We try to achieve a trade-off between the two measures similar to tf-idf score in Information Retrieval. $Inf(f)$ computes the inverse logarithmic feature frequency as shown in Equation 5.7 where N is the total number of entities in the knowledge graph G. The popularity (i.e., frequency) of value v of the feature f is computed by Equation 5.8. $Prop(f)$ and $Val(f)$ are two functions that return the property and the value of the feature f. Function $rank(f)$ facilitates selection of important features in the GRASP based summary generation as it can add higher profits for some features which are considered to be important in addition to the pairwise feature profit computed based on relatedness. Note that Equations 5.7, 5.8, and 5.9 are the ones we used in FACES and FACES-E approaches in Chapter 3 and Chapter 4.

$$Inf(f) = log(\frac{N}{|\{e|f \in FS(e)\}|}) \tag{5.7}$$

$$Po(v) = log|\{triple\ t|\exists\ e, f : t\ \text{“}appears\ in\text{”}\ G\ and\ t \equiv (e\ Prop(f)\ Val(f))\ and\ Val(f) = v\}| \tag{5.8}$$

$$rank(f) = Inf(f)\ *\ Po(Val(f)) \tag{5.9}$$

Relatedness of a Feature Pair

We calculate the relatedness of a feature pair by utilizing two measures. First, we employ semantics based measurement to analyze the relatedness between two properties by computing the overlap of terms that represent the two properties. Second, we utilize a graph and co-occurrence based measure to compute relatedness between two values (entities), specifically, using a vector space model similar to word embedding for graphs.

For the *semantics based relatedness measure*, we process the property of each feature, with the help of a lexical database, namely WordNet. For a given feature f, we get its property name (label of the property URL) and retrieve hypernyms from the lexical database. We also pre-process them (e.g., remove camel-case and stop words). Then we combine all the extracted terms and original terms for property label of the feature f into a set S_f. Then the semantics based relatedness $SemRel_p(f_i, f_j)$ of the two features f_i, f_j is computed by getting the Jaccard co-efficient of the two sets of the features S_{f_i} and S_{f_j} as shown in Equation 5.10. We chose to get hypernyms from the lexical database instead of synonyms or hyponyms because we need to compute the relatedness instead of strong similarity.

$$SemRel_p(f_i, f_j) = \frac{|S_{f_i} \cap S_{f_j}|}{|S_{f_i} \cup S_{f_j}|} \tag{5.10}$$

We consider a *co-occurrence based relatedness* to be computed between values of the features.

Similar to word embedding models like Word2Vec, we utilize a graph based model called RDF2Vec [Ristoski and Paulheim 2016] for this purpose. The model was developed using path based co-occurrence and showed promising results in data mining and similarity computation applications [Ristoski and Paulheim 2016]. We employ a pre-trained model on DBpedia knowledge graph and compute cosine similarity of any given two entities over their vector representation as shown in Equation 5.11. Given two features $f_{i,a}$ and $f_{j,b}$ belonging to entities e_i and e_j and their corresponding vector representation of their values ($Val(f_{i,a})$ and $Val(f_{j,b})$) shown as $\vec{Val}(f_{i,a})$ and $\vec{Val}(f_{j,b})$, respectively, and the relatedness measure $r(f_{i,a}, f_{j,b})$ is defined as in Equation 5.12.

$$GraphRel_v(\vec{Val}(f_{i,a}), \vec{Val}(f_{j,b})) = \frac{\vec{Val}f_{i,a} \cdot \vec{Val}f_{j,b}}{|\vec{Val}f_{i,a}||\vec{Val}f_{j,b}|} \tag{5.11}$$

$$r(f_{i,a}, f_{j,b}) = \frac{SemRel_p(f_{i,a}, f_{j,b}) + GraphRel_v(\vec{Val}(f_{i,a}), \vec{Val}(f_{j,b}))}{2} \tag{5.12}$$

Improving Feature Diversity in Entities

We implemented the GRASP algorithm presented by [Yang et al. 2013] and adapted it to fit our problem solution. The GRASP approach constructs random solutions and then improves upon them in the local search phase, in several iterations and returns the best result found so far based on the total profit. Recall that one of our objectives in this problem is to improve diversity of features selected for each entity. In order to achieve that, in addition to the introduction of negative profits, we make changes in the candidate feature selection step. The GRASP approach keeps a *candidate set* and *remaining set* of features for the collection of entities. The remaining set contains all the unselected features from the entity collection and candidate set is a random sample of this set. By introducing a threshold value η, we filter out features belonging to the remaining set where their maximum pairwise profit value with already selected feature is greater than η. That is, for a candidate feature f and the set of selected features S, we filter out f if $max(r(f, f_{i \in S})) > \eta$ where f,$f_i \in FS(e)$. With this modification, we are able to improve diversity in the results for each entity

by forcing the combinatorial optimization algorithm to not access similar features that have already been selected.

5.2 Evaluation

We discuss the details of our experimental settings and results below.

5.2.1 Implementation Details and Algorithm Settings

In our implementation of memory-based GRASP algorithm, we set $\gamma, \beta, \lambda, \sigma$ to 1, 3, 5, and 5, respectively. These values are suggestions from the authors of GRASP. We normalized profit values by dividing them using the maximum profit. We also added average similarity between the value of each feature and the entity collection to the diagonal of the profit matrix (to improve relatedness). In the greedy ranking function shown in Equation 5.5, we set $\tau = 1$ and $\phi = 0.5$. In the profit matrix, we used $\alpha = 2$, $\beta = 1$, and $\gamma = 1.5$. We set the threshold $\eta = 0.45$. The parameter values in the greedy ranking function and profit computation needed to be tuned for this task. We used a separate document sample to determine these values. Further, in this implementation, we consider feature weights to be uniform and equal to 1. Therefore, the length of the summary for each entity denotes the knapsack size for that entity. We used DBpedia (version 2016-04) encyclopedic dataset as our knowledge graph to retrieve entity descriptions and ran the RDF2Vec model on it. For the semantic relatedness measure, we used the WordNet lexical database.

Question	Wikinews					AQUAINT				
	Response: Mean (SD)			F(2,357)	LSD post-hoc	Response: Mean (SD)			F(2,147)	LSD post-hoc
	REMES	FACES	RELIN	(p-value)	(p <0.05)	REMES	FACES	RELIN	(p-value)	(p <0.05)
Q1: Summaries assisted me to get some relationships between the entities in the entity collection.	3.98 (1.16)	3.66 (1.08)	2.78 (1.18)	35.798 (6.772e-15)	REMES >FACES >RELIN	4.50 (0.65)	3.92 (0.92)	3.06 (1.13)	30.866 (6.427e-12)	REMES >FACES >RELIN
Q2: The facts in each summary are diverse.	4.12 (0.93)	3.93 (1.08)	3.79 (1.28)	2.747 (6.500e-2)	REMES >RELIN	4.26 (1.01)	3.98 (0.89)	3.22 (1.36)	11.879 (1.700e-5)	REMES, FACES >RELIN
Q3: The summaries helped me to better understand the document.	3.69 (0.71)	3.38 (1.41)	2.84 (0.71)	17.868 (4.022e-8)	REMES >FACES >RELIN	4.36 (0.60)	3.76 (0.96)	2.92 (1.32)	25.927 (2.267e-10)	REMES >FACES >RELIN
Q4: The summaries provide me an overview of the entire entity collection.	3.78 (1.07)	3.48 (1.07)	2.91 (1.20)	19.148 (1.260e-8)	REMES >FACES >RELIN	4.26 (0.63)	3.74 (0.80)	2.88 (1.19)	30.123 (1.086e-11)	REMES >FACES >RELIN
Q5: I like the summaries generated.	4.05 (0.89)	3.72 (0.91)	3.18 (1.20)	22.586 (5.805e-10)	REMES >FACES >RELIN	4.22 (0.68)	3.32 (1.04)	2.54 (1.20)	35.611 (2.447e-13)	REMES >FACES >RELIN

Table 5.1: Evaluating system summaries using questionnaire.

System	UCI		UMASS	
	Wikinews	AQUAINT	Wikinews	AQUAINT
REMES	**0.064**	**0.056**	**-0.301**	**-0.257**
FACES	-0.083	-0.259	-0.971	-0.428
RELIN	-0.221	-0.148	-0.984	-0.589

Table 5.2: Average coherency of different models

5.2.2 Datasets and Evaluation Setting

We evaluated REMES using qualitative and quantitative measures. For the qualitative evaluation, we requested a set of judges to rank systems on the Likert scale [1] 1 to 5 (1 for strongly disagree and 5 for strongly agree) for a given set of questions. For the quantitative evaluation, we evaluate the proposed approach against other systems for their level of relatedness. We use two document samples taken from two popular entity linking benchmark datasets: (i) Wikinews [2] (20 documents) and (ii) AQUAINT [3] (10 documents). We use object properties associated with the entities.

5.2.2.1 Qualitative Evaluation

We compared REMES with two state-of-the-art stand-alone entity summarization systems: FACES and RELIN. The goal of this evaluation is to measure how successful is each system in selecting summaries for each entity in a collection of entities to maximize inter-entity relatedness and intra-entity diversity and importance of features. We constructed 5 questions to evaluate on a Likert scale (1 strongly disagree and 5 strongly agree). We asked 13 judges to answer these questions for each dataset and each question had at least 5 different judges. The evaluation contains 850 question instances scored by the judges. The questions and the results are shown in Table 5.1. The proposed multi-entity summarization approach achieved higher mean scores (on the Likert scale) for all the questions used in the evaluation. We measured its statistical significance by first performing one-way

[1] `https://en.wikipedia.org/wiki/Likert\_scale`, accessed 04/10/2017

[2] `http://www.newsreader-project.eu/results/data/wikinews`, accessed 04/10/2017

[3] `http://www.nzdl.org/wikification/docs.html`, accessed 04/10/2017

ANOVA and then further investigating using Least Significant Difference (LSD) post-hoc analysis.

5.2.2.2 Quantitative Evaluation

To further evaluate the robustness of the proposed REMES model, we processed the summaries generated by the three systems and compared how effective they were in picking related features between entities. To measure the relatedness between features in the generated summaries, we measured semantic similarity of the entities (by processing their labels in the graph) in those features. In particular, we assessed the relatedness of these entities by employing two state-of-the-art NLP semantic similarity techniques, namely, UCI [Newman et al. 2010] and UMass [Mimno et al. 2011]. UCI was measured by a sliding window and the Point-wise Mutual Information (PMI) of all entity pairs. The entity co-occurrence counts were calculated utilizing a sliding window with the size 10. For every value pair the PMI is calculated on Wikipedia articles as shown in Equation 5.13.

$$UCI(W_i, W_j) = log\frac{p(W_i, W_j) + \epsilon}{p(W_i)p(W_j)} \tag{5.13}$$

where W_i, W_j are the labels of the entities e_i, e_j and the word probabilities (p(W)) are calculated by counting word co-occurrence in a sliding window over Wikipedia. On the other hand, UMass is measured based on document co-occurrence counts as shown in Equation 5.14.

$$UMass(W_i, W_j) = log\frac{D(W_i, W_j) + \epsilon}{D(W_i)} \tag{5.14}$$

where $D(W_i, W_j)$ counts the number of documents containing both W_i and W_j words and $D(W_i)$ counts the ones containing W_i, and ϵ is the smoothing factor. We used Palmetto [4] for measuring the UMass and UCI measures (using Wikipedia as the external corpus). Table 5.2 shows the semantic relatedness of the generated summaries for the three different systems based on the above metrics.

Summaries generated by REMES

Dmitry Medvedev	*Russia*
dbo:title-dbr:President_of_Russia	dbo:establishedEvent-dbr:Russian_Empire
dbo:otherParty-dbr:Communist_Party_of_the_Soviet_Union	**dbo:leaderName-dbr:Dmitry_Medvedev**
dbo:birthPlace-dbr:Saint_Petersburg	dbo:currency-dbr:Russian_ruble
dbo:predecessor-dbr:Valadimir_Putin	dbo:capitol-dbr:Moscow

Summaries generated by FACES

Dmitry Medvedev	*Russia*
dbo:title-dbr:President_of_Russia	dbo:establishedEvent-dbr:Russian_Empire
dbo:otherParty-dbr:Independent_(politician)	dbo:leaderName-dbr:Vladimir_Putin
dbo:almaMater-dbr:Saint_Petersburg_State_University	dbo:southwest-dbr:Black_Sea
dbo:deputy-dbr:Igor_Shuvalov	dbo:capitol-dbr:Moscow

Figure 5.2: Example entity summaries for two entities

5.2.3 Discussion

In the qualitative evaluation, REMES ranked higher than the other two systems for both the datasets, except for question 2, where p-value (0.18 for Wikinews and 0.20 for AQUAINT) was not significant enough to make a decision between multi-entity system and FACES. This is not totally unexpected because FACES system has shown superior capabilities in achieving diversity in generating entity summaries (by using a comprehensive hierarchical clustering approach). For all other questions, REMES outperformed the others and achieves higher mean scores, confirming its ability to generate summaries while maximizing inter-entity relatedness and intra-entity importance (and comparable to FACES in diversity). Figure 5.2 shows summaries generated for two entities using the REMES and the FACES systems. While REMES tries to make a connection between the entities (by selecting the leader for Russia), FACES could not get such relatedness. This is mainly because FACES cannot and does not consider other entities in the entity collection.

In the quantitative evaluation, we further confirmed that REMES generates summaries that maximizes relatedness of features for entity collections. We utilized an external knowledge source (Wikipedia) to capture relatedness of features selected for the summaries. The higher the semantic similarity score, the more related features are in the summaries generated for the entity groups. Clearly, the summaries generated by the proposed approach are more related according to both

[4] `http://aksw.org/Projects/Palmetto.html`, accessed 04/10/2017

measures that further confirms the achievement of our objective of creating relatedness based entity summaries for entity collections.

Finally, analyzing user feedback on the generated summaries, majority of the users (about 80%) mentioned that the multi-entity approach has the capability to pick related features in the summaries. We plan to further investigate how to select appropriate properties to improve the summaries.

Limitations

REMES approach depends on semantic relatedness measures (using RDF2Vec and hypernym-based computations) and optimization steps performed by GRASP algorithm. GRASP is an approximation to the optimal solution. The success of GRASP can be further evaluated and measured for improvements. Further, REMES only processed entity descriptions that come from the knowledge graph and no processing is done on the textual content surrounding the entities.

5.3 Conclusion

Summarizing a collection of entities is challenging since it involves processing all the entities in the collection simultaneously and in a coherent way. We proposed REMES approach to select related features among entities while keeping the diversity and saliency of features within each entity. REMES utilizes a graph-based RDF2Vec model to compute relatedness of two entities and semantic expansion based measure to compute relatedness of two properties. Further, we adapted a QMKP problem instance to match our task and used an implementation of a memory-based optimization algorithm called GRASP. The REMES approach has been evaluated against two state-of-the-art stand-alone entity summarization systems in two different settings: qualitative and quantitative. Extensive set of experiments using statistical tests (one-way ANOVA and LSD post-hoc) on two different datasets confirmed that REMES outperformed the others in generating high quality summaries for the collections of entities.

6

Enrichment and Usage of Structured Knowledge - two use-cases

The main focus of this dissertation is on entity summarization approaches that use structured knowledge. In this chapter, we discuss two illustrative applications of enriching and using structured knowledge: (i) identify equivalent properties between datasets (schema level) and (ii) use the structured knowledge to rank documents.

The two applications we discuss in this chapter can be used to support summarization. Identification of equivalent properties between linked datasets supports data integration. This can be used to integrate triples related to an entity scattered among datasets and then generate entity summaries. The second application in this chapter retrieves related documents. Documents are represented using extracted sets of triples. By using methods we discuss in this work, we can retrieve related sets of triples from a knowledge graph (representing a document corpus). Then we can summarize the retrieved triples (we have to adapt techniques presented earlier in this dissertation as they are

for handling features) to represent the original set of triples for efficient processing.

6.1 Identifying Equivalent Properties between Linked Datasets

We propose an approach to align properties between two different linked datasets [Gunaratna et al. 2013]. Our approach relies on utilizing the entity co-reference relationships (*ECR*) such as those formalized using *owl:sameAs* and *skos:exactMatch*. Our approach analyzes occurrences of equivalent subject and object values across datasets to align properties, e.g., given two matching subject and object pairs that are connected by *ECR* links, we check whether the associated property names have the potential to be equivalent. This is done in a robust manner by analyzing the aggregated statistical results related to matching subject-object pairs for a given pair of properties. Our work uses property extensions to glean equivalent properties that generalizes and improves upon the state-of-the-art. Using these results, we show how existing entity co-reference links between resources in LOD cloud can be used to align properties. The main contributions of this work are as follows:

- It introduces an efficient approach that utilizes property extensions and resource inter-links for property alignment, and
- It uses the notion of *Statistical Equivalence* to approximate *owl:equivalentProperty*.

6.1.1 Related Work

Property alignment, which is crucial for data integration tasks, has been addressed less often compared to concept and instance alignment. In contrast with instance and concept alignment [Shvaiko and Euzenat 2013], properties exhibit complex structure and meaning. Current techniques used for property alignment (including object) fall into three categories: (i) Syntactic/ dictionary-based, where predicate similarity is garnered via string matching or using WordNet, (ii) Schema dependent, and (iii) Schema independent, where instance level information is utilized. Some use a mix of these techniques and analyze properties of property alignment [Cheatham 2014].

Several efforts match data-type properties in ontologies [Nunes et al. 2013] [Tran et al. 2011]. [Nunes et al. 2013] utilized mutual information present at the instance level using genetic algorithms. They discuss matching of one to many complex relationships in different ontologies, but are limited to data-type properties. [Tran et al. 2011] discuss a cluster based similarity aggregation methodology for ontology matching where they focus on four different similarity measures to align properties. They calculate string, WordNet, profile, and instance similarities based on the domains and ranges of the properties. Even though [Tran et al. 2011] try to match object properties, they admit that the results are not strong enough to distinguish matching and non-matching property names. [Sleeman et al. 2012] incorporate a density estimation approach using Kernel Density Estimation (KDE) to map opaque properties (properties conveying the same meaning irrespective of their names) in ontologies. However, they transform values into numeric form to be compatible with KDE. This transformation is not easy in the LOD context where each instance contains many triples.

[Zhao and Ichise 2012] presented a graph based ontology analysis complementing their previous work related to building a mid-level ontology utilizing WordNet and string based similarity measures to group properties. The approach is not suitable for identifying equivalent properties since it groups any two related properties (using object value similarity) and does not take into account the effect of coincidental matches in initial grouping. SLINT [Nguyen et al. 2012] is an instance matching system that uses an IR based heuristic to calculate object values overlap. Both these approaches are coarse grained and hence not suitable for identifying equivalent properties as they aggregate properties on the overlap (not on the individual subject pairs): e.g., they can confuse conceptually different predicates “placeOfBirth” and “placeOfDeath”. These approaches are also different from ours in the sense that they use only object values and their overlap whereas we strictly try to match the property extensions minimizing false positives. TripleRank’s effort [Franz et al. 2009] in faceted browsing computes latent predicate similarity (i.e., similar properties within a dataset) indirectly as a byproduct of Singular Value Decomposition (SVD) and it is hard to verbalize the results in terms of extensions. Furthermore, it does not provide an evaluation on intra/inter dataset

property alignment in terms of precision and recall. The analysis of *owl:sameAs* networks and their implications for detecting schema-level inconsistencies and ontology alignment are discussed in [Ding et al. 2010]. Some of the alignment techniques and applications have used these networks showing their effectiveness in practice [Parundekar et al. 2012] [Zhao and Ichise 2012]. We also use a similar link traversal network model in our approach to match property extensions.

6.1.2 Approach

Property alignment is a non-trivial research problem in the Semantic Web domain. Progress on this problem can lead to significant advancements in data integration tasks. The objective of property alignment is to identify equivalent or sub-property relationships between a property pair P_1 and P_2, which may be in the same or different datasets. Since property names in different datasets have independent origin and relationships capture complex meaning in a triple [Sheth et al. 2004], calculating string similarity or synonym based measurements on property names alone does not suffice. To solve this problem, our "extensional" approach determines related properties (P) by finding similar triple patterns across datasets by matching subject (S) and object (O) values in triples of the form (SP_1O) and (SP_2O).

6.1.2.1 Property Alignment between Datasets

OWL [Dean et al. 2004] defines the concept of equivalent property (*owl: equivalentProperty*) as two properties having the same extension. For example, if property P is defined by triples $\{aPb, cPd, ePf\}$ and property Q is defined by triples $\{\ a\ Q\ b,\ c\ Q\ d,\ e\ Q\ f\ \}$, then they are equivalent properties because they have the same extension $\{\{a, b\},\ \{c, d\},\ \{e, f\}\}$. Since it is hard to expect exactly the same property extensions in real datasets, we approximate it by a significant overlap in matching subject-object pairs. For this purpose, we define *statistical equivalence* of properties on linked datasets. For example, if property P is defined by triples $\{\ a\ P\ b,\ c\ P\ d,\ e\ P\ f\ \}$ and property Q is defined by triples $\{\ a\ Q\ b,\ c\ Q\ d,\ g\ Q\ h\ \}$, then property extensions are not the same, but P and

Q have matching subject and object values two times out of three, providing statistical evidence in support of their equivalence. When we utilize evidence for extension matching, we need to overcome the potential problem of incorrect matches in complex data representation contexts.

We first determine the 'relatedness' between a property pair to decide a match, which also reduces the search space. Note that while SKOS [Miles and Bechhofer 2008] is a formal specification of concept relatedness in ontologies, there is no such specification for properties. Therefore, we now present some notions to help represent property alignment on linked data. We first define the notion of *candidate match* between two properties.

The following statement is true for all the definitions in this section. Let $S_1P_1O_1$ and $S_2P_2O_2$ be two triples in two different datasets D_1 and D_2, respectively, representing relations $P_1(S_1, O_1)$ and $P_2(S_2, O_2)$. *ECR* are the *entity co-reference links* described earlier.

Definition 7 ***candidate match*** *Let $S_1P_1O_1$ and $S_2P_2O_2$ be two triples. The two properties P_1 and P_2 are a candidate match iff $S_1 \overset{ECR^*}{\longleftrightarrow} S_2$ and $O_1 \overset{ECR^*}{\longleftrightarrow} O_2$. We say two instances are connected by an ECR^* link if there is a link path between the instances using ECR links (where, * is the Kleene star notation).*

Candidate matches can provide supportive evidence for property alignment. But there can be coincidental (spurious) matching of properties. Consider the following two triples in the datasets DBpedia(d) and Freebase(f):

d:Arthur Purdy Stout *d:place of birth* *d:New York City*

f:Arthur Purdy Stout *f:place of death* *f:New York City*

Arthur Purdy Stout is a person (in fact, a surgeon and pathologist in real life) who lived in New York City. Given that *d:Arthur Purdy Stout* is the same as *f:Arthur Purdy Stout* and *d:New York City* is the same as *f:New York City*, *d:place of birth* and *f:place of death* properties are a candidate match according to the definition. But clearly these two properties should not be treated

as equivalent because they have different intentional semantics. Therefore, this coincidental match is not an equivalent match.

To minimize mis-identification of coincidental matches as equivalent (ideally eliminating them), our approach aggregates additional evidence in support of a statistical match, to approximate equivalent match (defined formally using extensions). Therefore, we keep track of key statistical measures along with the candidate matches to compute statistical equivalence. For a candidate matching property pair (P_1,P_2), *Match Count* $\mu(P_1,P_2)$ and *Co-appearance Count* $\lambda(P_1,P_2)$ can be defined as follows.

Match Count $\mu(P_1,P_2)$ is the number of triple pairs for P_1 and P_2 that participate in candidate matches. That is,

$$\mu(P_1, P_2) = |\{S_1P_1O_1 \in D_1 \mid \exists\, S_2P_2O_2 \in D_2 \quad \wedge\ S_1 \overset{\text{ECR}^*}{\longleftrightarrow} S_2 \wedge O_1 \overset{\text{ECR}^*}{\longleftrightarrow} O_2\}| \tag{6.1}$$

Co-appearance Count $\lambda(P_1,P_2)$ is the number of triple pairs for P_1 and P_2 that have matching subjects. That is,

$$\lambda(P_1, P_2) = |\{S_1P_1O_1 \in D_1 \mid \exists\, S_2P_2O_2 \in D_2 \wedge S_1 \overset{\text{ECR}^*}{\longleftrightarrow} S_2\}| \tag{6.2}$$

Statistical equivalence in this work is measured by analyzing candidate matches over co-appearances of a property pair, which provides statistical evidence, i.e., it will have many matching subject-object pairs over common subjects. Therefore, the number of matching subject-object pairs in the property extensions and co-appearances of a property pair directly influence the decision function F (defined below) for selection (captured using a confidence threshold α). Also, a property pair must co-appear enough times (supporting evidence) to be picked as a match, to overcome coincidental matches, by achieving a sufficient match count μ. Therefore, this minimum number of match count μ should be greater than a constant k. This constant k filters out many incorrect random candidate matches. Now, statistically equivalent property pairs can be defined as follows.

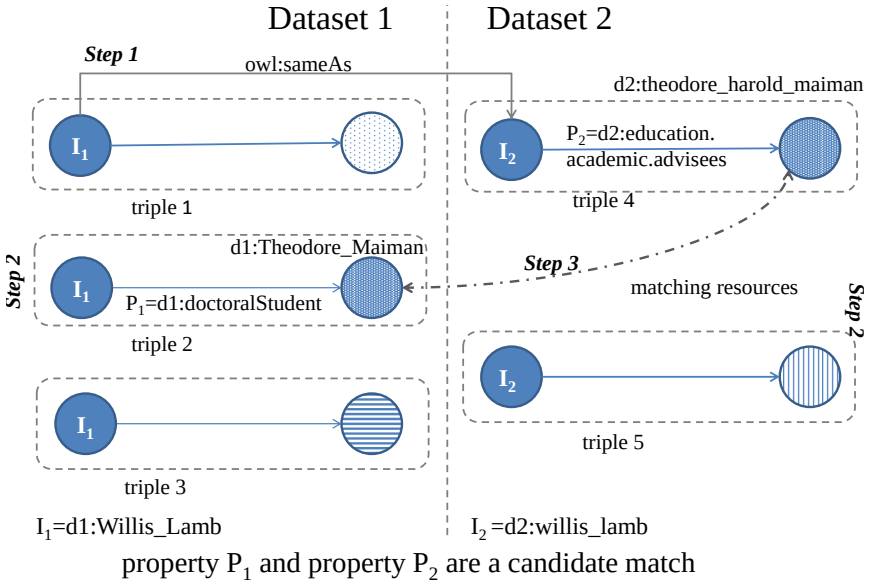


Figure 6.1: Process of Candidate Matching. Matching resources are in the same color/pattern.

Definition 8 ***statistically equivalent properties*** *The pair of properties P_1 and P_2 are statistically equivalent to degree (α, k) iff*

$$F = \frac{\mu(P_1, P_2)}{\lambda(P_1, P_2)} \geq \alpha \; and \; \mu(P_1, P_2) \geq k, \; for \quad 0 < \alpha \leq 1, \; k > 1 \tag{6.3}$$

Note that, a list of statistically equivalent properties can, strictly speaking, consist of equivalent properties, sub-properties and incorrectly mapped coincidental matches.

6.1.2.2 Resource Matching and Generating Property Alignments

Our algorithm is based on exploiting entity co-reference links that exist between instances in linked data for candidate matching of property pairs. This process is further illustrated in Figure 6.1 by matching an instance I_1 (Willis Lamb) with I_2 using *owl:sameAs* as the co-reference (ECR) link. This is achieved in three steps. In Step 1, the corresponding instance for I_1 is identified as I_2 by following an *owl:sameAs* link from I_1 to I_2. In Step 2, the subject instances are expanded using triples they consist of, i.e., dataset 1 has three triples for I_1 as the subject and dataset 2 has two

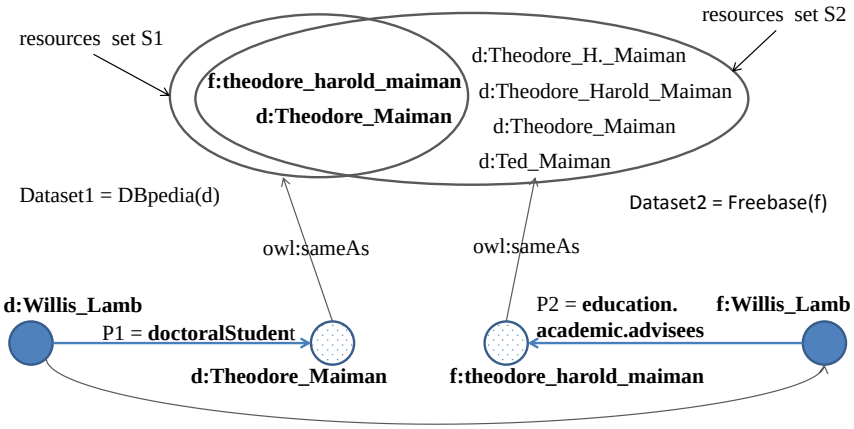


Figure 6.2: Property matching with overlapping sets of resources

triples for I_2 as the subject. In Step 3, since all subject values are matching for the triples of I_1 and I_2, finding two matching object values leads to a *candidate match*, i.e., in triples 2 and 4, object values can be matched by following an *owl:sameAs* link between them. Therefore both subject and object values match for triples 2 and 4 (also an extension match) leading to a *candidate match* for *doctoralStudent* and *education.academic.advisees* properties. This process is further outlined in Algorithm 3.

Comparison of the object values of triples is computed by checking for an overlap of ECR links, which includes checking for ECR^*. This is further illustrated in Figure 6.2 with only direct linking between them (one level deep path) for clarity. Further, this overlap based computation allows us to have link chains (ECR^*) to arrive (converge) at a decision. The process of candidate matching is presented in Algorithm 3 where the input to the procedure is a set X of subject instances of the first dataset and *namespace* identifier of the second dataset. The $GetCorsEntity$ procedure returns the relevant corresponding entity in the second dataset for a given subject instance in the first dataset (by following an ECR link and *namespace* identifier). The $ExtractPO$ procedure is for extracting all the property-object pairs for a given subject instance and returns a *map* data structure. The *map* data structure has object values stored for each property. The $GetECRLinks$ procedure returns all the available corresponding entities without considering any namespaces. If two

Algorithm 3 GenerateCandidateMatches(X, *namespace*)

Require: X, *namespace*

Ensure: λ, μ for each property pair

1: **for** $i = 1 \rightarrow Size(X)$ **do**
2: *subject* $S1 \leftarrow X[i]$
3: *subject* $S2 \leftarrow GetCorsEntity(subjectS1, namespace)$
4: *map* $l1 \leftarrow ExtractPO(S1)$
5: *map* $l2 \leftarrow ExtractPO(S2)$
6: **for** each *property* $p \in l1$ **do**
7: *object* $o1 \leftarrow l1(p)$
8: *Set* $o1\_ecr \leftarrow GetECRLinks(o1)$
9: **for** each *property* $q \in l2$ **do**
10: **if** p *exactmatch* q **then**
11: p *matches* q
12: **else**
13: *update* $\lambda(p, q)$
14: *object* $o2 \leftarrow l2(q)$
15: *Set* $o2\_ecr \leftarrow GetECRLinks(o2)$
16: $isMatch \leftarrow Match(o1\_ecr, o2\_ecr)$
17: **if** $isMatch$ **then**
18: *update* $\mu(p, q)$
19: **end if**
20: **end if**
21: **end for**
22: **end for**
23: **end for**

D1:Dataset 1	D2:Dataset 2	Matching Subject Pairs	Matching Object Pairs
S_1 doctoralStudent O_1	S_1' academic.advisees O_1'	(S_1 , S_1')	(O_1 , O_1')
S_1 birth_place O_2	S_1' place_of_birth O_3	(S_1 , S_1')	---
S_2 doctoralStudent O_4	S_2' influenced O_4'	(S_2 , S_2')	(O_4 , O_4')
S_2 birth_place O_5	S_2' place_of_birth O_5'	(S_2 , S_2')	(O_5 , O_5')
	S_2' place_of_death O_5'	(S_2 , S_2')	(O_5 , O_5')
S_3 doctoralStudent O_6	S_3' academic.advisees O_6'	(S_3 , S_3')	(O_6 , O_6')
	S_3' influenced O_7	(S_3 , S_3')	---
S_3 birth_place O_8	S_3' place_of_birth O_8'	(S_3 , S_3')	(O_8 , O_8')

Generated Candidate Matching Property Lists – Matches selected are in Boldface

[D1:doctoralStudent] ⟶ [**[D2:academic.advisees, 2:2]** , [D2:influenced, 1:2]] *list1*

[D1:birth_place] ⟶ [**[D2:place_of_birth, 2:3]** , [D2:place_of_death, 1:1]] *list2*

Property Pair	MatchCount	Co-appearanceCount
[D1:doctoralStudent, D2:academic.advisees]	2	2
[D1:birth_place, D2:place_of_birth]	2	3

Figure 6.3: Calculating *MatchCount* and *Co-appearanceCount* values

property names can be matched exactly (except namespaces), then they are considered as matched, as outlined in line 11 of the algorithm. As Algorithm 3 is performed on a sample instance set taken from dataset 1, candidate matches are found with μ and λ counts. Then applying function F with α and k in equation 6.3 for aggregated μ and λ for each property pair in the whole instance set will yield *statistically equivalent* property pairs for the two datasets. Note that $GetCorsEntity$ and $GetECRLinks$ procedures can be configured to use a specific ECR link or multiple types of ECR links (like *owl:sameAs, skos:exactMatch,* etc).

Figure 6.3 shows an example for calculating μ and λ. In this example, the algorithm used α=0.5 and k=2 to decide on matches from each list of candidate matching pairs and eliminate coincidental matches. Recall that α is the minimum fraction of matching triples over co-appearances and k is the minimum number of matching triples required to be considered for equivalence. The need for k together with α can be further explained using *list1* & *list2* in Figure 6.3, where "doctoralStudent" matched to "influenced" and "birth_place" matched to "place_of_death" satisfying α. The use of k avoids such coincidental (incorrect) matches. It also illustrates the nature of our bootstrapping

algorithm that decides on a matching property pair after analyzing the evidence from all matching subject-object pairs of the property extensions.

Complexity Analysis

Let the average number of properties for an entity be x and average number of objects for a property be j. Then, Algorithm 3 has to compare $j*j*x*x$ object value pairs for an entity pair. For n number of subjects, it would be $n*j*j*x*x$. To extract property-object pairs, it requires $2*n$ operations [1]. x and j are independent of n as number of properties per instance and number of object values per property do not change on average for larger or smaller n. Since $n > j$ and $n > x$, $O(n*j^2*x^2+2n)$ is n. The inherent parallel nature of Algorithm 3 allows us to adapt it to Map-Reduce paradigm to improve efficiency.

Implementation Details

The algorithm is implemented using Java, Apache Hadoop and Jena framework for processing RDF triples. Datasets are replicated locally using Virtuoso triple store except for Freebase dataset for which we used their public APIs.

In this experiment, we mainly used *owl:sameAs* and *skos: exactMatch* links as *ECR* (*owl:sameAs* was the dominant link of the two for the selected datasets). When searching for *ECR* links between entities, we investigated one level deep paths in this experiment (see Figure 6.2) since it was sufficient for the experimental datasets. But this can be extended further by examining more than one level deep link paths (ECR^*) for enhanced coverage. We also used exact matching of *rdfs:labels* of the two objects when comparing *ECR* links for object equivalence. This is used as an approximation to *ECR* links for the comparison of object values (line 16 of Algorithm 3) to improve coverage in absence of *ECR* links. This is because, we are looking for exact matching of labels for objects that belong to the same entity in two datasets. It is a good approximation as the subject (entity) is

[1] If comparisons are not restricted this way to each subject, a naive algorithm would need $n^2 * j^2 * x^2 + 2n$ comparisons.

guaranteed to be the same, since only *ECR* links are used for subject matching. This approximation also tries to compensate for sparseness of *ECR* links.

The process presented in Algorithm 3 requires a lot of comparisons and could grow for larger instance sets. But the algorithm was easily adapted to Map-Reduce framework by distributing subject instances among mappers. The specific implementation is explained below:

- **Map phase**
  - Let the number of subject instances in dataset 1 (D1) be X and namespace of dataset 2 (D2) be ns. For each subject $i \in X$, start a mapper and call GenerateCandidateMatches(i, ns) outlined in Algorithm 3.
  - Each mapper outputs (key,value) pairs as (p:q, μ(p,q):λ(p,q)) to reducer, where property p$\in$D1 and property q$\in$D2.
- **Reducer phase**
  - Collects output from all mappers and aggregates μ(p,q) and λ(p,q) values for each key p:q.

The process can be parallelized since computation of one subject instance is independent of the others. We implemented the algorithm in Hadoop Map-Reduce 1.0.3 framework and achieved significant improvements in running time on a 14 node cluster (in fact, a speedup of 833% compared to the desktop version on average). Moreover, we may achieve faster times when more resources are available to parallelize.

When the algorithm is run for a sample set of instances starting from dataset $D1$, we can generate candidate matches of property pairs. One property may be matched to many other properties because of similar extensions, as explained in Section 6.1.2.1. After recording candidate matches, each property in $D1$ has been mapped to a list of properties that are in dataset $D2$ with match counts (see Figure 6.3). The most probable matching properties will have higher match counts in the list and also have higher values for function F. We sort each list in descending order based on

match counts and then test the first property pair from each sorted list to see whether it satisfies threshold α and k for a match[2]. Applying these thresholds and sorting the lists remove many of the coincidental matches and retain probable matches. The algorithm outputs these as *statistically equivalent*. In this experiment, we were able to remove 76%, 87%, 67%, 83%, and 25% coincidental matches respectively from Experiments 1 to 5 [3]. These statistics also reveal that simple object value overlap or grouping mechanisms [Nguyen et al. 2012][Zhao and Ichise 2012] are not adequate for finding equivalent properties in the LOD context.

6.1.3 Evaluation

The objective of this evaluation is to show that *statistical equivalence* of properties can successfully approximate *owl:equivalentProperty* in the LOD context and the developed algorithm performs considerably better than the existing techniques used for property alignment, which includes string similarity and external hierarchies (i.e., WordNet) based approaches. To prove our claim, we chose DBpedia, Freebase[4], LinkedMDB[5], DBLP L3S[6], and DBLP RKB Explorer[7] datasets and extracted sample instance sets for the following reasons. (1) They have more or less complete information for instances. (2) They are well interlinked. (3) They have complex/opaque[8] properties. (4) They cover several dimensions of our evaluation as multi-domain to multi-domain, specific-domain to multi-domain and specific-domain to specific-domain dataset property alignment.

Our property alignment results are presented in Table 6.1. We randomly selected 5000 subject instances for each experiment, numbered 1 to 5 in Table 6.1. The experiments cover object properties in person, film, and software domains between DBpedia and Freebase, films between DBpedia and

[2]Assuming both datasets do not have duplicate properties. In our experiments, DBpedia had duplicate properties.

[3]From 732, 355, 221, 255 and 4 generated candidate property pairs respectively.

[4]http://freebase.com/

[5]http://linkedmdb.org/

[6]http://dblp.l3s.de/d2r/

[7]http://dblp.rkbexplorer.com

[8]Complex or opaque properties are semantically the same but have different word selections in describing the relationship.

LinkedMDB, and articles between DBLP RKB explorer and DBLP L3S datasets. DBLP RKB explorer and DBLP L3S are two separate datasets for DBLP that use two different ontologies. Furthermore, the DBLP RKB explorer project had different requirements, where it needed to achieve high precision and recall on search services compared to DBLP L3S dataset. DBpedia and Freebase are both multi-domain huge data hubs in LOD with overlapping information but have non-trivial differences in their schema. Freebase uses more blank nodes, increasing its complexity compared to DBpedia. LinkedMDB is a specialized dataset for movies and has its own schema which is completely different from DBpedia. Therefore, the datasets selected for the experiments from LOD are different from each other in both complexity and variety of data representation.

	Measure Type	DBpedia-Freebase (Person)[1]	DBpedia-Freebase (Film)[2]	DBpedia-Freebase (Software)[3]	DBpedia-LinkedMDB (Film)[4]	DBLP_RKB-DBLP_L3S (Article)[5]	Average
Our Algorithm	Precision	**0.8758**	**0.9737**	0.6478	0.7560	**1.0000**	**0.8427**
	Recall	**0.8089***	**0.5138**	**0.4339**	**0.8157**	**1.0000**	**0.7145**
	F measure	**0.8410***	**0.6727**	**0.5197**	**0.7848**	**1.0000**	**0.7656**
Dice Similarity	Precision	0.8064	0.9666	0.7659	**1.0000**	0.0000	0.7078
	Recall	0.4777*	0.4027	0.3396	0.3421	0.0000	0.3124
	F measure	0.6000*	0.5686	0.4705	0.5098	0.0000	0.4298
Jaro Similarity	Precision	0.6774	0.8809	**0.7755**	0.9411	0.0000	0.6550
	Recall	0.5350*	**0.5138**	0.3584	0.4210	0.0000	0.3656
	F measure	0.5978*	0.6491	0.4903	0.5818	0.0000	0.4638
WordNet Similarity	Precision	0.5200	0.8620	0.7619	0.8823	**1.0000**	0.8052
	Recall	0.4140*	0.3472	0.3018	0.3947	0.3333	0.3582
	F measure	0.4609*	0.4950	0.4324	0.5454	0.5000	0.4867

Table 6.1: Alignment results of object properties. Experiments are numbered 1 to 5.

Deciding α and k Values

Estimating the values for α and k was done based on the following facts. (1) Data in different datasets on LOD are not complete and contain similar but not identical information. (2) Representation of these data is not uniform due to multiple authors and naming preferences. (3) A resource is not guaranteed to be linked to all matching resources. Because of these reasons, the same property in two different datasets cannot be expected to have similar values for μ and λ for a higher F value closer to 1. Therefore, based on the above observations and our empirical evaluation, a threshold

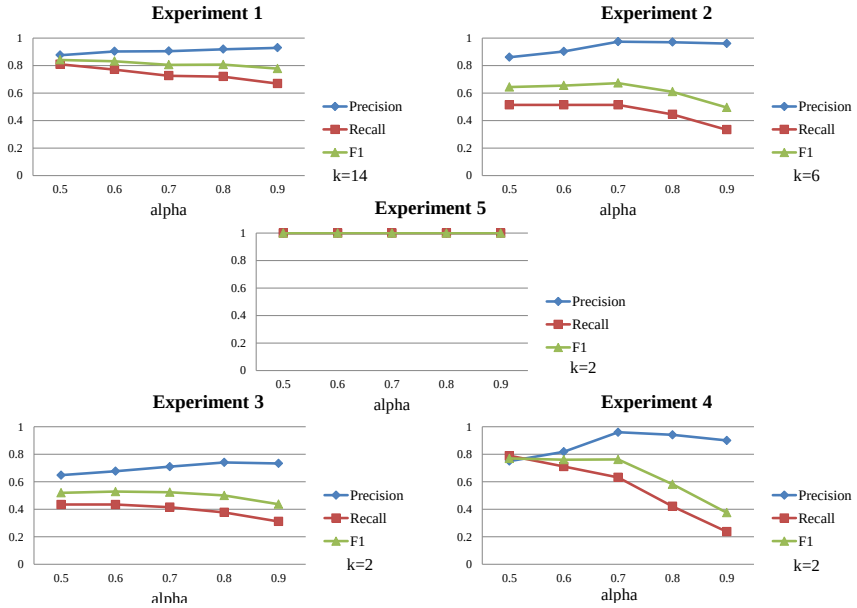


Figure 6.4: Precision, Recall and F measures for varying α values

value closer to 0.5 for α seemed to be appropriate considering F measure. Figure 6.4 further clarifies this claim in our empirical evaluation showing precision, recall, and F measure for varying α with the chosen k values. Results presented in Table 6.1 are using 0.5 for α except for experiment 2 where it is 0.7 for optimal F measure results.

The constant k is also affected by the above reasons and we followed a data driven approach to approximate a suitable value for k as follows. First the algorithm output is filtered using α=0.5 and k=2 which means, the lowest possible matching with a positive confidence level[9]. Then we get the μ values for property pairs matched using exact string matching, which are not identified by the algorithm and get the average of the counts as constant k. Our analysis over k suggests that most of the time, optimal performance for the algorithm is achieved by a value closer to this approximation of k (default set to 2). Following this approach, we used 14, 6, 2, 2, and 2 for k in the experiments numbered as 1, 2, 3, 4, and 5 respectively in Table 6.1. Increasing both these thresholds α and k yields high confidence matches as they demand a higher number of extension matching.

[9] α¡0.5 is a negative confidence level since less than half of subject-object pairs got matched from common subjects.

Experiment Settings

We compared our bootstrapping based algorithm with two string matching algorithms and WordNet similarity suggested by [Zhao and Ichise 2012]. We calculated WordNet similarity [10] by searching for at least one matching object value [Zhao and Ichise 2012] and then applying WordNet similarity on pre-processed terms of the properties. We tokenized and removed stop words from properties to calculate WordNet similarity and added Porter's stemming algorithm for string similarity. The results shown for these similarities are the optimal ones considering F measure metric. The threshold values used for string similarity and WordNet similarity algorithms were 0.92 and 0.80 respectively. For experiment 5, Dice and Jaro similarity didn't show any matching properties even for a threshold value of 0.5.

We used three independent reviewers to evaluate the matching of the properties and we chose the majority vote to determine the correct matches. The evaluators were given some sample matches (2-3), and if the meaning of the properties were hard to understand by their name (which is the case for most properties), they were provided with queries to execute and explore details about them, like instances they connect to and their domain and range. They were not asked specifically to distinguish between an exact match and a sub-property match in the alignment. For the experiment numbered 1 in Table 6.1, we could not find all possible matches because the total number of combinations is large for a manual evaluation($\sim$ 39k pairs). Therefore, we gave evaluators all the mappings found by the algorithm without any threshold applied (extracted from property lists as shown in Figure 6.3). Hence, recall and F measure are just approximations and they are marked with an * in Table 6.1 for experiment 1. Among the above mentioned comparable techniques, in every case, our bootstrapping based approach showed higher recall and F measures.

[10] http://www.sussex.ac.uk/Users/drh21/

Matching Category	**Dataset 1**	**Dataset 2**
String Similarity	db:nationality	fb:nationality
	db:religion	fb:religion
Synonymous	db:occupation	fb:profession
	db:battles	fb:participated_in_conflicts
Complex Matches	db:screenplay	fb:written_by
	db:doctoralStudents	fb:advisees

Table 6.2: Sample of matching properties under different categories. namespaces: db for DBpedia and fb for Freebase.

Types of Properties Identified

Our algorithm identified different kinds of matching property pairs. It can potentially subsume approaches that use techniques such as string similarity and synonym checking as shown in Table 6.2 using extensional matching. The sample pairs listed in Table 6.2 are all correctly identified by our algorithm. To explain example matching pairs in Table 6.2, consider the following observations.

"String Similarity Matches" can be identified using string similarity measures such as Dice's Coefficient or simple character comparison on property names. *"Synonymous Matches"* are mappings that can be identified by analyzing synonyms of the property names. For example, "occupation" and "profession" can be mapped using a dictionary, but interestingly the WordNet approach failed to match this pair for the provided threshold (showed very low similarity value of 0.5). In fact, both pairs were missed by the WordNet approach. *"Complex Matches"* are the last category shown in Table 6.2 and are harder to identify. All of the approaches outlined in the evaluation missed this mapping except our algorithm. This is because our approach exploits property extensions in aligning properties. There are also false positives in our result set. One such property pair is *http://dbpedia.org/property/issue* and *http://rdf.freebase.com/ns/people.person.children*, which happens to have similar extensions but has different intentions.

6.1.4 Discussion

The results show that our approach can effectively identify equivalent object properties by utilizing different statistical parameters presented above. It performs well even when complex properties exist, like in experiments 1 and 5 in terms of F measure metric. Our algorithm also performs well in terms of precision when datasets contain more literally similar property names as in experiments 2, 3, and 4, and discovers more interesting matches in every case, showing higher recall, e.g., experiment 5 is about aligning properties in the same domain (publication) with high overlapping information, but represents data using two completely different ontologies. These two ontologies do not have similar word selection or synonyms in their schema exemplifying complex data representation typically found on LOD. Because of this, string similarity based approaches do not perform well and synonym based approaches also show poor coverage over terms. Therefore, novel techniques such as ours for discovering equivalent properties on Linked Data (i.e., Complex Matches in Table 6.2) are indispensable [11].

We selected DBpedia, Freebase, LinkedMDB, and DBLP for our evaluation because of the existence of many entity co-reference links between their entities, specifically, DBpedia and Freebase provide diverse and complex property sets. Alignment between DBpedia and Freebase evaluates multi-domain property alignment whereas DBpedia and LinkedMDB covers multi-domain and specific-domain dataset alignment. The DBLP alignment evaluates our algorithm between specific-domain datasets. Therefore, our evaluation covers property alignment between different types of datasets that can arise in the LOD domain. When analyzing results of matching frequency for property pairs, it was observed that some matching properties do not appear frequently enough. Consequently, the algorithm's confidence in picking them as a match is low. This is mainly due to the nature of properties, as discussed in Section 3. When a random sample set is selected, it contains instances belonging to various types, e.g., a person type can have instances belonging to athlete, artist, etc., which have rare properties. We can run the algorithm iteratively for more

[11] More details can be found at `http://wiki.knoesis.org/index.php/Property_Alignment`, accessed 04/10/2017

subject instances that have these less frequent property pairs to improve precision. Our algorithm assumes that there are no duplicate properties in both datasets being aligned. If both datasets have duplicate properties, it requires additional inferencing mechanism to identify missing pairs, since only one matching pair from each candidate list is selected.

We observed that certain properties that are mapped as equivalent properties are actually sub-properties. This happens mainly because we do not distinguish between equivalent properties and sub-properties. For an example, *http://dbpedia.org/property/mother* and *http://rdf.freebase.com/ns /people.person.parents* are two such properties matched as equivalent, while, in fact, the first is a sub-property of the second. We would like to distinguish sub-properties in our future experiments for more fine grained matches. Also, we believe ECR^* links are useful in general, where many datasets are linked to central hubs like DBpedia and the algorithm will be able to discover more connections between resources by finding link paths via these hubs.

We believe our approach has resulted in high quality results compared to existing approaches because it, (1) does extension matching using entity co-reference links, (2) bootstraps from candidate matches and aggregates the results, and (3) makes final alignments using statistical measures analyzing aggregated confidence of the matching pairs. While, string similarity and synonym matching have been shown to be effective in the past (primarily for literals), they do not have sufficient coverage for resources as shown in our evaluation. We use *owl:sameAs* as an entity co-reference link but its use to link two semantically equivalent instances seems to be controversial in the LOD community [Halpin et al. 2010]. In spite of well-known shortcomings, (like equating "London" to "Greater London"), our algorithm is expected to be sufficiently robust for property alignment, because it is based on aggregating information from a large number of entity equivalence assertions. In other words, we believe that the effect of a few misused links will not affect the final result much, because our algorithm does not decide on a property match by analyzing a single matching triple. Even though the interlinking (entity co-reference) relationships are small compared to the size of similar instances between datasets today, we expect that these links will become prevalent as the

datasets evolve and are maintained as part of the "linked data life cycle" (with projects such as http://latc-project.eu/ and http://stack.lod2.eu/). Moreover, we can see other recent successful efforts in using *owl:sameAs* networks in the LOD context [Parundekar et al. 2012] [Zhao and Ichise 2012] in spite of these known issues.

We also identified some property pairs which are matched but have incorrect meaning in the property name. For example, *http://dbpedia.org/property/issue* and *http://rdf.freebase.com/ns/people.person.children* property pair between DBpedia and Freebase. This happened because of mixed values present in the DBpedia property *http://dbpedia.org/property/issue*, that has both integer values and names of children as object values. In this case, we can regard this kind of a property as ambiguous and having noise, which has a negative effect on the matching process. Another instance where the algorithm can go wrong is when it encounters special cases where the two datasets have enough facts for the process to identify two properties as a match, but actually they are not. For an example, for film domain, it maps *http://dbpedia.org/ontology/distributor* to *http://rdf.freebase.com/ns/film.film.production_companies*. This mainly happens because a number of production companies who produce also distribute films. Thus, they have multiple roles and the extensional equality may choose to match one of the many roles (extensions match but different intentions).

6.1.5 Future Work and Conclusion

We have developed and evaluated an extension-based approach to match object properties on linked datasets. We have defined a computable concept of *statistical equivalence* of properties to approximate *owl:equivalentProperty* by leveraging entity co-reference links. Our algorithm is unique and novel in how it computes extensional equivalence iteratively by building candidate matches in parallel. This approach ultimately determines statistically equivalent property pairs. The algorithm is easily parallelizable as evidenced by our straight forward Map-Reduce implementation. The empirical evaluation shows that our approach is superior to the state of the art using F measure metric (on

average, F measure gain is in the range 57% - 78%). This suggests that it is possible to align object properties on LOD datasets effectively. Our approach subsumes conclusions arrived at by string-based and WordNet-based similarity metrics and discovers more hidden and interesting matches well-suited for the LOD cloud. However, we can use string-based or WordNet-based approaches to further improve confidence in our results to minimize coincidental matches. In fact, our data driven approach can also be adapted to align data-type properties by using similarity metrics for object values in triples. Another beneficial side effect of object property alignment is that it may be used to generate or discover potential co-reference links between instances. Furthermore, we expect to test the algorithm on different levels (strength) of entity co-reference links. In future, we will extend the coverage to more types of data and discover domains in which properties exist for better alignment. The alignment problem can be further refined to determine sub-properties in the future that will help with fine grained data integration tasks.

6.2 Using Structured Knowledge for Document Similarity

Finding related documents is an interesting research problem in text and document retrieval. Keyword co-occurrence, matching combination of keywords, and cosine similarity of term vectors are some of the techniques used to match documents. In the simplest form, documents can be indexed using keywords and these keyword indices can be used to retrieve related documents, but this does not handle semantic similarity between documents. By semantic similarity, we mean matching that goes beyond lexical similarity computations like exact matching of keywords. Furthermore, keyword-based systems, including advanced systems like PubMed [12], can handle more than one keyword search query using keyword co-occurrence but we are interested in retrieval based on triples and not just concepts (i.e., keywords).

A semantically aware documents retrieval system can help a typical user who needs to get re-

[12] `https://www.ncbi.nlm.nih.gov/pubmed/`, accessed 04/10/2017

lated documents even when he is not completely sure of exactly what keywords or phrases to use for the search. Furthermore, if related documents can be fetched utilizing triples (i.e., semantic predications) which are extracted from document corpus, it can provide more precise and also semantically matched results. For example, a query to find documents that contain triples like "ASPIRIN TREATS HEADACHE" is expected to retrieve documents that have drugs related to ASPIRIN treating diseases like HEADACHE. This is different from a keyword-based search query where order of keywords does not matter and no semantic matching is performed.

We propose a document retrieval technique based on semantic matching of triples (here onwards referred to as *predications*) extracted from documents in the biomedical domain [Gunaratna 2016]. The Biomedical Knowledge Repository (BKR) [Bodenreider and Rindflesch 2006] is a repository of integrated biomedical data from literature, structured databases, and terminological knowledge sources like Unified Medical Language System (UMLS) [Bodenreider 2004]. BKR represents the integrated information in RDF using RDF triples. For example, the RDF triple (predicate) "lipoproteinx⇒ affects ⇒ inflammatory_cells" was extracted by a text mining tool called SemRep [Rindflesch and Fiszman 2003] from a MEDLINE journal article (with PubMed identifier PMID: 17209178) and states that lipoprotein (denoted as "subject" of the RDF triple) affects (denoted as "property" of the triple) inflammatory_cells (denoted as the "object" of the triple). Each document can be represented using a set of extracted predications like these. In this approach, we compute the similarity between sets of predications to derive the similarity between documents. The proposed approach enriches document retrieval by making it:

1. More precise - searches at the predication level rather than words.

2. More flexible - uses semantic similarity and hence covers more document matches than pure lexical similarity matches.

3. Semantically aware - our search mechanism can take into consideration the "context" in which the user searches related documents using sentences.

6.2.1 Related Work

Concept similarity computation is a popular topic in the biomedical research community. There exist several kinds of similarity measures in the literature that use distance between concepts (number of nodes/ edge counting), node features (e.g., number of shared ancestors), and information content (e.g., frequency of a concept in a given corpus) [Batet et al. 2011]. We consider using a simple, yet powerful measurement to capture ontological similarity of concept pairs. The use of a simple but effective similarity measure is needed because of the large number of similarity pair computations required in our application. [Batet et al. 2011] proposed a measure of dissimilarity based on the shared number of ancestors between two concepts. However, its value is not normalized, making it unsuitable for our application.

There are other systems that try to index and retrieve related predications. [Cohen et al. 2009] proposed an indexing mechanism for predications and a retrieval mechanism. Even though it has advanced retrieval capabilities like leaving part of a predication empty, it has no flexibility in matching related predications (i.e., no semantic similarity). TripleRank [Franz et al. 2009] is an authoritative ranking mechanism for triples based on the "popularity" of triples. It is related to our system as it ranks triples in a given context, but does not consider similarity or relatedness between triples. Our proposed approach is different from indexing systems and keyword based retrieval mechanisms as it consists of a flexible semantic matching component.

6.2.2 Approach

We are interested in computing similarity between documents using predications. A document can be represented as a set of predications. Furthermore, similarity between two sets of predications belonging to two documents can be used to compute the similarity between the two documents. Figure 6.5 shows how document - document similarity computation is decomposed into three stages of similarity computation.

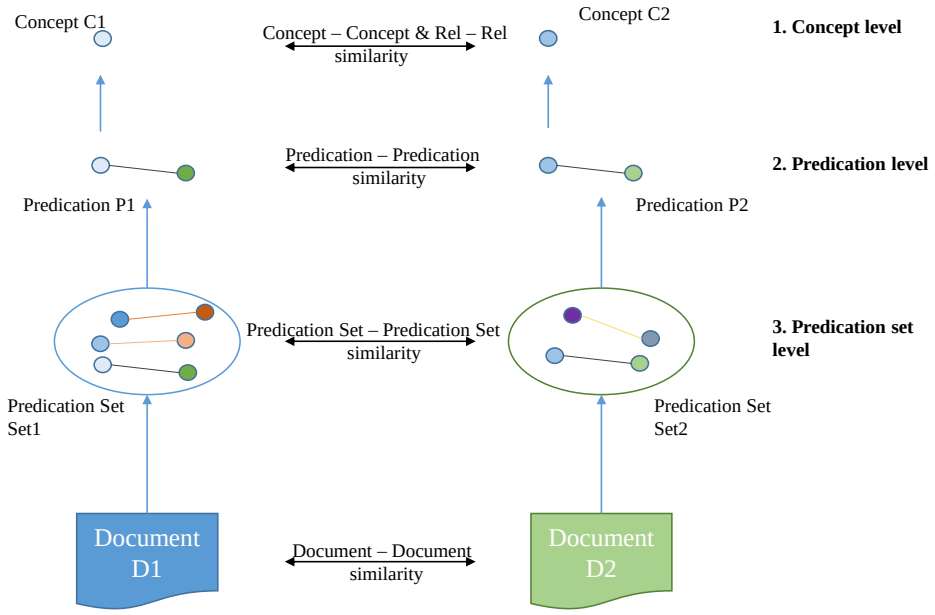


Figure 6.5: Overview of the approach.

1. Compute concept - concept and relationship-relationship similarity.

2. Compute predication - predication similarity.

3. Compute predication-set - predication-set similarity.

Predications in the BKR have concept (class) instances for subjects and objects. We are interested in finding out concept level similarity for predications and hence, we represent each subject and object of predications with its assigned concept.

Concept - Concept Similarity

Since there are many predications in the BKR, we try to use a simple similarity measure for concept concept similarity. The idea is to use the proportion of shared ancestors between two concepts as a measure of their similarity. We leverage hierarchical relations in the UMLS Metathesaurus, a terminology integration system, to compute the set of ancestors for each concept. We use the Jaccard coefficient to quantify the overlap between two sets of ancestors. The similarity can be

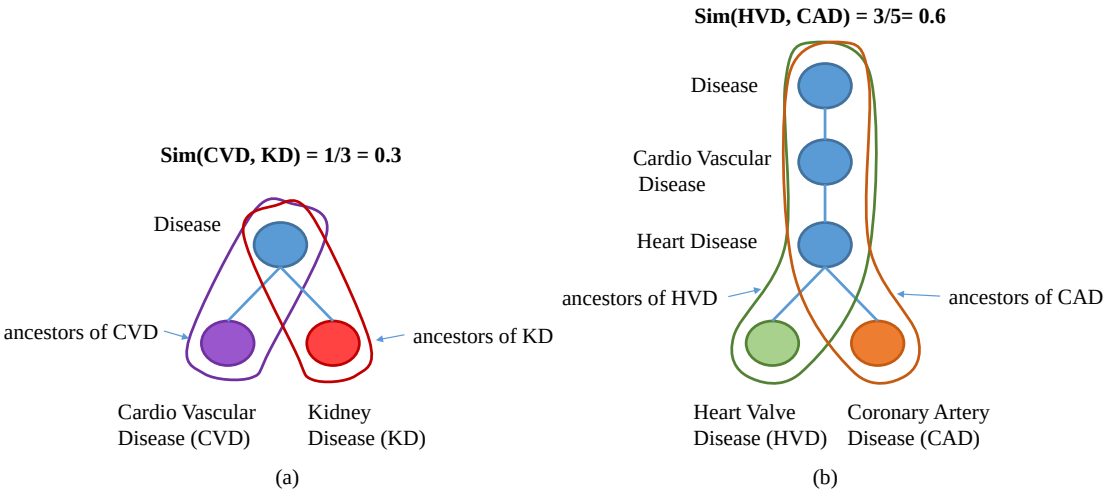


Figure 6.6: Jaccard similarity computation example for two concepts.

computed as shown in Equation 6.4. In this similarity computation, we add the concept itself to its set of ancestors in order to preserve high similarity for concepts that appear lower in the hierarchy. Figure 6.6 (a) and (b) show the behavior of Jaccard similarity computation for concepts. When concepts are higher in the concept hierarchy, they have very abstract meaning and hence, their similarity is expected to be lower (Figure 6.6 (a)) than the ones that appear lower in the hierarchy where they have very specific meaning (Figure 6.6 (b)). The Jaccard similarity value computed in this way varies between 0 and 1.

$$Jaccard(C_1, C_2)) = \frac{\#\ shared\ ancestors\ between\ C_1\ and\ C_2}{\#\ total\ concepts\ in\ C_1\ and\ C_2} \tag{6.4}$$

Predication - Predication Similarity

We compute the similarity between two predications as the average pairwise similarity of subject, predicate, and objects pairs. Similarity between a predication P_1 and predication P_2 denoted as $Sim(P_1, P_2)$ is computed as shown in Equation 6.5. W_s, W_r, and W_o are weights associated with similarity values between subjects, predicates, and objects, respectively. Since we get the average similarity over subject, predicate, and object pairs and each similarity value is between 0 and 1,

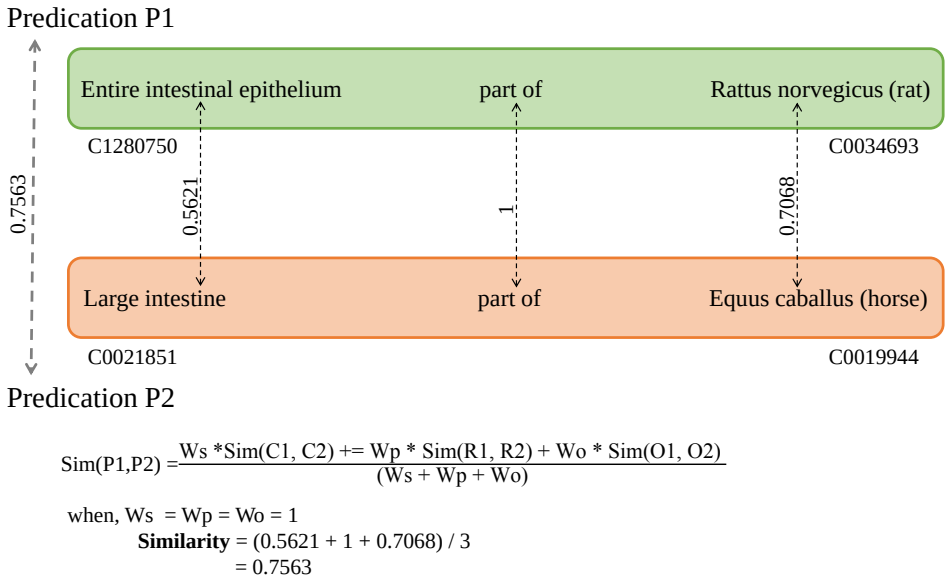


Figure 6.7: predication - predication similarity computation example for two predicates.

$Sim(P_1, P_2)$ is always between 0 and 1.

$$Sim(P_1, P_2)) = \frac{W_s * Sim(S_1, S_2) + W_r * Sim(R_1, R_2) + W_o * Sim(O_1, O_2)}{W_s + W_r + W_o} \quad (6.5)$$

An example similarity computation of two predications is shown in Figure 6.7. In the example, all weights are equal to 1.

Predication-set - Predication-set Similarity

Similarity between sets of predications is computed according to the formula shown in Equation 6.6. This is an accepted technique to compute the similarity between two sets [Azuaje et al. 2005] based on the similarity between their members. The intuition is to compute predication predication similarity between the two sets in both directions picking the maximum value for each predication and taking the average across the total number of predications.

$$Sim(SetS_1, SetS_2)) = \frac{1}{m+n} * (\Sigma_k \, max_p(Sim(P_k, P_p)) + \Sigma_p \, max_k(Sim(P_k, P_p))) \quad (6.6)$$

6.2.3 Evaluation

For our evaluation, we selected a subset of articles from MEDLINE and retrieved their predications from the 2013AB version of the BKR. Our evaluation is limited in size and only serves as a proof of concept for our proposed approach. The document sample for the evaluation is selected as follows. First, we randomly selected 30 documents from MEDLINE citations. Then, for each of these documents, we retrieved the top 30 related citations from the PubMed related citation search. Our document sample consists of 907 documents (when duplicate documents are removed).

Implementation Details

The prototype is developed using the Java programming language and we used the Virtuoso [13] triple store to store predications. Similarity values for concept and relationship pairs, predication pairs, and document pairs are stored in memory as key-value pairs using BerkeleyDB [14] database for rapid access.

Results

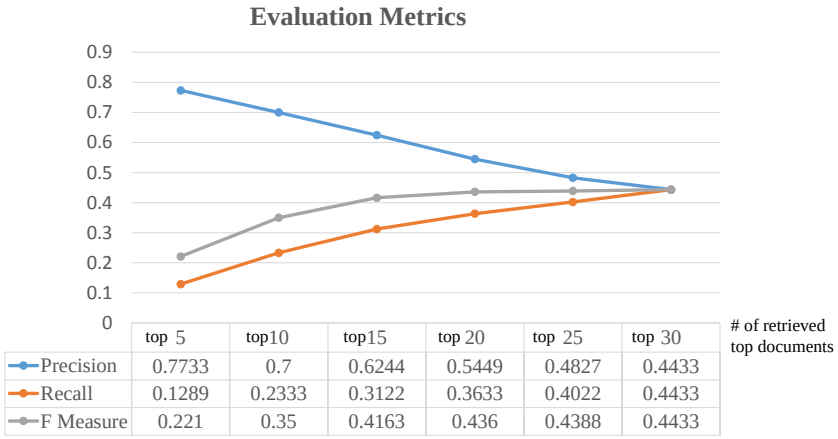
We measure precision, recall, and F-measure against the PubMed related citation gold standard. Computation of precision, recall, and F-measure are defined in Equations 6.7, 6.8, and 6.9, respectively.

$$precision = \frac{\#\ releveant\ documents\ \cap\ \#\ retrieved\ documents}{\#\ relevant\ documents} \tag{6.7}$$

$$recall = \frac{\#\ releveant\ documents\ \cap\ \#\ retrieved\ documents}{\#\ retrieved\ documents} \tag{6.8}$$

[13] `http://virtuoso.openlinksw.com/`, accessed 04/10/2017

[14] `http://www.oracle.com/technetwork/database/databasetechnologies/berkeleydb/overview/index.html`, accessed 04/10/2017



	top 5	top10	top15	top 20	top 25	top 30
Precision	0.7733	0.7	0.6244	0.5449	0.4827	0.4433
Recall	0.1289	0.2333	0.3122	0.3633	0.4022	0.4433
F Measure	0.221	0.35	0.4163	0.436	0.4388	0.4433

Figure 6.8: Results against the PubMed related citation gold standard using top n documents.

$$F\ Measure = 2 * \frac{precision\ *\ recall}{precision\ +\ recall} \tag{6.9}$$

The preliminary results against the PubMed related citation gold standard is shown in Figure 6.8. We measured precision, recall, and F-measure for the top n documents as shown in Figure 6.8. For the top 5 documents, 77% of the documents retrieved by our approach are relevant, demonstrating that it can retrieve top documents with high precision. The reason for lower recall is that we did not adjust weights in the predicationpredication similarity computation and prune similarity values that only matched part of the predications. Matching only part of the predications does not mean they are similar as they can be out of context.

6.2.4 Discussion

Our results showed promise for a new way of computing document similarity using semantic predications. Moreover, we can further improve the precision and recall as follows.

- Improving precision: We have not filtered out predication pairs that have very low similarity values. This artificially lowers the document - document similarity ranking. Introducing a threshold to filter out low similarity predication pairs can improve precision.

- Improving recall: We have not experimented with tuning weights for subject, predicate, and object pairs in predication predication similarity computation. Learning suitable weights for the related document retrieval use case can improve recall as it can remove noise in predication similarity computation.

Advantages

Our bag-of-predications approach has clear advantage over bag-of-words document retrieval systems as it is semantically-aware. It makes use of hierarchical relationships between concepts, as well as hierarchies of relationships, to measure concept similarity. Concept similarity adds flexibility in retrieving related documents, whereas bag-of-words approaches can only leverage word occurrence or co-occurrence (i.e., utilizing probabilities). Because it is based on predications, not keywords, our approach can also provide more precise results, as it captures the context of the query. For example, the predication "ASPIRIN TREATS HEADACHE" is not simply the concatenation of the three keywords "ASPIRIN", "TREATS", and "HEADACHE". It expresses the precise treatment relation between the drug and the disease. For this reason, our approach is more precise than traditional document retrieval models.

Furthermore, we could also use predication predication similarity to provide question answering or exploration capabilities to a user. For example, a user can ask a question like "give me related predications to ASPIRIN TREATS HEADACHE" or "find <what?> TREATS HEADACHE". In the first example, the user can explore related predications and in the second example, he can find out what drugs (or interventions) can treat headache. Further, this type of semantics-based ranking methods can be used in intelligent assistants [Nezhad et al. 2017] to make use of background knowledge.

Limitations

There are three different limitations with the current prototype. They are as follows.

1. Limitations with SemRep

   (a) SemRep uses a template-based predication extraction and hence it can miss extraction of some predications from articles.

   (b) Because SemRep does not handle co-reference resolution of named entities, it cannot extract predication across sentences and hence it can miss extraction of predications that span across sentences.

2. Limitations with similarity

   (a) ConceptConcept similarity needs to be evaluated in UMLS. The simple notion of shared proportion of ancestors between two concepts (measured with the Jaccard coefficient) has never been evaluated on UMLS hierarchies, even though it has been tested in other domain datasets like genes.

   (b) Weights of the predicationpredication similarity needs to be calibrated for better performance. As of now, we use the simplest representation of weights, with all weights equal to 1.

   (c) The scale of the current evaluation is small. Moreover, our evaluation is limited in scope (i.e., against PubMed related citation search results.)

3. The current prototype cannot handle the large number of similarity computations required to scale to the whole MEDLINE corpus.

6.2.5 Future Work and Conclusion

Future Work

We plan to evaluate our concept concept similarity metric on the UMLS, and to adjust and learn weights empirically for predication similarity. Also a large-scale evaluation with an independent test dataset (i.e., other than PubMed related citations) needs to be conducted. We also would like to

address the scalability issues related to computation by adapting cluster-based computations and use efficient data storage for large indices. It is worthwhile to investigate how to further improve precision and recall by adapting a threshold variable and weights adjustment.

Conclusion

We proposed a document retrieval approach that leverages semantic predications (in other terms, triples) extracted from these documents. We introduced the idea of using a bag-of-predications approach instead of bag-of-words approach for representing documents. We showed that our approach can provide precise and flexible results. It is also suggested that the outcome of predicationpredication similarity can be used for question answering and knowledgebase exploration purposes. With the suggested improvements in the near future, we believe that the proposed approach can make a significant real world impact by a use case implementation for document retrieval.

7

Conclusion and Future Work

This dissertation focuses on entity summarization at two different levels: (i) single entity summarization and (ii) multi-entity summarization.

7.1 Single Entity Summarization

We first proposed the FACES single entity summarization approach that considered diversity, importance, and uniqueness in selecting features for the summary. It makes use of schema level knowledge of the values of features (type assignments of the entities) and external lexical-database categorizations (hypernyms) to conceptually group features of a given entity. These groups are called facets and they are more semantically related than syntactically related compared to the state-of-the-art groupings available for triples/features. We adapted Cobweb clustering algorithm, which is conceptual, hierarchical, and incremental in nature, to create facets. These facets help FACES to create diverse entity summaries where the diversity comes at a conceptual level. For ranking the features, we used a tf-idf based ranking scheme that focused on informativeness of the feature and popularity (i.e. frequency) of the values of the features. We showed that FACES achieved excellent results compared to comparable state-of-the-art systems using object properties through a new gold standard dataset created using DBpedia and blinded user study.

One limitation that FACES has is that it can successfully group features belonging to object properties as they have types assigned to their values that FACES uses in creating facets. We extended FACES and proposed FACES-E that can group both object and datatype properties and select features for entity summaries. FACES-E computes types for literals whenever possible and enriches datatype properties with type semantics. We achieve this by: (i) exact and semantic matching of the focus term to class labels, and (ii) spotting entities related to the focus term and retrieving their types. This is a significant contribution in terms of making datatype properties inter-operable with object properties in knowledge graphs. Further, this can facilitate applications and use-cases like entity summarization, data integration, and dataset profiling. FACES-E improved upon the ranking criteria for features by extending ranking measures to both object and datatype properties and proposed to rank the facets before selecting the entity summaries. Special consideration was given in proposing feature ranking for datatype properties as their values are not represented by URIs. FACES-E was evaluated against comparable state-of-the-art systems by extending the gold standard used in FACES. The results confirmed that FACES-E could generate better quality entity summaries considering diversity, importance, and informativeness aspects for both object and datatype properties.

Broader Impact: The single entity summarization approach discussed in this dissertation (combining FACES and FACES-E) generates diverse and comprehensive summaries. These summaries can be used in applications like Web search to assist users to quickly understand the information contained in entities of interest. Commercial scale applications like Google Search and Bing already use entity summaries in their search interfaces and the work presented in this dissertation can be regarded as in the same level of interest. Further, the methods and ideas we presented to: (i) automatically and incrementally group features, (ii) type literals in knowledge graphs, and (iii) diversify entity summaries can be certainly used to complement existing techniques not limiting to entity summarization. For example, the conceptual grouping methods can be used to analyze and semantically group triples in datasets.

7.1.1 Future Work

We explored a simple tf-idf based ranking for FACES and FACES-E that could be complemented with complex graph-based ranking algorithms like PageRank and authority ranking measures for improved summary quality. The grouping algorithm adapted from Cobweb can work with streaming data. So, FACES and FACES-E can be used in streaming data environment to summarize dynamic entity facts like in news feeds. We have not specifically measured the grouping quality of the proposed approach and this can be further evaluated and improved, even to perform independently to entity summarization as a conceptual grouping framework for RDF triples.

We also proposed a component to compute types for literals in datatype properties in RDF datasets. Noise in these literals is a common problem and a way to filter out values that are noise is required. Further, some datatype properties are considered as labeling properties and they should not be typed (e.g., "familyName"). We manually filtered such properties in our experiments and an automatic way of identifying these properties will be a significant contribution not only to typing the literals but other data processing applications like dataset profiling and entity linking. We have not proposed a method to type numeric property values and also a proper ranking mechanism. Developing proper ranking measures for them can improve the summary quality and coverage of features.

7.2 Multi-Entity Summarization

Both FACES and FACES-E approaches deal with single entity summaries. That is, they cannot process more than one entity at a time to create the summaries. Sometimes, it is important to capture the context of a collection of entities in the entity summaries. Therefore, we proposed REMES that can process multiple entities and select related features to each other for entity summaries in addition to having diversity and importance of features preserved within each entity summary. Multi-entity summarization is computationally expensive as we have to deal with more than one

entity and probably belonging to different types. To achieve the aforementioned qualities in the entity summaries, we mapped the QMKP problem and adapted GRASP optimization algorithm. We modified how the pairwise profits are computed for feature pairs in QMKP and also modified some optimization steps to guarantee the intra-entity diversity. To measure the relatedness, we adapted RDF2Vec model and hypernym-based expansion proposed in FACES with Jaccard similarity. We compared the proposed approach with comparable single entity summarization approaches using a questionnaire based on Likert scale. The results showed that our approach performed better in creating multi-entity summaries.

Broader Impact: The relatedness-based multi entity summarization (REMES) approach discussed in this dissertation addressed the problem of generating multiple entity summaries considering a collection of entities, giving priority to intra-entity importance and diversity and inter-entity relatedness. This can be used to facilitate complex Web browsing needs such as analyzing and summarizing Web documents (and also multiple entity search queries) where multiple entities are found. Further, these methods could be used in information dissemination efforts (e.g., NEWS sites) to quickly analyze connections between the entities and also get quick understanding of them.

7.2.1 Future Work

Our multi-entity summarization approach mainly relies on the GRASP optimization algorithm (to solve QMKP) and relatedness and importance measures. Evaluating the performance of GRASP and further improving it to work for multi-entity summarization problem is a possible next step. Further, we used graph-based RDF2Vec model to get the entity-entity relatedness and we can try incorporating textual relatedness measures for improved summary quality. Because of the complexity involved in processing multiple entities, GRASP optimization algorithm provides reasonable summaries but the summary quality can be further improved when and if suitable processing techniques become available.

References

Antoniou, G. and Van Harmelen, F. 2004. *A semantic web primer*. MIT press.

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

Auer, S., Lehmann, J., and Hellmann, S. 2009. Linkedgeodata: Adding a spatial dimension to the web of data. In *Proceedings of the International Semantic Web Conference*. Springer, 731–746.

Auer, S., Lehmann, J., Ngomo, A.-C. N., and Zaveri, A. 2013. Introduction to linked data and its lifecycle on the web. In *Reasoning Web. Semantic Technologies for Intelligent Data Access*. Springer, 1–90.

Azuaje, F., Wang, H., and Bodenreider, O. 2005. Ontology-driven similarity approaches to supporting gene functional assessment. In *Proceedings of the ISMB'2005 SIG meeting on Bio-ontologies*. 9–10.

Batet, M., Sánchez, D., and Valls, A. 2011. An ontology-based measure to compute semantic similarity in biomedicine. *Journal of biomedical informatics 44,* 1, 118–125.

Berners-Lee, T. 1989. Information management: A proposal.

Berners-Lee, T. 2006. Linked data-design issues.

Berners-Lee, T., Fischetti, M., and Foreword By-Dertouzos, M. L. 1999. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. HarperInformation.

Berners-Lee, T., Hendler, J., Lassila, O., et al. 2001. The semantic web. *Scientific american 284,* 5, 28–37.

Bodenreider, O. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research 32,* suppl 1, D267–D270.

Bodenreider, O. and Rindflesch, T. 2006. Advanced library services: Developing a biomedical knowledge repository to support advanced information management applications. *Lister Hill National Center for Biomedical Communications, National Library of Medicine, Bethesda, Maryland*.

Brickley, D. and Guha, R. V. 2004. {RDF vocabulary description language 1.0: RDF schema}.

Cheatham, M. 2014. The properties of property alignment on the semantic web. Ph.D. thesis, Wright State University.

Cheng, G., Jin, C., and Qu, Y. 2016. Hieds: A generic and efficient approach to hierarchical dataset summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Cheng, G., Tran, T., and Qu, Y. 2011. Relin: relatedness and informativeness-based centrality for entity summarization. In *The Semantic Web–ISWC 2011*. Springer, 114–129.

Cheng, G., Xu, D., and Qu, Y. 2015a. C3d+ p: A summarization method for interactive entity resolution. *Web Semantics: Science, Services and Agents on the World Wide Web 35*, 203–213.

Cheng, G., Xu, D., and Qu, Y. 2015b. Summarizing entity descriptions for effective and efficient human-centered entity linking. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 184–194.

Cohen, T., Schvaneveldt, R. W., and Rindflesch, T. C. 2009. Predication-based semantic indexing: permutations as a means to encode predications in semantic space. In *Proceedings of the AMIA*.

Collins, M. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics 29,* 4, 589–637.

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. 2004. Owl web ontology language reference. *W3C Recommendation February 10*.

Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., and Kolari, P. 2005. Finding and ranking knowledge on the semantic web. In *The Semantic Web–ISWC 2005*. Springer, 156–170.

Ding, L., Shinavier, J., Shangguan, Z., and McGuinness, D. 2010. Sameas networks and beyond: analyzing deployment status and implications of owl: sameas in linked data. *The Semantic Web–ISWC 2010*, 145–160.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 601–610.

Ell, B., Vrandečić, D., and Simperl, E. 2011. Labels in the web of data. In *Proceedings of the International Semantic Web Conference*. Springer, 162–176.

Fang, Y., Si, L., Somasundaram, N., Al-Ansari, S., Yu, Z., and Xian, Y. 2010. Purdue at trec 2010 entity track: A probabilistic framework for matching types between candidate and target entities. In *Proceedings of the TREC*.

Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine learning 2,* 2, 139–172.

Franz, T., Schultz, A., Sizov, S., and Staab, S. 2009. Triplerank: Ranking semantic web data by tensor decomposition. In *The Semantic Web-ISWC 2009*. Springer, 213–228.

Freitas, A. and Curry, E. 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM, 279–288.

Freitas, A., Curry, E., Oliveira, J. G., and O'Riain, S. 2012. Querying heterogeneous datasets on the linked data web: challenges, approaches, and trends. *IEEE Internet Computing 16,* 1, 24–33.

Freitas, A., Sales, J. E., Handschuh, S., and Curry, E. 2015. How hard is this query? measuring the semantic complexity of schema-agnostic queries. *IWCS 2015*, 294.

Gallo, G., Hammer, P. L., and Simeone, B. 1980. Quadratic knapsack problems. In *Combinatorial optimization*. Springer, 132–149.

Gambhir, M. and Gupta, V. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review 47,* 1, 1–66.

Gennari, J. H., Langley, P., and Fisher, D. 1989. Models of incremental concept formation. *Artificial intelligence 40,* 1-3, 11–61.

Gruber, T. R. et al. 1993. A translation approach to portable ontology specifications. *Knowledge acquisition 5,* 2, 199–220.

Gunaratna, K. 2016. Document retrieval using predication similarity. *arXiv preprint arXiv:1604.05754*.

Gunaratna, K., Cheng, G., Thalhammer, A., and Liu, Q. 2016. Results of the 2016 entity summarization evaluation campaign (ensec 2016).

Gunaratna, K., Lalithsena, S., and Sheth, A. 2014. Alignment and dataset identification of linked data in semantic web. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4,* 2, 139–151.

Gunaratna, K., Thirunarayan, K., Jain, P., Sheth, A., and Wijeratne, S. 2013. A statistical and schema independent approach to identify equivalent properties on linked data. In *Proceedings of the 9th International Conference on Semantic Systems*. ACM, 33–40.

Gunaratna, K., Thirunarayan, K., and Sheth, A. 2015. Faces: Diversity-aware entity summarization using incremental hierarchical conceptual clustering. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Gunaratna, K., Thirunarayan, K., Sheth, A., and Cheng, G. 2016. Gleaning types for literals in rdf triples with application to entity summarization. In *Proceedings of the Extended Semantic Web Conference*. Springer, 85–100.

Gunaratna, K., Yazdavar, A. H., Thirunarayan, K., Sheth, A., and Cheng, G. 2017. Relatedness-based multi-entity summarization. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*.

Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. 2013. Evaluating entity linking with wikipedia. *Artificial intelligence 194*, 130–150.

Halpin, H., Hayes, P. J., McCusker, J. P., McGuinness, D. L., and Thompson, H. S. 2010. When owl: sameas isnt the same: An analysis of identity in linked data. In *The Semantic Web–ISWC 2010*. Springer, 305–320.

Han, L., Kashyap, A., Finin, T., Mayfield, J., and Weese, J. 2013. Umbc ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Vol. 1. 44–52.

Hassanzadeh, O. and Consens, M. P. 2009. Linked movie data base. In *Proceedings of the LDOW*.

Hitzler, P., Krotzsch, M., and Rudolph, S. 2009. *Foundations of semantic web technologies*. CRC Press.

Jones, K. S. 2007. Automatic summarising: The state of the art. *Information Processing & Management 43,* 6, 1449–1481.

Joshi, A. K., Hitzler, P., and Dong, G. 2013. Logical linked data compression. In *Proceedings of the Extended Semantic Web Conference*. Springer, 170–184.

Joshi, A. K., Jain, P., Hitzler, P., Yeh, P. Z., Verma, K., Sheth, A. P., and Damova, M. 2012. Alignment-based querying of linked open data. In *Proceedings of the OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 807–824.

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.

Mani, I. 2001. *Automatic summarization*. Vol. 3. John Benjamins Publishing.

Manola, F., Miller, E., McBride, B., et al. 2004. Rdf primer. *W3C recommendation 10,* 1-107, 6.

Mena, E., Kashyap, V., Sheth, A., and Illarramendi, A. 1996. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems*. IEEE, 14–25.

Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*. ACM, 1–8.

Michalski, R. S. 1980. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *Journal of Policy Analysis and Information Systems 4,* 3, 219–244.

Miles, A. and Bechhofer, S. 2008. Skos simple knowledge organization system reference. *W3C Recommendation*.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography 3,* 4, 235–244.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 262–272.

Nadeau, D. and Sekine, S. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes 30,* 1, 3–26.

Nenkova, A. and McKeown, K. 2012. A survey of text summarization techniques. In *Mining Text Data*. Springer, 43–76.

Newman, D., Lau, J. H., Grieser, K., and Baldwin, T. 2010. Automatic evaluation of topic coherence. In *Proceedings of the 2010 Conference of the North American Chapter of the ACL: Human Language Technologies*. ACL, 100–108.

Nezhad, H. R. M., Gunaratna, K., and Cappi, J. 2017. eassistant: Cognitive assistance for identification and auto-triage of actionable conversations. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 89–98.

Nguyen, K., Ichise, R., and Le, B. 2012. Slint: A schema-independent linked data interlinking system. *Ontology Matching*, 1.

Nunes, B. P., Mera, A., Casanova, M. A., Fetahu, B., Leme, L. A. P. P., and Dietze, S. 2013. Complex matching of rdf datatype properties. In *International Conference on Database and Expert Systems Applications*. Springer, 195–208.

Parundekar, R., Knoblock, C. A., and Ambite, J. L. 2012. Discovering concept coverings in ontologies of linked data sources. In *The Semantic Web–ISWC 2012*. Springer, 427–443.

Paulheim, H. and Bizer, C. 2013. Type inference on noisy rdf data. In *The Semantic Web–ISWC 2013*. Springer, 510–525.

Qian, R. 2013. Understand your world with bing. *Bing search blog, Mar*.

Rindflesch, T. C. and Fiszman, M. 2003. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of biomedical informatics 36,* 6, 462–477.

Ristoski, P. and Paulheim, H. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *Proceedings of the International Semantic Web Conference*. Springer, 498–514.

Semantic-Web. 2017. The semantic web. `https://www.w3.org/standards/semanticweb/#w3c_content_body`. [Online; accessed April-05-2017].

Shah, I. and Sheth, A. 1999. Infoharness: managing distributed, heterogeneous information. *IEEE Internet Computing 3,* 6, 18–28.

Sheth, A., Arpinar, I. B., and Kashyap, V. 2004. Relationships at the heart of semantic web: Modeling, discovering, and exploiting complex semantic relationships. In *Enhancing the Power of the Internet*. Springer, 63–94.

Sheth, A., Avant, D., and Bertram, C. 2001. System and method for creating a semantic web and its applications in browsing, searching, profiling, personalization and advertising. US Patent 6,311,194.

Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., and Warke, Y. 2002. Managing semantic content for the web. *IEEE Internet Computing 6,* 4, 80–87.

Sheth, A., Bertram, C., and Shah, K. 1999. Video anywhere: a system for searching and managing distributed heterogeneous video assets. *ACM SIGMOD Record 28,* 1, 104–114.

Sheth, A. and Thirunarayan, K. 2012. Semantics empowered web 3.0: managing enterprise, social, sensor, and cloud-based data and services for advanced applications. *Synthesis Lectures on Data Management 4,* 6, 1–175.

Sheth, A. P. 2000. Semantic web and information brokering: Opportunities, commercialization, and challenges.

Shvaiko, P. and Euzenat, J. 2013. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering 25,* 1, 158–176.

Singhal, A. 2012. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*.

Sleeman, J., Alonso, R., Li, H., Pope, A., and Badia, A. 2012. Opaque attribute alignment. In *Proceedings of the 28th IEEE International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 17–22.

Sleeman, J. and Finin, T. 2013. Type prediction for efficient coreference resolution in heterogeneous semantic graphs. In *Proceedings of the Seventh IEEE International Conference on Semantic Computing (ICSC)*. IEEE, 78–85.

Sydow, M., Pikuła, M., and Schenkel, R. 2013. The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *Journal of Intelligent Information Systems 41,* 2, 109–149.

Thalhammer, A., Knuth, M., and Sack, H. 2012. Evaluating entity summarization using a game-based ground truth. In *The Semantic Web–ISWC 2012*. Springer, 350–361.

Thalhammer, A., Lasierra, N., and Rettinger, A. 2016. Linksum: using link analysis to summarize entity data. In *Proceedings of the International Conference on Web Engineering*. Springer, 244–261.

Thalhammer, A. and Rettinger, A. 2014. Browsing dbpedia entities with summaries. In *The Semantic Web: ESWC 2014 Satellite Events*. Springer, 511–515.

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

Tran, Q., Ichise, R., and Ho, B. 2011. Cluster-based similarity aggregation for ontology matching. *Ontology Matching*, 142.

Tylenda, T., Sozio, M., and Weikum, G. 2011. Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery. In *Proceedings of the 20th international conference companion on World wide web*. ACM, 273–276.

Web-Statistics. 2017. The size of the world wide web (the internet). `http://www.worldwidewebsize.com/`. [Online; accessed April-05-2017].

Xu, D., Cheng, G., and Qu, Y. 2014. Facilitating human intervention in coreference resolution with comparative entity summaries. In *The Semantic Web: Trends and Challenges*. Springer, 535–549.

Yan, J., Wang, Y., Gao, M., and Zhou, A. 2016. Context-aware entity summarization. In *Proceedings of the International Conference on Web-Age Information Management*. Springer, 517–529.

Yang, Z., Wang, G., and Chu, F. 2013. An effective grasp and tabu search for the 0–1 quadratic knapsack problem. *Computers & Operations Research 40,* 5, 1176–1185.

Zhao, L. and Ichise, R. 2012. Graph-based ontology analysis in the linked open data. In *Proceedings of the 8th International Conference on Semantic Systems*. ACM, 56–63.