What Search Engines Can't Do.
Holistic Entity Search on Web Data

Von der Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von Silviu Homoceanu
geboren am 01.09.1982
in Bacau, Rumänien

Eingereicht am:
10.11.2014

Mündliche Prüfung am:
27.04.2015

Referent:
Prof. Dr. Wolf-Tilo Balke

Korreferent:
Prof. Dr. Ansgar Scherp

Technische Universität Braunschweig

2015

ABSTRACT

More than 50% of all Web queries are entity related. Users search either for entities or for entity information. Still, search engines do not accommodate entity-centric search very well. They only provide keyword search and return a list of links to articles. It falls on the users' shoulders to browse through the returned Web pages, and to pick the relevant information. In contrast, database systems master this task without any difficulties because the data is structured and integrated. Thinking along the same lines, the mainstream approach for accommodating entity search was to build a structured Web. Linked data and schema.org represent the most prominent initiatives for a structured Web. But our analysis shows that neither of the two can reliably serve the purpose of entity search. In our opinion, the problem of entity search on Web data cannot be solved by manually building a global schema. Instead, we propose data-driven approaches, dynamically extracting schemata shaped for each entity type. Obviously, these approaches have to be tailored for the various types of entity-centric queries.

Building on the concept of the *semiotic triangle* from cognitive psychology now established also in information theory, which models entity types in terms of intensions and extensions, we identified three types of queries for retrieving entities: *type-based queries* - searching for entities of a given type, *prototype-based queries* - searching for entities having certain properties, and *instance-based queries* - searching for entities being similar to a given entity. For *type-based queries* we present a method that combines query expansion with a self-supervised vocabulary learning technique built on both structured and unstructured data. Our approach is able to achieve a good tradeoff between precision and recall. For *prototype-based queries* we propose ProSWIP, a property-based system for retrieving entities from the Web. But since the number of properties given by the users can be quite small, ProSWIP relies on direct questions and user feedback to expand the set of properties to a set that captures the user's intentions correctly. Of course the number of questions has to be kept small. ProSWIP cleverly solves this problem with the help of information theory concepts choosing to ask user feedback on but a few properties showing the highest information gain. Our experiments show that within a maximum of four questions the system achieves perfect precision of the selected entities. In the case of *instance-based queries* the first challenge is to establish a query form that allows for disambiguating user intentions without putting too much cognitive pressure on the user. Learning from standard example-driven entity search tasks like *related entity finding* (REF) and *entity list completion* (ELC) we propose a minimalistic instance-based query comprising the example entity and intended entity type. With this query and building on the concept of *family resemblance* we present a practical way for retrieving entities directly from the Web. Our approach is able to retrieve semantically meaningful entities even for entity types, which have proven problematic for REF and ELC.

Providing information about a given entity, *entity summarization* is another kind of entity-centric query. Google's Knowledge Graph is the state of the art for this task. In our quest for enabling instance-based search we observed that entity types from manually curated knowledge bases like Wikipedia may group together heterogeneous entities. This is problematic especially for entity summarization systems, as the resulting entity overviews end up being too general. Unfortunately, Google's Knowledge Graph is also affected by this problem. Relying entirely on manually curated knowledge bases, this also excludes all new and less known entities. We propose not to rely on prearranged schemas, but to use a data-driven approach. Our approach intelligently blends the homogeneity/heterogeneity of entity types with schema integration techniques in the light of facts extracted directly from the Web. Our experiments on real-world entity classes representing different degrees of class homogeneity show that this approach is indeed superior to both, frequency-based statistical approaches and the Knowledge Graph, in terms of precision and recall.

Our results show that *type-based*, *prototype-based, instance-based,* and *entity summarization* queries can successfully be answered with data-driven approaches, and that the results are superior to corresponding state of the art methods. We are confident that mastering these four query types enables holistic entity search on Web data for the next generation of search engines.

ZUSAMMENFASSUNG

Mehr als 50% aller Web Suchanfragen sind entitätsbezogen. Benutzer suchen entweder nach Entitäten oder nach Entitätsinformationen. Dennoch werden entitätsbezogene Anfragen von Suchmaschinen nicht gut unterstützt. Sie stellen nur eine Liste von Links zu Artikeln bereit. Es ist die Aufgabe der Benutzer die Suchergebnisse nach relevanten Informationen zu durchsuchen. Im Gegensatz dazu beherrschen Datenbanksysteme solche Anfragen, da die Daten strukturiert und integriert vorliegen. Folglich war der Hauptansatz um Entitätssuche im Web zu unterstützen, ein strukturiertes Web zu bauen. Linked data und schema.org stellen die wichtigsten Versuche für den Bau eines strukturierten Webs dar. Allerdings zeigen unsere Analysen, dass keiner der beiden Ansätze zuverlässig das Entitätssuchproblem lösen kann. Unsere Meinung nach kann das Entitätssuchproblem auf Web Daten nicht durch das manuelle Erstellen eines globalen Schemas gelöst werden. Stattdessen stellen wir in dieser Arbeit datengetriebene Ansätze, welche entitätstypenangepasste Schemata dynamisch extrahieren vor. Offensichtlich müssen solche Ansätze an die verschiedenen Typen von entitätszentrischen Anfragen angepasst werden.

Aufbauend auf dem Konzept des *semiotischen Dreiecks* aus der kognitiven Psychologie, haben wir drei Anfragetypen zur Entitätssuche identifiziert: *typbasierte Anfragen* – Suche nach Entitäten eines gegebenen Typs, *prototypbasierte Anfragen* – Suche nach Entitäten mit bestimmten Eigenschaften, und *instanzbasierte Anfragen* – Suche nach Entitäten die ähnlich zu einer gegebene Entität sind. Für *typbasierte Anfragen* haben wir eine Methode entwickelt die *query expansion* mit einer *self-supervised vocabulary learning Technik* auf strukturierten und unstrukturierten Daten verbindet. Unser Ansatz liefert einen guten Kompromiss zwischen Precision und Recall. Für *prototypbasierte Anfragen* stellen wir ProSWIP vor. Dies ist ein eigenschaftsbasiertes System um Entitäten aus dem Web abzurufen. Da aber die Anzahl der Eigenschaften die durch die Benutzer bereitgestellt werden relativ klein sein kann, baut ProSWIP auf direkten Fragen und Benutzer Feedback um die Menge der Eigenschaften zu einer Menge welche die Intentionen der Benutzer korrekt erfasst zu erweitern. Man kann natürlich nicht erwarten das Benutzer bereit sind viele Fragen zu beantworten. ProSWIP lost dieses Problem mit Hilfe von informationstheoretischen Konzepten. Benutzerfeedback wird nur bei Eigenschaften mit dem größten Informationsgewinn verlangt. Unsere Experimente zeigen dass mit maximal vier Fragen eine perfekte Precision erreicht wird. In dem Fall von *instanzbasierten Anfragen* besteht die Schwierigkeit darin eine Anfrageform zu finden die die Benutzerintentionen eindeutig macht ohne dem Benutzer eine zu hohe kognitive Last aufzuerlegen. Wir stellen eine minimalistische instanzbasierte Anfrage, die aus einem Beispiel und dem entsprechenden Entitätstypen besteht vor, indem wir Erfahrungen aus standard Beispielgetriebenen Entitätssuchaufgaben wie related entity finding (REF) und entity list completion (ELC) benutzen. Mit Hilfe des Konzepts der *Familienähnlichkeit* entwickeln wir eine praktische Lösung um Entitäten mit Bezug zur der Anfragenentität direkt aus dem Web

abzurufen. Unser Ansatz erzielt sogar für Entitätstypen, die für REF und ELC problematisch waren, gute Ergebnisse.

Entitätszusammenfassung ist ein anderer Typ von entitätszentrischen Anfragen, der Informationen bezüglich einer Entität bereitstellt. Googles Knowledge Graph ist der Stand der Technik für solche Aufgaben. Auf dem Weg *instanzbasierte Anfragen* zu ermöglichen haben wir festgestellt dass Entitätstypen aus manuell erstellten Knowledgebases wie Wikipedia heterogen sein können. Besonders für Entitätszusammenfassungssysteme ist dies problematisch weil die erzielten Entitätsüberblicke zu allgemein werden. Leider ist auch Googles Knowledge Graph von diesem Problem betroffen. Das Zurückgreifen auf manuell erstellte Knowledgebases schließt alle neuen und weniger bekannten Entitäten aus. Wir schlagen vor sich nicht auf vorbestimmte Schemata zu verlassen, sondern datengetriebene Ansätze zu nutzen. Unser Ansatz verbindet die Homogenität/Heterogenität von Entitätstypen mit Schemaintegrationstechniken auf direkt aus dem Web extrahierten Fakten. Unsere Experimente auf Entitätsklassen die unterschiedliche Grade von Klassenhomogenität aufweisen, zeigen dass dieser Ansatz besser in Bezug auf Precision und Recall ist als frequenzbasierte statistische Ansätze und der Knowledge Graph.

Unsere Ergebnisse zeigen dass *typbasierte Anfragen*, *prototypbasierte Anfragen, instanzbasierte Anfragen* und *Entitätszusammenfassungsanfragen* erfolgreich durch datengetriebene Ansätze beantwortet werden können. Die Ergebnisse sind entsprechenden state-of-the-art Methoden überlegen. Wir sind überzeugt dass das Bewältigen dieser vier Anfragetypen eine holistische Entitätssuche auf Web Daten für die nächste Generation von Suchmaschinen ermöglicht.

Acknowledgements

First I would like to express my sincere appreciation to Prof. Dr. Wolf-Tilo Balke for his useful critiques, enthusiastic encouragement and patient guidance of this research work. He gave me the possibility to join one of the most ambitious institutes of this university, he introduced me to research in the field of information systems and he always took time for discussing new research ideas. He was a true mentor for me. For all this he has my deepest gratitude.

I got to know Prof. Dr. Ansgar Scherp through his valuable research work on linked data. In fact his analysis of the schema structures on the Linked Open Data cloud motivated me to analyze the feasibility of property-based entity retrieval from linked data, now an important part of this thesis. I would like to thank him for being my second examiner.

There were many interesting discussions and collaborations with my colleagues from the Institute of Information Systems at the Technical University of Braunschweig and the L3S Research Center. I would like to thank Joachim Selke for our interesting and useful discussions. Learning from his vast experience was extremely valuable for me especially during my early years. I would also like to thank Christoph Lofi for sharing his experience on paper writing with me. Furthermore, I would like to thank my colleagues, Sascha Tönnies, Benjamin Köhncke, Jose Maria Gonzales Pinto, Philipp Wille, Simon Barthel, Kinda El Maarry, Felix Geilert, Jan-Christoph Kalo, Michael Loster und Jewgeni Rose for the wonderful collaboration.

I would also like to extend my thanks to Thomas Mack for always ensuring the needed hardware and software support. I greatly appreciated his help since many of the experiments I performed required complicated setups which he quickly organized. Furthermore, I would like to thank Regine Dalkiran for giving me feedback on my writing on numerous occasions. Her valuable input helped me improve my writing skills in both German and English.

Finally, I would like to thank my family for their support and patience. I am especially grateful to my wife for supporting and encouraging me to continue my work even if this meant working longer hours or weekends.

Table of Contents

Chapter 1

Introduction

With the widespread use of the Web as primary information source, entity-centric search has become a common task for many people. For decades, Web search engines have been the de facto tools for searching the Web. *Searching for something? Google it!* Entity-centric search is no exception. In fact, according to studies performed on the Yahoo! query logs ([73, 74]), entity related search has already surpassed the mark of 50% of all Web queries.

Typical entity types in this context are ‘people’, ‘places’ or ‘products’. But generally speaking, an entity is defined as any existing or real thing. In consequence, things like ‘diabetes’, ‘hypertension’ or ‘the flu’ are entities just as ‘Angela Merkel’, ‘Berlin’ or an ‘iPhone 6’ cell phone are. Entities have various properties and corresponding property values. For instance, amongst many others, a person has properties like ‘gender’, ‘nationality’, or ‘profession’. Only this way, can abstract queries like retrieving all female German Chancellors be answered. Not only do entities show different properties, they are also categorized in classes, the entity types. Entity types span over various degrees of granularities. They range from all-encompassing types like ‘thing’ representing all possible entities, to basic entity types like ‘person’ or to more fine granular entity types like ‘female German Chancellors’, constructs obtained by fixating the values for one or more of the entity properties.

With entity data in mind, the functionality of entity search refers to methods which enable search engines to answer entity-related queries on Web data, by understanding and satisfying the user intent. Entity-related queries are in this case all queries on entities, properties or entity types. Entity-related data is abundantly available on the Web in three main forms of representation:

- Embedded in text documents
- Stored as facts in linked data stores
- Embedded in text documents and annotated with global semantic vocabularies

These three forms of data representation have emerged as a consequence of the purposes they were designed to serve. For instance text documents are the standard means for communicating information between people using the Web as a medium. Since the text documents were written for humans, entities are described with just enough information for people to understand the meaning. For this reason, this data representation form heavily relies on common knowledge and contextual information. For example when writing about a certain entity, like the new ‘G63 AMG’ car from Mercedes most properties of this entity are not required to be explicitly mentioned in the document. Since it is a car, it obviously has a body, an engine, wheels and all the other properties a car has. Only things that set it apart from other cars, like the fact that it has six wheels[1] have to be made explicit. But since most of the information is implied and has to be decoded by the reader with the help of background information, understanding and retrieving entity-meaningful information implied in text documents is a difficult task for machines.

Driven by the Semantic Web initiative to make entity data on the Web accessible to machines, a large amount of data has been published online in linked data stores. Building together the Linked Open Data (LOD) cloud, data stores contain billions of facts about entities. Facts are extracted from text documents with the help of information extraction techniques or produced and maintained by large data providers. Storing even the most basic facts, this is an important form of entity data representation for entity search because all information is made explicit. But the data is still difficult to query in integrated form, because each data store has its own vocabulary for structuring the data. This way, entities, properties or entity types may bear different names or identifiers in different data stores making it difficult to join facts from different sources.

To make all entity data available to machines in an integrated fashion, semantic vocabularies for annotating in-text entity data were proposed. Schema.org is the most popular collection of such entity annotation vocabularies. For documents annotated with schemata from schema.org, all annotated entity data is this way available in structured form and can theoretically be queried with structured query languages, like a huge Web database. Having the same structure, all data can be queried in an integrated form. Regarding implicit information, the main assumption is that even the most basic information is available somewhere on the Web, annotated with schema.org. Unfortunately, there is not nearly enough data annotated with such vocabularies. In fact as we will extensively discuss in Chapter 2, none of these data representation forms managed to facilitate entity search on Web data.

The problem of searching for entities on the Web is obviously important and neither the appearance of the LOD nor schema.org has solved it for now. Although a fully structured Web building on a global schema would definitely empower entity search, we believe that such an approach is not feasible. Instead, in this thesis we claim that independent of the way entity data is represented on the Web, entity search should consider how humans manage entity data. Perhaps the best example for this, is the way information is handled on an abstract, conceptual level. For instance, in terms of data modelling, during the conceptual modelling phase data engineers have a mental representation of ‘things’ they require for describing entities. Those ‘things’ are a generalization of the mind, for a category of real world entities. They are mere abstractizations, or concepts, as they are referred to in cognitive psychology [114]. In the context of information theory, a concept represents a set of entities of a similar kind. Such entities may be similar in terms of structure, i.e. sharing a certain set of attributes. The may also be similar in terms of attribute

[1] http://www.autocar.co.uk/car-review/mercedes-benz/g63-amg-6x6

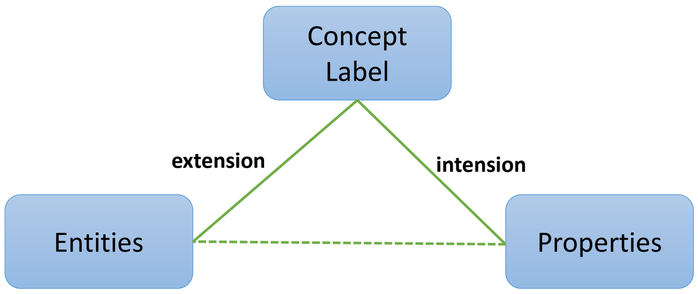


Fig. 1. Semiotic triangle.

values, i.e. sharing both attributes and the corresponding values. But a more elaborate view over the meaning of concepts is provided by linguists Ogden and Richards in [92]. In this work the authors introduce the *triangle of reference*. Also known as the *semiotic triangle* (see Fig. 1), this is now an established model also in information science [114]. In this model, concepts are represented in language through words or labels. They are defined by their *extensions* and *intensions*. The extension of a concept is the set of entities that fall under it while the intension determines the concept via its properties or attributes. Going back to entities, with the semiotic triangle in mind, the word or symbol is, in this case, the *entity type* (e.g. 'actor') that the user thinks of. For entity type 'actor' the *extension* would comprise all actors that ever existed and its *intension* could be 'person' and 'played in at least one movie'.

Considering the semiotic triangle as a model of how people handle entity data, when searching for entities, users are searching for the set of entities (the extension) of a certain entity type that they have in mind. In exploratory search, and especially in product search, this is the first step towards compiling a list of candidates to choose form (see [107] for details). The next step, also relevant for entity search, requires paying closer attention to some of these candidates. Entity summaries of a certain entity come in handy for this purpose. In order to get either the set of entities or the entity summary, users have to express their information need somehow. As a query, they may provide for instance the entity type; or they may provide an intensional description, a prototype of the expected entities; or some partial extension, instances, most probably well-chosen examples of entities that one is searching for. Taking all this into consideration, the following types of queries are relevant for, and should be supported by systems offering entity-centric search:

- **Entity type query. Input:** the entity type. Expected **output**: the extension; the set of entities of the given type. An example of such a query would be searching for all entities of type 'movie'. Search engines, which are the de facto information retrieval systems for the Web, were not built for, and are not able to accommodate such queries. A try on one of the leading search engines[2] on query 'movie' returns links for IMDb, YouTube, AppleTrailers, RottenTomatoes but not even one movie.

- **Prototype-based query. Input:** the intension; a set or attributes describing the entity type. Expected **output**: the extension; the set of entities of the *intended* type. Since it relies on attributes/properties to describe the entity type, this type of query is also called a prototype based query, and the properties are expected to offer a prototypical sketch of the expected entities. An example of such a query would be searching for all entities having 'title', 'year', 'director', 'actor' and 'genre' as attributes. Due to the co-occurrence of terms that have a high characteristic power for movie Web pages, for query *'title', 'year', 'director', 'actor', 'genre'* search engines are able to return two IMDb pages of two movies[3]. An important problem for this type of queries is the fact that providing a complete intensional description of the intended entity type is not a trivial task for the user. Instead, systems that support such a query type will have to take into consideration that the user may provide a subset of the intension, a small number of properties he/she is aware of.
- **Instance-based query. Input:** partial extension; a few examples of entities best representing what the user is searching for. Expected **output**: the extension; the set of entities of the same type as the example entities. An example of such a query would be searching for all entities similar to 'The Matrix', 'Avatar' and 'Inception' as examples. This kind of queries have been previously researched in the context of multimedia databases, e.g. query by humming for audio [45], or by sketch for images and videos [66, 90]. As expected also in this case, Web search doesn't provide any satisfying results. Major concerns for this query type are the fact that the examples have to be well chosen and that users will not be able to provide more than a handful of examples. For reasons of practicality one can expect that the user provides at best a maximum of five examples.
- **Entity summary. Input:** an entity. Expected **output**: an entity summary; the intension plus values for the attributes in the intension. Knowledge Graphs, introduced by Google and recently adopted by Bing, provide this kind of functionality. But they rely on manually curated knowledge bases that prevents them from accommodating many new or more obscure entities available on the Web.

[2] https://www.google.de/?gfe_rd=cr&ei=zdmeU66iKaWK8QeIqoHoBQ&gws_rd=ssl#q=movie

[3]https://www.google.de/?gfe_rd=cr&ei=zdmeU66iKaWK8QeIqoHoBQ&gws_rd=ssl#q=%E2%80%98title%E2%80%99%2C+%E2%80%98year%E2%80%99%2C+%E2%80%98director%E2%80%99%2C+%E2%80%98actor%E2%80%99%2C+%E2%80%98genre%E2%80%99+

Taking all this into consideration, our goal is to research data-driven methods, which independent of the data representation forms, are able to answer these typical entity search query types.

Boundaries of the thesis. Two types of entity related queries will not be handled in the course of this work: user/user group specific entity types that require personalization and entity related question answering.

All these query types are quite flexible, allowing users to express all kinds of entity types. For example one could search for clear-cut categorical entity types like 'movie' or 'actor'. Users can also search for entity types where the expected result set enjoys the consensual acceptance of the large majority of users e.g. 'science fiction movie'. But, as introduced in [114], one can also search for more individual, user/user group specific entity types like for instance 'good mood movie'. In the course of this thesis we will consider only entity types where there is consensus w.r.t. the expected result and user profiling or personalization is not required. Of course personalized entity search is very interesting for product search and it may be interesting for systems like Amazon's A9 product search engine. In fact, there is a compelling amount of research in the field of recommender systems ([85, 100, 101]) focusing on exactly these problems. But especially in a time when privacy concerns have reached critical levels and users are advised to use alternative methods (see https://duckduckgo.com/privacy for more information) to avoid profiling from Web search engines, this type of queries has lost, relevance for Web search.

The task of "Web-based Question Answering" where users search for a specific piece of information about a certain entity e.g. 'Amazon customer service phone number' is also not a subject of this thesis. In this field extensive research ([26, 27, 78, 134]) has been done. Systems like the well-known IBM Watson [120] stand as a proof of their success.

Thesis structure. In this thesis we propose a solution to entity-centric search on Web data. Our approach builds on the assumption that different types of entity-centric query types require different methods. In Fig. 2 we present a simplified view of our system for holistic entity-centric search. The heart of this system is represented by the four core components corresponding to the entity-centric query types. They rely on Web data and query type specific methods to enable entity-centric search. Discussed and evaluated individually, they are the central part of this thesis. In the middle, there is a basic component whose only purpose is to forward the query to the appropriate component according to the query type, and present the result back to the user.

The *entity type* query type for entity-centric search poses the least cognitive burden to users. One just needs to state the entity kind and the system will do the rest. But precisely because entity-type queries are concise, simple keywords, it makes the work of a system that supports them, much more difficult. While the information transmitted by such a keyword is easy to understand for people due to common sense and background knowledge, machines require complicated methods and large

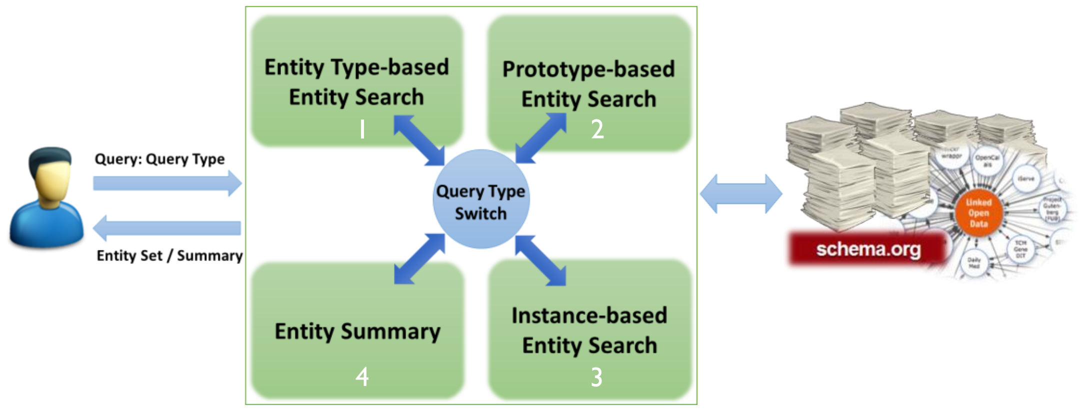


Fig. 2. System overview, with the four main components of the system being presented in detail in chapters 3 to 6.

amounts of data to mine this information from. The details on how to build such methods are presented in Chapter 3.

The *prototype-based* query type requires that the user provides a set of properties that intensionally define the intended type of entity. But, as we have found out in [59] users are rarely keen on providing more than four properties. This leads to the fact that in most cases, a system accepting this kind of query has to be able to work with only a subset of the actual intentional description. For instance, a user may provide 'Title', 'Director', 'Genre', 'Language' as an intension. But this description, is rather broad, as it may refer to movies, audio books or even video games. In Chapter 4 we present a method that is able to cope with such cases.

Also in the case of the *instance-based* query type, one cannot expect that the user will provide more than a few examples of entities to point out what his/her information needs are. But as we will discuss in more detail in Chapter 5, providing examples may leave room for ambiguity. For instance, with "Ronald Reagan" as a query entity and "Clint Eastwood" as additional example, the user will be referring to American actors rather than American presidents. However, he/she might also have more restricted entity types in mind like Western actors, actors from California, American actors with political ambitions, and so on. The more examples, the better a query can be disambiguated, however increasing query complexity.

Next, we present an approach for providing a data driven *entity summary* that is not limited to popular entities already present in manually curated databases. But what should such a summary comprise and how long should it be? The answer to these questions is presented in Chapter 6.

Finally, in Chapter 7 we present our conclusion on entity-centric search together with an outlook to future work.

But before proceeding to introduce our take on entity-centric search, in the next chapter we provide an extensive overview on existing approaches and efforts being made to enable entity search on the Web.

Chapter 2

Related Work

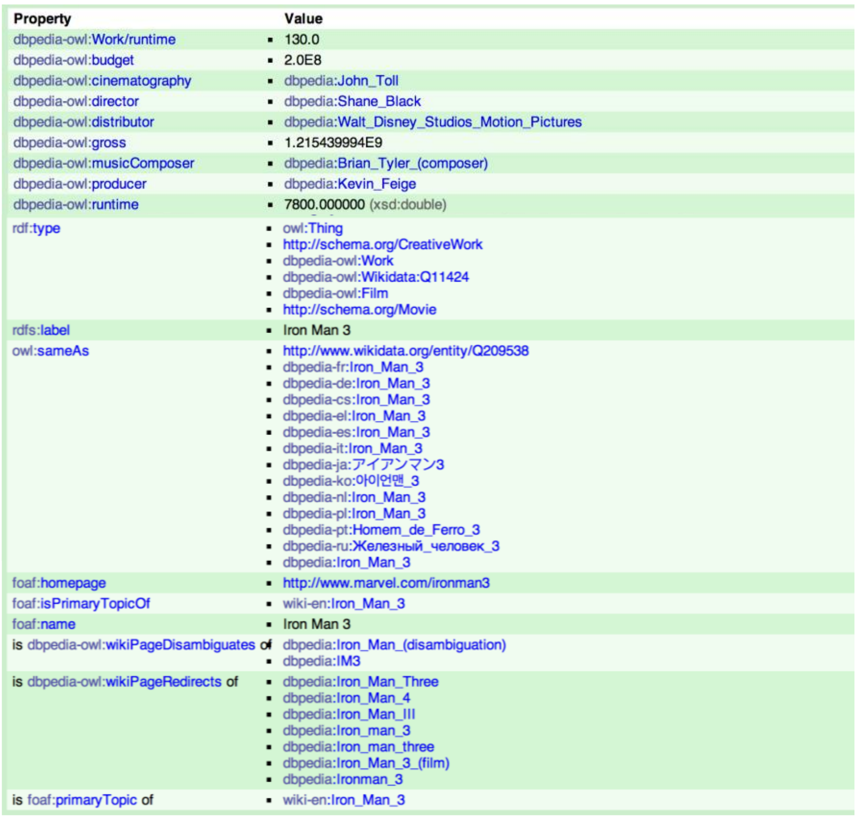
The storage, management, and retrieval of entity-related data has always been among the core applications of database management systems. Such systems can master entity queries because the underlying data is provided in structured form on a model that matches the query needs. However, entity search has moved to the Web. And on the Web, entities are not only characterized by structured data alone but to a large extent, by unstructured information. Indeed, unstructured data is a rich source of information on the Web. In order to exploit it best for entity-centric search, the research community has focused on building capabilities for extracting structured information out of it. Named entity recognition (NER) [31, 80], entity detection (ED) [7], or the extraction of precise facts from unstructured data known as open information extraction (OIE) [37] are just a few of the information extraction techniques that reached maturity. With these tools, large repositories of facts in the form of *(subject, predicate, object)* triples could be extracted. But without proper structure, like a global ontology or schema, querying entities in such triple stores remains a challenge. In fact, the problem of algorithmically structuring information on the Web has been extensively researched, see e.g., [22, 24, 116]. However, current automatic approaches still face quality problems and require considerable effort for extracting, transforming and loading data. Thus, from a practical perspective they are not yet mature enough to keep up with the volume and velocity, at which new data is published on the Web.

Another way of approaching this problem emerged from how data surfaces the Web. Most websites are dynamically generated from some structured data source. For instance, the IMDb page of the movie 'Iron Man 3'[4] is obviously a dynamically generated page. It is most probably built with some HTML template engine and server sided scripts accessing the needed data from a database. This is convenient for human use, but not for machines. However, making the data directly available online, in structured form, in a data store, would allow machines to perform proper entity search, much beyond the naïve information retrieval (IR)-style keyword search. Instead of trying to automatically extract the structured data, the Linked Open Data (LOD) [9, 13] initiative tried a different approach. It offers technology for information providers to directly publish data online in structured form and interlinked with other data. In Fig. 3 we present a small selection of the data on movie

[4] http://www.imdb.com/title/tt1300854/

'Iron Man 3' available on DBpedia.org[5]. Such data would most definitely allow us to perform some entity-centric search. For starters, *entity summary* queries seem to be fairly simple: just by searching an entity, in this case a movie, one can get all information about it. Or not? Unfortunately, some of the important information is missing in the case of the 'Iron Man 3' movie. For instance, no information about the full cast or the movie genre is provided on DBpedia. However, this data is available on Freebase. After all, this is the point in *linked data*: to join information from multiple data stores for a unified view of an entity. In LOD entities are matched to one another through the *owl:sameAs* property. Hence, for the example presented in Fig. 3, we would require that an *owl:sameAs* link matches the entity from DBpedia to its counterpart on Freebase. But this link is missing. As discussed in [33], the cross-linkage in the LOD is not nearly as extensive as one would hope. At the same time, automatic instance matching solutions present important quality problems [57].

Furthermore, it is worth noticing that, in the example presented in Fig. 3, various vocabularies (e.g. DBpedia's *dbpedia-owl*, *foaf*, *owl*, *rdf*, *rdfs* and *dbprop*) are being used



Property	Value
dbpedia-owl:Work/runtime	▪ 130.0
dbpedia-owl:budget	▪ 2.0E8
dbpedia-owl:cinematography	▪ dbpedia:John_Toll
dbpedia-owl:director	▪ dbpedia:Shane_Black
dbpedia-owl:distributor	▪ dbpedia:Walt_Disney_Studios_Motion_Pictures
dbpedia-owl:gross	▪ 1.215439994E9
dbpedia-owl:musicComposer	▪ dbpedia:Brian_Tyler_(composer)
dbpedia-owl:producer	▪ dbpedia:Kevin_Feige
dbpedia-owl:runtime	▪ 7800.000000 (xsd:double)
rdf:type	▪ owl:Thing
	▪ http://schema.org/CreativeWork
	▪ dbpedia-owl:Work
	▪ dbpedia-owl:Wikidata:Q11424
	▪ dbpedia-owl:Film
	▪ http://schema.org/Movie
rdfs:label	▪ Iron Man 3
owl:sameAs	▪ http://www.wikidata.org/entity/Q209538
	▪ dbpedia-fr:Iron_Man_3
	▪ dbpedia-de:Iron_Man_3
	▪ dbpedia-cs:Iron_Man_3
	▪ dbpedia-el:Iron_Man_3
	▪ dbpedia-es:Iron_Man_3
	▪ dbpedia-it:Iron_Man_3
	▪ dbpedia-ja:アイアンマン3
	▪ dbpedia-ko:아이언맨_3
	▪ dbpedia-nl:Iron_Man_3
	▪ dbpedia-pl:Iron_Man_3
	▪ dbpedia-pt:Homem_de_Ferro_3
	▪ dbpedia-ru:Железный_человек_3
	▪ dbpedia:Iron_Man_3
foaf:homepage	▪ http://www.marvel.com/ironman3
foaf:isPrimaryTopicOf	▪ wiki-en:Iron_Man_3
foaf:name	▪ Iron Man 3
is dbpedia-owl:wikiPageDisambiguates of	▪ dbpedia:Iron_Man_(disambiguation)
	▪ dbpedia:IM3
is dbpedia-owl:wikiPageRedirects of	▪ dbpedia:Iron_Man_Three
	▪ dbpedia:Iron_Man_4
	▪ dbpedia:Iron_Man_III
	▪ dbpedia:Iron_man_3
	▪ dbpedia:Iron_man_three
	▪ dbpedia:Iron_Man_3_(film)
	▪ dbpedia:Ironman_3
is foaf:primaryTopic of	▪ wiki-en:Iron_Man_3

Fig. 3. Data from dbpedia.org about the Iron Man 3 movie.

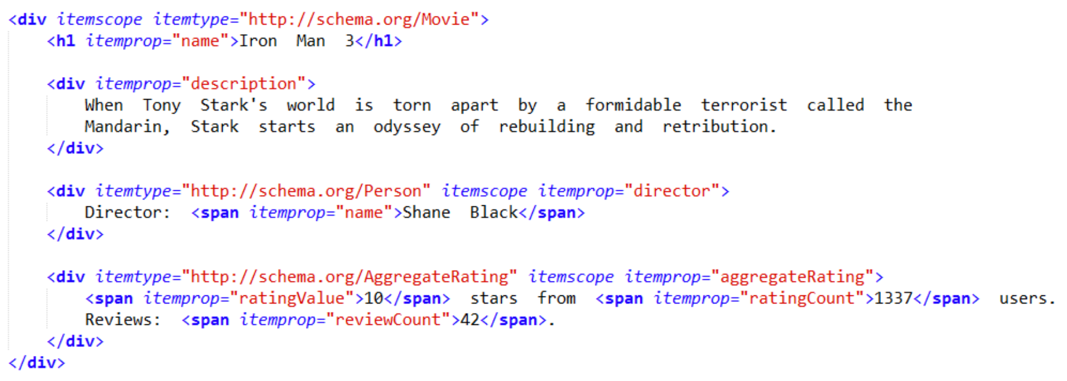
[5] DBpedia [14], is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. It is the heart of the LOD cloud as most other data stores link their data to DBpedia. The data on movie 'Iron Man 3' can be found at: http://live.dbpedia.org/page/Iron_Man_3

for describing the structure of the data. Indeed, the LOD is very flexible since it even allows for each data publisher to define its own structure. But, as we have shown in [59], this flexibility comes at a price: although data stores may overlap in terms of the data stored, the vocabulary used for structuring (and thus querying) may seriously differ. Ontology alignment has been proposed as a remedy, but the quality of results is still not convincing [62, 63]. An in depth analysis on the benefit of LOD for entity-search, the problems that arise and possible solutions are presented in Section 2.1.

To avoid all these problems while improving their query capabilities, major Web search engine providers went a slightly different way. Their managed approach builds on a collection of ready-made schemas accessible on schema.org, which are centrally managed by Bing, Google, Yahoo! and Yandex. These schemas are used as a vocabulary to be embedded in the HTML source code of a page using microdata. An example of a microdata annotation integrated into the Web page of the 'Iron Man 3' movie on IMDb is presented in Fig. 4. The main incentive for page owners to use schema.org is that once a Web page features content annotated with schema.org's vocabulary, any search engine can present it as a rich snippet. Furthermore, the Web page has a higher chance of being found by users interested in that very specific content, too. Indeed, motivating page owners to annotate their data with schema.org vocabulary has multiple advantages:

- The effort is spread over many shoulders reducing the effects of volume and velocity at which new data comes to the Web;
- annotations are of high quality – the one creating the data should understand its semantic meaning best;
- the structure is centrally managed and data can be queried globally without complicated alignment operations like in the case of LOD;
- entity-centric queries with Web data are enabled, ultimately fostering semantic search for the next generation Web.

For entity-centric search, schema.org represents an important advantage because every entity type is described by one global schema. There is no need to solve complicated schema mapping, or instance matching problems to get a unified view of



```
<div itemscope itemtype="http://schema.org/Movie">
    <h1 itemprop="name">Iron  Man  3</h1>

    <div itemprop="description">
        When  Tony  Stark's  world  is  torn  apart  by  a  formidable  terrorist  called  the
        Mandarin,  Stark  starts  an  odyssey  of  rebuilding  and  retribution.
    </div>

    <div itemtype="http://schema.org/Person" itemscope itemprop="director">
        Director:  <span itemprop="name">Shane  Black</span>
    </div>

    <div itemtype="http://schema.org/AggregateRating" itemscope itemprop="aggregateRating">
        <span itemprop="ratingValue">10</span>  stars  from  <span itemprop="ratingCount">1337</span>  users.
        Reviews:  <span itemprop="reviewCount">42</span>.
    </div>
</div>
```

Fig. 4. Schema.org annotation for movie Iron Man 3 in microdata format.

some entity. Getting all entities of the same type, or based on some properties is also straight forward. One just needs to index Web data with microdata aware indexes like for instance Sindice (see [36]) is doing[6]. But an in depth analysis on schema.org reveals, that the number of annotations is in fact very small [56]. This means that, until schema.org gains traction, its practical use for entity search is quite limited. More on this topic is presented in Section 2.2.

Unstructured data, linked data and data annotated with global vocabularies like schema.org are the main representation forms for entity data on the Web. Building on representation form specific query functionality (see Fig. 5), systems like the Google Knowledge Graph or the recently published Google Knowledge Vault [35] have been proposed. The Knowledge Graph provides a short and concise summary of an entity. After typing some entity name into Google's search field, an entity summary is displayed on the right hand side of the search results, if the Knowledge Graph contains the entity. A sample entity summary for 'Chuck Noris' is shown in Fig. 6. According to Google's official blog[7], the Graph mainly relies on manually curated data sources like the Wikipedia infoboxes[8], Google's Freebase[9], and schema.org[10] annotations on the Web. But the Knowledge Graph has a major shortcoming: it doesn't cope with the number of new entities published daily on the Web.

It only provides information on well-known entities already having a Wikipedia article, Freebase record or sufficient schema.org annotations. Our extensive evaluation presented into more detail in Chapter 6, shows that this is indeed rather limited. For instance, from a list of 14,199 common diseases (according to World Health

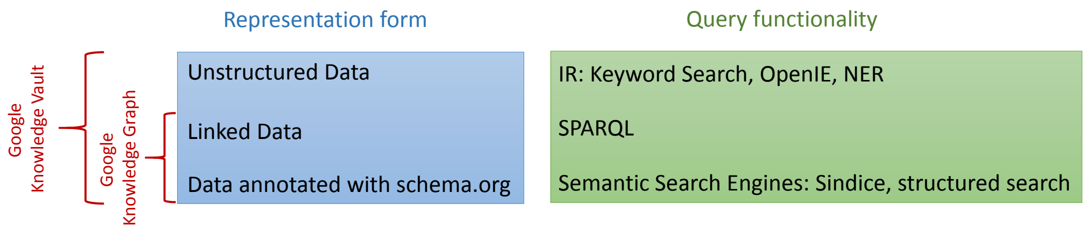


Fig. 5. Entity data on the Web. Representation forms, query functionality and entity-centric search systems.

[6] An example of retrieving all schema.org annotations for movies with Sindice: http://sindice.com/search?q=schema&nq=&fq=class%3Ahttp%3A%2F%2Fschema.org%2FMovie%20format%3AMICRODATA&interface=guru&facet.field=domain

[7] http://www.googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html

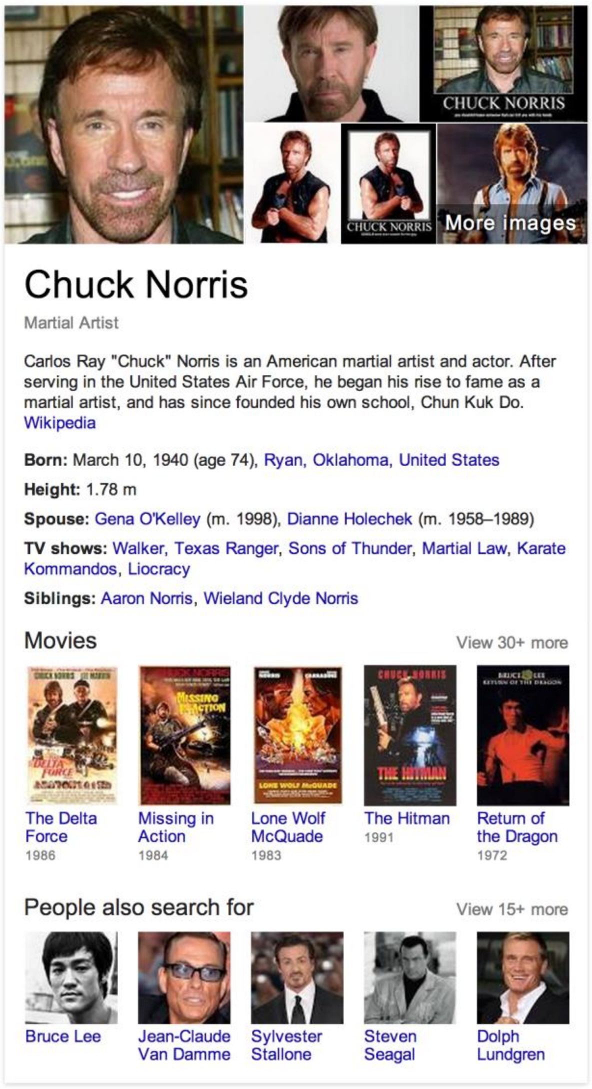
[8] An infobox is a fixed-format table designed to be added to the top right-hand corner of articles to consistently present a summary of some unifying aspect that the articles share and sometimes to improve navigation to other interrelated articles.

[9] Freebase ([16]) is a community-curated database of well-known people, places and things.

[10] Schema.org is an initiative of Bing, Google, Yahoo! and Yandex to create and support a common set of schemas for structured data markup on web pages.

Organization – the International Statistical Classification of Diseases[11]) we found about 7,000 being covered by the Wikipedia diseases category, with only about 3,000 of them also featuring an actual article on Wikipedia or Freebase. As we will discuss in Section 2.2, schema.org has not really gained traction. Considering its low acceptance of only about 1.5% of the websites, schema.org doesn't contribute much to extending the knowledge base either.

This way, the majority of entities (in particular, new or more obscure entities) not present in the manually curated Web resources used by Google's Knowledge Graph, cannot benefit from data summarization.



Chuck Norris

Martial Artist

Carlos Ray "Chuck" Norris is an American martial artist and actor. After serving in the United States Air Force, he began his rise to fame as a martial artist, and has since founded his own school, Chun Kuk Do. Wikipedia

Born: March 10, 1940 (age 74), Ryan, Oklahoma, United States

Height: 1.78 m

Spouse: Gena O'Kelley (m. 1998), Dianne Holechek (m. 1958–1989)

TV shows: Walker, Texas Ranger, Sons of Thunder, Martial Law, Karate Kommandos, Liocracy

Siblings: Aaron Norris, Wieland Clyde Norris

Movies View 30+ more

The Delta Force 1986

Missing in Action 1984

Lone Wolf McQuade 1983

The Hitman 1991

Return of the Dragon 1972

People also search for View 15+ more

Bruce Lee

Jean-Claude Van Damme

Sylvester Stallone

Steven Seagal

Dolph Lundgren

Fig. 6. Knowledge Graph - result for 'Chuck Noris'.

[11] http://www.who.int/classifications/icd/en

The Knowledge Vault is a Web-scale probabilistic knowledge base. It combines information extracted from unstructured data from the Web obtained via techniques like OpenIE and NER, tabular Web data, page structure elements, data manually annotated with schema.org and prior knowledge derived from existing knowledge repositories from the LOD cloud. All these information sources are fused together with machine learning methods to build a unified and representation form independent data source for entity search. Published just few weeks ago, the system is the latest development from Google. It is meant to replace the Knowledge Graph and it follows the same basic idea as proposed in this thesis, i.e. tapping all information sources by abstracting from the various data representation forms and fusing them into an integrated data repository. More on our system for data extraction and representation is provided in Chapter 5. But like any system that integrates data from multiple sources, the Vault, suffers from problems regarding entity reconciliation and duplicate detection. This problems are known from core database research and have recently been discussed with respect to interlinking data in the LOD cloud. Proposed solutions seem to have reached maturity and show promising results. We discuss this topic in more detail in Section 2.3.

2.1. Linked Data

The term "Linked Data" was coined by Tim Berners-Lee, the director of the World Wide Web Consortium in a design note ([9]) discussing issues around the Semantic Web project. His vision was that intelligent agents will one day be able to handle all the "day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines." [10]. But for this to be possible, computers have to become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers.

Linked Data describes a method for of publishing structured data on the Web in such a way that it can be interlinked and become more useful. The building blocks are standard Web technologies such as the Hypertext Transfer Protocol (HTTP) [38], the Resource Description Framework (RDF) data model [93] and Uniform Resource Identifiers (URIs) [11]. But rather than using them to serve web pages for human readers, it extends them to share information in a way that can be read automatically by computers. This enables data from different sources to be connected and queried.

The core of publishing linked data is outlined by Tim Berners-Lee through four *design principles* published in [9], paraphrased along the following lines:

1. Use URIs to denote things.
2. Use HTTP URIs so that these things can be referred to and looked up ("dereferenced") by people and user agents.
3. Provide useful information about the thing when its' URI is dereferenced, leveraging standards such as RDF.

4. Include links to other related things (using their URIs) when publishing data on the Web.

Since 2006, when the concept of linked data has been introduced, the amount of data published on the Web in linked form, has gradually increased. In May 2007 it comprised 12 data stores. They all published data in *subject, predicate, object* RDF triple format, accounting together for about 500 million RDF triples. In September 2011, when the diagram (presented in Fig. 7) of the LOD cloud was last updated, there were about 31 billion triples. Currently, more than 53 billion triples in over 300 data stores are available in the largest Virtuoso-based Semantic Web data cache[12].

LOD covers a wide range of domains, governmental data representing the largest portion of data that is being published online in linked form. In **Table 1** we provide an overview of the various domains, number of data stores and amount of triples for the LOD cloud from Fig. 7. Cross-domain data sources concentrate information on all kinds of entities. This makes them a good data source for entity type-based entity search. With 41 data sets and more than 4 billion triples, cross-domain data stores are also well represented in the LOD could. These statistics don't even include Freebase, which by itself covers information on almost 45 million entities,

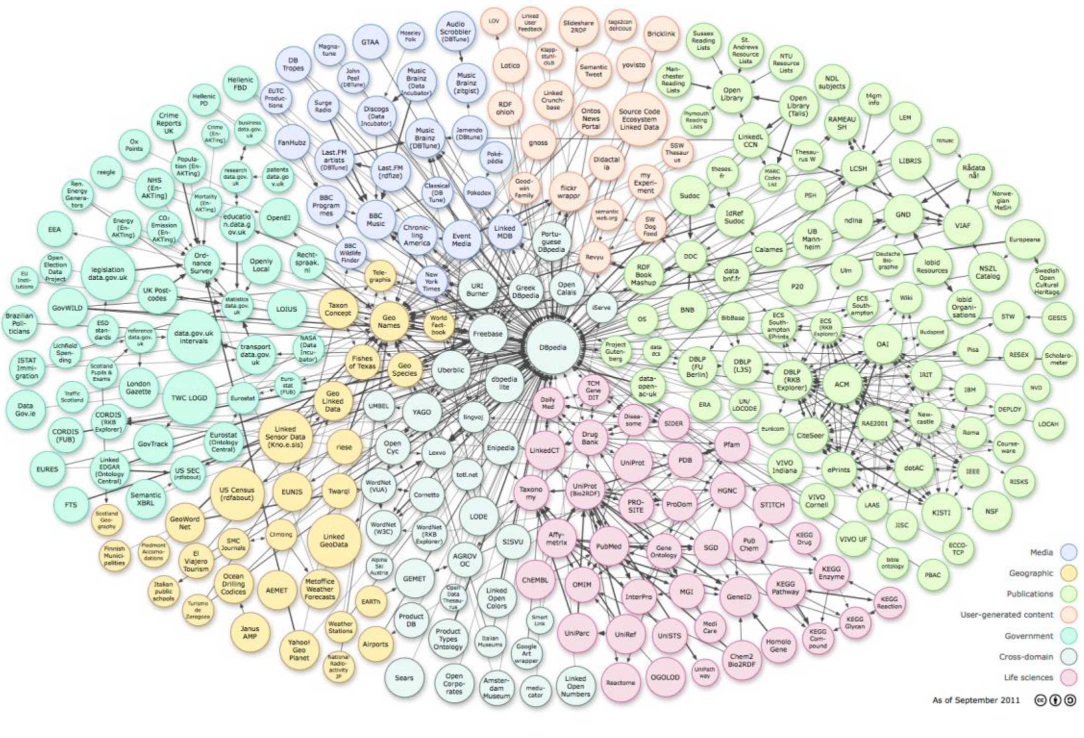


Fig. 7. Linking Open Data cloud diagram as of September 2011. By Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/.

[12] The Virtuoso SWDB is accessible at http://lod.openlinksw.com/sparql/ through a SPARQL endpoint.

partly gathered from other LOD sources. Typical examples of cross-domain data stores are DBpedia, OpenCyc [77] or YAGO [116]. DBpedia for instance, stores data about all kinds of entity types, be it personalities, organizations, medical conditions, movies, books, music, video games and so on. In total it provides information on about 4 million entities. The cross-domain data sources are well inter-linked, but have fewer out-links. They are mostly used as references by other data stores, having many more in-links (as observed on the example on DBpedia in Fig. 7).

Table 1: Domain based overview of the amount data stores, triples and RDF links that are set from data sources within a domain to other data sources. LOD state as of September 2011. Data from http://lod-cloud.net/state/.

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

There is no central authority to manage data published in linked form. Each data store is responsible for publishing, storing and managing its own data. In consequence, working with such data, requires that one first searches for relevant data sources and possibilities to access the respective data. Fortunately, DataHub[13], which is a data management platform offered by the Open Knowledge foundation[14], provides data owners with the tools for registering the published datasets, for tagging them with metadata and corresponding description, and adding SPARQL endpoints for data access. It also allows data consumers to search for data sets based on keywords, to download samples, even complete data dumps, or to query data if SPARQL endpoints are provided.

On a logical level, in data stores, the data is usually stored in *subject, predicate, object* RDF triple format. An excerpt of some triples for movie 'Iron Man 3' from DBpedia is presented in Fig. 8. The subject is always an URI representing an entity,

[13] http://datahub.io/

[14] The Open Knowledge foundation (https://okfn.org/) is a worldwide non-profit network of people passionate about openness, using advocacy, technology and training to unlock information and enable people to work with it to create and share knowledge

```
<IMURI> <http://www.w3.org/2000/01/rdf-schema#label> "Iron Man 3"@en
<IMURI> <http://dbpedia.org/property/name> "Iron Man 3"@en
<IMURI> <http://dbpedia.org/ontology/director> <http://dbpedia.org/resource/Shane_Black>
<IMURI> <http://dbpedia.org/ontology/producer> <http://dbpedia.org/resource/Kevin_Feige>
<IMURI> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing>
<IMURI> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/CreativeWork>
<IMURI> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Movie>
<IMURI> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Film>
<IMURI> <http://www.w3.org/2002/07/owl#sameAs> <http://rdf.freebase.com/ns/m.0bc1yhb>
<IMURI> <http://www.w3.org/2002/07/owl#sameAs> <http://www.wikidata.org/entity/Q209538>
```

Fig. 8. RDF excerpt comprising 10 triples from DBpedia for movie Iron Man 3. <IMURI> is short for <http://dbpedia.org/resource/Iron_Man_3>.

the predicate is an URI representing a property, and the object is either the URI of an entity or a literal. For the example in Fig. 8, the subject of all these triples is obviously the DBpedia URI of the movie 'Iron Man 3'. Some interesting predicates are the <http://www.w3.org/2000/01/rdf-schema#label> (known as, and further denoted with rdfs:label since *rdfs* is a predefined prefix of http://www.w3.org/2000/01/rdf-schema#) property indicating the human friendly name of a resource, or the <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> (known as rdf:type again as a result of predefined prefix structure) linking an entity to its type. For our previous example, the movie 'Iron Man 3' is defined as having the types <http://www.w3.org/2002/07/owl#Thing>, <http://schema.org/CreativeWork>, <http://schema.org/Movie> and <http://dbpedia.org/ontology/Film>. Is this entity type information useful for empowering entity search on types? Being a property of all entities, in the following sections we discuss about the contribution of LOD data to entity-centric search by focusing on the example of the rdf:type property. But our observations are valid for any other property and entity search task.

2.1.1. Entity Type Queries using the rdf:type Property

The semantic Web research community has invested much effort in providing entities with proper type information. This is exactly the kind of information required for entity search based on types: for instance, when searching for movies, one could get all movie URIs and their titles from DBpedia only, by getting all entities of type <http://dbpedia.org/ontology/Film> with the following SPARQL query[15]:

```
SELECT DISTINCT ?MovieTitle, ?MovieURI WHERE{
  ?MovieURI rdf:type dbpedia-owl:Film .
  ?MovieURI rdfs:label ?MovieTitle .
}
```

where *rdf*, *rdfs* and *dbpedia-owl* are the predefined prefixes for the rdf syntax, the rdf schema and the DBpedia ontology name spaces. This query returns all 77,600 movie entities available on DBpedia. Since DBpedia, like the other data repositories in the LOD is manually curated, we can assume that all these entities are indeed movies. Precision is in this case not an issue. However, recall is an issue, since these are not

[15] The SPARQL endpoint for DBpedia is available at http://dbpedia.org/sparql

all the movies available in the LOD cloud. Other sources, like YAGO, Freebase, Linked Movie DataBase[16] (LinkedMDB), etc., also store movie entities. In consequence, to fully answer the entity type based queries one would have to search and return all relevant entities from all relevant data stores. Three basic steps have to be performed for this:

- First, one has to identify relevant sources and corresponding SPARQL endpoints. This is done by means of keyword search on DataHub.
- Second, one has to run SPARQL queries on each identified endpoint. The goal here is to get all relevant entities based on the entity type. To make matters simpler, one can use the endpoint[17] provided by OpenLink which centralizes and caches all data in the LOD cloud in one single large triple based Virtuoso repository. This way all data can be accessed through a single endpoint. However, this only solves part of the problem: running the SPARQL query from the previous example will return all movies from DBpedia only, even if executed on the OpenLink Virtuoso cache comprising all triples from the LOD. The reason for this behavior is the fact that the query specifies that all entities of type 'dbpedia-owl:Film' be returned. At the same time, other data sources use other URIs for the movie entity type. At the core of the problem lies one of the basic design principles of LOD: the fact that each data store is responsible of publishing and maintaining its data. As such, each source may develop its own vocabulary and ontologies for structuring the data. This flexibility comes at a price: querying the LOD as a unified source of data is impossible due to the lack of global or aligned structure. In the case of entity type-based search, this means that, for the same entity type, each data source may have its own data type URI. For movies for instance, the URI for the 'movie' type on Freebase is <http://rdf.freebase.com/ns/film.film>. Although it represents the same type of entities as the DBpedia movie type, the Freebase movie type is represented by a different URI. This is not an isolated case: later on, in this section we provide an in depth analysis showing the complexity of the problem and possible solutions on the example of various types of entities.
- Third, obviously, data sources in the LOD cloud may overlap in terms of stored entities. In the final step, one has to integrate the result obtained from querying all relevant data stores. This basically means identifying and eliminating duplicates. Normally, this should be a simple task since all entities are represented through unique identifiers: URIs. Ideally one could just use the DISTINCT clause from SPARQL to get the set of all unique entities for the given entity type. However, the advantage of having independent data stores with a flexible vocabulary strikes again since one entity may be represented

[16] linkedmdb.org/

[17] http://lod.openlinksw.com/sparql/

with different URIs in different data stores. Known as *instance matching*, the problem of finding all URIs that refer the same real-world entity has yet to be solved. An in-depth analysis over this problem is provided in Section 2.3.

2.1.2. Analyzing the Practical Utilization of rdf:type in Linked Data

The basic hypothesis is that one can use the rdf:type property to successfully perform entity search based on the type. The rdf:type property, defined in the RDF Schema 1.1 of the W3C recommendation[18] is an instance of the rdf:Property that is used to state that an entity is an instance of a class. For this reason, rdf:type is the most simple way of getting entities of a certain type: identify the type URI (a task that could simply be performed with the help of an inverted index with keywords as keys and entity type URIs as values) and get all entities having that URI as a type. But as we have seen on the example of movies, the same type may have different URI representations. To assess the dimension of this problem we performed an in-depth analysis on the type URIs assigned to entities of the same type.

Presented in [59], we conducted an analysis on the well-known Billion Triple Challenge corpus (BTC) from 2012 ([50]). BTC comprises about 1.4 billion quads of the form *(subject, predicate, object, source)* crawled from major LOD data stores like Datahub, DBpedia, Freebase, YAGO and others during May and June 2012. We extracted a list of movie types from the BTC data set, by bootstrapping on a seed of movies from the LinkedMDB and manually inspecting the resulting types. With this approach we found 4,336 URIs all for the type 'movie'. This surprisingly large number is mostly due to the very fine classification provided by YAGO. This way, besides basic types, most specific movie types e.g. Animation, FrenchFilm, or even 1910DramaFilms are introduced. Using the same method, we found 2,919 URIs representing the type 'book' and 14,190 URIs for type 'music album'. Overall, we found 169,469 unique URIs for movies in BTC. The number of unique movies is obviously smaller because of existing duplicates, but it is hard to establish manually. With the DBpedia movie type URI, one can reach only about 20% of the entities. A list of most comprising URIs for 'movie' type is presented in **Table 2**. For entities of type 'book' and 'music album' results are similar. To be self-contained, we include them in the Appendix A chapter in **Table 19** and **Table 20** respectively. In consequence, with the current state of linked data and without further intervention entity search based on the rdf:type property doesn't really work on a large scale. While precision is perfect, since the type information is manually curated, the problem is the low recall indirectly caused by the vocabulary flexibility.

We did not aim for, and raise no claim for completeness for this experiment. But the sheer number of URIs identified for the same type, corroborated with the fact that one type URI reaches only a subset of entities of a given type, shows that getting all entities of a certain type using LOD is not trivial. Ontology alignment has been

[18] http://www.w3.org/TR/rdf-schema/#ch_type

proposed as a remedy, but the quality of results is still not convincing [62, 63]. However, as the results from **Table 2** suggest, it would be enough to get to the few URIs comprising most entities of the type.

Table 2: Top 30 most comprising URIs for the 'movie' entity type from various data stores in the BTC corpus.

URI	Nr.	%
<http://rdf.freebase.com/ns/film.film>	118,731	70.06
<http://schema.org/Movie>	34,455	20.33
<http://dbpedia.org/ontology/Film>	34,455	20.33
<http://data.linkedmdb.org/resource/movie/film>	31,141	18.38
<http://dbpedia.org/class/yago/Movie106613686>	8,406	4.96
<http://dbpedia.org/class/yago/English-languAgeFilms>	8,036	4.74
<http://dbpedia.org/class/yago/Black-and-whiteFilms>	6,011	3.55
<http://dbpedia.org/class/yago/AmericanFilms>	5,941	3.51
<http://umbel.org/umbel/rc/Movie_CW>	5,091	3.00
<http://dbpedia.org/class/yago/IndianFilms>	3,121	1.84
<http://dbpedia.org/class/yago/ShortFilms>	1,693	1.00
<http://dbpedia.org/class/yago/Hindi-languAgeFilms>	1,507	0.89
<http://dbpedia.org/class/yago/BritishFilms>	1,368	0.81
<http://dbpedia.org/class/yago/Spanish-languAgeFilms>	1,320	0.78
<http://dbpedia.org/class/yago/SilentFilms>	1,307	0.77
<http://dbpedia.org/class/yago/DramaFilms>	1,236	0.73
<http://dbpedia.org/class/yago/IndependentFilms>	1,102	0.65
<http://dbpedia.org/class/yago/ItalianFilms>	1,035	0.61
<http://dbpedia.org/class/yago/FrenchFilms>	1,017	0.60
<http://dbpedia.org/class/yago/Italian-languAgeFilms>	892	0.53
<http://dbpedia.org/class/yago/ArgentineFilms>	883	0.52
<http://dbpedia.org/class/yago/2007Films>	867	0.51
<http://dbpedia.org/class/yago/ComedyFilms>	866	0.51
<http://dbpedia.org/class/yago/2006Films>	858	0.51
<http://dbpedia.org/class/yago/2008Films>	821	0.48
<http://dbpedia.org/class/yago/Tamil-languAgeFilms>	811	0.48
<http://dbpedia.org/class/yago/French-languAgeFilms>	792	0.47
<http://dbpedia.org/class/yago/JapaneseFilms>	774	0.46
<http://dbpedia.org/class/yago/2005Films>	774	0.46
<http://dbpedia.org/class/yago/2009Films>	774	0.46

For instance, when starting from the DBpedia URI for the 'movie' type, connecting that URI to its' counterpart from Freebase and LinkedMDB would already account for more than 90% of entities. Can one identify and connect different URIs representing the same type to ultimately improve entity search on types in LOD? Ontology alignment doesn't seem to work. However, another possible approach is to perform type alignment by means of witnesses, with the help of the owl:sameAs property: For instance, starting from the type dbpedia-owl:Film, one would identify movie entities from DBpedia like the Iron Man 3 movie with URI <http://dbpedia.org/resource/Iron_Man_3>. This entity is represented in Freebase through URI <http://rdf.freebase.com/ns/m.0bc1yhb>. This fact is stored amongst the DBpedia data of movie Iron Man 3 through triple:

```
<IMURI> <http://www.w3.org/2002/07/owl#sameAs> <http://rdf.freebase.com/ns/m.0bc1yhb>
```

Following on the type of entity URI <http://rdf.freebase.com/ns/m.0bc1yhb> in Freebase, one finds five type URIs presented in **Table 3**. Out of these, URI <http://rdf.freebase.com/ns/film.film> grants access to 70% of the movie entities in the BTC corpus. However, connecting the DBpedia URI to any other of the five URIs, would have disastrous effects on the results: it would include all kinds of entities be it movies, music, video games or books, that have a common topic or have been awarded some prize. Filtering this type URI only, without manual intervention, is possible if there is enough evidence, that the URI for 'movie' type from DBpedia is the same as the one from Freebase and less similar to the other Freebase type URIs. A large number of witnesses confirming this connection is strong evidence enough. Therefore, it is crucial to have a high number of, in this particular case movie URIs, from one source linked through owl:sameAs predicates to their counterparts from other data stores. Unfortunately, as discussed in [33], today owl:sameAs linkage is not nearly as extensive as one would hope. Many links are missing and from the ones available, a large part are trivial links like for instance the internal links of DBpedia linking URIs across different languages.

Known under the problem of *entity reconciliation* or *instance matching*, the problem of matching the same entity in different data sets has been extensively studied ([15, 61, 64, 89, 118, 124]) and seems to have reached a level of maturity. In [57] we pay closer attention to systems that promise to create high quality owl:sameAs links automatically. This has two advantages: On the one side it allows us to connect all URIs of a certain type, and in the following step it allows to remove duplicate URIs

Table 3: Type URIs for entity <http://rdf.freebase.com/ns/m.0bc1yhb> from Freebase.

<http://rdf.freebase.com/ns/film.film>
<http://rdf.freebase.com/ns/common.topic>
<http://rdf.freebase.com/ns/award.award_nominated_work>
<http://rdf.freebase.com/ns/award.award_nominee>
<http://rdf.freebase.com/ns/award.award_winning_work>

of the same real world entity, reducing the result set to one URI per unique entity. But, as we will discuss in more detail in Section 2.3, instance matching systems still struggle when it comes to high precision instance matching, making them useless for type alignment tasks.

Considering all this, we believe that rdf:type is a simple and effective way of getting all entities of a given type from one data store. But this represents only part of the entities to be found in the LOD cloud. If multiple stores are targeted, or if the whole LOD is to be used as a data sources, the flexibility of each data store being allowed to have its own vocabulary strikes. The same entity and entity type, have different URI representations in different stores. The built in mechanism that should help solve this problem is based on the owl:sameAs and the owl:equivalentClass properties. owl:sameAs matches instances of the same real world entity, and owl:equivalentClass matches URIs of the same class, in different data sources. Unfortunately, only a fraction of the data is linked with these properties. There are two main possible approaches at this point:

- One approach is to enforce a single vocabulary, a global schema that is centrally managed, and used by all data providers and publishers, so that there is only one type URI per type. This is the approach followed by Web search engine providers with their proposal of schema.org.
- The other approach, followed by the semantic Web community is to keep the schema flexibility, but to improve on instance matching systems that automatically create high quality links connecting data across multiple stores. The evolution of the systems presented in the Ontology Alignment Evaluation Initiative[19] (OAEI) with its Instance Matching[20] (IM) tracks stand as proof of the efforts that are being made in this direction.

These two approaches are discussed in more detail in the following sections.

2.2. Schema.org

Schema.org is built on the idea of a global vocabulary that is centrally managed by some authority, and that data producers and publishers use to annotate their data with. But the idea of semantically annotating data on the Web started much earlier. The first standardized concept that implemented the approach of open data annotations is called microformats[21].

Microformats started as a project in 2005, with the goal to integrate semantic information into HTML code for making it machine readable. For this purpose microformats use existing HTML tags e.g. *class*, indicating the class or format name, *rel*

[19] http://oaei.ontologymatching.org/

[20] http://www.instancematching.org/oaei/

[21] http://microformats.org/

(relationship) providing the description of the target address in an anchor-element and *rev* (reverse relationship) providing the description of the referenced document. Out of these, class is by far the most useful tag for microformats. As the name suggests, microformats are organized in formats. Each format covers one specific topic. Currently, there are 35 different formats[22]. Most of them cover locations, products or social annotations. A classic example of microformats (presented in Fig. 9) is the annotation of an address, with the help of the *adr* format specified with the HTML *class* tag. There are however, disadvantages to using microformats. They require the implementation of additional markup throughout the websites. Furthermore, since there are a limited set of data types supported by microformats, the data type may not even be fully supported.

```
<div class="adr">
  <span class="street-address">42 Sample Street</span>
  <span class="locality">Random City</span>
  <span class="postal-code">12345</span>
  <span class="country-name">UK</span>
</div>
```

Fig. 9. Example of microformats in HTML code.

Another technology for annotating Web data is Resource Description Framework in Attributes (RDFa). It is a W3C specification[23] for integrating RDF data into several web-formats, such as HTML, XHTML or XML. It was introduced in 2008 and refined in 2012 to version RDFa 1.1 which is now fully supported even by HTML7. RDFa is built on top of RDF, the well-known semantic web basic technology and it represents the vision of the semantic web community on how to semantically annotate data in Web pages. An example of an RDFa annotation for a person using schema.org as a source for the 'person' vocabulary is presented in Fig. 10. In contrast to microformats, RDFa uses vocabularies that allow users to create their own schemata. RDFa is quite flexible, offering for instance the possibility to combine multiple vocabularies, for mixing types of entities based on the needs. While it offers much more flexibility, RDFa was originally too complex. For this reason, in 2009, another technology designed as a simple subset of RDFa and microformats was proposed. Called microdata, this technology is primarily focusing on the core features of RDFa.

```
<div vocab="http://schema.org/" typeof="Person">
  <span property="name">John Doe</span>
  <img property="image" src="sample.jpg" />
  <span property="gender">male</span>
  <span property="telephone">0123456789</span>
</div>
```

Fig. 10. Example of RDFa in HTML code with schema.org vocabulary.

[22] http://microformats.org/wiki/Main_Page#Specifications

[23] http://www.w3.org/TR/rdfa-syntax/

Microdata[24] was introduced in 2009 as a part of HTML5 specifications. It uses hierarchical vocabularies as a data model to provide the semantics, or meaning of an item. It also allows Web developers to design a custom vocabulary or use vocabularies available on the Web. It was built for HTML5 and uses the following HTML5 tags for semantic annotations:

- *itemscope* – Indicates the annotation of the item. The descendants of this element contain information about it.
- *itemtype* – A valid URL of a vocabulary that describes the item and its properties context.
- *itemid* – Indicates a unique identifier of the item.
- *itemprop* – Indicates that its containing tag holds the value of the specified item property. The properties name and value context are described by the items vocabulary. Properties values usually consist of string values, but can also use URLs using the *a* element and its *href* attribute, the *img* element and its *src* attribute, or other elements that link to or embed external resources.
- *itemref* – Properties that are not descendants of the element with the itemscope attribute can be associated with the item using this attribute. It provides a list of element ids (not itemids) with additional properties elsewhere in the document.

On the example of an event annotation with schema.org vocabulary, in Fig. 11 we show how these tags are used in microdata.

All three technologies have advantages and disadvantages. Microformats introduces the least complexity but it is also the least powerful. RDFa 1.1 is the most flexible but the most complex. Microdata seems to offer a tradeoff although some constructs like an item having multiple types (a business can be both an 'AutoPartsStore' and a 'RepairShop') are difficult to express.

At the same time when these technologies were being introduced, web search companies were struggling to get past the 10 blue links, they had offered for the last

```
<div itemscope itemtype="http://schema.org/Event">
  <a itemprop="url" href="event_site.html">
    <span itemprop="name">A space odysee</span>
  </a>
  <meta itemprop="startDate" content="2001-07-05T13:37">
    <span>Thu, 07/05/01 1:37 p.m.</span>
  <div itemprop="offers" itemscope itemtype="http://schema.org/AggregateOffer">
    <span itemprop="offerCount">42</span> tickets left
  </div>
</div>
```

Fig. 11. Example of microdata in HTML code with schema.org vocabulary.

[24] http://www.whatwg.org/specs/web-apps/current-work/multipage/microdata.html#microdata

decade, to provide for better user experience. Proposed in 2008, Yahoo! Search Monkey was the first service which allowed web site owners to use structured data to make Yahoo! Search results more useful and visually appealing, and drive more relevant traffic to their sites. Yahoo! Search Monkey relied on both microformats and RDFa technologies for data acquisition. Following on Yahoo!'s initiative, in mid-2009 Google introduced Rich Snippets, a search result presentation technique designed to summarize the content of a page making it easier for users to understand what the page is about. An example of a rich snippet generated for the IMDb page of movie 'Iron Man 3' is presented in Fig. 12. Like Yahoo's tool also Google's snippets relied on the Microformats, RDFa and Microdata data annotations embedded in HTML code.

The benefits of annotated data became clear to search engine companies very soon. However, neither Microformats nor RDFa or Microdata gained real traction amongst web site owners. Furthermore, the vocabularies used were heavily fragmented: different sources may use different vocabulary to describe the same data. This hindered the process of automatically interpreting or querying data as a whole. To increase the acceptance of such technologies, and solve the problem of fragmented vocabularies, in 2011 Bing, Yahoo and Google joined forces. They launched the schema.org initiative to "create and support a common set of schemas for structured data markup on web pages."[25,26,27] Schema.org is a set of extensible schemata, shared vocabularies which webmasters can use to mark up their pages in ways that can be understood by the major search engines. The main incentive for page owners is a better presentation of content in the web search result list. This way, the Web page has a higher chance of being found by users interested in that very specific content. For search engines, motivating page owners to annotate their data with schema.org vocabulary has the advantage that complex queries with Web data are possible, ultimately replacing keyword driven, with semantic enabled search. For entity centric search, schema.org presents the advantage that the structure is centrally managed and data can be queried globally without complicated alignment operations like in the case of LOD.

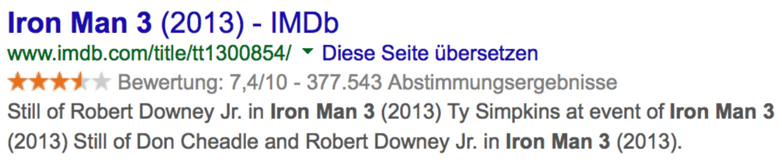

Iron Man 3 (2013) - IMDb
www.imdb.com/title/tt1300854/ ▾ Diese Seite übersetzen
★★★½☆ Bewertung: 7,4/10 - 377.543 Abstimmungsergebnisse
Still of Robert Downey Jr. in **Iron Man 3** (2013) Ty Simpkins at event of **Iron Man 3** (2013) Still of Don Cheadle and Robert Downey Jr. in **Iron Man 3** (2013).

Fig. 12. Google rich snippet for movie 'Iron Man 3' for IMDb page.

[25] http://blogs.bing.com/search/2011/06/02/introducing-schema-org-bing-google-and-yahoo-unite-to-build-the-web-of-objects/

[26] http://googleblog.blogspot.de/2011/06/introducing-schemaorg-search-engines.html

[27] http://www.ysearchblog.com/2011/06/02/introducing-schema-org-a-collaboration-on-structured-data/

But the fact that the schemata are managed by a central authority, a board of members representing the major search engine companies, has also attracted criticism from the semantic web community. To show flexibility, schema.org is continuously being updated and anybody can suggest a new schema or extend an existing one. For instance, in early-2013 there were about 300 schemata on schema.org. One year later more than 500 schemata were available. Furthermore, experts from various fields were actively engaged in developing schemata. For instance, schemata related to medicine have been developed on the model of the Medical Subject Headings (MeSH)[28], and with support from the US National Library of Medicine. Schemata related to movies have been developed with the help of IMDb who was afterwards amongst the early adopters of schema.org.

Overall, the underlying technology has reached maturity and schema.org is used as the de-facto vocabulary for microdata markup, which is already used by all the major search engine. It covers a large number of hierarchically organized entity types[29], extensively described with an average of about 33 attributes, and a maximum of 71 attributes (for the ExercisePlan schema[30]). Since the main result of using schema.org is annotating and structuring entities on the Web with a global and controlled vocabulary, schema.org is interesting for any type of entity-centric search. For instance, for type-based search one just needs to know the schema.org URL of an entity type e.g. http://schema.org/Movie for 'Movie' and to retrieve all annotations from a schema.org aware index similar to the one maintained by Digital Enterprise Research Institute (DERI) at sindice.com. The precision is still high, because data is annotated by people, and there are no vocabulary issues affecting recall like in the case of LOD. So is schema.org the technology that enables entity search on the Web? It certainly seems so. But is schema.org being used? Are there enough Web pages with schema.org annotations to justify its use for entity type search built on schema.org? In the following sub-section we analyze the web-scale acceptance of schema.org.

2.2.1. The Acceptance of Schema.org

There have been other studies analyzing the acceptance of various vocabularies for annotating data on the Web like for instance the work done by Muelheisen and Bizer in [86] or Mika and Potter in [83]. However, our work presented in [56] is the first study that focuses on schema.org.

To assess the acceptance of schema.org we analyzed ClueWeb12, a publicly available corpus comprising English sites only. ClueWeb12 is part of the Lemur Project[31]

[28] http://www.nlm.nih.gov/mesh/

[29] There were 529 schemata available on schema.org in November 2013

[30] https://schema.org/ExercisePlan

[31] http://lemurproject.org/

initiated in 2000 by the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts Amherst and the Language Technology Institute (LTI) at the Carnegie Mellon University. It comprises 733 million English Web pages, crawled between February 10th 2012 and May 10th 2012. These pages come from about 51 million domains. Considering that the size of the Web is currently estimated at about 60 billion websites[32], ClueWeb2012 comprises about 1,5% of the indexed Web, a reasonable size for a Web scale analysis.

There are also other options for large Web corpora like the Common Crawls Corpus[33], which comprises about 3 billion pages crawled from about 40 million domains. Common Crawls was also crawled in the first half of 2012. Unfortunately, there is not too much information about the kind of Web pages it comprises or the rationale behind the crawling process. For this reason, ClueWeb12 is usually preferred for scientific analysis and has been also our first choice.

ClueWeb12 comprises pages of broad interest: the initial seeds for the crawl consisted of 3 million websites with the highest PageRank from ClueWeb09, a previous Web scale crawl. To maintain a certain level of quality, several filters were applied to the crawled pages: first, websites blacklisted for malware, phishing, spyware, virus, file hosting, file sharing or pornographic content activities were all removed. Second, all non-English pages have been removed. Finally, pages larger than 10MB or containing more than 1 million terms were also removed. All other pages were saved in the web archive file format (.warc), split into chunks of 1GB and comprising several thousands of compressed Web pages each. In total, ClueWeb12 requires about 6TB hard-drive space when compressed, and about 60TB when uncompressed.

The analysis has been performed on a small enterprise mid-range server with 2 x 8 core x 3.1-3.8 GHz CPU running up to 32 parallel processes with hyper threading, 384 GB of DDR3 RAM and 9 x 480 GB SSD drives. Since there was not enough space to accommodate the full, uncompressed size of ClueWeb12, the analysis process was performed in small iterations focused on each archive file: Basically, each file was loaded into RAM, uncompressed, analyzed for microformats, RDFa, and microdata annotations, and then removed. Every annotation found was also persistently stored on the SSD based hard drive for later use. Since archives are independent from one another, the process could be parallelized up to the hardware limit and took about 12 days to complete.

Our analysis shows that about a year after its introduction, only 1.56% of the web pages from ClueWeb12 used schema.org to annotate data. The numbers of pages annotated with schema.org using the three mainstream data annotation techniques found in the ClueWeb12 documents are presented in **Table 4**. The use of different standards reflects the chronology of their adoption: microformats are the most

[32] http://www.worldwidewebsize.com/

[33] http://commoncrawl.org/

spread followed by RDFa, microdata and schema.org. It's interesting to notice that while microdata was introduced just a year after RDFa, there is a noticeable difference between their usage rates. The reason for this behavior is that when it was introduced, RDFa was presented as the prime technology for semantic annotation. Many content providers adopted it. Further developments brought by Google in 2009 have been regarded as yet another annotation method. It was only in mid-2011 when microdata became the main annotation technology for the newly proposed schema.org that microdata started gaining momentum. In fact, 12/15 million documents annotated with microdata (representing approximatively 80%) are schema.org annotations.

Table 4: Distribution of annotations by technology in ClueWeb12.

Data	Found URLs
Microformats	97,240,541 (12.44%)
RDFa	59,234,836 (7.58%)
Microdata	15,210,614 (1.95)
schema.org	12,166,333 (1.56%)

Out of the 296 schemas available in mid-2012 when ClueWeb12 was crawled, 244 schemas have been used in the annotations. To retrieve the state of schema.org at that time we used the Internet Archive[34]. The number of annotations per schemas (**Table 5**) follows a power law distribution with just 10 highest ranking schemas being used for 80% of the annotations and 17 schemas making for already 90% of all annotations. From the low occurring schemas in the long tail, 127 schemas occur less than 1000 times and 96 schemas occur even less than 100 times.

Schemas on schema.org are quite extensive. They include on average 34 attributes. "Thing" is with 6 attributes the smallest schema while "ExercisePlan" with 71 attributes is the most extensive. The annotations however are by far not as extensive as the structure allows. On average over all annotations, only 4.7 attributes were used. This accounts for about 10% of the attributes available in the corresponding schemas despite remaining data and existing matching attributes. It seems users are satisfied with just annotating some of the attributes. Most probably, this behavior is driven by the fact that rich snippets can only present a few attributes. In consequence users annotate only those few attributes that they consider should be included in the rich snippet. This way, from a user perspective, both the effort of annotating additional information and the risk that the rich snippet would present a random selection out of a broader number of annotated attributes are minimized.

[34] web.archive.org/web/20120519231229/www.schema.org/docs/ full.html

Table 5: Top-20 schema.org annotations from the ClueWeb12 corpus.

Schemata	Occur-rences	Average Nr. of Attributes	Percentage (Schema.org)
http://schema.org/Blog	5,536,592	5.56	19.57%
http://schema.org/PostalAddress	3,486,397	3.62	12.32%
http://schema.org/Product	2,983,587	2.28	10.54%
http://schema.org/LocalBusiness	2,720,790	3.29	9.62%
http://schema.org/Person	2,246,303	4.97	7.94%
http://schema.org/MusicRecording	1,580,764	2.77	5.59%
http://schema.org/Offer	1,564,257	1.32	5.53%
http://schema.org/Article	1,127,413	1.04	3.99%
http://schema.org/NewsArticle	823,572	3.81	2.91%
http://schema.org/BlogPosting	767,382	3.32	2.71%
http://schema.org/WebPage	659,964	4.11	2.33%
http://schema.org/Review	470,343	3.20	1.66%
http://schema.org/Organization	407,557	1.35	1.44%
http://schema.org/Event	400,721	2.69	1.42%
http://schema.org/VideoObject	396,993	0.47	1.40%
http://schema.org/Place	380,055	2.50	1.34%
http://schema.org/AggregateRating	342,864	1.66	1.21%
http://schema.org/CreativeWork	232,585	2.30	0.82%
http://schema.org/MusicGroup	223,363	1.15	0.78%
http://schema.org/JobPosting	168,542	4.38	0.60%

In conclusion, relatively to the number of Web pages in ClueWeb12, there are not too many annotations. The existing ones are also not very extensive. Therefore, an entity search system relying on schema.org will suffer from massive recall problems returning just a fraction of all relevant entities. Still, the absolute number of annotations per schema is quite large. Furthermore, all attributes of all schemata found in ClueWeb12 have been covered, in different annotations. Hence, we have reason to believe that automatically recognizing and maybe annotating unstructured data with schema.org schemata is possible. Can we learn models from the annotated data to empower high quality annotations? We discuss such a possibility in the following sub-section.

2.2.2. Learning to Annotate Unstructured Data with Schema.org

Unfortunately, as we have seen, schema.org annotations are not yet used broadly. The main reason invoked insistently on technology blogs on the Web is that the actual process of annotating Web data with schema.org is quite demanding[35]. In particular, the structure is centrally managed by schema.org and not at the liberty of annotators. This means that when annotating pages one has to:

a) repeatedly switch between the Web page to annotate and schema.org,
b) to browse through hundreds of schemas with tens of attributes each trying to find those schemas and attributes that best match the data on the Web page,
c) and finally to write the microdata annotation with corresponding schema.org URL resources into the HTML code of the page.

With such a complicated process it's no wonder that 1.1% of all found annotations are erroneous. Most frequent errors were bad resource identifiers caused by misspelled schemas or attributes or by schemas and attribute names incorrectly referred through synonyms.

But given the large number of annotations found in ClueWeb12, one could exploit these annotations to extend the coverage of schema.org. This would solve the recall problem entity type search with schema.org support. But the downside is that automatic methods may introduce "false negatives" (data that is being annotated incorrectly). The goal is in this case to increase recall, but without losing too much precision. The idea is to automatically match schemata form schema.org to relevant parts of unstructured data from web pages. If possible, such a system could even automatically map schema attributes to corresponding values from the text. All this is feasible if we are able to match schemata to pieces of page content. Imagine a system with the following basic workflow: given a Web page, it finds matches between schemata and pieces of page content, using models that have been trained with machine learning techniques on data annotated in ClueWeb12. Theoretically, any selection comprising consecutive words from the page content is a possible candidate for the matching. But considering all possible selections of page content is not feasible. Fortunately, the layout expressed through HTML elements says much about how information is semantically connected. With the help of the Document Object Model (DOM) API the HTML page is represented as a logical structure that connects HTML elements to page content in a hierarchical DOM tree node structure. These nodes envelop the pieces of content that are matched to the schemas. The matching method follows a greedy strategy, finding the smallest DOM nodes that best match a certain schema. Starting from the most fine-granular nodes (nodes are processed in the reversed order of the depth-first search) the content of each node is checked for possible match with all schemata from schema.org. The process continue until all nodes have been considered.

The fundamental task that needs to be solved first is matching schemata to unstructured data.

[35] See for example http://readwrite.com/2011/06/07/is_schemaorg_really_a_google_land_grab

Matching Schemata to Unstructured Data

From a sequence of words (the content of a DOM node) and the list of schemas from schema.org, the matching process finds those schemas that “best” match the content. More formally, given $W_n=\{w_1, w_2, \ldots, w_k\}$ the sequence of words representing the content of node *n*, and *S*, the set of URIs for schemata from schema.org, the schemata that best match W_n are:

$$S_{W_n} = \{s_i \mid s_i \in S \wedge match(W_n, s_i) \geq \theta\} \qquad (1)$$

where θ is a quality regulating parameter (for our experiments θ was set to 0.5), and *match*:{Words × URIs} → [-1,1] is the function for computing the confidence that a certain schema matches the given set of words. The expression of this function depends on the method that is chosen to perform the matching. There are various such methods. For instance, given that schemata published on schema.org describe various types of entities, one of the first approaches that come to one's mind for binding these schemata to unstructured data is entity recognition and named entity recognition. This has proven to work well for some entity types like products, persons, organizations or diseases [128]. However, considering the popular entities annotated on the ClueWeb12 corpus (**Table 5**), most of them describe more abstract entities e.g. “Blog”, “Review”, “Offer”, “Article”, “BlogPosting”, etc. In fact, out of the top-20 entity types, entity recognitions systems like OpenNLP[36] or Standford-NER [39] recognize less than half of them. Given an observation W_n, and the annotations extracted from ClueWeb12 as a training set comprising a large number of observations whose category of membership is known (the annotated schema) this becomes a problem of identifying the class for observation W_n. Machine learning methods like Naïve Bayes classification or Support Vector Machines have proven successful for text classification tasks even for more abstract entity types ([55, 58]).

Naïve Bayes classifiers rely on probabilities to estimate the class for a given observation. It compares the “positive” probability that some word sequence is the observation for some schema to the “negative” probability that the same word sequence is an observation for other schemas. In this case the matching function is:

$$match_{Bayes}(W_n, s) = \mathrm{P}(s|W_n) - \mathrm{P}(\bar{s}|W_n) \qquad (2)$$

But neither of the two probabilities can be computed directly from the training set. With the help of Bayes’s Theorem $\mathrm{P}(s|W_n)$ can be rewritten in computable form as $\mathrm{P}(s|W_n) = \frac{\mathrm{P}(W_n|s)*\mathrm{P}(s)}{\mathrm{P}(W_n)}$. Since W_n is a sequence of words that may get pretty long ($W_n=\{w_1, w_2, \ldots, w_n\}$), and this exact same sequence may occur rarely in the training corpus, to achieve statistically significant data samples “naive” statistical independence between the words of W_n is assumed. The probability of W_n being an observation for schema *s* becomes: $\mathrm{P}(s|W_n) = \frac{\prod_{j=1}\mathrm{P}(w_j|s)*\mathrm{P}(s)}{\prod_{j=1}\mathrm{P}(w_j)}$, and all elements of this formula

[36] opennlp.apache.org

can be computed based on the training set: P(s) can be computed as the relative number of annotations for schema s, $\mathrm{P}(\mathrm{w_j}|s)$ the number of annotations for schema *s* that include w_j relative to the total number of annotations for s, and $P(w_j)$ as the relative number of annotations including w_j. The negative probability $P(\bar{s}|W_n)$ is computed analogously and the matching function on the Bayes classifier can be rewritten as:

$$match_{Bayes}(W_n, s) = \prod_{\mathrm{j=1}} \mathrm{P}(\mathrm{w_j}|\mathrm{s}) * \mathrm{P(s)} - \prod_{\mathrm{j=1}} \mathrm{P}(\mathrm{w_j}|\bar{\mathrm{s}}) * \mathrm{P}(\bar{\mathrm{s}}) \quad (3)$$

Being common to all matching involving W_n, $\prod_{\mathrm{j=1}} \mathrm{P}(\mathrm{w_j})$ can safely be reduced without negative influence on the result. Probabilities for all words from the training set comprising annotations from ClueWeb12 (excluding stop words) build the statistical language models for all schemas, which are of course efficiently pre-computed before performing the actual Web site annotations.

Support Vector Machines use a different approach for classification. For each schema, a training set is built. It comprises annotations of the schema ("positive annotations") and annotations of other schemas ("negative annotations") in equal proportions. Each training set is represented in a multidimensional space (the Vector Space Model) with terms from all annotations as the space axes and annotations as points in space. In this representation, SVM finds the hyperplane that best separates the positive from the negative annotations for each schema. In the classification process, given observation W_n, and a schema *s*, SVM represents W_n in the multidimensional term space and determines the side W_n is positioned in with respect to the hyperplane of *s*. If it's the positive side then there is a match. The normalized distance from W_n to the hyperplane reveals the confidence of the assignment. The closer W_n is to the hyperplane of s, the less reliable the assignment. In this case, the *match* function is:

$$match_{SVM}(W_n, s) = distance(W_n, H_s) \quad (4)$$

Evaluation

For evaluating the schema matching functionality we prepared two data sets. Each has about 60,000 annotated web pages randomly harvested from ClueWeb12 comprising annotations with about 110 different schemas each. One of them is used as a training corpus for the classification methods. The other one is used as a test set. The test set is stripped of all annotations and provided to the system. We compare the pages annotated by the system for both Naïve Bayes and SVM, to the pages from the original test set and measure the schema matching effectiveness in terms of precision and recall.

On inspection over the results, about 5% of the schemas were not detected at all. The reason for this behavior is the fact that these schemas are present in the test set but they have no or almost no occurrences (up to 10) in the training set. Increasing the size of the training set helps reducing the number of undetected schemas. In fact, initial experiments with 10,000 and 30,000 web pages as training sets, with smaller schema annotation coverage, showed higher numbers of undetected schemas.

Overall, on the 110 schema annotations the system achieves on average 0.59 precision and 0.51 recall for Naïve Bayes and 0.74 precision and 0.76 recall for SVM, Simply matching schemas at random, as a comparison method, results in precision and recall lower than 0.01. The result values vary strongly from schema to schema. In **Table 6** we show the results for 15 schemas. The goal of the schema matching system is to improve recall while maintaining precision under control. For this reason, the 15 schemas presented in **Table 6** are chosen to cover the whole spectrum of F_2-measure values, given that the F_2-measure weights recall twice as much as precision.

No correlation between the number of occurrences in the training set and results could be observed. Having hundreds of schema annotations seems to lead to results similar to having tens of thousands of annotations. A few schemas, especially in the case of Naïve Bayes, have catastrophic precision and recall values (less than 0.01), despite occurring more than 4,000 times in the training set. These are schemas with very broad meaning e.g. "WebPage" or "Thing". Overall, SVM does better than Naïve Bayes. But it is interesting to notice that for many schemas the two approaches seem to complement each other: schemas where the Bayes achieves bad results are handled much better by SVM and vice versa. Taking this into consideration, it is

Table 6: Precision and recall values for the matching of schemata with Bayes and SVM.

		Naïve Bayes				SVM					
Nr.	Schemata	Prec.	Rec.	#	F_2	F_2	Prec.	Rec.	#	Schemata	Nr.
1	JobPosting	0.80	0.91	4,129	0.89	0.95	0.94	0.95	10,622	Blog	10
2	Event	0.63	0.78	15,856	0.74	0.85	0.79	0.87	4,129	JobPosting	1
3	LocalBusiness	0.86	0.71	65,691	0.74	0.82	0.83	0.82	126,220	Product	9
4	Offer	0.49	0.82	73,907	0.72	0.81	0.72	0.84	65,691	LocalBusiness	3
5	MusicRecording	0.42	0.77	12,848	0.66	0.77	0.69	0.79	34	EducationEvent	14
6	Movie	0.38	0.38	3,407	0.38	0.68	0.63	0.69	10,395	Place	8
7	Review	0.53	0.32	4,551	0.35	0.68	0.59	0.70	4,551	Review	7
8	Place	0.47	0.30	10,395	0.32	0.58	0.54	0.60	1,153	MobileApplication	13
9	Product	0.92	0.26	126,220	0.30	0.57	0.52	0.59	14,398	Article	12
10	Blog	0.11	0.16	10,622	0.15	0.46	0.44	0.46	3,407	Movie	6
11	Organization	0.52	0.10	10,558	0.12	0.46	0.41	0.47	12,848	MusicRecording	5
12	Article	0.06	0.05	14,398	0.05	0.45	0.43	0.46	4,725	WebPage	15
13	MobileApplication	0.01	0.04	1,153	0.02	0.41	0.41	0.41	15,856	Event	2
14	EducationEvent	0.00	0.25	34	0.01	0.39	0.38	0.40	10,558	Organization	11
15	WebPage	0.00	0.00	4,725	0.00	0.16	0.17	0.16	73,907	Offer	4

probable that approaches relying boosting meta-algorithms like the well know AdaBoost [40] will provide even better results. But for the moment, such a system is able to match schemata correctly on average in 2 out of 3 cases. However, the overall quality of the results doesn't encourage us to believe in the feasibility of a fully automatic system extending the coverage of schema.org. Instead we believe that interactive user support is necessary. Following on this idea, in [56] we present SASS (Schema.org Annotation Support System) an approach that goes beyond the GUI tools offered by Google's Structured Data Markup Helper or state-of-the-art systems like presented in [67] to offer users the necessary support for annotating data with schema.org.

In conclusion, until schema.org gains traction, its contribution to entity search is minor. Even so, like in the case of LOD, using annotated data for entity search requires that duplicate entities can be identified. There are quite a few systems which have achieved remarkable results for this classical instance matching task ([89, 133]), subject we discuss in more detail in the following section.

2.3. Instance Matching

The problem of finding identity links (owl:sameAs) between identifiers of the same entity in various data stores has been heavily researched (see [15, 61, 64, 89, 118, 124]). Various systems have been proposed and the reported results are, with precision and recall of over 0.9 very promising. As such, we are encouraged to believe that the problem of instance matching can be solved with any of these "out-of-the-box" systems. There is plenty to choose from, so we started looking for an instance matching approach to integrate into our entity retrieval system.

At their core, instance matching systems build on one or more of the following techniques: probabilistic matching, logic-based matching, contextual matching, or heuristic matching based on natural language processing (NLP). Each approach shows strengths and weaknesses. But these particularities are hard to assess, since each system was evaluated on different data samples. The choice of data for the evaluation has a big influence on the results. For instance, there is a large number of class equivalence links between DBpedia and YAGO. If these two data sources build a large portion of the evaluation data then approaches like the one presented in [15] are favored. The verbose nature of the URIs also helps shallow NLP techniques favoring for in-stance the system presented in [89]. The situation is different for other selections like LinkedMDB and YAGO since the URIs provided in LinkedMDB are more cryptic and links to and from YAGO are rare.

Of course, instance-matching approaches have to be able to work with all kinds of entities from multiple data-stores. Again, this may boost the performance of some systems, since different aspects of an entity can be learned iteratively from various stores. On the other hand it can be detrimental to the overall data quality, since the more entities and entity types are available, the more probable it becomes for systems to generate incorrect identity links. Take for instance LINDA [15] which heavily exploits transitive links to support the inter-linking process. When it was evaluated on the Billion Triple Challenge corpus comprising entities from various stores the respective precision was about 0.8. For relaxed similarity constraints the precision even drops to 0.66. But with every third identity link being incorrect, this level of quality does not seem satisfactory. It will lead to wrong type connections in the LOD heavily affecting the precision of retrieved entities and it will falsely remove entities it wrongly identifies as duplicates. In contrast, SLINT+ [89] reports an average precision of 0.96 on DBpedia and Freebase data.

But does this really mean that SLINT+ performs better? The respective precision was achieved on a biased set, representing a highly inter-linked extract from DBpedia and Freebase! It is therefore impossible to directly compare the performance of the two systems. To make systems comparable to one another, the Ontology Alignment Evaluation Initiative (OAEI) organizes a yearly evaluation event including an Instance Matching track. For the last year's evaluation there were evaluation tests involving data value differences, structural heterogeneity and language heterogeneity. With small data value and structure alterations and involving a small extract (1744 triples and 430 URIs) from a single high quality data source (DBpedia), we will show that the tests do not accurately reflect the problems encountered in real-world data.

Actually, judging by the 2013's OAEI evaluation results (showing again a sustained precision of over 0.9), instance matching systems seem to work very well. But considering the modest precision achieved by systems like LINDA on real-world data, this raises the question: Is instance matching ready for reliable data inter-linking? To answer this question, we perform extensive real-world experiments on instance matching using a system which has proven very successful in OAEI tests. Published in [57], ours is the first study that provides an in depth analysis over how effective instance matching systems are on real-world data.

Putting Instance Matching to the Test

Instance matching is about finding and reconciling instances of the same entity in heterogeneous data. It is of special interest to LOD because the same entity may be identified with different URIs in different data stores and the owl:samesAs property useful for interlinking URIs of the same entity is not as wide-spread as needed.

In the context of LOD, given multiple sets of URIs D_1, D_2, …, D_n, with each set comprising all unique URIs of a data store, *matching* two instances of an entity can formally be defined as a function *match*:*URI*×*URI*→{false, true} with:

$$match(URI_i, URI_j) := \begin{cases} true, if\ sim(URI_i, URI_j) > \theta \\ false, otherwise \end{cases} with\ URI_i \in D_i\ , URI_j \in D_j \qquad (5)$$

where $1 \le i, j \le n$, and *sim()* is a system dependent, complex similarity metric involving structural, value-based, contextual and other similarity criteria, and θ is a parameter regulating the necessary quality level for a match.

Based on this function, instance matching systems build an *equivalence class* for each entity. An equivalence class comprises all URIs used by any source to refer to some corresponding unique entity. For instance, considering only DBpedia, Freebase, YAGO and LinkedMDB, the equivalence class for the entity "Martin Scorsese" is:

{http://dbpedia.org/resource/Martin_Scorsese,
http://yago-knowledge.org/resource/Martin_Scorsese,
http://rdf.freebase.com/ns/m.04sry,
http://data.linkedmdb.org/resource/producer/9726,
http://data.linkedmdb.org/resource/actor/29575,
http://data.linkedmdb.org/resource/editor/2321}.

It's worth noticing that in contrast to general purpose knowledge bases like Freebase or DBpedia, specialized data stores like LinkedMDB have finer granularity, differentiating between Martin Scorsese as actor, editor, or producer. According to the owl:sameAs property definition in the OWL standard, all URIs referring to the same real world object should be connected through owl:sameAs. In consequence, all six URIs from the previous example should be linked by owl:sameAs relations. Of course one could argue that finer, context-based identity is required and that "Martin Scorsese, the producer" may not be the same as "Martin Scorsese, the actor". For further discussions regarding context-based similarity and identity see [48]. Here we adopt the definition as provided by the OWL standard for the owl:sameAs property.

Instance matching is an iterative process. Once some of the instances are matched either manually or by some system and owl:sameAs links have been established, more identity links can be found by exploiting the transitivity inherent in identities: Given that URI_A and URI_B represent the same real world object, the same applying for URI_B and URI_C implies that also URI_A and URI_C represent the same real world entity. Consequently, an owl:sameAs link between URI_A and URI_C can be created. However, the actual process of discovering sameAs links *is based on some similarity function and not on identity*. Similarity functions, however, are usually not transitive!

Let us give a simplified example where the matching function relies on the Levenshtein distance on the rdfs:label property as similarity metric. Consider that a URI with rdfs:label "Scorsese, Martin" referring to the well-known movie producer, is matched with a URI with rdfs:label "Scorsese, Cartin" (which could be a typo). This last URI matches a URI with rdfs:label "Scorsese, Chartin" and the match process goes on up to a URI with rdfs:label "Scorsese, Charles". Charles Scorsese is an actor known for his role in Goodfellas and actually Martin's father. This problem is well known in the area of single link clustering: similarity clustering may lead to chains of URIs where neighboring URIs in the chain are similar, but for long enough chains the ends of the chain have almost nothing in common. Linking the URIs of Martin and Charles Scorsese with owl:sameAs would obviously be incorrect. Of course this example is constructed, but the danger of transitively matching unrelated

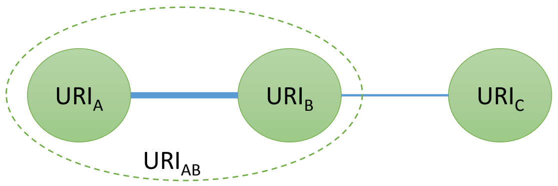


Fig. 13. Three URIs matching in a chain (URI$_A$ and URI$_C$ don't match). The similarity between URI$_A$ and URI$_B$ is stronger than the similarity between URI$_B$ and URI$_C$.

instances in the context of large amounts of data is real. In consequence, evaluation data involving triples from multiple stores is necessary for exposing such weaknesses.

From the instance matching systems we found that only LINDA specifically addresses the problem of transitivity and selects only those matches consistent with transitivity as follows: on the example in Fig. 13, considering that $sim(URI_A, URI_B) > sim(URI_B, URI_C)$, the equivalence class of URI_A comprises only URI_B and vice versa, i.e. both URIs refer the same entity and all properties valid for URI_A are also valid for URI_B and all properties valid for URI_B are also valid for URI_A. To express this we can denote the entity referred by URI_A and URI_B through URI_{AB}. Even though URI_A and URI_C don't show a large enough similarity, they are considered to refer the same entity if $match(URI_{AB}, URI_C)$ is true. Then, URI_C will also be added to the equivalence class. The process of finding identity links continues iteratively up to convergence.

Borrowing from hierarchical clustering, also the complete-linkage criteria could for instance be easily adopted to enforce transitivity. Assume after pairwise comparing all URIs we find three URIs matching in a chain like presented in Fig. 13. Any set of *n* linked URIs satisfies the complete-linkage criteria, iff all *n* URIs match in a pairwise. Obviously this is not the case for chains. In consequence, chains are broken up by removing the weaker links. In the case of links of equal strength one of them is broken at random. Consider $sim(URI_A, URI_B) > sim(URI_B, URI_C)$. Since $match(URI_A, URI_C)$ is false, the link between URI_B and URI_C has to be removed. As a rule, the list of URIs being *weakly linked* to an URI_x is:

$$\begin{aligned} WL_{URI_x} = \{URI_y \mid \exists\, z: match(URI_x, URI_z) = true \\ \wedge\ match(URI_y, URI_z) = false \\ \wedge\ sim(URI_x, URI_z) \geq sim(URI_x, URI_y)\}. \end{aligned} \tag{6}$$

After all weak links are broken for all URIs, the *equivalence class* of an URI is given by a function *E:URI→{URIs}* where:

$$E(URI_k) := \{URI_l \mid match(URI_k, URI_l)\} \tag{7}$$

Instance Matching - State of the Art

Instance matching is crucial for several applications like data integration, identity recognition and more important, for entity type alignment. Recognizing the lack of evaluation data, OAEI provided a reference benchmark for ontology alignment since

2004. Probably fostered by advances in Linked Data, four years later, [3] is one of the first publications to address this problem for instance matching. The authors discuss the particularities of instance matching and name main challenges. Based on these challenges, they design a benchmark with movie data from IMDb that emphasizes on data value differences, structure and logical heterogeneity. Finally, they compare the results for two instance matching algorithms to show the applicability of the data set.

In 2009, OAEI introduced an instance matching track and provided first generated benchmarks[37]: one comprising three datasets with instances from the domain of scientific publications built on Digital Bibliography & Library Project (DBLP), one with three datasets covering several topics, structured according to different ontologies from DBpedia and one generated benchmark obtained by modifying a dataset according to the data value, structure and logical heterogeneity criteria introduced in [3]. Evaluation data has gradually improved and last year's benchmark comprised five test cases: One for value transformation, where the value of five properties was changed by randomly deleting or adding characters; one for structure transformation, where the length of property paths between resources and values has been changed; a languages test where comments and labels were provided in French instead of English; one set combining value and structure transformation using French text and one where besides the value, structure and language challenges, some entities have none or multiple counterparts (a cardinality test). The data for the tests was extracted from DBpedia: it comprised 1744 triples, 430 URIs and only 11 predicates. It involves only one type of entity: personalities from the field of computer science like Alan Turing, Donald E. Knuth, or Grace Hopper and is limited to triples having such personalities as a subject. Four instance matching systems have been evaluated on this benchmark. Out of the four, SLINT+ [89] and RiMOM [118, 119, 133] achieved outstanding results with an average precision and recall over all test of more than 0.9.

While these results are quite promising, similar systems have proven weaker performance on real-world larger in size and involving multiple data stores. To assess the performance of such systems with real-world data, we built an evaluation set comprising 90,000 entities, from four domains, extracted from five data stores. In contrast to the OAEI test cases, all domains were included in all tests rendering cross-domain false positive matches (e.g. person being matched to movie) possible. The data stores were all-purpose knowledge bases like DBpedia and Freebase as well as domain focused stores like LinkedMBD and DrugBase. Some sources have cryptic URI naming conventions while some are more explicit. Also the granularity of properties varies between sources. We believe this is a more appropriate way of measuring the success of instance matching algorithms.

[37] http://oaei.ontologymatching.org/2009/instances/

Evaluation

For evaluating instance matching systems we rely on real-world data comprising entities of types Person, Film, Drug and Organization. The data was extracted from five stores: Freebase, DBpedia, LinkedMDB, DrugBase and NewYork Times. A detailed description of the data set is presented in **Table 7**. Instance matching systems are quite resource demanding ([15, 89]). For this reason, the evaluation data has a manageable size of about 90 thousand entities. This translates to about 4.9 million triple representing all relations having one of the selected entities as a subject. Such volume can be matched in a matter of minutes on commodity hardware. A similar number of entities was selected from each data store. The size difference between entity types was considered, too: overall, the data set comprises about 35 thousand entities of type person, 30 thousand entities of type film, about 15 thousand drug entities, and about 8 thousand organizations. To emphasize data value problems, entities were selected after alphabetically ordering them on their labels. This way, almost all entities have labels starting with the letter 'A'. Due to the small number of entities, DrugBase and NewYork Times have been selected in full. The number of properties per entity type is, with a maximum of 2,537 unique properties for persons, notably higher than in the OAEI tests. This stresses out structure heterogeneity of real-world data. The ontology differences between data sources, different aggregation levels introduced by LinkedMDB, or the fact that persons are being matched with actors add to the challenges this data set poses. Furthermore, in contrast to OAEI tests, having data form multiple stores increases the risk of building wrong transitive links. At the same time, the fact that multiple domains are compared, the possibility of creating bad links between entities of different types also exists. Finally, the selected data is not heavily interlinked. There are 5,855 owl:sameAs links between entities in our data set. 5,264 of them are between DBpedia and Freebase entities, 548 between DBpedia and LinkedMDB entities and 43 between entities from DBpedia and the NewYork Times.

To assess the quality of instance matching systems, we performed instance matching on the data presented in the previous chapter and measured *sampled precision*.

Table 7: Number of entities and properties per data store and entity type.

Types	Freebase	DBpedia	LMDB	NYT	DrugBase
	#entities / properties				
Person	10,000 / 1,006	10,000 / 2,537	10,000 / 10	4,979 / 11	0
Film	10,000 / 465	10,000 / 565	10,000 / 48	0	0
Drug	5,000 / 435	5,000 / 247	0	0	6,712 / 36
Org.	5,000 / 641	0	0	3,044 / 11	0
#entities	30,000	25,000	20,000	8,023	6,712
#triples	1,749,433	2,461,263	264,902	90,850	314,108

Table 8: The number of owl:sameAs links, the number of owl:sameAs links between entities of different types, and the precision of the links created with SLINT+ and by computing the transitive closure of links created by SLINT+, respectively.

θ	SLINT+			cl_{TR}		
	#sameAs	Inter-domain	Prec.	#sameAs	Inter-domain	Prec.
0.95	8,020	33	0.91	2,055	89	0.20
0.75	16,739	119	0.71	5,498	216	0.15
0.50	17,436	230	0.76	7,038	396	0.09
0.25	25,113	1,734	0.67	14,879	2,408	0.02

We computed the transitive closure of the resulting owl:sameAs links and measured the quality of the newly created links. We paid special attention to the resulting equivalence classes as well as to entities of different types that have been matched. All tests were performed for high to low similarity thresholds. Since one of the characteristics of the data set was that it is not highly interlinked, there were not enough owl:sameAs links available to also measure recall.

The instance matching system is a black box from our perspective. Any domain independent system can be used. SLINT+ is one of the systems to achieve exceptional results in instance matching tasks. It is training-free and domain-independent. It builds on thorough predicate alignment and selection, shallow NLP and correlation based instance matching. It has already been successfully tested on selections from DBpedia and Freebase and it is available online for download[38].

For a similarity threshold of 0.95, SLINT+ creates 8,020 owl:sameAs links (see **Table 8**). 33 of them link drugs or movies to persons. They are obviously wrong. Overall, we observed a sampled precision of 0.91 for this threshold. The lower the similarity threshold, the more links are found. For a similarity threshold of 0.25, 25,113 links are found. Even for such a low similarity threshold the precision is with a value of 0.67 quite impressive. According to the OWL standard, owl:sameAs links are transitive. Like most instance matching systems, SLINT+ ignores this aspect, probably because few bad links may lead to an explosion of bad links through transitivity. On the other hand completely ignoring transitive links is dangerous since any query engine using the links created by SLINT+ may transitively link sources to solve join queries. Computing the transitive closure of the owl:sameAs relations discovered by SLINT+ for a threshold of 0.95 we obtained an additional 2,055 links. However, the precision measured for these transitive links is only 0.20.

But how is this possible? Due to the non-transitive nature of the similarity function, long chains of entities belonging to the same equivalence class may be created. The longer the chain, the higher the probability that URIs that are far apart in the

[38] http://ri-www.nii.ac.jp/SLINT/index.html

chain refer different entities. Even for high precision oriented similarity thresholds like 0.95, SLINT+ produces 11 equivalence classes with more than 10 URIs each. Actually, the largest equivalence class has 23 URIs, while for lower similarity thresholds there are equivalence classes with 38 URIs (see **Table 9**). One false owl:sameAs link connecting two smaller equivalence classes in such a large class creates a huge explosion of false links. Assuming two equivalence classes each having 10 URIs, one false link created by SLINT+ connecting the two classes may generate up to 100 incorrect links (all pairwise combinations developing between the two classes: C_2^{20}-$2 \cdot C_2^{10}$). Considering the high precision for 8,020 links but the low precision for all transitive links, the real, overall precision achieved by SLINT+ for a threshold of 0.95 is $\frac{8{,}020*0.91+2{,}055*0.20}{8{,}020+2{,}055} = 0.77$ and thus quite comparable to LINDA and not acceptable of connecting type URIs in LOD.

Not knowing all owl:sameAs links for all entities from our data set it is impossible to accurately measure *recall*. However, if we take into consideration that 25,113 entities were found with a precision of 0.67 and that an additional 14,879 were found with a precision of 0.02, we can assume that the data set should have, when correctly interlinked, at least 17,123 links (25,113 * 0.67 + 14,879 * 0.02). Assuming that 8,020 * 0.91 + 2,055 * 0.20 = 7,709 correct links have been discovered for a threshold of 0.95, this translates into a recall of at best 0.45. This is much lower than the results observed on the OAEI benchmark. We observed similar results for other instance matching systems too. In **Table 10** we present the precision values obtained by PARIS [115], another leading instance matching system.

Table 9: Number of equivalence classes per number of URIs in the equivalence class, for various similarity thresholds.

#URIs per class	# equivalence classes			
	θ=0.95	θ=0.75	θ=0.5	θ=0.25
2	4,168	5,054	7,008	8,180
3	529	1,160	2,023	2,781
4	54	222	315	648
5	15	110	136	303
6	7	49	67	167
7	1	24	38	89
8	4	22	22	52
9	5	12	17	43
10	2	11	12	27
11	2	4	8	13
12	2	8	9	9
13	0	1	3	12
14	0	3	1	7
15	1	6	4	6
16	1	1	3	5
17	0	1	3	7
18	1	1	2	4
19	1	1	2	1
20	1	2	2	4
21	1	1	2	2
22	0	1	2	3
23	1	1	1	2
24	0	0	2	1
27	0	1	0	1
29	0	1	1	1
31	0	0	0	1
38	0	0	1	1

Table 10: The number of owl:sameAs links created with PARIS, the corresponding precision value, and the links obtained by computing the transitive closure of links created by PARIS, respectively.

θ	PARIS		cl_{TR}	
	#sameAs	Prec.	#sameAs	Prec.
0.95	24,771	0.93	2,194	0.54
0.75	29,098	0.84	3,401	0.54
0.50	34,077	0.78	5,427	0.38
0.25	36,423	0.64	6,523	0.25

To sum up, on first sight, the results for today's instance matching systems seem quite impressive. But if the problem of transitivity is not properly considered, even for very high similarity thresholds the precision on links obtained through transitivity is catastrophic. This has a high impact on the overall quality of the created links, making instance matching useless for LOD type alignment purposes as well as entity duplicate detection.

2.4. Conclusions

Entity search on structured data is a trivial task. LOD and schema.org are the prime candidates that come to mind when it comes to structured Web data. However, some particularities of these data sources make entity centric search on LOD and schema.org more difficult than anticipated:

- For LOD, access to entity data is provided through URIs. Unfortunately, URIs are not unique within the LOD. For this reason it is difficult to retrieve entities from the LOD as a hole. Our analysis on the example of entity type queries shows that without proper type alignment, for a given type, the user ultimately receives but a small fraction of the entities. Assuming that type alignment is possible for example through ontology alignment or type witnesses, another problem is reconciliation the resulting entities: coming from different data sources which may overlap in terms of entities, duplicates represented with different URIs may exist. They have to be detected and eliminated. Unfortunately, although it was recently claimed to be a solved problem, instance matching still faces major issues in that respect.

- Schema.org did not gain any traction. Machine learning techniques have long been used to compensate the lack of human input in tasks like automatic text classification. One could imagine a solution to extend the coverage of schema.org by learning from existing annotations. However, our experiments show that human input is still required in order to provide for a higher level of quality. Furthermore, like in the case of LOD, in order to put together a set of resulting entities, one has to eliminate duplicates. But without reliable instance matching, this task represents an important source of errors.

In consequence, until LOD is more strongly interlinked and the various vocabularies are better integrated, entity search on LOD should be limited to single data stores. Since data stores offer internal data consistency (URIs should be unique within one data store), and types are curated manually, this will result in high precision (ideally 100%), but probably low recall results for type based search. Schema.org could be interesting for entity search, if it reaches web scale. As we have shown in [56] machine learning is quite useful in that respect if used together with user feedback. But even so, the use of schema.org will always be hindered by instance matching issues that are even more difficult to solve that in the case of LOD because there are no owl:sameAs links and multiple instances of the same entity are not connected in any way between different Web sites.

Overall, we believe the approach followed by the Knowledge Vault to integrate data from all available Web sources including unstructured data to build a Web-scale knowledge base is a more promising approach. Each fact is associated with confidence values, allowing for questionable facts to be manually checked in a crowdsource fashion. Building on state-of-the-art OpenIE systems, in this thesis we follow a similar approach to prepare Web data for entity-centric search. A detailed description of the system is presented in Chapter 5.

Chapter 3

Entity Search Based on the Entity Type

Entity type queries are the most basic form of entity-centric search. The users state the entity kind and the system has to retrieve all entities of the given type. Running such queries on structured data is trivial. Initiatives like the LOD or schema.org aim to provide for a machine readable source of structured information that captures as much Web data as possible. But as we have seen in the previous chapter, both approaches have shortcomings. Furthermore, they both rely on ontologies to structure data. At the top level, their ontologies have few, abstract and all-encompassing entity types. For instance the top level in the DBpedia and the schema.org ontologies comprises just one entity type: ‘Thing’. Each type is narrowed down in hierarchical fashion (see Fig. 14) to a more detailed entity type in the next level. According to the theory established by Eleanor Rosch [81, 103–106], one can distinguish between three concept levels and therefore three entity types: the *superordinate* level or the level of categories of entities e.g. ‘furniture’, ‘vehicle’, ‘communication device’; the *basic* level e.g. ‘chair’, ‘car’, ‘cell phone’; and the *subordinate* level further specifying the categories of entities below the basic level e.g. ‘kitchen chair’, ‘sports car’ or ‘business cell phone’. Connecting this hierarchical representation to the principle of *economy and informativeness trade off* introduced by Loyd K. Komatsu in [69] for concept hierarchies, there are few entity types at the superordinate level (increasing economy). They are general entity types that group together entities having few things in common (decreasing informativeness). At the subordinate level, entities of the same type show similar characteristics, (increased informativeness). However, at this level there are much more entity types (decreasing economy).

Indeed, if we pay closer attention to the hierarchical structure of schemata on schema.org, the 406 schemata are organized on 5 levels. About 84% of the schemata (levels 4 and 5 in Fig. 15) are subordinate entity types. They comprise very specific

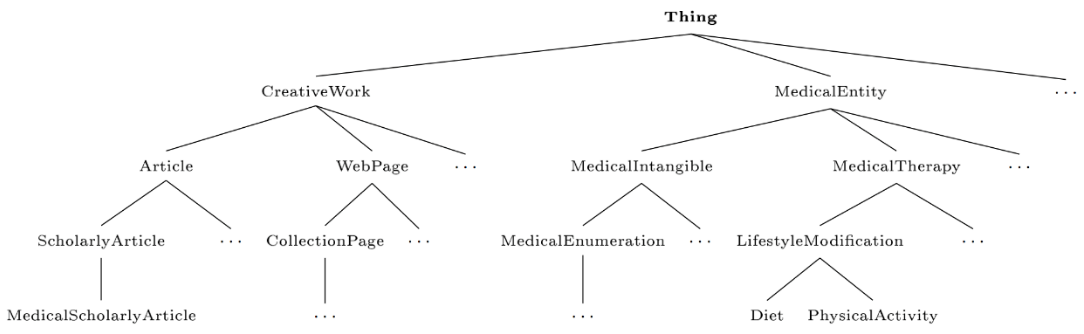


Fig. 14. Excerpt from the schema.org hierarchy.

entity types like ‘medical scholarly article’ or ‘day spa’. But despite the fact that schema.org is strongly oriented towards structuring product data for enabling product aware rich snippets, its structure provides no subordinate level entity types for any of our previous examples i.e. ‘kitchen chair’, ‘sports car’ or ‘business cell phone’. And this is not an isolated case. The large number of possible subordinate entity types, makes it is impossible to include all or most such entity types in the ontology.

But is there a need to support subordinate entity types? To answer this question, in [53] we studied the AOL Web search query logs (comprising logs of all searches done by 650,000 AOL users over the course of three months in 2006), with regard to our sample domain of cell phones. In particular, we extracted 21,650 cell phone relevant entries through the use of regular expressions. After manual inspection we classified all queries into six base categories (see Fig. 16). The resulting categories deal with:

- Products: represents about 22% of the queries. It contains queries related to brands, product prices, product features, specifications, and types e.g., ‘Motorola Razr’, ‘cell phone battery’, or ‘cell phone for kids’.
- Telecom & Pricing Plans: for example ’Verizon cell phones’ or ‘compare cell phone plans’. This category represents about 30% of the cell phone related queries.
- Accessories: represents 17% of the queries, and refers to products for cell phones e.g., ‘sexy phone wallpaper’ or ‘ringtones’.

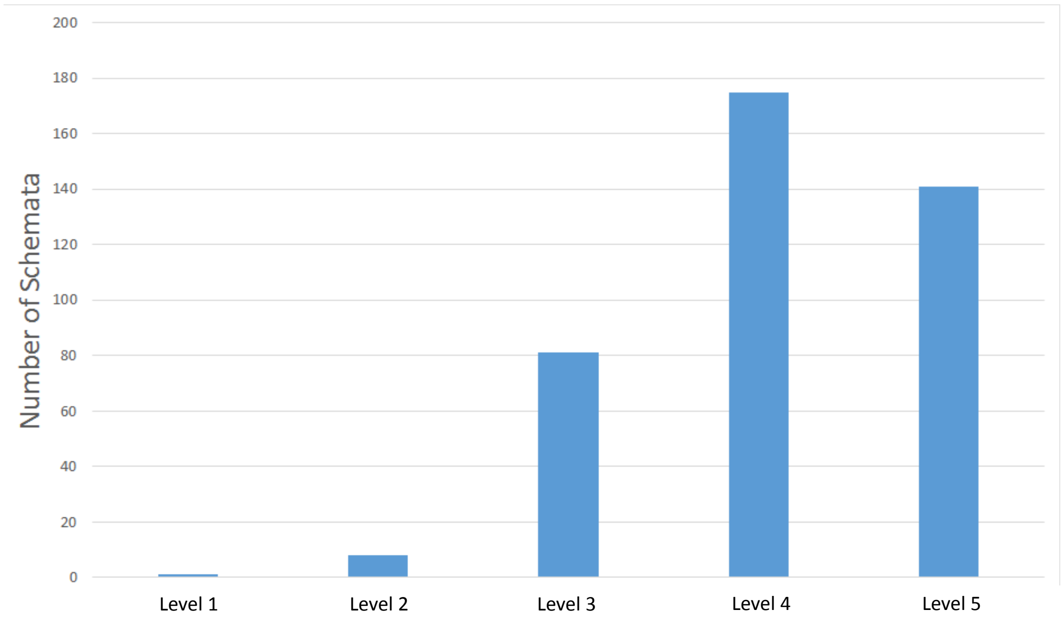


Fig. 15. The distribution of schemata from schema.org on hierarchy levels.

- Phonebook: 13% in size refers to cell phone numbers, or reverse phonebook lookups, e.g., 'cell phone number lookup'.
- Unspecific Queries: about 15% of all queries representing too general queries, usually simply 'cell phones'.
- Other: about 3% containing more exotic queries like 'help finding lost cell phone', or 'cell phone health risk'.

Focusing on the category with references to products, we observed that the majority of queries are either concerned with a specific brand (e.g., 'Motorola' or 'Sony Ericson phone') or the price (e.g., 'Nokia 5300 price'). Still, the amount of queries on subordinate entity types e.g., 'cell phone for kids', 'cell phone for seniors', 'business cell phone', 'fashion cell phone', 'camera cell phone', etc. is about 30% of all product queries for the 'cell phone' domain.

To summarize, on one side, the principle of *economy and informativeness trade off* strikes and sources for structured data on the Web can only provide for partial coverage of the sheer number of subordinate entity types, but on the other side, users do search for such entity types in volumes that cannot be neglected.

Another relevant finding is that some of the subordinate entity types represent simple attribute/value constraints over basic entity types. For instance the entity type 'science fiction movie' is composed out of the basic type 'movie' and the 'science fiction' genre constraint. If genre information is available, such a type can be constructed with the help of the 'movie' basic type. But things can get complicated pretty fast: such types can span over multiple attributes, without a clear definition of the weights of each attribute. For instance, the 'business' aspect of a 'cell phone' as an entity type is most certainly related to technical capabilities like the 'organizer/calendar', the 'email client', the 'battery life' or the 'qwerty keyboard'. But while

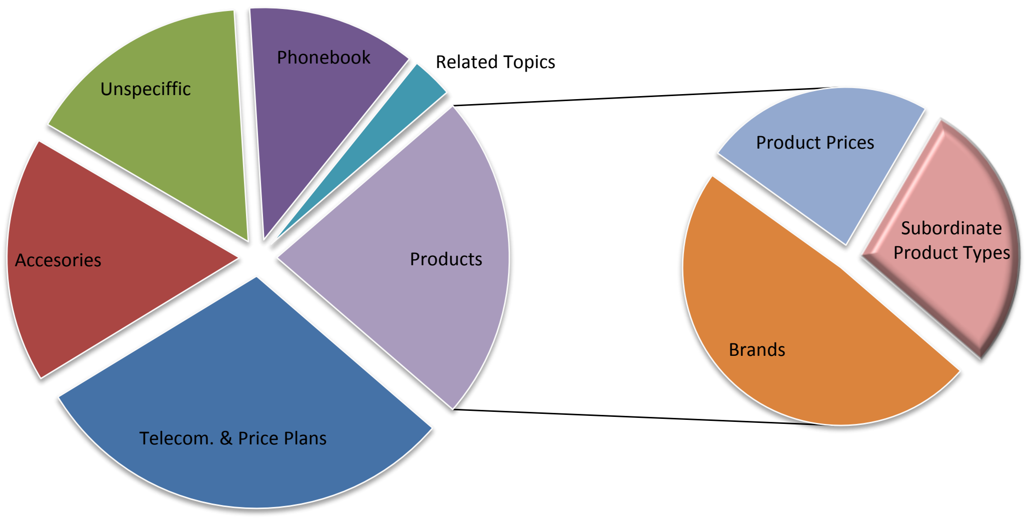


Fig. 16. Types of queries relevant to the 'cell phone' domain from the AOL query log.

users agree on which cell phones are fit for business use and which not, it is practically impossible to automatically construct the 'business cell phone' entity type from a structured data source comprising for instance technical specifications of 'cell phones' alone. We argue that a more sophisticated method mining entity types from both structured and unstructured data from the Web is required for solving this problem. In the following section we provide an elaborate description of a system that is able to cope even with such difficult entity types.

3.1. Mining Entity Types from the Web

In this section we analyze the possibility of using besides structured, also unstructured data to improve results. The main assumption is that recall can be improved by bridging structured data with the massive amount of unstructured data available on the Web. Furthermore, this allows for a more flexible approach in terms of unknown entity types: in the case of subordinate entity types, the problem for LOD and schema.org data is that most such types were not present in the underlying ontologies or vocabularies. With the integration of unstructured data, we expect that such types are also being used and appear in unstructured data.

Entity-type queries are simple and concise keywords. They bear information that is easy to understand for people due to common sense and background knowledge. But machines require complicated methods and large amounts of data to mine this implicit information from. This is especially important for supporting subordinate entity types, where concepts imply additional constraints over a basic entity type. In this chapter, we introduced the basic problem of supporting queries on implicit information on the example of the cell phone domain, where discussion boards regularly refer to flowery subordinate entity types like 'ideal for social networkers', 'perfect for fashionistas', 'tough as nails', or 'multimedia marvels'. But this is a general problem in entity centric search: recent results have been observed also in other domains like the predominant tagging of explicit media features, in contrast to the high number of queries on implicit (usage-based) features in online image repositories or music stores (see [12]). Therefore, being able to transform implicit information needs into explicit terms for querying is generally of vital importance for supporting entity type queries.

The challenge of implicit information needs has been discussed before and is directly addressed by some retrieval paradigms. However, experiments on real world data have shown that classical IR techniques like the vector space retrieval model (VSM [108]) and latent semantics (LSI [32]) don't achieve satisfying results [29]. On the other hand, query expansions, i.e. augmenting user queries with relevant semantically related terms, show promising results, if only the expansion terms are chosen in a sophisticated manner. While first approaches only focused on synonymy and term disambiguation, today, domain knowledge is incorporated. Expansion algorithms range from using simple lexical databases [125] like WordNet [84], existing domain ontologies [113] like the MeSH controlled vocabulary (a thesaurus used for indexing medicine related articles), to extracting language models e.g., probabilistic models based on term co-occurrence [97] directly from text.

The main challenge here to find those terms implied by the query entity type, which are appropriate for expansion. For instance, following on our running example from the domain of cell phones, a clear semantic connection between the 'business cell phone' concept and technical features like 'email clients', 'organizer', 'calendar', 'notepad' and 'file browser' could be established. Revisited in this section in detail, in [53] (extended in [54]) we present a novel query expansion method, which is able to solve the expansion problem for entity centric search based on entity types. With recall in mind, at its core, our approach builds on a self-supervised learning technique using both structured and unstructured data. But before going into more detail, in the following section we will lay the foundation of our approach providing a formal description of the problem at hand.

3.1.1. Task Definition

We rely on shallow semantics, a few simple, yet suitable, heuristics taking advantage of entity-related data available online in both structured and unstructured form. By unstructured data we mean entity related documents on the Web. In the best case these should be articles describing one entity each. For structured data sources we already met LOD and schema.org. The plan is to connect structured and unstructured data, for a complete view of entities, to ultimately boost recall. But, connecting linked data to Web documents, a process similar to the approach of matching schemata to pieces of text is, as we have shown in Section 2.2, an error prone process. Introducing errors at such an early stage, in the source data, would have severe impact on the end results and has to be avoided at all costs. Regarding schema.org, if it would be adopted on large scale, we wouldn't have this problem in the first place. Unfortunately, as previously shown, there is not too much annotated data.

However, looking at Web pages presenting various entities like for instance the description of some product, a cell phone, one can observe that more often than not, tables comprising information in structured form complement the textual description of the entities. Such tables are easy to recognize due to the HTML tag making it easy to extract. Web scale table extraction systems like the one presented in [24] confirm the success of this approach and large Web tables' corpora, comprising more than 147 million tables, are already available for public access[39]. In consequence, for this approach, we consider unstructured data in form of some textual description of an entity and structured data in form of tables corresponding to the same entity. The entities represent in this way the common ground between its unstructured and structured data.

[39] http://webdatacommons.org/webtables/

The task at hand is to support subordinate entity type search with a focus on recall, while keeping precision under control. Information in subordinate entity type search is implied through concepts. Although many real world queries use conceptual information, it is difficult to define what a concept actually is, and how it can be reliably spotted in queries. Psychology defines concepts as a cognitive unit of meaning, typically associated with a single meaning of a term [87]. Any term can therefore be the representation of a concept. The major importance of specific concepts in practical life comes with the generally consensual notions humans connect with some concepts: each concept carries connotations that immediately create an intuition about what is meant, and thus enable efficient communication. For example asking about a 'cell phone for kids' will immediately bring up ideas like robustness, ease of use, fun colors, security features, and parental control pricing plans. Explicitly adding exactly these connotations to a query is what makes applying a semantic query expansion technique so promising for good retrieval quality. However, lacking a clear definition, detecting conceptual features in queries is a serious problem. Whereas for explicit features like 'weight', 'size', or 'display type', new developments in declarative query languages already allow a mapping of previously unknown attributes to actually existing attributes in the underlying data (e.g., using malleable schemas [135]), the recognition of *implicit conceptual features*[40] like 'for kids', 'portability' or 'design' is much harder. Still, even if implicit conceptual features cannot be clearly defined and the exact disambiguation is beyond the scope of this thesis, detecting queries on entity types with implicit conceptual features is important for dealing with subordinate entity types.

With structured and unstructured data in mind, implicit features obviously can never be attribute names of structured data (otherwise they are simple, explicit constraints on entity types like 'science fiction movie'). Also in the respective set of values they should rarely occur for the same reason. Similarly, in unstructured text documents an implicit concept should occur not too often, either. But since texts are the usual way to communicate connotations and tie concepts to entities, any important implicit concept definitely should occur at least sometimes. In the following we consider that any noun (<N>) and nominal phrase (<NP>) from the query, is an implicit conceptual feature if it complies with Observation 1.

Observation 1: *Implicit Conceptual Feature*

Let x be any query term, S be the collection of structured data, $f_S(x)$ be the percentage of entities for which x occurs in values of structured data, D the set of documents presenting entities and $f_D(x)$ the percentage of documents grouped by entities explicitly mentioning x.

[40] We call such features *implicit conceptual features* because their main merit is that of implying and transmitting information to the user. It is exactly this implied information (product features in our running example), which has to be mined and made explicit to reach the relevant entities.

An implicit conceptual feature q is any query term for which $0 \leq f_S(q) \leq r$ and $s \leq f_D(q) \leq t$, where r, s and t are domain specific parameters.

For our cell phone example and the later experiments, we tried different values for r, s, and t. We found that occurrences under 5% in structured data and occurrences in between 2% and 10% of unstructured reviews are sufficient for detecting most implicit conceptual features without generating too many false positives. These parameters are collection/domain-specific, minor adjustments being necessary for other data collections and domains. Such adjustments may be for example performed on the run, by manually inspecting frequently posed queries from the query log.

Entity-related data available online comprises both structured and unstructured data. This only seems to complicate the problem to solve. But for our retrieval task this is not a problem, but rather a feature. This is on one hand because most concepts will only be explicitly tied to entities in unstructured texts, thus descriptive vocabularies can be derived by co-occurrence analysis. But also because many concepts are to some degree affected by certain structural characteristics, thus the statistical analysis and exploitation of value distributions can point to similar entities (e.g., 'portable' items will definitely show a bias towards smaller sizes and lighter weight). In fact, starting with a seed vocabulary for some relevant entities, and learning their structured characteristics to find similar entities, which in turn are used to expand the vocabulary and learn even more about the structural bias, will lead to a boosting like, cyclic improvement of a model that subsequently can be used for effective querying.

In summary, for implementing the query expansion of some initial implicit query term our approach requires the extraction of terms relevant to the intended concept from the underlying data, and thus has to bridge the gap between structured and unstructured information. The retrieval task can be formalized as follows:

Problem Statement: *Query Expansion for Implicit Features*

Given: A relational database S containing data with respect to entities $P_1, \ldots, P_N$. For each entity P_i, there also are text documents $D_{i,1}, \ldots, D_{i,n(i)}$ describing P_i.

Task: Given a user query Q containing an implicit conceptual feature q, derive an expanded query $Q' := q \cup \{q_1, q_2, \ldots, q_k\}$, where $q_1, q_2, \ldots, q_k$ are terms from S and D which explicitly describe q (with corresponding weights $w_1, \ldots, w_k$).

3.1.2. The Query Expansion Process

The problem of querying for implicit conceptual features is typically solved by using a query expansion technique. The key task however, is the selection of the right terms for expanding the query. An intuitive approach would be to consider for the expansion all the terms occurring together with the queried concept in the entity data. But the number of such terms is quite high, and although the query expanded in this manner leads to high recall, the precision is catastrophic with almost any product qualifying as a result. In consequence, to avoid such behavior, we first choose a set of candidates that appear together with the query in entity data, then we calculate the weight of each candidate term based on a function similar to the term co-occurrence and finally we select only those terms with the highest weights.

Choosing the Candidate Terms for Query Expansion

Each entity is described through one or more text documents and one or more Web tables. Of course any term appearing in documents or the tables could be of interest for the expansion. But particularly in the case of the documents, it's obvious that many of the terms have no relation with the query. Actually, in the case of our 'business cell phone' example, concepts mostly relate to some product features. Thus any term expressing a feature and co-occurring with the queried concept in the entity data is considered a candidate for the query expansion. Two steps have to be performed for choosing the candidate terms: first, *select the query relevant entity data* (data in which the queried concept appears) and second *extract the entity features* from the selected data.

a) *Selecting the query relevant entity data.* A document is likely to be relevant if the query is mentioned in it. But not the same can be said about the structured data. For products, we found numerous cases where a product manufacturer would include some task based concept in the product name, model or series, although the product is not a good candidate for the concept. However, if the query is explicitly mentioned in a document, then the technical specification of the corresponding product is also relevant with respect to the query.

The process of selecting query relevant data works as follows: documents containing the query are selected and considered relevant. Then the entities described by those documents are selected as well together with the corresponding structured data. This way, relevant documents, entities and structured data are selected. In a second pass, in a boosting fashion, unstructured data being similar (by means of classic string similarity functions, specified later) to documents already found as relevant, are also added to group in transitive fashion along with the corresponding entities and their structured data. All data is separated this way in two classes: class *c* representing data - documents (D_c) and tuples from the structured data (S_c) - being highly relevant with respect to the query and $\bar{c}$ representing the remainder of the data. In technical terms, in order to distinguish between relevant and irrelevant documents we represent each document as a vector according to the vector space model: terms of all documents represent axes of the space and projections of documents on each axis are computed with the help of the Term Frequency-Inverse Document Frequency (TF-IDF) measure [44]. The similarity between two documents is computed according to the well-known cosine metric (further denoted as *cos*) [108].

This being said, we can proceed to elaborating on D_c and S_c:

$$D_c = \{d_i | d_i \in D \wedge \exists d_j \in D'_c \text{ s.t. } cos(d_i, d_j) \geq \theta\} \quad (8)$$

where θ is a collection specific parameter regulating the precision of *c*, and *cos* represents the cosine similarity measure;

$$D'_c = \{d_j | d_j \in D \wedge d_j \text{ contains } q\} \quad (9)$$

P_c is the set of entities whose textual descriptions have been found as relevant to the query and S_c are the corresponding structured data:

$$P_c = \{p_i | p_i \in P \wedge \exists d \in D_c \; with\; d\; describing\; p_i\} \quad (10)$$

$$S_c = \{s_i | s_i \in S \wedge \exists p_i \in P_c \text{ s.t. } s_i \; tech.specs.of\; p_i\} \quad (11)$$

Accordingly $\bar{c}$ comprises $D_{\bar{c}} = D - D_c$ and $S_{\bar{c}} = S - S_c$.

b) Extracting entity features. In the case of unstructured data product features are usually represented through nouns and nominal phrases [60]. Some adjectives can also imply product features e.g., 'heavy' may imply the 'weight', but these are rather infrequent cases. Consequently, in order to extract the candidate terms, we applied standard natural language processing (NLP) techniques like part-of-speech tagging (POS) and chunking. Word inflections have been eliminated by means of stemming.

In structured data, entities are described through table attributes and the corresponding values. While all attributes are entity features, from the values we only considered the ones corresponding to categorical attributes. Obviously all values in define a certain aspect of the entity but the categorical attributes bear most of the differentiating force. Typical examples of such values are 'nokia', 'apple', etc., for the 'brand' attribute, or 'candy bar', 'clam shell' for the 'form factor' attribute. Numerical values like in the case of the 'price' or 'weight', have dynamically been reduced to the ordinal values 'low', 'average' and 'high'. We established the 'average' interval of the values for an attribute as being between [average of the values – one standard deviation, and average of the values + one standard deviation]. We then set the 'low' and 'high' intervals accordingly. Although they are not candidate terms, and will not be included amongst the query expansion terms, these ordinal values allow us to establish the weight of their corresponding attribute. In this manner we can for example find out that the 'weight' is an important factor since most of the devices which are explicitly relevant toward the conceptual query, fall into just one of these intervals (say 'low weight'), while the remaining products are spread amongst, or fall into the other two intervals.

Finally, after establishing what *entity features* and *query relevant data* stand for, we can formally define the set of candidate terms:

Definition 1: *Candidate Terms (CT)*

Let CT_D and CT_S be the set of query expansion candidate terms from documents and structured data respectively, with:

$CT_D = \{t_i | (POS(t_i) =< N > \vee POS(t_i) =< NP >) \wedge (t_i \subseteq d, \text{with } d \in D_c)\}$ and

$CT_S = \{t_i | t_i\ table\ attr.from\ S\} \cup \{v_{t_i} | (\exists v_{t_i}\ value\ of\ attr.t_i\ in\ S_c) \wedge$

$\wedge\ (t_i\ categorical\ attr.)\}$

where, *POS (t$_i$)* represents the part of speech of term *t$_i$*, and <*N*> and <*NP*> tags represent the noun and respectively nominal phrase parts of speech.

We define the set of *candidate terms* as: $CT = CT_D \cup CT_S$.

Calculating the Weight of Candidate Terms

Associating the candidate terms with the *right* weights is crucial for the entire process. The weight of a term must reflect the term's contribution to describing the queried concept. We estimate the weight of a candidate term by relying on a document classification approach introduced in [131]. The basic idea is to give higher weight to candidate terms appearing quite often in data from *c* and not that often in data from $\bar{c}$.

Definition 2: *Weighting Function (W)*

Let ct_i be any candidate term from the candidate list *CT*. $n_c(ct_i)$ and $n_{\bar{c}}(ct_i)$ represent the number of documents (if ct_i was extracted from unstructured data) or tuples (if ct_i was extracted from structured data) that contain ct_i from *c* and respectively $\bar{c}$.

The weight of ct_i, denoted $W(ct_i)$ is estimated by calculating the difference between the normalized frequencies of ct_i in *c* and $\bar{c}$:

$$W(ct_i) = \frac{n_c(ct_i) - min_c}{max_c - min_c} - \frac{n_{\bar{c}}(ct_i) - min_{\bar{c}}}{max_{\bar{c}} - min_{\bar{c}}}, \qquad (12)$$

where the components of the normalizing factors max_c and min_c are the number of documents, or by case tuples, containing the most frequent and respectively least frequent entity feature from c. $max_{\bar{c}}$ and $min_{\bar{c}}$ are analogously defined, with the most frequent and respectively least frequent feature from $\bar{c}$.

NB: For the candidate terms which were extracted from structured data, the weight of an attribute is calculated by considering also the corresponding attribute values. *ct$_i$* is being extended in this case to the attribute-value pair. To clarify, the weight of the 'price' attribute, which may be selected as a candidate term, will be calculated as the maximum out of three weights, one for 'low price' one for 'average price' and one for 'high price'. Of course in the case of numerical attributes, this is only possible if the values have previously been transformed to ordinals based on their average values and standard deviation (as previously discussed in this section).

The key factor in the weighting function is that the weight of each term is normalized with respect to typical terms (the most frequent entity features) from both c and $\bar{c}$. This is critical because $|c| \ll |\bar{c}|$. In this way important candidate terms with implicit connection to the queried concept aren't severely penalized despite appearing also in $\bar{c}$.

But since we have split the data into two classes why not apply classical supervised machine learning techniques on this automatically generated training set and train a classifier? As argued in [43] and as shown later on in the evaluation section, classical IR techniques like VSM are not able to retrieve many of the eligible products. Therefore both $D_{\bar{c}}$ and $S_{\bar{c}}$ contain data which is implicitly relevant regarding the query. For this reason, classifiers like SVM or decision trees are not an option (see [131] for further details). Furthermore, typical weight measures associated with *discriminative feature weighting* like term co-occurrence, mutual information or information gain tend to excessively penalize important terms due to the noisy initial classification.

Selecting the Expansion Terms
Having calculated the weight of all candidate terms we are now ready to choose the most appropriate terms for query expansion. Taking a closer look at the weighting function, the candidate terms are associated values between [-1; 1]. As intuitively expected, there are few very week candidate terms, with weights close to -1, many general terms, with similar normalized appearances in c and $\bar{c}$ and weights close to 0, and some strong candidate terms with values closer to 1. For the query expansion, we chose the candidates with the highest weights according to the 'three-sigma rule' [95] (average plus three standard deviations).

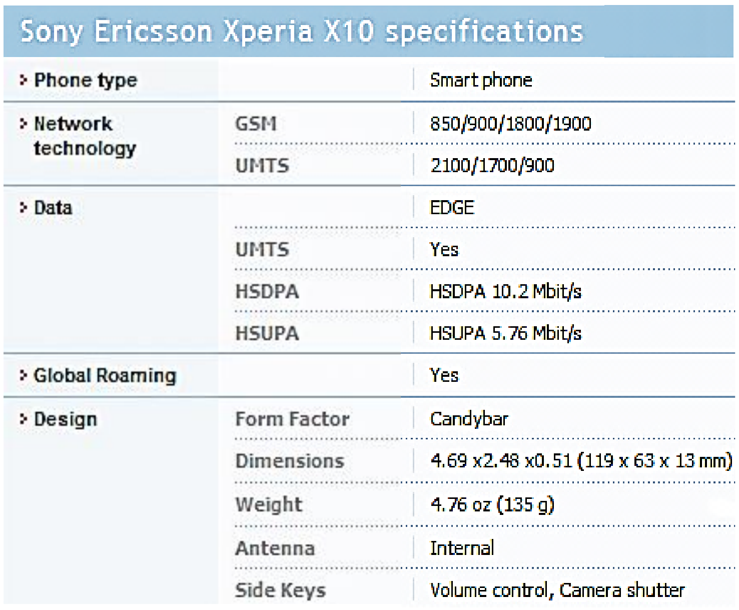
3.1.3. Evaluation

Evaluation methodology
Query expansion is a classical method for improving the retrieval performance of IR techniques. For evaluating purposes, we compared results with the well-known VSM featuring TF-IDF with cosine similarity. LSI is a promising technique for indexing and retrieving documents in a low-dimensional concept space by making use of semantic connections between terms. We address queries containing implicit concepts and as such we considered LSI is an important reference for our tests. Since the proposed approach is a query expansion technique, we also compared our method against Stephen Robertson's best match (BM25) [102] with Kullback-Leibler Divergence (KLD) [72] as probabilistic term weighting scheme, a widely accepted approach as being the standard method for weighting terms in query expansion. As metric we relied on the well-known Precision/Recall curves [123] emphasizing on precision/recall ratios at *k* (with *k* iterated from 1 to the number of all returned entities).

The evaluation process was the following: for each conceptual query, candidate terms for expansion were extracted according to Definition 1. All candidates were weighted with the function presented in Definition 2 and only those terms having weights greater than average plus three standard deviations were considered for the query expansion. With the query in expanded form, all products were ranked based on their relevance to the expanded query. The relevance of a product was computed as the sum of weights of the query expansion terms appearing in the unstructured data associated with the entity. As a gold standard we had domain experts tagging products with respect to prevalent concepts in the respective domain.

Evaluation Data

Consistent with our running example for subordinate entity types, for evaluating the query expansion method introduced in the previous section, we rely on product data from the field of cell phones. The structured part of the data comprises in this case, technical specifications of the products (an example of such data is presented in Fig. 17). For the unstructured data, text documents come in more flavors like for example editor's reviews, user reviews or blogs. Analyzing these information sources we observed that they offer different perspectives of the products. If editor's reviews presented the features and facts in a more objective manner, with extensive but field-relevant vocabulary, the user comments were smaller in size, concentrated on a reduced number of features, and were strongly influenced by the user's interests and point of view towards the entity. Blogs were even more emotional than user reviews making sentiment analysis an absolute requirement. Sentiment analysis however remains very unreliable when the text uses slang, sarcasm, emoticons, prolonged letter usage, capitalization, punctuation, etc. For this reason, we performed the query expansion process on a collection of 350 products with the corresponding technical specifications and 500 editor's reviews. The data has


Sony Ericsson Xperia X10 specifications

Phone type		Smart phone
Network technology	GSM	850/900/1800/1900
	UMTS	2100/1700/900
Data		EDGE
	UMTS	Yes
	HSDPA	HSDPA 10.2 Mbit/s
	HSUPA	HSUPA 5.76 Mbit/s
Global Roaming		Yes
Design	Form Factor	Candybar
	Dimensions	4.69 x2.48 x0.51 (119 x 63 x 13 mm)
	Weight	4.76 oz (135 g)
	Antenna	Internal
	Side Keys	Volume control, Camera shutter

Fig. 17. Structured data table comprising technical specifications for Sony Ericsson Xperia X10 extracted from phonearena.com.

been crawled from phonearena.com, a top Web publication in the field.

To test the quality of the expansion terms for different queries, we built a gold standard, comprising 50 products with corresponding technical specifications and 200 user reviews crawled from CNET[41]. We chose to evaluate on the more challenging user reviews to question the suitability of the expansion terms for non-expert typical user language. These products were manually labeled by experts in the field, as either being relevant or not with respect to three most important[42] subordinate entity types: ‘business cell phone’, ‘social networking cell phone’ and ‘camera cell phone’ based on their user reviews and technical specifications. We chose these features to cover different levels of clarity regarding the meaning of the query terms: ‘business’ represents ambiguous, classical concepts; ‘social networking’ stands for emerging concepts with well-defined use and finally ‘camera’ represents clear cut technical characteristics.

Discussion of the Results

Baseline: first we tested the base line methods i.e. VSM with TF-IDF, LSI and BM25 with the available data. To our surprise, LSI always obtained poor results even compared to VSM (see Fig. 18). Varying the number of dimensions for LSI (we evaluated with 10, 20 and 100 dimensions which according to [32] typically provide for good results) for all our test scenarios, didn’t bring any improvements. The reason for this behavior is the small amount of data available for training the LSI. The collection of

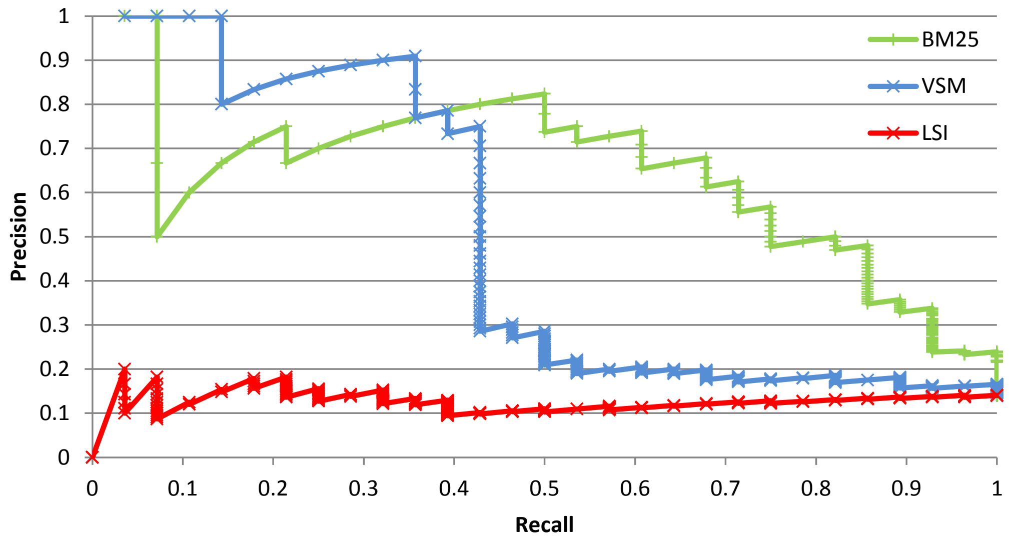


Fig. 18. ‘Business Cell Phone’ – LSI vs. VSM vs. BM25.

[41] http://www.cnet.com - a leading technology oriented Web site offering large amounts of both editor and user reviews for different products

[42] According to field experts, article presented on msn.com at http://tech.uk.msn.com/features/photos.aspx?cp-documentid=149711759

500 documents seems rather limited for the latent semantics needs. Editor's reviews are rather scarce resource, so we then increased the document base for LSI to 6000 documents, supplementing with user reviews. However, user generated documents do not offer similar advantages as editor's reviews do. Even with this large collection, LSI is still unable to achieve notable results. Collecting editor's reviews over long periods of time is also not a solution. The cell phone domain is a great example in showing how fast concepts evolve with time.

The TF-IDF based VSM retrieved all the products for which the conceptual query is explicitly mentioned in the description of products. This provided for quite good precision for low recall rates. But the precision deteriorated heavily in the case of products for which the query concept is only implied in the description. In the case of conceptual query 'business' presented in Fig. 18, VSM achieved good precision up to a recall of about 40%. The behavior of VSM becomes clear after taking a closer look at the data: 43% of the reviews the experts labeled as relevant towards the 'business' concept, explicitly mentioned the conceptual feature. VSM identified with a high precision exactly these documents. It is interesting to notice that there was a drop in precision at a recall of about 15%. The reason for this drop is that the word 'business' appeared in the description of some non-business products, e.g., "allows you to locate businesses nearby" tricking VSM into retrieving the product as relevant.

The BM25 ranking model assigns weights to all terms according to the KLD probabilistic weighting scheme. Only terms having the KLD weight above a certain threshold are used for expansion. In order to establish this threshold, we conducted a series of tests. Expanding the query with terms weighting more than the average KLD weight of all terms, provided the best results in terms of precision and recall for BM25. For the case presented in Fig. 18, BM25 with KLD identified 192 expansion terms out of which the top 10 terms were: 'business', 'bold', 'nexus', 'pure', 'webo', 'she', 'exchange', 'control', 'offer' and 'storm'. In terms of precision and recall, BM25 achieved less precision than VSM in the low recall area (up to 40% recall), but compensated more than enough by obtaining pretty high precision (about 50%) for recall rates as high as 80%. Since LSI doesn't even come close to the results of the other two techniques, in any experiment we performed, in the following graphs we will display only the more successful VSM and BM25.

Also worth mentioning is the 'saw-tooth shape' effect [19], common in precision/recall curves.

The query expansion technique: our query expansion comprises terms which have orthogonally been extracted from structured and unstructured data. But is using both of the underlying sources boosting recall or is using structured data just as good? To answer this question, we evaluated the results obtained by expanding the query with terms originating from structured data only, then from unstructured data only, and then from both data sources. In **Table 11** we present the query expansion terms

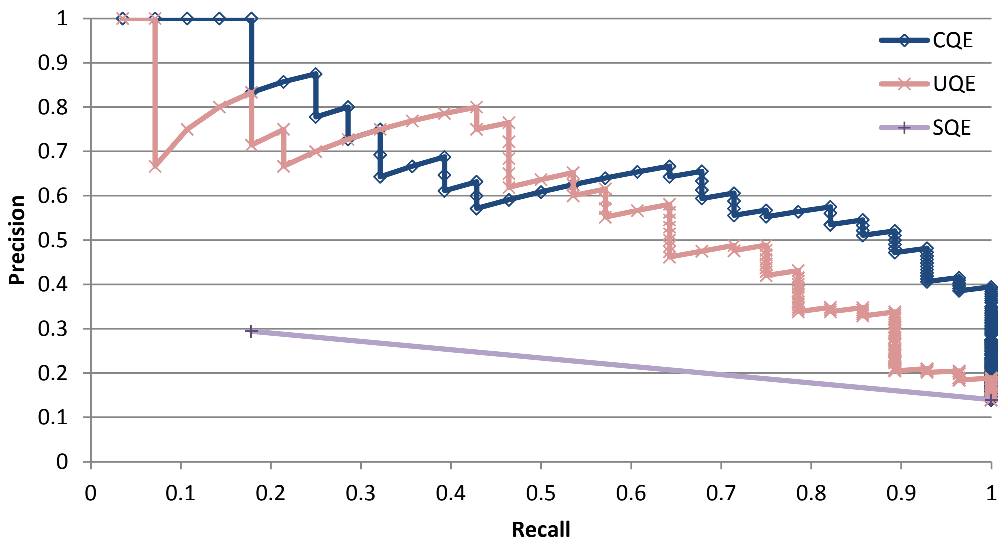


Fig. 19. 'Business Cell Phone' – CQE vs. UQE vs. SQE.

extracted from the technical specifications, along with the top 10 out of a total of 153 terms extracted from the unstructured data.

As shown in Fig. 19, expanding the query only on the technical specifications (Structured data Query Expansion, further denoted as SQE), leads to poor results in terms of precision and recall. The same test performed with the expansion terms from the unstructured data (Unstructured data Query Expansion, further denoted as UQE) already delivers much better results. Finally, since the structured and unstructured data cover different aspects of products, by considering both data sources, Conceptual Query Expansion (further denoted as CQE) achieved even better results. Not only did the precision for low recall values drastically improve, but

Table 11: Query expansion terms.

Structured Data	Unstructured data
Phone type	Windows mobile
Smart phone	Business
Phonebook features	Work
Picture id	Letters
Multiple numbers	Notes
Phonebook capacity	Fileds
	Qwerty keyboard
	Navigation
	Outlook
	Task

it was also maintained above 50% up to a recall above 90%. Also worth mentioning is the fact that at 100% recall, precision was of approximately 40%. In fact, CQE has consistently achieved better results than considering only structured or unstructured data alone for all experiments. This confirms our assumption that bridging structured and unstructured data will have a positive effect on the results. Taking this into consideration, for the subsequent experiments we present the results for CQE only.

Comparing the results to the baseline methods (Fig. 20), besides some marginal cases in low recall conditions, VSM was always dominated by CQE. On the other hand, BM25 achieved results that were quite comparable with our approach. Between the recall rates of 30% to 60% (middle area of the recall range) it even managed to obtain higher precision. However, for the low (up to 30%) and high (above 80%) recall areas, CQE was superior. Taking a closer look at the results we observed that the behavior of BM25 was much more similar to the results we obtained by expanding the query based only on the unstructured data (UQE in Fig. 19). By considering also the structured data, the precision is then improved in low and high recall areas, at the cost of precision in the middle recall area.

The positive behavior of BM25 confirms that query expansion is indeed a suitable and most powerful technique for dealing with more sophisticated queries as is the case for concepts. However, as we will present in the following section, BM25 doesn't always achieve such good results.

The 'social networking cell phone' query is an exceptional example of how the syntactical representation of some concepts can be misleading. From a linguistic perspective this type is represented by a nominal phrase with strong syntactic relation

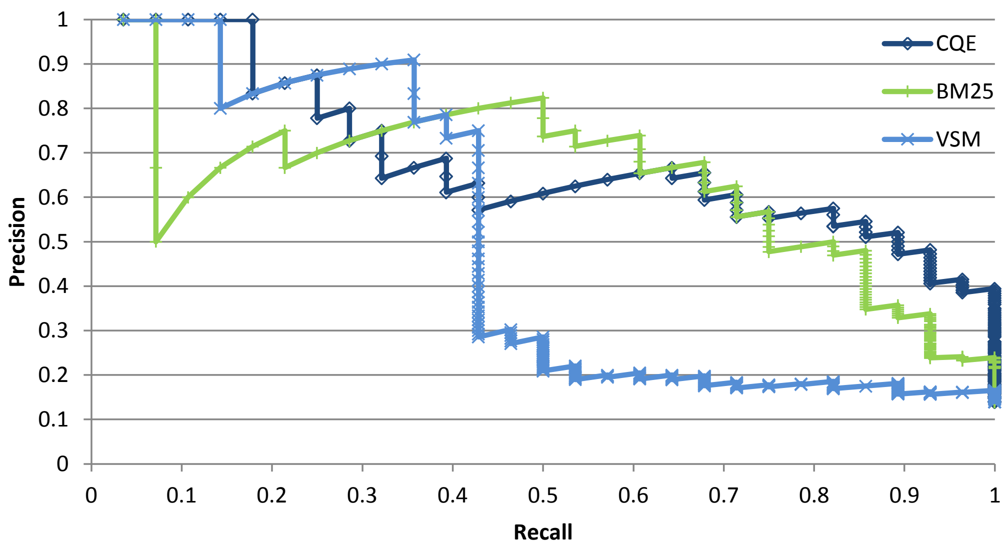


Fig. 20. 'Business Cell Phone' – CQE vs. VSM vs. BM25.

to the 'networking/network' technical feature. This relation however doesn't reflect human perception. For instance, the concept of 'social networking' and the 'UMTS network' technical specification show no semantic connection whatsoever. In such cases, both VSM and BM25 have a very difficult time in providing for correct results (Fig. 21). In fact looking into the behavior of both methods, we observed a very powerful topic drift towards the 'networking' features of products.

Since the number of such conceptual queries relying on nominal phrase constructs is not neglectable (e.g., 'tough as nails', 'packed with value', 'multimedia marvel' to mention just a few) we decided to take a closer look at the 'social networking' case. In the case of VSM, every product containing the terms 'social' or 'networking' in its textual description was considered relevant. Of course, products for which both terms co-occur in the textual description were ranked higher. This way VSM was able to identify the explicit cases, achieving some precision for the top 20% of the entities. Unfortunately, the remainder of the entities was ranked based on their mentioning of the term 'networking'. This led to catastrophic precision.

BM25 was also not able to provide notable results. Similar to the case of VSM, most of the products were considered relevant, due to their description containing the term 'networking'. As a consequence, the query expansion terms seemed to have been selected randomly. Besides 'social' and 'networking', other top expansion terms were 'nexus', 'hero', 'release', 'widget' and 'bluetooth'.

In the case of CQE first, the set of documents containing the complete concept were selected. In a boosting fashion, this set of documents was expanded to include all other documents being highly similar to them. At this stage however the topic drift doesn't take place for two reasons: on the one side documents selected in the

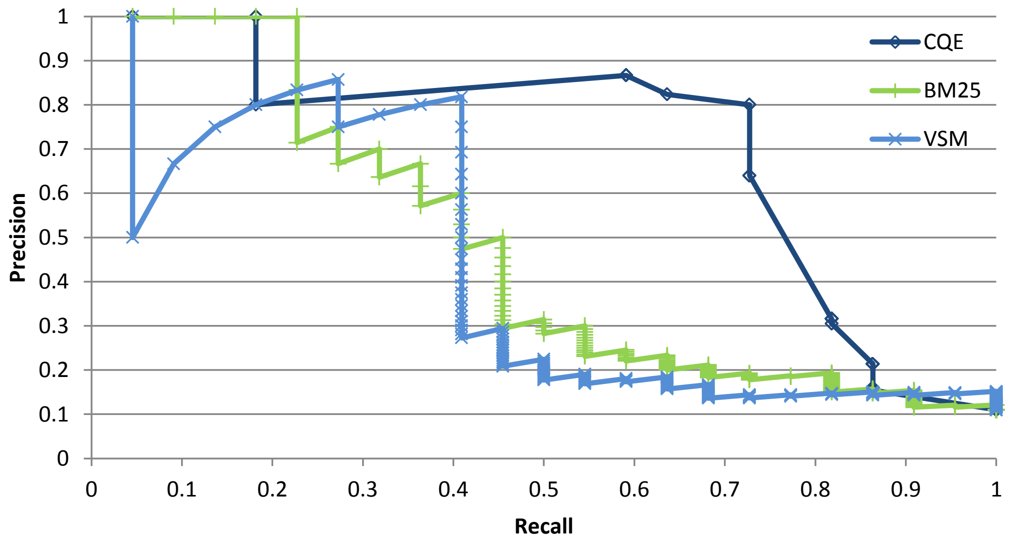


Fig. 21. 'Social Networking Cell Phone' – CQE vs. VSM vs. BM25.

first stage are relevant since they include the complete concept as a noun phrase and not a part of it; on the other side the highly selective threshold θ (from Eq. 8) for the similarity between selected documents and the rest, prohibits from expanding the relevant document base with product descriptions which are only vaguely similar to relevant documents.

As shown in Fig. 21, our results were, in this tricky case indeed much better than the ones achieved by VSM and BM25. The curve is also different from the 'business' concept. This is due to the fact that more user reviews share the same strength, i.e. the recall was improved without notably lowering precision. Actually, it is a consequence of the reduced number of terms selected for query expansion, which characterizes this entity type.

Finally, inspired by the contradicting terms obtained when considering also the 'network' feature as seed for expansion, the last of our tests, investigated a query purely based on a technical feature. The results show that our approach is at the present time indeed limited to expanding implicit conceptual features (see Fig. 22). The retrieval performance for technical features was merely comparable to VSM and BM25. The reason is that technical features are always explicitly mentioned in most of the editor reviews, as well as the technical specifications, regardless of the product. For example, the 'camera' technical feature was present in 80% of the documents from the collection used for query expansion. This clearly calls for standard techniques and our approach cannot offer any additional benefits here.

Performance results: since query expansion should be conducted in real time, we inspected the feasibility of the proposed method also in terms of performance. As

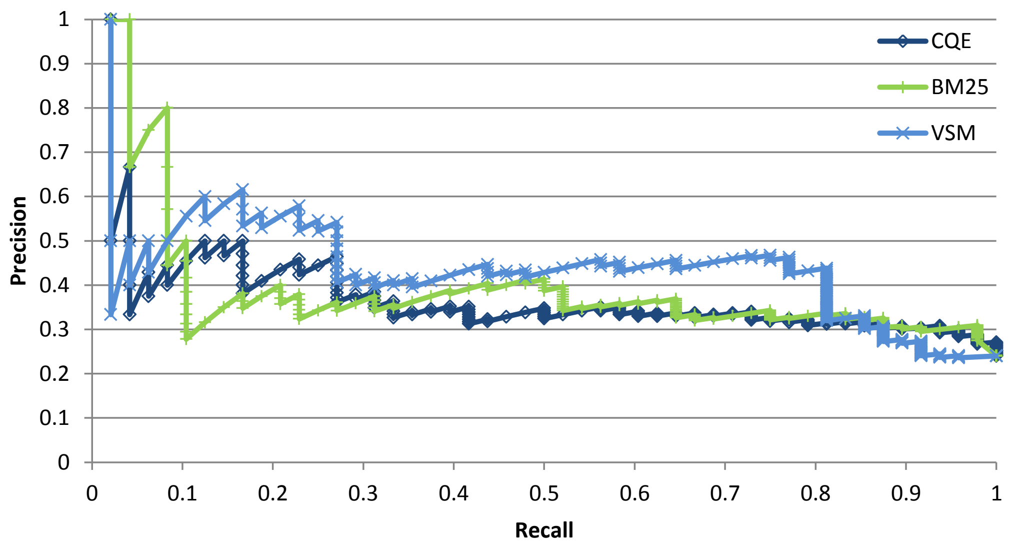


Fig. 22. 'Camera Cell Phone' – CQE vs. VSM vs. BM25.

expected, the NLP techniques, respectively the chunking process and the POS detector represent important performance killers. Considering a collection of 500 documents with an average of 1500 words each, the parsing process took about 100 seconds, which is not acceptable for real-time constraints. By comparison BM25 needed about 7 seconds to prepare the documents (The preparation includes text tokenization, word stemming and building inverted indices). Even if by comparison the time of 7 seconds seems quite good, it still doesn't fulfill real-time expectations.

Surely by optimizing the implementation of the NLP components or by means of parallelization, one could achieve better performance. The solution we propose is a system which makes use of the caching principle. Two major components are necessary: an on-line retrieval component which establishes the workflow and performs the actual Web search, and one off-line component, which maintains a database of entities together with structured and unstructured data. It goes beyond the scope of this work, but such a database could easily be put together by applying WebTables (the system presented in [24]) to the ClueWeb12 corpus. The off-line component also performs NLP tasks on the documents, storing the resulting noun phrases into the database. This reduces the computation time of a query expansion model to less than 2 seconds on regular hardware (for our tests we used a Core I7 QM with 2.4 Ghz and 16 GB RAM). Run on the same collection of preprocessed documents, also BM25, also needed 2 seconds for query expansion.

Building further on the caching solution, one could even store the expansion models (expansion terms and corresponding weights) of the most queried concepts, just by periodically inspecting query logs for the most frequent terms complying with *Observation 1*. This reduces the on-line process to ranking new entities based on pre-cached models, operation which can be easily executed in real-time.

3.1.4. State-of-the-Art

Recently, several search engines have been proposed, which can retrieve entities, and especially products even if the query keywords don't match the product tuples in the database [2, 91]. Such engines extract the entities which co-occur with keywords from the query, in documents on the web. But for concept driven querying this approach is likely to suffer from *incompleteness* since most of the concepts are mentioned only in a few documents. The reason is that concepts are rather implied by means of related terms. We tackle this problem by further expanding the query with terms related to the concept. Such search engines may also suffer from *impreciseness* of the results. In some of the documents the concept may be present but with a different meaning than the one intended by the user. Searching for a 'business' cell phone, one would also encounter cell phones with a description similar to '…it has GPS, so you can locate businesses nearby!'. By adding weights to the query expansion terms we are able to maintain a higher precision even for high recall.

On the other hand, approaches like [43, 109] follow a query transformation technique. They translate the user query to a SQL statement to be executed on the product database. The query terms are mapped to predicates on the table attributes. This approach is able to tackle queries like 'small IBM laptop' with clear meaning (map on the size and brand attributes). However complex concepts i.e. 'business' for which the meaning is rather ambiguous, are associated with a textual predicate ('contains') over attributes like the product name or description. Again this approach suffers from *incompleteness* and *impreciseness*.

Our work is also related to the field of product feature extraction. In this context, Hu and Liu [79], introduce a method for considering product features implied through adjectives like 'heavy', or 'big'. For this purpose, they use a human labeled training set, and generate rules with association rule mining for the features and adjective mappings. As in the case of approaches translating the user query to SQL, this method is only feasible for queries where a clear-cut mapping between the query and table attributes can be performed. This is not the case for conceptual queries.

Turning to the field of concept extraction, in [127], Weld Hoffman and Wu propose Kylin, a self-supervised open information extraction technique. Kylin relies on information from Wikipedia to learn extractors for concepts. Wikipedia is only used as a seed, with the extractors being learned by means of bootstrapping on the Web and with the support of WordNet providing for the semantic term relations. But the extracted concepts are rather general and cannot cope with the closed vocabulary of product descriptions.

An interesting approach is presented in [28]. The authors build on the theory of Formal Concept Analysis introduced by Rudolph Wille in [42] and caching mechanisms to improve precision and recall for conceptual queries. This approach assumes that a shared, domain-specific vocabulary is available and known to the user. However, in the context of web search, and especially in the case of users who can't express their needs in clear cut feature language, such vocabularies have to be extracted first. The method we presented in this section is not affected by such problems as it dynamically extracts the needed vocabulary if enough structured and unstructured data is available.

3.2. Conclusions

In this chapter we have presented an in-depth analysis on the topic of *entity type-based* query type for entity-centric search. Such a query type is essential for entity retrieval systems as it poses the least cognitive burden on users. One just needs to state the entity kind and the system does the rest. Structured data is, if available, always the first choice for querying data. Also in the case of entity search, data sources like linked data or schema.org represent a great opportunity. The process of retrieving entities of a certain type from this data sources is simple enough: the user gives an entity type, and the entity search system building on either LOD or schema.org or even both, to return all entities of that type.

However, there are some problems severely affecting the benefits of the structured data on the Web for entity-centric search tasks: in the case of linked data, each data source may use its own vocabulary. This has the effect that entities and types are represented through different URIs in different sources. We observed thousands of different URIs for the same type. For this reason, it is difficult to retrieve entities based on type from more than one sources. In the case of schema.org, our web scale analysis shows that, one year after being launched, with an acceptance rate of about 1.5% of the Web pages in ClueWeb12, it didn't gain traction. Extending its coverage with for example machine learning tasks also didn't produce satisfactory results. Another problem that is relevant for entity type-based queries is the support for *subordinate* entity types. As discussed in this chapter, structured data on the Web is organized with the help of type hierarchies covering the few superordinate and basic entity types. However, there are so many possible subordinate entity types, that one can hardly expect to include all of them in this static hierarchy.

Overall, we believe that at the present time, the contribution of schema.org for entity type based search is completely neglectable. Search on structured data on the Web is in our opinion limited to one data store search only and to *superordinate* and *basic* entity types. This has sever effects on the recall and on the number of supported queries. In the future, we believe that, boosted by manual (crowd-sourced) or semi-manual effort to align types and interlink entity instances, linked data will play a major role in accessing entities from the Web. This would have positive impact in the recall. But we don't believe that *subordinate* entity types, will ever be properly supported by static vocabularies. Instead, we think that a system that is able to mine new, unknown types out of Web data makes more sense. It has been shown times and again, that for information retrieval, especially for keywords carrying considerable knowledge easy to understand by people, query expansion is the best approach. Combined with a self-supervised vocabulary learning technique built on both structured and unstructured data, such an approach is able to achieve a good tradeoff between precision and recall, with about 70% precision for 70% recall. Perfect recall can also be reached at the cost of precision (about 40% precision for perfect recall). The evaluation presented in this section was focused on the example of cell phones, however we have observed similar results on experiments also for the laptops and cars domain. This doesn't allow us to claim the generality of this approach for all kind of entities. But at least for entities having abundant structured and unstructured data on the Web, like it is the case for consumers' products, such an approach seems promising.

Reviewing the discussed possibilities, an interesting approach would be to combine the strength of linked data with the flexibility of query expansion. Such a hybrid system would benefit from the high precision that isolated data sources can deliver, while entity retrieval on query expansion on the Web could cater for better recall values and support for ad-hoc types not known to the LOD vocabularies. But in order to compile a list of resulting entities, duplicate detection is required, a problem that instance matching has yet to master. For this reason, we believe that for the moment, the decision between tapping linked data as a source or querying the Web with IR techniques like proposed in this chapter depends on the user requirements. If precision on common entity types is required, than tapping Freebase or DBpedia is more sensible. If subordinate entity types are the target and a tradeoff between precision and recall is acceptable, than our take on query expansion is definitely the better choice.

Of course entity type-based entity search is but one way of searching for entities. In the following chapter we continue our quest for approaches enabling entity-centric search by focusing on the entity properties. Our experience from working with both structured and unstructured data, elaborately presented in the course of this chapter has shown that both types of data sources are valuable. Combined they bring added value to the quality of the results. For this reason, we believe it is important that solutions for the remaining entity-centric query types be able to work with both types of data. In consequence, all other systems presented in the subsequent chapters will work on triples of the form *(subject, predicate, object)*. This is perfectly aligned with linked data which can directly be used as a data source. For unstructured data, we rely on an information extraction system we have developed to extract facts from text and store it as a triple based knowledge base. This triple extraction system is presented in detail in Chapter 5.

Chapter 4

Property-based Entity Search

When searching for a certain type of entity, users have some mental representation of "things" they are looking for. Discussed in Chapter 1 in detail, it is common knowledge in cognitive psychology (imported in information science [114]) that concepts take the place of thoughts. They are represented through symbols (words, sounds, etc.), defined intensionally by a set of properties, and extensionally by a set of entities. The goal in entity search on Web data is to easily access the entities that correspond to a concept the user thinks of. This concept may easily be expressed by its symbol, a word label a subject we have extensively discussed in the previous chapter.

Another way of expressing the user information needs in entity search is through the intension: the set of properties prototypically defining the kind of entity that the user is searching for. We call such queries *property-based entity search* queries. But there is not much research involving property-based entity search, or any kind of data access building on properties for that matter. Recently, there have been some proposals to adopt property-based models for accessing data for programming purposes. Sketched first in [110] and extended in [111] the entity type information for the newly proposed programming paradigm is given by properties: a type is defined by a set of required properties, and every entity with at least those properties is part of that type. But besides these visionary publications, not much has been done to show how property-based entity access can actually be implemented. So why isn't there more being done in this respect? Are properties too weak for entity search? In [70] the authors measure the conditional probability that an entity has a specific property set given a certain type, and vice versa. The analysis was performed on the BTC'12 data set and it shows that properties bear more information than types. So this cannot be the reason.

Obviously, providing a property based description of the entities to be searched for is not as easy as providing types. Maybe the cognitive load put on the user is too high to make such queries feasible. After all, our analysis on the acceptance of schema.org (presented in Section 2.2 on the example of articles in ClueWeb12) shows that people used at best an average of about 4.6 attributes, which is about 15% of the attributes available for annotation. This is despite the fact that the content comprised data for many more attribute annotations. This shows users are not keen on providing too many properties even though a more accurate description would have clear and direct benefits like directing more users to own pages. In consequence, one can safely assume that also for entity search by means of properties, users will most probably provide on average about 4 properties as a description of the intended entity type. Indeed, the cognitive load can be a problem for this kind of entity search. Consequently, a system accepting this kind of queries has to be able to work with only a subset of the actual intentional description. For instance, a user with movies in mind, may provide {'Title', 'Director', 'Genre', 'Language'} as an intension. But this description, is rather broad, as it may refer to movies, audio books or even video games. Ultimately, the quality of the retrieved entities is poor.

Nonetheless, we believe that property based entity retrieval is a promising take on the problem of entity centric search. The first step towards a working solution is to acknowledge that the provided properties are a mere subset of an intensional definition of the entity type, and that the choice of properties has severe impact on the quality of the retrieved entities. Building on these observations, our work presented in [59], is the first research paper to address the problems of property-based entity retrieval. Starting from properties provided by users as a query, we propose a method for estimating the quality of the selected entities and if necessary, identifying additional properties that have a high positive impact on the quality. The system we propose, works in an iterative fashion to assist users to extend the property-based type definitions while checking that the extended definition still matches their intentions.

The contribution of our work, discussed in more detail in this chapter, can be summarized as follows: an extensive inspection of property-based entity search; the presentation and evaluation of a quality metric enabling transparency for this entity search approach; and the presentation and evaluation of a property selection method for improved data quality.

4.1. Use Case

To assess the feasibility of the property-based paradigm for retrieving entities from the Web, we conducted an experiment on the example of movies. When searching for movies, users will most probably query on a few properties (up to four if we consider people's behavior observed for annotating data with schema.org attributes) they usually associate with movies. These properties are used in the property-based entity access approach as filters such that all entities having those properties are considered to represent movies. Inspecting the selected entities one can measure the quality of the property-based entity search approach by computing the percentage of entities which actually are movies. But first, what are the properties people typically associate with movies?

Our analysis on schema.org annotations presented in Section 2.2 on the example of articles in ClueWeb12 is again helpful also for answering this question. For movies, we observed about 40,000 annotations. On average, each movie annotation comprises 4.6 movie properties. Amongst the properties used to annotate data, the 'Title', 'Description' an 'URL' of the movie page, the 'Director' and the 'Genre' are the most frequently used. In **Table 12** we present a list of the properties appearing in at least 30% of the movie annotations. We assume that these properties or at

Table 12: Top ‘Movie’ properties (with frequency above 30% of all movie annotations) from the ‘Movie’ schema from schema.org used for annotating movie data on Web pages from ClueWeb12.

Property	Movies annotated with property
Title	78 %
Description	56 %
URL	44 %
Director	39 %
Genre	38 %
Actors	38 %
AggregateRating	33 %

least most of them, were not annotated by chance with such high frequency, but rather because they are typically associated with movies. In this case, for the ‘Movie’ entity type, most probable property-based definitions are a subset comprising three, four or at best five properties from the most frequent ones, e.g. {Title, Description, URL}, {Title, Description, URL, Director}, {Title, Description, URL, Director, Genre}.

According to the idea of property-based entity retrieval, all entities fulfilling these properties represent movies. To put this approach to the test we inspected its use on the BTC12 linked data corpus, introduced in the previous chapter in Section 2.1. The process of selecting data for a set of properties provided in natural language works as follows:

- Property URIs are identified for each property. For this purpose, all subjects from tuples of the form (*, rdfs:label, *p*) are selected for each property *p* (* is a wildcard which may be substituted by any URI). Synonym sets provided by WordNet or obtained through the owl:sameAs predicate are used to extend the coverage of each property (more details in Section 4.2.1).
- With *p’* as the URI of each property *p*, the entities to be selected are the set of all distinct subjects *s* for which there are tuples of the form (*s*, *p’*, *) in the BTC dataset (* is a wildcard which may be substituted by any URI or literal in this case).

An overview of the selectivity for different property sets is provided in **Table 13(a)**. While Title, Description and URL are quite general (1.5 million entities of various types are selected), Director, Actors and especially Genre notably reduce the number of relevant entities.

Table 13: Number of entities from the BTC12 data corpus fulfilling each property set (a). The corresponding precision and recall values (b).

Property Set	(a) Nr. of Entities from BTC	(b) Precision / Recall
{Title, Description, URL}	1,447,813	0.018/0.3
{Title, Description, URL, Director}	29,328	0.78/0.26
{Title, Description, URL, Director, Genre}	2,266	0.35/0.01
{Title, Description, URL, Director, Actors}	21,531	0.92/0.23

Precision and recall are the standard measures for evaluating the quality of retrieved information or, in our case, the quality of selected entities. Precision represents for this use case the proportion of entities being movies out of all retrieved entities, while recall represents the proportion of retrieved movies out of all movies present in the BTC dataset. Computing precision and recall is not trivial in this case since it requires recognizing entities that are movies. A problem we already raised awareness on, in linked data, URIs are not unique over all data stores. In consequence, the rdf:type property which connects entities to their types, is in this case impossible to rely on without further action. In the previous chapter, in Section 2.1 we invested much effort to manually build a long list of URIs comprising 4,336 URIs all representing the 'Movie' entity type. With these types we identified a total of 169,469 movies in the BTC dataset. It's important to mention that this is just an approximation of the exact number of movies in BTC, therefore the recall presented in **Table 13(b)** has to be taken as an estimation of the actual value.

As shown in **Table 13(b)**, the choice of properties has notable impact on the quality of the selected entities: precision increases from a mere 0.02 to 0.78 by adding one single property to the definition of 'Movie' entity type. Precision values of 0.92 are possible if the "right" properties are chosen. Recall is, with 0.3 for the first three most frequent properties, quite low. The main reason is the sparseness of the data. This becomes extreme in the case of 'Genre' with just a few movies having this property.

Overall, the property-based entity retrieval can lead to high quality/high precision entity selection if properties are well chosen. A major obstacle in the process is the lack of transparency: the user has no idea about the quality of the selected entities. Properties belonging to the entity type definition are mandatory: since they are provided by the user one can be certain of their correctness. In consequence, none of the entities missing on any of these properties should be retrieved. But this has a high impact on recall. Combined with the sparse nature of Web data, the more elaborate the definition, the smaller the number of selected entities. In the case of entity type-based queries, and especially for linked data, we mostly focused on recall, since high precision was ensured by the fact that data was manually curated. As we have seen in the use case presented in this section, in the case of property-based entity retrieval this doesn't apply anymore as the choice of properties and property based type definition has a high impact on precision. We believe that if precision and selection quality can be controlled, one can improve recall by building on structural similarity or by extending the selection to cover types discovered with high precision through the property-based retrieval. For this reason, we focus here on improving the quality of the selection throughout precision first, by extending the entity type definition with a set of well-chosen properties.

4.2. Property-based Entity Retrieval – System Description

Starting from a property-based entity type definition with properties expressed in natural language and a large collection of data organized as (subject, predicate, object) triples, the entity retrieval system that we propose, helps the user to obtain high quality selections: relying on a measure of property-based data homogeneity, it measures the quality of the entities that fulfill the property-based definition. If the quality is low, key properties contributing the most to better data quality are found. The user has to finally decide if those properties are part of the entity type or not. The definition of the intended type is extended to include the user feedback and the process is repeated until the quality reaches a satisfactory level. For this purpose, the following functionality is required:

- identify and select those entities that fulfill the property-based type definition;
- compute the quality of a collection of entities;
- find properties that, if added to the set of properties defining the type, considerably improve the quality of the selected data.

4.2.1. Property-based Data Access

As previously mentioned we aim for a system that is able to work with both structured and unstructured data. But working with triples like provided in linked data is enough since (as we will show in Chapter 5) information presented in text form can be extracted into triples to build an additional data store made available in the LOD cloud. In consequence, in the following we discuss about data as being organized only in triple form and refer the whole data through LOD.

According to the property-based entity retrieval approach, the system selects all entities from the data having all properties from a given set. In our triple stores, like in linked data, properties are represented through URIs. Hence, a *mapping* between the properties in natural language and the URIs is necessary. For this mapping, we rely on the rdfs:label property, an instance of rdf:property providing a human-readable name for a resource. For better coverage, and especially for data coming from text, where there are no owl:sameAs relations, each property is automatically extended beforehand with a list of synonyms from WordNet.

Definition 3 (mapping): Given a property *p* ∈ Properties, P_{SYN_p} its set of synonyms from WordNet (including *p*) and *LOD* a large set of 3-tuples of the form (subject, predicate, object), we define *map* as a function *map* : Properties → ℘(URIs) with:

$$p \longmapsto \{s \mid \exists p_i \in P_{SYN_p}: (s, \text{rdfs: label}, p_i) \in LOD\} \tag{13}$$

For some entities the rdfs:label property may be missing. Furthermore, the same property may be present in different data stores under different URIs, possibly connected to each other through the owl:sameAs property. In consequence, in a dictionary-like fashion, each property is actually mapped to a set of URIs all considered synonyms. This is especially important for the triples extracted from text.

Mapping Expansion Algorithm:

With $\Delta_{p,1} \coloneqq map(p)$ define

$$\begin{aligned}\Delta_{p,i+1} \coloneqq \{s_j{}' \mid \exists s_j \in \Delta_{p,i}: (s_j, \text{owl: sameAs}, s_j')\\ \in LOD \vee (s_j{}', \text{owl: sameAs}, s_j) \in LOD\}\end{aligned} \tag{14}$$

$$Dictionary(p) \coloneqq \bigcup_{i=1}^{\infty} \Delta_{p,i} \tag{15}$$

By repeatedly linking elements through synonyms, two or more properties from the definition set may end up being represented by the same set of URIs. Such cases are reported to the user.

At the very core of the property-based approach, an entity is relevant with respect to a specific property if there is a statement or fact asserting that the entity has this property. In the context of linked open data, we define the binary relevance of an entity w.r.t. a property as a *hit* function:

Definition 4 (hit): Given some entity $e \in E$ represented by its URI, a property in natural language *p* ∈ Properties and *LOD* defined as above, we define *hit* as a function
hit : (URIs × Properties) → {0, 1} with:

$$hit(e,p) = \begin{cases} 1 & \text{iff } \exists\, p' \in Dictionary(p): (e, p', *) \in LOD \\ 0 & otherwise \end{cases} \tag{16}$$

where * is a wildcard that may be substituted by any literal or URI.

According to the *semiotic triangle*, concepts and thus entity types are defined intensionally by a set of properties, and extensionally by a set of entities. Aiming for simple yet effective access to entities we define concepts expressing entity types, as the set of properties that intensionally define the entity type given by the concept.

This type definition may iteratively evolve based on user feedback. Because the user feedback may be negative w.r.t. to some properties (by negative we mean properties that all entities corresponding to the type definitely shouldn't possess), we define an entity type as follows:

Definition 5 (type): Given a concept *c*, representing an entity type extensionally defined through the set of entities given by their URIs, E_c, we define the *type* of concept *c* denoted T_c as the set of properties $T_c = P_{c_+} \cup P_{c_-}$ with P_{c_+} the set of positive properties and P_{c_-} the set of negative properties ($P_{c_+} \cap P_{c_-} = \emptyset$), such that:

$$\begin{aligned} &(i) && \forall e \in E_c, p \in P_{c_+}: hit(e,p) = 1 \\ &(\text{ii}) && \forall e \in E_c, p \in P_{c_-}: hit(e,p) = 0 \\ &(iii) && \forall e \notin E_c\ \exists p \in P_{c_+}: hit(e,p) = 0 \end{aligned} \tag{17}$$

While here all properties (initial as well as positive and negative extensions) are treated equivalently, the fact that not all properties extending the definition are required is a starting point for future work. As in the case of properties, in the LOD cloud the same entities may end up having multiple URIs. For the sake of simplicity, we refer to one entity as being uniquely identified by an URI.

More often than not, the number of properties provided by users to describe some type of entity is much smaller than the intension of that entity type. Extensive experiments presented in Section 2.2 show that on average only 4.6 (out of an average of 34 existing) properties have been used to describe entities. This suggests that the user provides a sub-set of properties meant to represent the intended (to us hidden) type. This set of properties is one of the many possible super-types of the intended type. Starting from a property set that builds a type or a super-type for some concept, all entities having all these properties are selected as being relevant for the type or super-type:

Definition 6 (property-based data access): Given a set of properties $T_c = P_{c_+} \cup P_{c_-}$ representing either a type or super-type for a concept *c* as before, the set of entities selected according to the property-based data access paradigm E_c is the set of entity URIs that fulfill all properties from T_c:

$$E_c = \bigcap_{j=1}^{|T_c|} E_j \tag{18}$$

where $\boldsymbol{E_j = \{e | hit(e,p_j) = 1\ if\ p_j \in P_{c_+} \wedge hit(e,p_j) = 0\ if\ p_j \in P_{c_-}\}}$

In the ideal case, for a concept *c* the set of properties T_c represents the intension of *c*. Then, the set of selected entities E_c also extensionally defines concept *c* and should fit the user needs. However, there is a high probability that T_c is a subset of the intension. Since the type intended by the user is hidden to the system and entities have no clear types, there is no trivial way for checking if the selected entities correspond to the intended concept and thus entity type. The user also has no feedback whatsoever regarding how good the retrieved entities match the intended entity type. This has grave effects on the applicability of the property-based entity retrieval approach. Aiming for better transparency of the whole approach, in the next section we introduce a measure of quality for the selected entities.

4.2.2. Quality of the Selected Entities

We measure the quality of entities selected through the property-based model as a function of entity homogeneity. The basic assumption is that the user describes simple concepts (like 'Movies' or 'Books') with all corresponding entities having the same or almost the same properties and not ad-hoc or composed concepts (like "all things having a geo-location"). Consider for example that the user provides three properties: {'Title', 'Description' and 'Genre'}. Based on these properties a set of eight entities is selected. Besides the three properties, each entity is described by other additional properties like in **Table 14**. Properties p_4, p_5 and p_6 may be, for instance, 'Duration', 'Actors' and 'Director' while p_7, p_8 and p_9 could represent 'ISBN', 'Pages' and 'Editor'. As you may have intuited, entities e_1, e_2, e_3 and e_4 represent movies while the remaining entities represent books. Properties in Web data and in the LOD cloud may be missing. This is reflected also in this artificial example with movies e_1, e_3 and e_4 providing no values for properties p_4 and respectively p_6. Analogously, for the entities representing books. The rest of the missing values are attributed to the fact that properties p_4, p_5 and p_6 are proper to movies while p_7, p_8 and p_9 are proper to books.

Table 14: On rows - the entities that are selected for the property set {p_1, p_2, p_3}. On columns – all properties describing any of the selected entities.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
e_1	✓	✓	✓	✗	✓	✓	✗	✗	✗
e_2	✓	✓	✓	✓	✓	✓	✗	✗	✗
e_3	✓	✓	✓	✓	✓	✗	✗	✗	✗
e_4	✓	✓	✓	✓	✓	✗	✗	✗	✗
e_5	✓	✓	✓	✗	✗	✗	✓	✓	✓
e_6	✓	✓	✓	✗	✗	✗	✓	✗	✓
e_7	✓	✓	✓	✗	✗	✗	✓	✗	✓
e_8	✓	✓	✓	✗	✗	✗	✓	✓	✗

More generally, starting from the set of properties, the system selects a set of entities as described in the previous section. In a relational sense, together with the union of all their corresponding properties (stop properties like rdfs:label, owl:sameAs, rdf:type, etc. are first removed) these entities form a relational schema (as in **Table 14**). Especially in the field of schema extraction and discovery, the number of null values has successfully been used for establishing the quality of the schema [22] – the better the schema, the fewer null values, the more homogeneous the data. Thus, if the data is homogeneous in terms of structure - their properties - these properties intensionally define a single entity type. As a measure of homogeneity we measure the property-based similarity between all entities. But there is a problem: Entities may be selected from different data sources (DBpedia, LMDB, etc.). Entities with the same type and from the same source tend to share the same properties, usually due to the focus of each data store. Different sources have different sizes, and small data sources with many properties can introduce null values. These null values are artificially amplified by the size of the data source. To handle this problem, we reduce all entities having the exact same properties to just one *witness*. This way, for the example presented in **Table 14**, e_3 and e_4 are both represented by one witness: $w_{e_3 e_4}$ having the same properties as e_3 or e_4. The same for e_6 and e_7. The rest are their own witnesses. Based on this observation we define the quality of a set of entities as follows:

While the Jaccard index is most suitable for measuring structural similarity between entities, any other similarity measure may be used here.

Definition 7 (quality): With the notations of T_c and E_c as above and W_c as the set of witnesses represented by URIs of entities from E_c, the quality of the selected entities is a function, $Q : \wp(\text{URIs}) \rightarrow [0, 1]$ with:

$$Q(E_c) = \frac{1}{C_2^n} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Sim(w_i, w_j) \qquad (19)$$

$\forall \boldsymbol{w_i}, \boldsymbol{w_j} \in W_c$, $n=|W_c|$ and $\boldsymbol{sim(w_i, w_j)} = \frac{|P_{w_i} \cap P_{w_j}|}{|P_{w_i} \cup P_{w_j}|}$ is the Jaccard index [10]. $\boldsymbol{P_{w_i}}$ is the set of properties of $\boldsymbol{w_i}$ and $\boldsymbol{P_{w_j}}$ is the set of properties of $\boldsymbol{w_j}$.

For the example introduced in **Table 14**, the quality of the selected entities is 0.55. If however, additional information were provided, like the fact that the entity type that the user has in mind also has property p_5, or doesn't have property p_7, the entities selected by the property based model restrict to movies only (entities 1 to 4). The quality in this case increases to 0.78, the result being slightly affected by the noise (missing values) in the data. In the following subsection we present how to find properties better separating various types of entities in the result set.

4.2.3. Property Selection

Finding the list of properties best distinguishing different types is similar to the problem of induction of an optimal decision tree in data classification, which is a hard task. It has been shown that finding a minimal decision tree consistent with the set of labeled entities provided as data is NP-hard [49]. Consequently, greedy algorithms like the C4.5 are applied for solving this problem [98]. When it comes to selecting some property that better discriminates between different types of entities, *information gain* from the field of information theory is the standard measure for deciding the relevance of a property [99]. Generally speaking, the information gain is the change in information entropy from a prior state to a state that takes some information as given. Computing this entropy change is only possible for entities that have class labels (entity types) attached. Types are provided in the LOD cloud by means of the rdf:type property, however entities may have multiple types partly with different granularities e.g., the movie "Gangs of NewYork" has types owl:Thing, schema.org/CreativeWork, dbpedia-owl:Film, yago:VictorianEraFilms and 15 other types. For other movies, types owl:Thing, or schema.org/CreativeWork are missing. All these types are obviously related to each other but without an upper ontology or global type hierarchy, it's difficult to make use of the type property to compute the information gain.

But as shown in [47], the type information strongly correlates with the entity properties: in the example presented in **Table 14**, it's obvious that entities having properties 'Duration', 'Actors' and 'Director' on top of 'Title', 'Description' and 'Genre' are movies while entities having 'ISBN', 'Pages' and 'Editor' are books. The type information is latent in the properties. But the missing values for some entities, as well as the heterogeneity of data sources make it difficult to fold all movies together to just one witness – a property set representing the movie type. Actually what happens is that more witnesses, with more or less similar properties, exist for a single type. The problem of reducing similar witnesses to a dominant type is similar to the problem of dimension reduction.

Principal component analysis (PCA) is the best, in the mean-square error sense, linear dimension reduction technique [65]. In essence, PCA is a basis transformation that seeks to reduce the dimensionality of the data by finding a few orthogonal linear combinations (called principal components) of the original variables capturing the largest variance. Given E_c the set of entities selected according to the property-based data access paradigm, and W_c the set of witnesses of entities from E_c, let X be a $n \times p$ matrix, where n and p are the number of entity witnesses and the number of properties of all witnesses, respectively. Let the matrix decomposition of X be

$$X = UDV^T \tag{20}$$

Y=UD are the principal components (PCs), where the $p \times p$ matrix U is the matrix of eigenvectors of the covariance matrix XX^T, matrix D is a $p \times n$ rectangular diagonal matrix of nonnegative real numbers on the diagonal with customary descending order, and the $n \times n$ matrix V is the matrix of eigenvectors of $X^T X$. The columns of V are called loadings of the corresponding principal components. Usually the first PCs (capturing the highest data variance) are chosen to represent the dominant dimensions.

For the example introduced in **Table 14** (first all data is reduced to binary values and centered on the columns such that the mean of each column is equal to 0), the first PC shows the strongest variance of 1.16. This PC is also considered relevant. The next two components show a variance of 0.2 and the rest are 0 or close to 0 and can be neglected. With respect to the properties, the coefficients of the first PC are clustered together according to their variance (**Table 15**). For this example, the three property clusters that build on the relevant PCs show the existence of two dominant types that differentiate in terms of properties p_4, p_5, p_6 and p_7, p_8, p_9. Showing no variance, properties p_1, p_2 and p_3 can be ignored since they belong to both dominant types.

In general, depending on the selected entity set, more PCs may be relevant. To dynamically establish which of them show notable variance, we rely on the ISODATA algorithm, an automatic thresholding approach [5] that identifies thresholds in one dimensional spaces that best separate a set of data points. With the PCs that show variances above the threshold, one dimensional clusterings (agglomerative hierarchical clustering with average inter-cluster similarity) on the coefficients are built for each PC. This way each property is assigned to one cluster for each relevant PC. Each set of properties belonging to the same clusters on all relevant PCs are grouped together and represent abstract dominant types we will further refer to as *latent types*. For the example in **Table 15**, considering that only PC$_1$ is relevant, the extracted latent types are $t' \equiv \{p_1, p_2, p_3, p_4, p_5, p_6\}$ and $t'' \equiv \{p_1, p_2, p_3, p_7, p_8, p_9\}$. With these types we can now label entities according to the property-based model. This way, e_2 will be labeled with t' and e_5 with t''. In this manner a set of labeled entities

Table 15: Coefficient values (component loadings) for each property, for the first three principal components.

PCs / Props.	PC$_1$	PC$_2$	PC$_3$
p$_1$	0.00	0.00	0.00
p$_2$	0.00	0.00	0.00
p$_3$	0.00	0.00	0.00
p$_4$	0.35	-0.71	0.00
p$_5$	0.50	0.00	0.00
p$_6$	0.35	0.71	0.00
p$_7$	-0.50	0.00	0.00
p$_8$	-0.35	0.00	0.71
p$_9$	-0.35	0.00	-0.71

is created. Entities that fulfill properties for multiple types ('Audiobooks' in the context of our example) are automatically associated with multiple labels.

With the set of labeled entities, the information gain for a property can be computed as follows:

Definition 8 (information gain): With the notations of T_c and E_c as previously defined and P_U the set of all properties of all entities from E_c, the *information gain* of a property $p \in P_U - T_c$ w.r.t. the entity selection E_c is:

$$Gain(p, E_c) = H(E_c) - \sum_{v \in \{0,1\}} \frac{|E_c|p^v|}{|E_c|} \cdot H(E_c|p^v) \qquad (21)$$

where $E_c|p^v = \{e \in E_c | hit(e,p) = v\}$.

The entropy (denoted H) represents a measure of the amount of uncertainty in the data and is usually computed as follows:

$$H(E_c) = -\sum_{i=1}^{n} p(t_i) \log p(t_i) \qquad (22)$$

where n represents the number of latent types and $p(t)$ represents the probability (relative frequency) of latent type t in E_c.

However in our case, an entity may have multiple types. Known as the multi-label learning problem, this poses difficulties for most learning and classification methods. The information gain - entropy based approach from the C4.5 decision tree algorithm is no exception [121]. To overcome this problem, we employ a modified version of the entropy proposed in [30] that considers multiple labels by introducing the probability of an entity not belonging to a certain type:

$$H(E_c) = -\sum_{i=1}^{n} ((p(t_i) \log p(t_i)) + (q(t_i) \log q(t_i))) \qquad (23)$$

with n and $p(t)$ as before and $q(t_i) = 1 - p(t_i)$ the probability of not having type t_i.

4.2.4. System Evaluation

The system presented in this chapter has two major objectives: to provide transparency regarding the quality of the entities retrieved through the property-based paradigm and to improve the quality of the selected data by iteratively, and with user feedback, extending the property-based type definition with chosen properties. To evaluate how well these objectives have been fulfilled we performed the following experiment: starting from different entity types presented in structured form with schemata on schema.org, as in the use case presented in Section 4.1, we build an initial type definition for each concept. This initial definition embodies typical properties most users associate with each entity type. It comprises the first four properties that have been most frequently annotated in ClueWeb12 for the corresponding schema.org schemata. The property-based data access is applied to these four properties and a set of entities from the BTC data corpus is selected. The quality score, precision and recall are computed for the selected entities. If the quality score is lower than 0.65 (our experiments have shown that a threshold of 0.65 brings satisfying data quality), a property is chosen based on its information gain. The user is asked whether this property belongs to the entity type or not. We simulate the user feedback by relying on information from schema.org: If the property with the highest information gain is part of the schema that describes the corresponding concept on schema.org (considering synonymy), then the user feedback is positive. The type definition for the entity type is in this case extended with this property and all entities having this property are kept. If, however, the property with the highest information gain is not part of the schema, then it is considered a negative property and all entities not having this property are kept. The process is repeated until the quality score reaches the quality threshold. Using schema.org to simulate user feedback is convenient but it has some drawbacks that will be addressed in future work: some properties that are part of schema.org may be irrelevant from a human perspective. At the same time, schema.org doesn't claim full completeness. In consequence one can't be sure that properties not being part of schema.org are negative properties.

In order to measure precision and recall, a gold standard is required. The gold standard represents, in this case, clear type information w.r.t. the entity types: In the context of movies, is a given entity a movie or not? We build the gold standard by bootstrapping on a set of 1,000 seed entities that we know are of the entity type: We extract all rdf:type types for each of the seed entities. On average, about 500 types are found. Types that are not related to the concept or that are too general (e.g. owl:Thing or schema.org/CreativeWork) are manually pruned. In a second iteration, all entities having those types are selected and 100 entities are randomly chosen. Only those entities that, on manual inspection show the correct type are kept. Their rdf:type types are extracted, and unrelated or general types are again manually pruned. The process is repeated one more time. The resulting list of rdf:type values represents the description of a concept type according to the rdf:type property. Any entity that has one of the types in the list is considered to be of the respective type. Of course, only a subset of the actual expressions of a certain type is found. As a result, the precision and recall values computed on this gold standard underestimate the actual values.

Our system chooses key properties to improve the type definition based on information gain. As a baseline, we built Rand, a system choosing properties at random (without replacement). The randomization process is repeated 10 times for each property selection step. Average quality, precision and recall values are considered for each iteration. The property that is closest to the average scores of all 10 random picks is chosen to extend the definition for the next iteration.

We evaluated our property based entity retrieval system (named ProSWIP – short for **Pro**perty based **S**emantic **W**eb **I**nteractive **P**rocessing) on multiple concepts from various fields, with different characteristics. In **Table 16** we present the results on the example of three chosen concepts. The base iteration (0) is common to both systems and corresponds to the most frequent four properties used for annotating the corresponding schema in ClueWeb12. For Movie, this iteration already produces good precision but it is quite restrictive in terms of recall. ProSWIP requires in this case 4 iterations to reach quality above 0.65 and perfect precision. With 0.93, precision is already very good after the first iteration. Further iterations isolate well defined movies from the ones with missing values. This in turn affects recall. Benefitting from high quality entity selection from the base iteration (78% of entities selected from the start are movies), the random approach is also able to obtain good results. Primarily guided by average scores and with high quality semantic feedback, the baseline method achieves 0.95 precision and a quality score of 0.59 after 4 iterations. Recall however is severely affected by the random choice of properties. For Music the base iteration is, with a precision of 0.44, of lower quality. Various types of entities are selected. The probability for the random selector to choose some irrelevant property is higher in this case. This is also reflected in the poor performance of Rand for Music. In contrast, ProSWIP achieves the desired level of quality after only two iterations. For Books, the base also has low precision with negative consequences on the performance of Rand. The quality metric we introduced is highly correlated to precision on all experiments (Pearson's linear correlation coefficient of 0.94) denoting its expressiveness for the quality of the data selection. Precision rapidly increases towards values above 90%, showing the success of the whole approach.

Table 16: Quality, precision and recall for three chosen entity types and multiple iterations.

Iteration	Quality(Q)		Precision		Recall	
Movies	ProSWIP	Rand	ProSWIP	Rand	ProSWIP	Rand
0	0.49	0.49	0.78	0.78	0.26	0.26
1	0.57	0.5	0.93	0.78	0.25	0.26
2	0.55	0.51	0.91	0.74	0.12	0.03
3	0.58	0.53	0.96	0.89	0.11	0.03
4	0.65	0.59	1	0.95	0.07	0
Music						
0	0.34	0.34	0.44	0.44	0.82	0.82
1	0.58	0.34	0.99	0.43	0.82	0.78
2	0.67	0.34	0.99	0.43	0.62	0.78
Books						
0	0.21	0.21	0.37	0.37	0.71	0.71
1	0.32	0.21	0.83	0.38	0.07	0.07
2	0.52	0.22	0.93	0.39	0.07	0.07
3	0.59	0.25	0.89	0.43	0.04	0.07
4	0.65	0.25	1	0.43	0.03	0.07

From a technical perspective, ProSWIP is a component implemented in Scala[43], which maps variable names to properties from the BTC data set. While classical relational databases are not suitable for querying on RDF data, graph databases like Neo4j[44] have limited performance for our approach. In comparison, Lucene[45] has proven much faster in both the time needed for initially loading the data (building the index) as well as in terms of querying. With an off-the-shelf commodity computer with Intel I5-3550 quad-core CPU with 3.3 GHz. 32 GB RAM and 8.5 ms access hard drive, the index creation for the complete BTC data set took about 39 hours (only one core was used). The resulting index was about 1T in size including data. One simple entity search takes about 16 seconds. But the complete process of property-based data access may take up to hours as multiple queries, entity and property retrievals are being performed. It was possible to speed up the process by introducing caching mechanisms, for instance for the property synonymy dictionaries. Computing the quality, principal components, latent types and information gain for all properties on large data samples takes under 2 seconds. Nonetheless, we believe that in order to realize all operations in real-time a Lucene-based distributed index leveraging Hadoop like for instance Elastic Search[46] is necessary.

4.2.5. State-of-the-art in Property-based Entity Retrieval

Property-based entity selection has recently been discussed in [111, 112] in the context of programming the Semantic Web. Challenges and open questions concerning a property based approach are discussed in these papers. Sharing their view, we inspect the practical feasibility of such an approach and address one of the main challenges: The data quality problem.

Property-based data access has been an important research topic especially for programming purposes. For instance, property-based interfaces have been studied for object oriented languages [46] or extensible record systems for different language settings [18, 68]. But additional challenges like discovering, comprehending and extending property sets to match the intended use arise in the context of retrieving entities from Web data.

From a broader perspective, systems like Tipalo [41] performing automatic typing for DBpedia entities are also relevant to our approach. Tipalo extracts types for entities based on their corresponding Wikipedia pages. But there are several entities in the LOD cloud having no article on Wikipedia that would hence remain untyped

[43] http://www.scala-lang.org/

[44] http://www.neo4j.org/

[45] http://www.lucene.apache.org/

[46] http://www.elasticsearch.org/

(there are about 14,199 diseases according to the International Statistical Classification of Diseases[47] most of them documented through PubMed but only about 3,000 of them featuring an actual article on Wikipedia). High precision knowledge bases like YAGO [116] relying on the Wikipedia category system and Infoboxes suffer from the same problem. In contrast, we build on structural similarity independent of all-encompassing information sources to find latent, contextually relevant types.

4.3. Conclusions

In this chapter we presented an in-depth analysis on the topic of *property-based entity search*. The user expresses his/her information needs through the intension, respectively the set of properties prototypically defining the kind of entity the user is searching for, and the system retrieves all entities having the given properties. While not as simple as querying by entity type, property-based entity search has the advantage that properties carry more information than types do. If proper types are not provided or cannot be extracted than properties may be an interesting option. But the cognitive load put on the user is higher, and as we have seen on the example of Web data annotations, users are not keen on providing too much properties. Therefore, such an entity retrieval system has to be able to work with a small subset of properties defining the intended entity type to be accepted by users.

Retrieving entities based on a small subset of properties and not on a full blueprint of the entity type is challenging. Our experiments show that all kinds of entities can easily make their way into the result set. To make matters worse, the user doesn't even have a clue of the bad quality of the entity selection, and simple property-based entity retrieval systems can't detect such problems since type information is missing or it is not reliable.

Motivated by positive examples in the field of programming, were *duck typing*[48] has already been successfully applied, we believe that property-based data access represents a cornerstone in retrieving entities from the Web. The key to its success is extending the property-based type definition with well-chosen properties that lead to high quality entities, while keeping the user informed on the quality of the selection. For this purpose we propose ProSWIP a system that builds on user feedback in order to ensure that it captures correctly the user's intentions. ProSWIP asks the user if some property is, or is not relevant regarding the intended entity type and extends the property set for the entity type definition in this way. However, entities have hundreds of properties. Asking more than a handful of questions is not feasible. ProSWIP cleverly solves this problem with the help of information theory concepts

[47] http://www.who.int/classifications/icd/en/

[48] A style of typing in which an object's methods and properties determine the valid semantics and not – the naming is attributed to James Whitcomb Riley who coined the duck test: "When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."

choosing to ask user feedback on but a few properties showing the highest information gain. Our experiments show that within a maximum of four iterations the system achieves very good quality. With an entity homogeneity-based quality metric the level of quality for the selected data can also be measured. Being highly correlated to precision, the quality measure we introduced provides for transparency. With additional feedback on chosen properties, precision easily reaches values above 0.9, confirming the success of this approach.

For the time being, all entities not showing a certain property that has been included in the entity type definition are not included in the result set. Corroborated with the sparse nature of data extracted from the Web, this severely affects recall. But even an entity doesn't possess all properties, defining the entity type, it can still be relevant if it possess at least those properties being *typical* for entities of the intended kind. Leveraging high quality property-based definitions, together with the concept of attribute typicality (introduced in Section 6.3.1) the recall problem can be tackled: we plan to use properties that have been found suitable to extend the concept definition, not as filters, but as features for entity ranking on structural similarity. Applied only to properties not being typical for the intended entity type, this condition relaxation should increase the robustness against missing values and have a positive effect on recall.

Another way of searching for entities is through instance-based search. Instance-based search is also known under the name of *query by example* and it has primarily been used in multimedia information retrieval systems. Systems like Shazam[49] or better yet Midomi[50] like systems, stand as a proof of their success. In the following chapter we analyze different possibilities for supporting this kind of queries for entity search.

[49] http://www.shazam.com/ is a mobile app that recognizes music being played around the user starting even from a few seconds of noisy piece of sound

[50] http://www.midomi.com/ is a system for audio query by humming. The user can hum or whistle a song and midomi finds the respective song in a large library of music pieces

Chapter 5

Instance-based Entity Search

When searching for entities, users may think of some examples best representing the entity types that they search for. For instance, when searching for 'sports cars', some users may already visualize a red Ferrari or an orange Lamborghini. One can exploit such automatic associations between entity types and representative entities to empower yet another type of entity search: instance-based entity search. For this type of search, users provide one or more example of entities and the retrieval system returns entities being similar to the ones given by the users.

But there are two major concerns with this type of queries: first it is very important that the examples are well chosen. To pick up on the example of American presidents, with "Ronald Reagan" as a query entity and "Clint Eastwood" as additional example, the user will be referring to American actors rather than American presidents. However, the user might also have more restricted entity types in mind like Western actors, actors from California, American actors with political ambitions, and so on. The more examples, the better a query can be disambiguated, however increasing query complexity. This brings us to the second main concern which is, how exactly should the query look like? How many examples should it contain? Obviously, there is a tradeoff between better disambiguation strength (the more examples the better the disambiguation) and the cognitive load put on the user (it's not realistic to expect that users will be able or willing to provide more than a handful of examples).

Learning from the experience of state-of-the-art entity search tasks presented in more detail in Section 5.1, in Section 5.2 we discuss the problem of query formulation and propose a satisfying solution requiring low effort from the user. Finally, in Section 5.3 we introduce our system for instance-based entity search along with the underlying theoretical foundations and corresponding evaluation. But before going into any more detail, in the following section we give an overview of the state-of-the-art in example-driven entity search.

5.1. State-of-the-art in Instance-based Entity Search

In contrast to all other entity-centric query types, instance-based entity search has been extensively researched and it is today one of the core tasks of information retrieval. Acknowledging its importance, the **T**ext **Re**trieval **C**onference[51] (TREC),

[51] http://trec.nist.gov/tracks.html

an on-going series of workshops focusing on important information retrieval challenges (tracks), introduced an entity track starting 2009. The aim of the entity track is to encourage research on entity-centric search on Web data. It proposed a standard set of queries and corresponding results extracted from ClueWeb09 and later the Sindice-2011 corpora [25] to make the system results comparable to one another.

When the entity track was first introduced, the entity search task was defined as follows: "Given an input entity, by its name and homepage, the type of the target entity, as well as the nature of their relation, described in free text, find related entities that are of target type, standing in the required relation to the input entity". This task is known today as Related Entity Finding (REF). Some example of REF queries from TREC 2011 are presented in Fig. 23. As observed in these examples, entity URLs representing the entity homepages are extracted from the ClueWeb09 corpus. To offer better entity description, in REF-LOD (related entity finding on linked data) the entity and homepage are provided as URI from the LOD and referenced from the Sindice-2011 corpus.

Many approaches have been proposed as a solution to REF. Systems like TongKey [94], PRIS [126], FDWIM [34], and many others, stand as a proof of the effort invested in supporting this kind of queries. However, the quality of the results was poor. Out of 14 systems submitted until 2011 to the REF challenge, the best achieve a disappointing mean normalized discounted cumulative gain (mean nDCG) of under 0.4. An overview of the results achieved by these systems is presented in Fig. 24. Analyzing the problem in more detail, on a query basis, some queries have been supported much better than others (see Fig. 25). For instance, query 36, focused on searching for companies that build parts for Ford cars has shown the worst results

```
<query>
<num>71</num>
<entity_name>Montana State University</entity_name>
<entity_URL>clueweb09-en0009-75-04753</entity_URL>
<target_entity>university department</target_entity>
<narrative>Montana State University departments that offer a PhD program</narrative>
</query>

<query>
<num>72</num>
<entity_name>Abraham Verghese</entity_name>
<entity_URL>clueweb09-en0033-79-30130</entity_URL>
<target_entity>publication</target_entity>
<narrative>publications in which Verghese's writing has appeared</narrative>
</query>

<query>
<num>73</num>
<entity_name>National Spelling Bee</entity_name>
<entity_URL>clueweb09-en0012-05-10759</entity_URL>
<target_entity>television program</target_entity>
<narrative>television programs with an episode referencing the National Spelling Bee</narrative>
</query>
```

Fig. 23. Example of REF queries from TREC 2011[52].

[52] http://trec.nist.gov/data/entity/11/11.topics-2.txt

while query 51 searching for national institute of health organizations was supported better by all systems. It seems that the <narrative> section describing some kind of an association relation between the query entity and the result has a big influence on the quality of the result. If it is complex and has to be reasoned out of data, like in the case of query 36 then the results are poor. However, if the required entities are already linked to the query entity with the given relation on some page on the Web, like in the case of query 51[53], the results are much better.

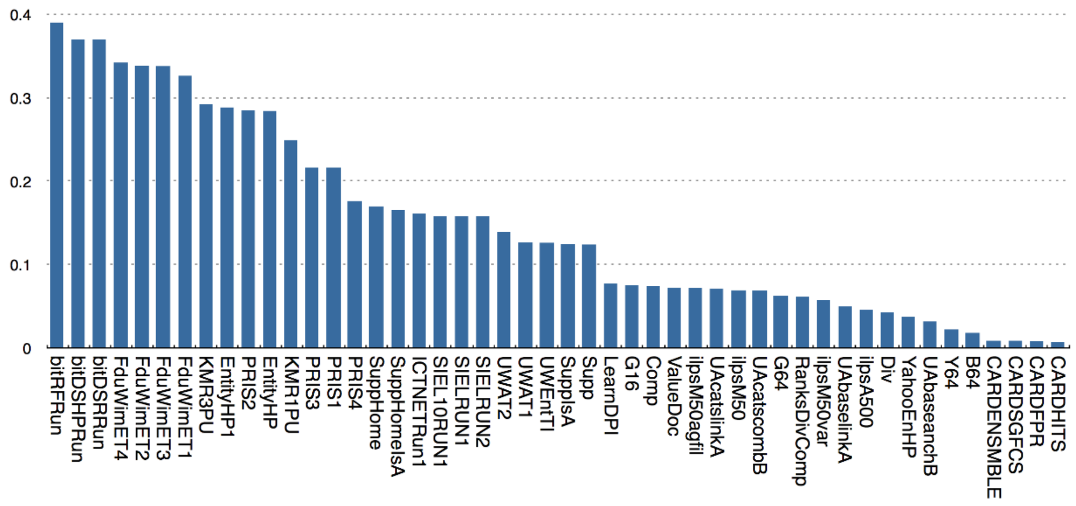
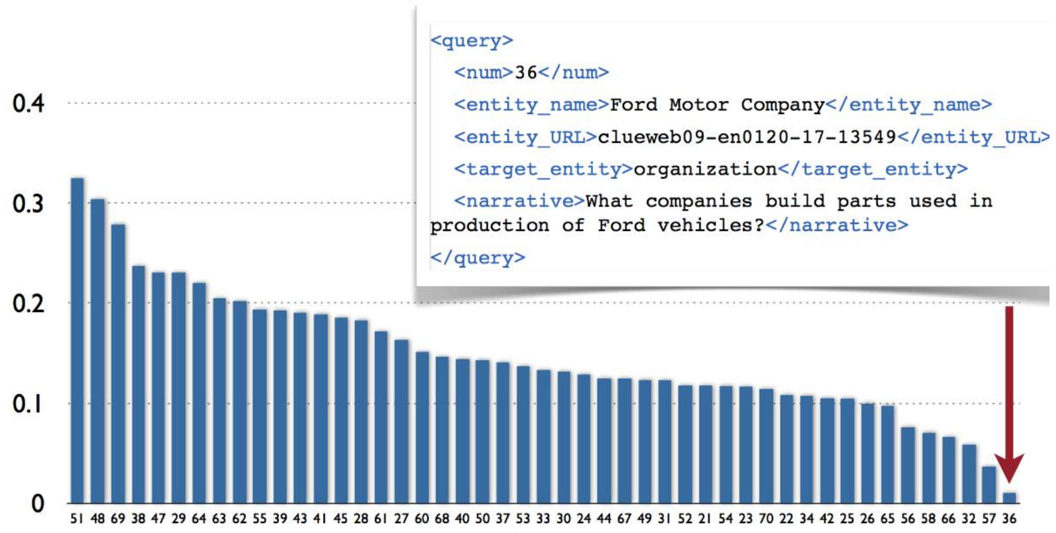


Fig. 24. Mean nDCG of 14 REF systems on TREC REF queries[54]. Results per system. – Balog et al. [6].



```
<query>
  <num>36</num>
  <entity_name>Ford Motor Company</entity_name>
  <entity_URL>clueweb09-en0120-17-13549</entity_URL>
  <target_entity>organization</target_entity>
  <narrative>What companies build parts used in
production of Ford vehicles?</narrative>
</query>
```

Fig. 25. Mean nDCG of 14 REF systems on TREC REF queries[54]. Results per query – Balog et al. [6].

[53] http://en.wikipedia.org/wiki/National_Institutes_of_Health#Institutes_and_Centers

[54] http://krisztianbalog.com/files/talks/smer2011-ref.pdf

The entity relation given in the <narrative> tag is complex and difficult to consider for entity selection. To provide for more information, a pilot task called Entity List Completion (ELC) was introduced in 2010 for the entity track of TREC. In addition to the information provided in REF queries, ELC queries comprise examples of target entities. The basic idea is that the positive examples of result entities be of some help in mining the user intentions if the narrative part of the query is too complex. Furthermore, besides the broad entity type of the target entities, a more specific type, usually from the DBPedia Ontology is also provided to narrow down the result list. An example of an ELC query from TREC 2011, comprising besides the query entity also seven exemplified target entities, is presented in Fig. 26. With this additional information, systems extended to support ELC queries like PRIS [132], or LIA-iSmart [17] already show better results with an average precision at 10 retrieved entities (further denoted as precision@10) of about 0.7.

In conclusion, queries formulated like in the case of ELC, comprising many more entities and having a specific target entity type, allow for results of better quality than REF. However, looking back at our problem of supporting entity centric search by means of instances, our experience in developing systems building on user input and user feedback ([56, 59]) shows that it is improbable that users provide such elaborated queries. With this in mind, our task is more similar to REF. But without

```
<query>
<num>25</num>
<entity_name>U.S. Supreme Court</entity_name>
<entity_homepage id="clueweb09-en0012-87-19363">http://www.supremecourtus.gov/</entity_homepage>
<target_entity>organization</target_entity>
<target_type_dbpedia>EducationalInstitution</target_type_dbpedia>
<narrative>From what schools did the Supreme Court justices receive their undergraduate degrees?</narrative>
<examples>
  <entity>
    <homepage id="clueweb09-en0005-75-28524">http://www.college.harvard.edu/</homepage>
    <name>harvard college</name>
  </entity>
  <entity>
    <homepage id="clueweb09-en0006-36-06636">http://www.cornell.edu/</homepage>
    <name>cornell university</name>
  </entity>
  <entity>
    <homepage id="clueweb09-en0008-46-00371">http://www.holycross.edu/</homepage>
    <name>college of the holy cross</name>
  </entity>
  <entity>
    <homepage id="clueweb09-en0008-52-12611">http://www.georgetown.edu/</homepage>
    <name>georgetown university</name>
  </entity>
  <entity>
    <homepage id="clueweb09-en0011-14-00284">http://www.stanford.edu/</homepage>
    <homepage id="clueweb09-en0040-19-16003">http://www.stanford.edu./</homepage>
    <name>stanford university</name>
  </entity>
  <entity>
    <homepage id="clueweb09-en0012-56-27384">http://www.uchicago.edu/</homepage>
    <name>university of chicago</name>
  </entity>
  <entity>
    <homepage id="clueweb09-en0132-25-55950">http://www.princeton.edu/rc/</homepage>
    <name>princeton university</name>
  </entity>
</examples>
</query>
```

Fig. 26. Example of ELC queries from TREC 2011[55].

[55] http://trec.nist.gov/data/entity/11/11.elc.topics.txt

examples of the expected entities, systems performing REF seem to have issues in fully understanding the requirements.

However, REF is more general than what we intend with our instance-based entity search system. With REF, one can freely express various constraints that are difficult to understand without further support. In our case, we are interested in finding entities of the target type, showing a high similarity towards the provided entity (the user given instance). In this respect, we consider that instance-based entity search, is a simplified REF task. Is this enough to provide for good quality results? Learning from the experience of both REF and ELC, in the next section we provide a detailed description of an instance-based entity search query that is both user friendly and effective in transmitting the user intentions.

5.2. The Instance-based Entity Query

The task in instance-based entity search is to find entities that are similar to a given example. With this task in mind, there are few important things to be learned from REF and ELC:

- Both REF and ELC are built to search for entities that are in a given relation to a given entity. Even if complex, in the case of ELC, this relation is made accessible with the help of positive examples of entities having that relation towards the query entity. For our task, we are interested in just one relation: entities that are highly similar to the query entity. If for instance, the user were to search for sports cars, with 'Ferrari 599' as an example, one would expect other Ferrari, Lamborghini, Aston Martin, or Maserati models to be returned rather than the Volkswagen Golf or the Ford Mondeo. In consequence, there is no need for a <narrative> tag or any other specification of the relation between the query entity and the target entity, as this is a constant component of the query.
- Both REF and ELC were meant to be used by advanced users with deeper knowledge of the Semantic Web or by information retrieval systems to improve search results. But neither of the two was intended for the large proportion of normal Web users. As such, the required information is not really user friendly: every piece of information that one has to provide in order for entity search to work, increases the cognitive pressure on the user. Our goal is to keep this load as low as possible. For this reason, we aim at a system that is able to work even with queries comprising just one entity. As learned from ELC, examples of target entities help to better understand the user intentions. Using a frequentist approach, systems like PRIS performing ELC, use these positive examples to derive a bag of words like vocabulary that is representative for these positive examples. Using this vocabulary as a model, such systems search for other entities using more or less the same vocabulary. At its core, this approach extracts the essence of the target entities and builds a prototypical representation. We argue that, especially for our task

of instance-based entity search, one does not require multiple examples of entities. Instead, we believe that one well-chosen example, which people usually associate with the intended entity type, is enough for a system to retrieve suitable entities.

- Both REF and ELC require that the user provides a target entity type. The entity type is also beneficial for disambiguation purposes. Can we spare the user from providing the entity type? According to our experiments it seems that without the entity type, it is difficult to disambiguate the user intentions. Even providing more than one entity as an example is less beneficial than providing an entity and the type. This problem will be discussed in more detail in the experimental section.

Picking up on the example of query 36 (presented in Fig. 25) from the REF track, which is not well supported by entity search systems, adapted to our task of instance-based entity search, the query in natural language would be to search for 'organizations like the Ford Motor Company'. The instance is 'Ford Motor Company' and the target entity kind is 'Organization'. The user query becomes in this case 'Ford Motor Company: Organization'.

The instance-based entity search query has the pattern 'Instance: Entity Type' and no further relation, narrative, or target entity examples are required. The relation is always the same (similar to), and since the instance is typical for the entity type, no other examples should be necessary. But if the entity type is provided, why also demand an entity? One could just retrieve all entities of the entity type given in the query like presented in Chapter 3. Indeed entity types are very useful in organizing entities. Actually, all information in Wikipedia, whose articles describe entities, is organized based on a hierarchical category system[56] built on entity types. This way, articles describing entities of the same type belong to the same category, bearing the entity type name. Manually inspecting articles on Wikipedia having the same entity category it can be observed that for entities of the same type, the structure of Wikipedia articles is often very similar with nearly identical first-level headings. Encouraged by this observation we analyzed a larger number of entities. For instance, starting from the list of 3,000 diseases featuring an article on Wikipedia we extracted the headings of all articles (purely structural headings of Wikipedia like "References" and "External Links" were pruned). Indeed, even over large samples of entities, a common structure can be extracted (see Fig. 27). A similar result holds in the case of American presidents, yet with lower percentages. Both entity types form homogeneous groups. However, this is not always the case: the same experiment performed on all companies from the S&P 500 list shows that, with the exception of only two headings (Products and Acquisitions) there is no common article structure for this category. Going a step further and inspecting the article headings and topics

[56] http://en.wikipedia.org/wiki/Wikipedia:Categorization

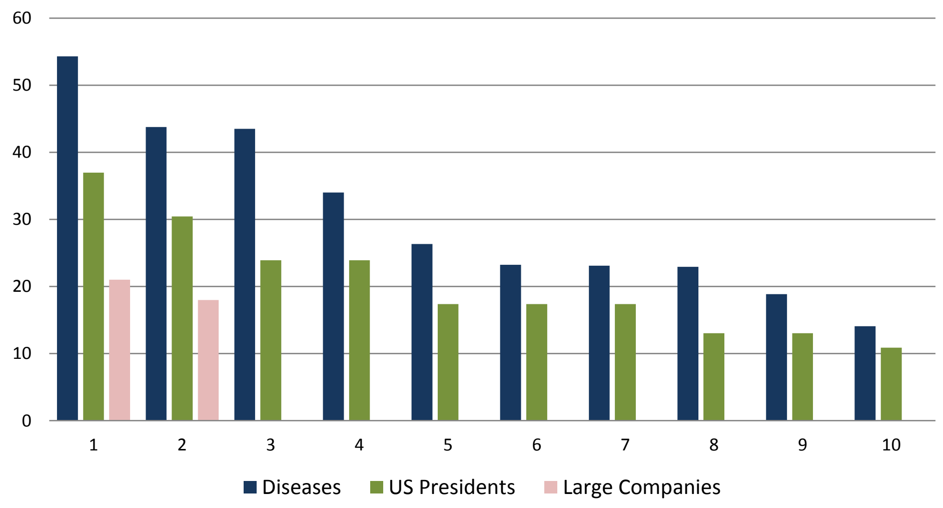


Fig. 27. Wikipedia article structure – percentage of entities (y-axis) belonging to the same category and sharing a certain heading (x-axis, values less than 10% are omitted).

for companies from the same business field, the structure becomes more homogeneous. For instance, articles for automotive companies often cover topics like 'Alliances' or 'Motorsport', in contrast to articles for pharmaceutical companies, where topics like 'Clinical Trials' and 'Litigation' are more common.

In consequence, entity types, and especially superordinate or basic entity types which are more general, may group heterogeneous entities together. Providing only the entity type, e.g., 'Organization' will return all kinds of companies and not only 'Ford Motor Company' and alike. In this case, in order to find a sweet spot between the more general entity type, and the user information needs, an entity example is required. Similar to an anchor, the entity fixates the focus of the user needs even within heterogeneous entity types. One could argue that tailored, subordinate entity types, like 'Car making companies' produce better results. Unfortunately, the principle of *economy and informativeness trade off* introduced by Loyd K. Komatsu, discussed in Chapter 3, strikes. The more specific the types the less chance that entities are categorized labeled or associated with them. In consequence less entities can be extracted from Web data. While systems like presented in Section 3.1, building on entity mining, have proven quite successful for subordinate entity types, it's difficult to know beforehand what kind of entity type the user will provide. We believe that in this respect, an example entity is valuable for fixating the user intention while allowing the system to be more robust and tolerant towards user input. Considering all this, in this chapter we inspect the value of an entity example plus entity type, for dealing also with more general entity types referring heterogeneous entities.

Of course the success of this approach is influenced by the choice of the query entity. While there is room for flexibility (e.g. providing 'Apple: Fruit' or 'Pear: Fruit' wouldn't make much difference), it's difficult to systematically support such queries if the entities provided by the user are but poor examples of the intended entity type (e.g. 'Olive: Fruit' or 'Tomato: Fruit'). Therefore, avoiding outliers, or better yet, aiming for entities that are typical for the entity type is in our opinion the basic requirement for instance-based entity search to work. Fortunately, experiments concerning typicality from the field of cognitive psychology have shown that people have no difficulty in remembering examples of typical entities if the entity type is known to them ([105]), so this requirement should not pose any difficulty to users.

With the query given as presented in this section, we are now ready to proceed to describing our instance-based entity search system.

5.3. System Description

The task is to find entities similar to a user provided example, and the query provided by the user is of the form 'Instance: Entity Type'. This is a simplification of the more general REF task. Our claim is that with this simplified type of query, but with a well-chosen entity as an input, being typical for the intended entity type, one can obtain acceptable results similar to ELC systems and much better than achieved by systems performing REF tasks, even with simple heuristics.

At the core of this claim lays the concept of typicality, which we will discuss in more detail in the following subsection.

5.3.1. Theoretical Foundations

Since its formal introduction in [104], the psychological concept of typicality has been widely researched and is now well established in cognitive psychology literature. It has been shown times and again that some instances of a semantic domain are more suitable than other instances to represent that domain: For example Jimmy Carter is a better example of an American president than William Henry Harrison. Leading the quest for defining the psychological concept of typicality, Eleanor Rosch showed empirically that the more similar an item was to all other items in a domain, the more typical the item was for that domain. In fact, the experiments show that typicality strongly correlates (Spearman rhos from 0.84 to 0.95 for six domains) with family resemblance a philosophical idea made popular by Ludwig Wittgenstein in [129]. For family resemblance Wittgenstein postulates that the way in which family members resemble each other is not defined by a (finite set of) specific property(-ies), but through a variety of properties that are shared by some, but not necessarily all members of a family. Based on this insight, Wittgenstein defines a simple family-member similarity measure based on property sharing:

$$S(X_1, X_2) = |X_1 \cap X_2| \qquad (24)$$

where X_1 and X_2 are the property sets of two members of the same family. However, this simple measure of family resemblance assumes that a larger number of common properties increase the perceived typicality, while larger numbers of distinct properties do not decrease it. But large numbers of distinctive properties for a family member should definitely not lead to the member's selection as a good example. Therefore, the model proposed by Tversky in [122] suggests that typicality increases with the number of shared properties, but to some degree is negatively affected by the distinctive properties:

$$S(X_1, X_2) = \frac{|X_1 \cap X_2|}{|X_1 \cap X_2| + \alpha|X_1 - X_2| + \beta|X_2 - X_1|} \tag{25}$$

where α and β ≥ 0 are parameters regulating the negative influence of distinctive

Definition 9: Family. *Let Q be an instance-based entity query, Q:=X: T(X), with X the instance entity, and T the entity type. Let C be the set of entities of type T. The family of X w.r.t. entity type T, denoted $F_{X,T}$, is a subset of entities from C, with:*

$$F_{X,T} = \{Y | Y \in C \land S(X, Y) > \theta\}$$

where $\boldsymbol{S(X, Y)}$ represents the similarity between entities X and Y (see eq. 25) and θ is a family specific threshold.

properties. In particular, when measuring the similarity of a family member X_2 to the family prototype X_1, a choice of α ≥ β poses the same or more weight to the properties of the prototype itself. For α = β = 1 this measure becomes the well-known Jaccard coefficient. For α+β ≤ 1 more weight is given to shared features, while for α+β > 1 diverse properties are emphasized, which is useful when dealing with more heterogeneous families.

Using either model allows to determine the pairwise similarity between all family members. The typicality score for each member is obtained by summing up its similarity values to all other members but this is not the focus here. The goal is to identify those entities that are similar to the entity provided in the query given that the query entity is typical for the intended entity type. Relating to the concept of family resemblance coined by Wittgenstein, the goal is to find the *family* of the entity provided as a query. To do this, we rely on Tversky's similarity measure (eq. 25) to find those k-nearest neighbors to the query entity. These entities not only have the same entity type as the query entity, but they also share similar structure.

The value of θ has to be established dynamically, based on the start entity and the entities falling into the same category. For this purpose, we employ automatic thresholding methods, in particular the ISODATA algorithm [5], a 1-dimensional form of the k-means clustering algorithm. Applied to the entities falling into the same category as the query, this method identifies the similarity threshold that splits the entities in two groups: one comprising homogeneous entities with high similarity to the query entity, which is the entity set that we want to return to the user, and one containing all the less similar entities.

The instance-based entity search is now reduced to finding the family of a given entity being representative for the intended entity type. With the task defined, in the following subsection we proceed to describing the entity retrieval system in detail.

5.3.2. System Architecture

The main task to be handled by the system is the computation of the family set according to the heuristics described in Definition 9. But from a technical stance, the main challenge lies in preparing the available data such that the results are satisfactory. As shown in [20] on the example of ELC tasks, approaches relying on both structured and unstructured data achieve results that are far superior to using only structured data for entity search. Our evaluation on entity type queries with both structured and unstructured data presented in Section 3.1.3, supports this observation. Building on state-of-the-art information extraction, for the prototype system we invest considerable effort to extract information from unstructured data and make it available in triple form, to ultimately ensure the best setting for instance-based entity search. Fig. 28 shows an overview of the system. In brief, the system works as follows:

- **Information Extraction:** Documents crawled from the Web, are processed with Open Information Extraction (OpenIE) methods. This results in a large number of facts represented as triples (subject, predicate, object). The subject and the object usually contain entities, while predicates represent attributes of the entities, see [82]. Since the same entity can be expressed in multiple forms, an *Entity Dictionary* listing unique entities and their possible string representations is kept and updated. Similarly, some attribute may be expressed by synonymous predicates. Therefore, unique attributes and possible representations are also stored as a *Paraphrase Dictionary*. All

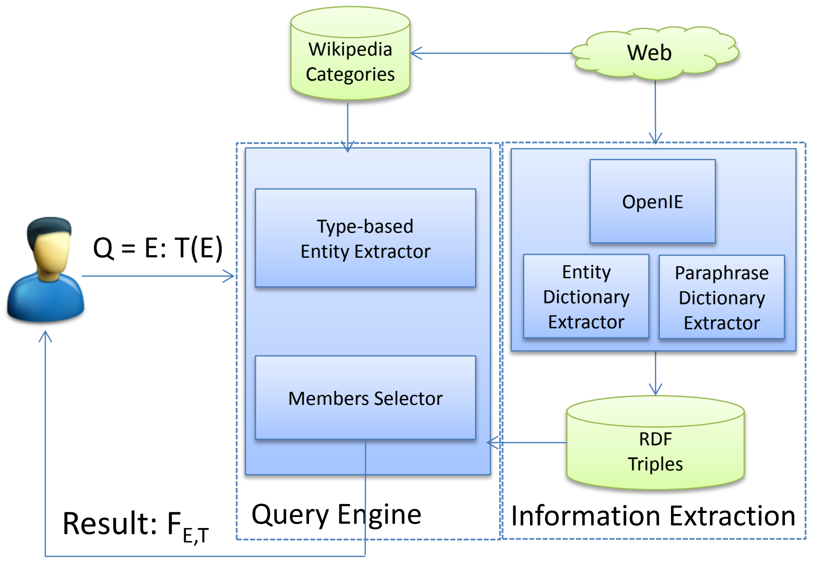


Fig. 28. Instance-based entity search based on family resemblance. System architecture.

extracted facts are cleaned based on these dictionaries. Then, they are stored in a knowledge base (we use a Virtuoso RDF database in our prototype).

- **Query Engine:** The query engine allows users to provide a query entity together with the corresponding entity type. Starting from the provided entity type all known entities belonging to this type are extracted either directly from Web documents or, like in the case of ELC tasks, from special dictionaries like Wikipedia. In a filtering step only entities being most similar to the query entity are selected to form the query's *family*. This set of entities is returned to the user.

The remainder of this section will provide a more detailed description of the architecture's main components.

Information Extraction
The information extraction (IE) module is responsible for processing documents crawled from the Web and for providing a relatively clean triple collection with disambiguated entities and predicates. Being query-independent, all these operations are performed offline. While considerable efforts are being made in recent developments like the PATTY system [88], until now there is no readily available framework that provides this complete functionality. In consequence, we employ state-of-the-art OpenIE tools to solve this task. For our implementation we used ReVerb [37], but basically any OpenIE tool that fulfills a basic level of quality requirements can be used. The IE component needs two types of dictionaries to work properly: one for uniquely identifying entities and one for uniquely identifying predicates.

Entity Dictionary: In Web documents (including news, blogs, tweets, etc.) the same entity is often represented by various strings. Two problems have to be discussed here: *synonymy*, i.e. every entity can have more than just one string representation form, e.g. "Barack Obama", "B. H. Obama", etc. and *ambiguity*, i.e. every string can refer to different entities e.g. "Clinton" may refer either to "Bill Clinton" or to "Hillary Clinton".

For synonymy, we assume that a mapping from different strings to the entity is provided. For simplicity, our prototype is restricted to Wikipedia entities, and uses the different string representation forms each entity has been labeled with in Wikipedia. For better coverage, different thesauri like WordNet or MeSH can be used.

Solving the problem of ambiguity is known as Entity Disambiguation. In order to solve this, we follow the rule of thumb that any ambiguous reference to some entity, say "Clinton", is preceded in the document by some clear entity reference like "President Clinton" or "Mrs. Clinton". If no such clear-cut reference can be found, we relax our assumption by following on the approach introduced in [82] assuming that each entity string is uniquely addressing exactly one entity within a document. In this way, on a document basis, we fill up the entity dictionary with string representations and the corresponding unique entity identifiers. The entity dictionary is used to clean all facts extracted by OpenIE by replacing all entity names with unique, disambiguated identifiers.

Paraphrase Dictionary: As in the case of entities, predicates are also expressed by means of synonym terms (president_of, won_elections_in, was_elected_president_of, etc.). However, in the case of predicates an acceptable mapping between a meaning and its representation forms has yet to be developed. The field of paraphrase discovery is concerned with this problem [8]. State-of-the-art methods rely on a class of metrics called *distributional similarity metrics* [75] built on the assumption that similar objects appear in similar context (known as the distributional hypothesis [52]). In the context of paraphrase discovery, this hypothesis is applied as: two predicates are paraphrases of each other, if they are similarly distributed over a set of pairs of entity-types. Furthermore, in contrast to the entity ambiguity problem, a simplifying assumption is made: predicates can't have multiple meanings (single-sense assumption [130]). Following these insights and similar to approaches like for example in [51], we applied hierarchical clustering to the predicate/entity-type pairs distributions. As a similarity measure we have used the well-known cosine metric with mean linkage as criteria. Still, despite experimenting with different similarity thresholds, the success of the paraphrasing process is rather limited. While in manual inspection the clusters prove good precision, just about 7% (for 0.9 similarity threshold) actually build clusters. Even lowering the similarity threshold to 0.7 only increases this number to 16%. The rest of the predicates build single node clusters although a substantial number of cases show obvious paraphrases. This is consistent with results from the literature [130], where even with enhanced information, the recall barely reaches 35%.

For each predicate, the corresponding cluster representative is determined as the most frequent predicate of the respective cluster. Each predicate and cluster representative is then inserted into the paraphrase dictionary. The system uses the paraphrase dictionary to clean the triples regarding predicate synonymy.

Query Engine

The query engine module represents the online part of the system. It is responsible for extracting the entity family. It accomplishes two tasks: it extracts entities of the given type, and it restricts the extracted entities to a small collection of entities having most similar structure to the query entity.

The first step in this direction is to identify all entities of the type given through the query. To do this, a mapping between the entities and the corresponding entity types is needed. Such mappings can be extracted directly from text with state-of-the-art entity class extraction methods. Such approaches build on lexico-syntactic patterns, like "…an *X* such as *Y*…" or "… all *X*, including *Y*…" expressing "is-a" hierarchies between entity category *X* and entity *Y*. As the focus of this thesis is not the technical process of extracting categories, in the same fashion as ELC systems, we also rely on the entity-category mapping provided by Wikipedia to find other entities in the RDF triples having the same category as the query entity. In the second step, the system filters the entities according to Definition 9.

For better overview of how the quantification of attribute typicality is performed we present the pseudo-code of our system's algorithm in Algorithm 1.

Runtime Analysis

Like all other REF systems, for our tests, we also rely on ClueWeb09 as a data set. ClueWeb09 contains 500 million Web documents. All documents were processed offline by our IE module and billions of noisy triples were extracted. After filtering out triples that are infrequent or have low confidence values according to ReVerb, only approx. 15 million triples, with about 2.2 million entities and 0.6 million predicates remain. On average, the IE module needs about 1 minute to process 8,000 sentences. On commodity hardware, the complete process for all documents took about 11 days.

For the online part (Algorithm 1), our system requires even for broad entity types with thousands of entities about 25 seconds per query. For instance, in the case

Algorithm 1: Algorithm for selecting the entity family.

Input: X - query entity, C - set of entities of type T, ϕ - attribute quality threshold, RDF triple collection

Output: F - set of entities forming the family of X

```
1: $F \leftarrow \{X\}$; $x\_attr \leftarrow$ ATTRIBUTES($X$, RDF)
2: foreach $Y$ in $C$ do
3:     $y\_attr \leftarrow$ ATTRIBUTES($Y$, RDF)
4:     $sim \leftarrow$ similarity($x\_attr$, $y\_attr$)     // Tversky's similarity, eq. 25
                             // computed only once then stored in memory for later use (**)
5:     $S \leftarrow S \cup sim$
6: end for
7: $\theta \leftarrow$ THRESHOLD($S$)
8: foreach $Y$ in $C$ do
9:     $y\_attr \leftarrow$ ATTRIBUTES($Y$,RDF)
10:    $sim \leftarrow$ similarity($x\_attr$, $y\_attr$)     // (**)
11:    if $sim \geq \theta$ then
12:        $F \leftarrow F \cup Y$
13:    end if
14: end for
15: return $F$

16: function THRESHOLD($S$)                 // the ISODATA method
17:    $\theta \leftarrow$ avg($S$); $found \leftarrow false$
18:    while $found = false$ do
19:        $low \leftarrow$ avgLower($S$, $\theta$)          // average of values < $\theta$
20:        $high \leftarrow$ avgHigherEqual($S$, $\theta$)   // average of values ≥ $\theta$
21:        $\theta_{new} \leftarrow (low + high)/2$
22:        if abs($\theta_{new} - \theta$) < $\varepsilon$ then
23:            $found \leftarrow true$
24:        end if
25:        $\theta \leftarrow \theta_{new}$
26:    end while
27: return $\theta$
```

entities of type medical condition or disease, there are 3,513 entities in the disease category on Wikipedia. For query "hypertension", our system needs 22.472 seconds to extract the family of the query entity. This covers the following parts: Extracting all attributes for all entities (13.350 seconds – an average of 3.8 milliseconds per entity); pairwise comparing the 1,329 found in the extracted statements (8.917 seconds – an average of 6.7 milliseconds per comparison); computing the family threshold (21 milliseconds). All other operations (assignments, logical, arithmetical operators) for the family computation require 184 milliseconds.

All tests have been performed single threaded. Since all major operations allow for parallelization, we have reason to believe that parallelization on a cluster with a few hundred CPU cores will reduce query time to less than a second.

5.3.3. Evaluation

REF and ELC are the standard tasks which build on example-driven entity search. But systems performing REF achieve poor results with an average precision@10 of about 0.4. Profiting from a more elaborate query, comprising numerous target entity examples, ELC systems achieve acceptable results of about 0.7 average precision@10. But the query is more complex, and not acceptable for the casual Web user. The problem for the poor results of REF systems is the complex relation expressed in natural language that such systems have to support. However, for instance-based entity search supporting such relations is not needed. Actually there is just one relation, that all target entities are highly similar to the entity given as example. We believe that in this case, one can achieve results that are comparable with ELC, without having to impose complex queries on the user. In Section 5.2 we claimed that a query comprising a single entity and its type is enough. But is the entity type necessary or is the entity by itself, or maybe together with a few examples already enough for a system to disambiguate according to the user intention? In the following we present an experiment focused on establishing the minimalistic form of the query. Afterwards, we proceed to evaluating the quality of the entities extracted by means of family resemblance, for various entities.

Experimental Setup

Dataset: Presented in the previous section under runtime analysis, all our experiments are conducted on the English part of the ClueWeb09, a standard corpus for entity search tasks in TREC.

Queries. We experiment with three practical types of entities identified in [74] as most popular entity-centric queries on the Web. Since named entities are of special interest for most applications, we use two types of named entities in our tests: persons (in the sense of American presidents) and organizations (in the sense of companies). In total we experiment over 544 queries split as follows: 44 American presidents and 500 companies.

Establishing Ground Truth: TREC provides data samples and gold standards for evaluation purposes. These data samples are built for either REF or ELC, and they comprise various complex relations between the provided entity and the target entities. In most cases, the query entity and the target entity don't even share the same type. As a consequence the samples from TREC are not useful for our evaluation and no other suitable evaluation dataset is available. We rely in this case on human assessment.

Measures. For the disambiguation experiments no entity types are provided. Given that in family resemblance entities are extracted based on structural similarity, we compare the success of various query types based on the quality of the extracted structure. For this purpose we measure the quality of the results in terms of precision@10 on attributes. Precision@10 is also the measure for the entity retrieval experiments presented in the second part of the evaluation section, only this time the quality of the entities found to be similar to the query entity is measured.

Disambiguation of Queries

As previously stated a query consists of two parts: an entity and the entity type for disambiguation. This is because a single entity is not nearly enough to understand what the user intended. Still, allowing users to give some examples might also help disambiguation. The more examples, the better a query can be disambiguated, however increasing query complexity. On the other hand, following on the example of REF and ELC systems, a user provided entity type leads to easy and high quality disambiguation. To establish which query form is better, we performed three experiments:

a) Users provide an entity, without additional information.
b) Users provide five entities of a similar kind - most users are able to provide three to five examples. The cognitive burden increases heavily beyond that.
c) Users provide an entity together with the entity type.

Evaluation Method: Since our approach relies on resemblance based on structure, i.e. shared attributes, without an entity type, for experiments a) and b) there is no entity set to filter the family from. A possible approach would be to consider all kinds of entities to start with, but with millions of entities on the Web such an approach is not really practical. Still, in order to evaluate the usefulness of the query, we measure the quality of the entity structure being extracted starting from a single entity. Drawing on the ELC literature favoring frequentist methods, we assume that attributes frequently appearing together with either the query entity for experiment a), with all entities provided as query for experiment b), and with all entities of the entity type given in the query for experiment c) define a good structure for their entity type. Relying on the infrastructure as provided by the information extraction described in section 5.3.2 we thus implemented the frequency-based baseline approach.

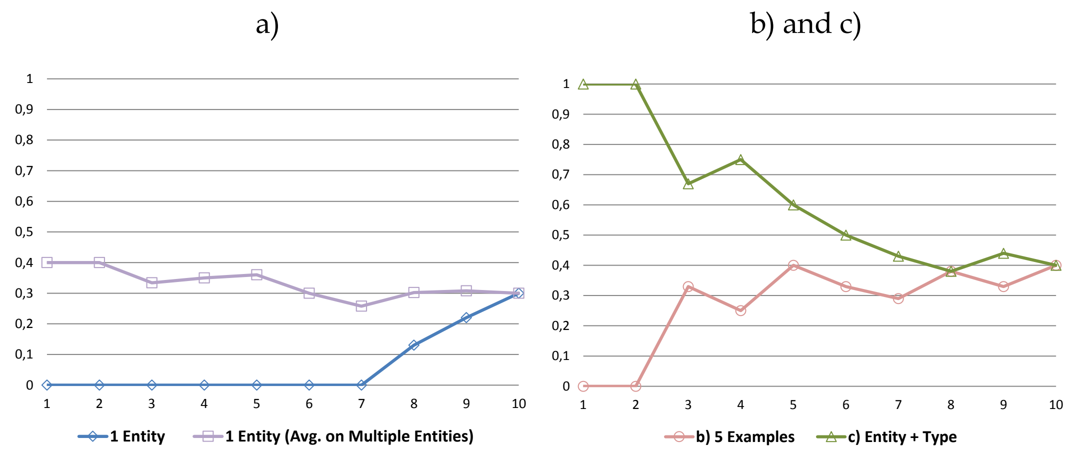


Fig. 29. Disambiguation of queries: query comprising a single entity - (a) on the left hand side, five entities - (b) on the right hand side or an entity and its entity type - (c) on the right hand side.

In Fig. 29 we present the top 10 attributes for our running example: American presidents. For just one entity (Fig. 29.a), the frequency-based method's precision proves really poor. No disambiguation can be performed and thus, all kinds of attributes are considered. Barack Obama (the blue line in Fig. 29.a) has proven to be an unlucky choice for the frequency-based approach. Averaging the precision over multiple American presidents shows an overview of the results (the magenta line in Fig. 29.a). In the case of five entity examples for disambiguation, the quality of the results is average at best (the red line in Fig. 29.b). Again, the reason is proper disambiguation. Finally when the category of American presidents is also provided (the green line in Fig. 29.c), a single entity is enough for extracting high quality structure.

Hence a query consisting of some entity and a respective category leads to better disambiguation, while requiring also less effort than a list of examples.

Family-based Entity Extraction

Through this chapter we argued that our approach based on family resemblance is particularly useful for handling the more difficult queries with entity types grouping together heterogeneous entities. Our experiment on the Wikipedia article content, has shown that 'organizations' is such an entity type. In consequence, in this section, we evaluate the quality of the retrieved entities on the example of the S&P 500 list of companies and ClueWeb09 as a data source.

The query always has the same structure: 'company name: Organization'. For each query, the system compares the query entity with all other 499 companies from the S&P list on structural similarity to ultimately select the family of entities highly similar to the query entity. For companies like "Toyota Motor Corporation", "Renault S.A.", or "Volkswagen A.G.", which are typical car makers, our systems builds families with 17 to 24 entities, clearly focusing on car companies (30% - 50% of the selected family members are car makers). For queries like "Apple Inc.", "Google

Inc." or "Microsoft" the results are similar, with 40% - 60% of IT companies in the selected family. The same behavior has been observed also for companies from the energy, finance, food, medical, and other sectors. This is consistent with our analysis on Wikipedia, which showed that articles already feature homogeneous content for companies from the same activity field. Driven by a single entity as an example, our self-tuning approach focusing around the family of the query entity is quite successful in finding a semantically meaningful sweet spot between the more general superordinate or basic entity types, and the very specific types.

Our system is not directly comparable with systems performing REF or ELC because of the different focus in terms of the relation between the query and target entities. But since these standard entity search tasks are the closest to solving the problem of instance-based entity search, they may still be useful as reference. To measure the results of our approach, we computed precision@10 for various entities by manually inspecting on the entities belonging to the same field as the query entity. In **Table 17** we present the results for entities from the field of automotive and IT. With an average precision@10 of 0.72 and 0.8 for automotive and IT companies respectively, the results are far superior to the average 0.4 achieved by bitRFRun (the system achieving the best results for the REF TREC task - Fig. 24). Even with more simple user friendly queries, our results are more similar to the 0.7 average precision@10 of Pris and LIAiSmart for the ELC TREC task.

Table 17: Precision@10 for organizations from the fields of automotive and IT.

Query	Precision@10	Field
BMW: Organization	0.6	Automotive
Fiat: Organization	0.7	
Toyota: Organization	0.7	
Renault: Organization	0.8	
Volkswagen: Organization	0.8	
Apple: Organization	0.8	IT
Google: Organization	0.8	
IBM: Organization	0.8	
Microsoft: Organization	0.8	
Yahoo: Organization	0.8	

It’s interesting to notice that even though we queried on different entities, the corresponding company field was always correctly identified. A level of flexibility regarding the query entity is provided. Throughout this chapter we argued that entities showing high typicality towards the intended entity type are more helpful for grounding the user information needs. Indeed, our experiment shows that for companies like Honda, an automotive company better known for its motorcycle manufacturing (Honda has been the world's largest motorcycle manufacturer since 1959) with a value of 0.1 achieved precision@10 is rather low. The family is a mix of companies, and Harley Davidson seems to be more related to it than any other car manufacturer out of the S&P 500 list of companies. As expected and according to the concept of typicality introduced by Rosch et al, some atypical examples of entities can be misleading causing poor selection results.

5.4. Conclusions

In this chapter we presented an in-depth view of *instance-based entity search*. This type of search is inspired by query by example, a type of search where users give one or more examples and the retrieval system returns similar objects. Adapted to entity search, users provide examples of target entities. Up to a handful of examples, it should not be a problem for most users, since one already visualizes suitable entities when performing the search anyway. Obviously the number of examples plays an important role: requiring fewer examples, the system is more user friendly. But fewer examples also capture less information about the intended entity type. This ultimately affects the power to disambiguate the exact user intentions. In consequence our prime concern regarding this query type is to find an instance-based entity search query structure that is user friendly, but at the same time allows for proper disambiguation.

The most basic and user friendly form of instance-based query comprises one single entity. But our experiments on the example of American presidents show that a query comprising one single entity leads to poor results. Furthermore, the choice of the entity greatly influences the outcome in terms of disambiguation. Even for queries comprising up to five example entities the results were only average. From our previous work we have learned that only few users are able and willing to provide more than five examples. In consequence, an instance-based query comprising examples only, is not sufficient.

Standard entity search tasks based on examples like REF and ELC have more elaborate queries. They comprise the example entity, the target entity type, the relation between the source and the target entity and in the case of ELC also multiple examples of target entities. Such queries are obviously not adequate for end users. For our task, from all these elements, the entity type seems to bring the most. Actually, as we have seen in Chapter 3, the entity type by itself is already enough to perform entity search. Our experiment on the Wikipedia entity types (which are the standard types used by ELC systems) has shown that type search works well if the types provided by the user are homogeneous, classifying together entities that are highly similar. But the success of the retrieval process should not be tied to the entity type being homogeneous or not. The system has to be flexible enough such that if the entity type is heterogeneous, the result still matches the user intentions. Our experiments show that even a single example of a target entity, is very valuable in this respect. All things considered, we have reason to believe that the minimalistic form of the instance-based entity search query should comprise only an example and the corresponding entity type. This is much less than standard example-based entity search tasks, and offers more flexibility than just entity type-based search.

But the example entity is only valuable if it captures at least to some extent the user intentions. As a constraint, outliers have to be avoided. Assuming that the entity example is well-chosen, we build on the concept of family resemblance and provide a practical way for computing families of entities. These entities are of the type provided by the user, and together with the example entity form a homogeneous group. Such an approach achieves impressive results being able to retrieve semantically meaningful entities even for entity types, which have proven problematic for REF and ELC (Fig. 25).

All in all, our analysis on instance-based queries for entity search has shown that up to five examples are not enough for proper disambiguation. With a query comprising an entity and corresponding entity type, and a simple similarity-based system the results are already practical. Furthermore, the system is more robust, as it can be more tolerant with the user, in terms of the chosen entity or entity type, as long as the entity is not an outlier and the entity type still makes sense.

This chapter concludes our presentation of methods for searching for entities. But lately, in the wake of systems like the Google Knowledge Graph, another kind of entity-centric search, focused on providing concise entity summaries has received a lot of attention. In the following chapter we pay closer attention also to this kind of queries.

Chapter 6

Entity Summarization

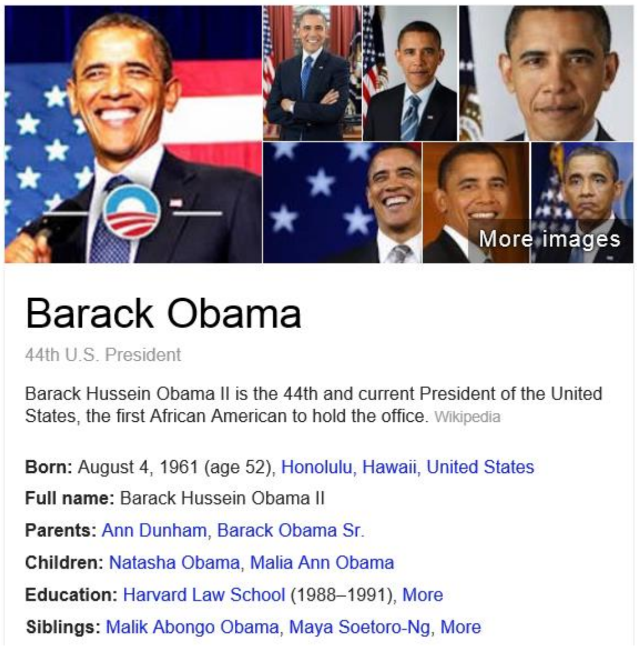
Most queries on Web data focus on searching for entities. We have covered this kind of queries extensively in the last three chapters. But lately, entity summarization which is another kind of entity-centric search, has strongly emerged: want to get some idea about a celebrity, a company or even a disease? Google it! Indeed Google evolved to accommodate this kind of queries. Today, the popular search engine features new entity summarization functionality called the Knowledge Graph. Integrated directly into the Web search page it summarizes knowledge of common interest using some fixed schema to provide a good entity overview. After typing some entity name into Google's search field, an entity summary is provided on the right hand side of the search results, if the Knowledge Graph contains the entity. A sample entity summary for 'Barack Obama' is shown in Fig. 30.

According to Google's official blog[57], the Graph mainly relies on manually curated data sources like Wikipedia Infoboxes, Google's Freebase, and schema.org annotations on the Web. But the Knowledge Graph has a major shortcoming: it doesn't cope with the number of new entities published daily on the Web. It only provides information on well-known entities already having a Wikipedia article, Freebase record or sufficient schema.org annotations. Our extensive evaluation on the example of diseases shows that, with just 3,000 out of 14,199 diseases featuring a Wikipedia article or Freebase entry, this is indeed rather limited. Considering its low acceptance of only about 1.5% of the websites, schema.org doesn't contribute much to extending the knowledge base either. This way, the majority of entities (in particular, new or more obscure entities) not present in the manually curated Web resources used by Google's Knowledge Graph, cannot benefit from data summarization.

We argue that a *data-driven approach*, of building entity summaries drawing not only from existing knowledge bases but also directly from unstructured data on the Web, is more suitable for entity-centric search. Inspecting ClueWeb09, approximately 11,000 statements regarding Barack Obama can be extracted using entity recognition and NLP techniques[58]. But the volume of information is huge and hardly appropriate for giving an overview of an entity. With the information needs of the majority of users in mind, when browsing through the large variety of *attribute: value*

[57] http://www.googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html

[58] Like in previously presented analysis on ClueWeb09, the statements are structured as triples of the form (*subject*, *predicate*, *object*). Predicates represent *attributes* and objects represent the corresponding *values*.



Barack Obama

44th U.S. President

Barack Hussein Obama II is the 44th and current President of the United States, the first African American to hold the office. Wikipedia

Born: August 4, 1961 (age 52), Honolulu, Hawaii, United States

Full name: Barack Hussein Obama II

Parents: Ann Dunham, Barack Obama Sr.

Children: Natasha Obama, Malia Ann Obama

Education: Harvard Law School (1988–1991), More

Siblings: Malik Abongo Obama, Maya Soetoro-Ng, More

Fig. 30. Knowledge Graph – results for Barack Obama.

pairs for 'Barack Obama', we found that many of them, e.g., visit: Israel, love: Broccoli or spent_vacation_in: Hawaii, seem irrelevant for satisfying common information needs. Can attributes like visit or love be recognized as irrelevant and pruned to obtain a suitable, yet concise structure?

The first idea that comes to mind is a frequency-based solution. Approaches like the count of witnesses as a measure for the importance of attributes have often proven efficient [27, 82]. Together with the Knowledge Graph, they serve as baseline for evaluating new approaches. But browsing through the triples for Barack Obama, one can observe that some of the information is common to all American presidents. For instance they all share features like their year of election, term in office, being members of some party, etc. One could say they form a *small world* built on characteristics that are typical for American presidents. Intuitively, a data-driven entity summary for "Barack Obama" as an "American president" would comprise a few, good descriptive properties selected from these shared characteristics.

Taking a closer look at how the attributes extracted for Barack Obama are actually shared among the 44 American presidents (see Fig. 31) a typical power law distribution can be observed. While the attributes that are common and important for this small world of presidents fall into the head of the distribution, the tail mostly comprises trivia about individual presidents. That means, by simply chopping off the tail, one might already identify common attributes of good quality. Is such a distribution valid for *all types* of entities, i.e. can the lessons learned from the small and

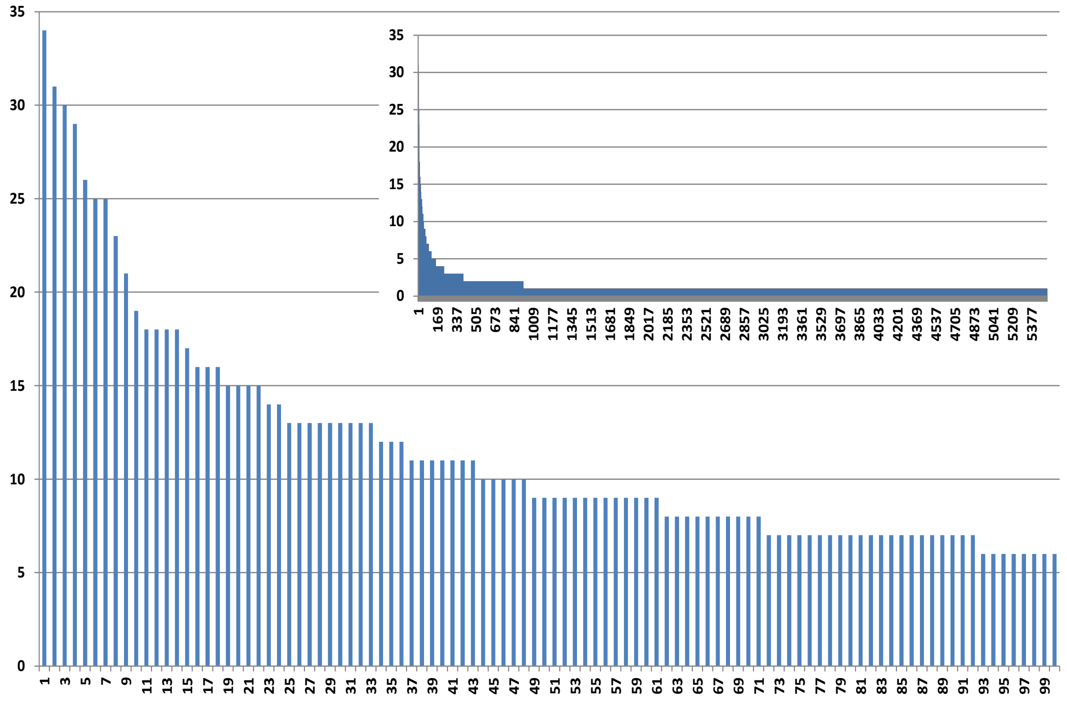


Fig. 31. Distribution of extracted attributes (x-axis) sorted by how many American presidents (y-axis) share each attribute (with zoom-in on the first 100 attributes).

homogeneous set of 44 American presidents be generalized? And, how can this distribution *efficiently* be derived and pruned, i.e. can this also be performed for classes with thousands of entities?

Motivated by these observations, in this chapter, we present *ARES* (AttRibute selector for Entity Summaries) a system for extracting data-driven structure for entity summarization. Regarding the user query, we first considered a query comprising just the entity of interest. However, our experiments presented in Chapter 4 in Section 5.3.3, show that disambiguation is best when besides the entity also the entity type is provided. For this reason, and similar to SCAD [4], the query for the task of entity summarization presented in this chapter comprises both the entity and entity type, e.g., “Barack Obama: American President”. After all, under the bonnet, the Knowledge Graph enriches the query entity with the Wikipedia category system in a similar way as we do. For such queries, ARES delivers highly typical attributes for the query entity in the context of the provided type. Finding the attribute values to complete the entity summary becomes straight-forward once the structure has been extracted [4, 134].

Exploiting facts extracted from the Web the main task of ARES is to derive a *common entity structure* with high quality attributes, typical for entities of the same or at least similar kind. In the previous chapter we built on the concept of typicality to extract *families* comprising entities showing high similarity amongst one another. Following on this idea, in this chapter we extend the concept of typicality and define *attribute typicality* together with a novel and practical rule for actually calculating it. We evaluate the quality of extracted attributes in terms of *precision* and *recall* deploying the basic structure from matching Wikipedia articles as *ground truth* together with human assessment.

But what should such a summary comprise and how long should it be? These questions will be addressed in Section 6.1, followed by the discussion of related work (Section 6.2) and the presentation of ARES, its theoretical foundation and the evaluation of the entity summaries produced by ARES for various entities and entity types (Section 6.3).

6.1. Reverse-Engineering Google's Knowledge Graph

The Knowledge Graph is the main reference system when it comes to entity summarization. Before going any further, we believe it is important to understand how this system works and more important what data sources the Graph relies on. While no scientific works have been published on this topic, according to Google the Knowledge Graph mainly relies on the Wikipedia Infoboxes, Freebase and schema.org annotations. As we have shown in Section 2.2 schema.org did not gain traction. There were but few annotations, and most of them (more than 66%) referred to products, news articles, movies or music. Unfortunately, for entities not related to e-shopping and especially for entities of broad interest like medical conditions the number of annotations didn't reach critical mass. But having at least some annotations for each entity is crucial for the Knowledge Graph to provide high quality summaries: the reliability of a piece of information is questionable if found in just few annotations from some unknown websites.

Schema.org is hardly used on the Web. In consequence it can't really contribute to the Knowledge Graph. It seems that the Knowledge Graph is mostly limited to entities from Wikipedia and Freebase. Similar observations have been made by technology blogs[59] that browsed through entities featuring a Knowledge Graph snippet. But only entities that reach a certain level of interest make it into Wikipedia or Freebase. This means that the Knowledge Graph is not able to scale with the thousands of entities hitting the Web each day.

Wikipedia and Freebase largely overlap in terms of the covered entities and entity structure. Freebase is mainly focused on providing structured information (same as Wikipedia Infoboxes but more extensive). Wikipedia additionally provides better, more accurate textual description, each entity being presented in a comprising article. Infoboxes are fixed-format tables built on one or more hierarchical Infobox templates. The purpose of Infoboxes is to consistently present a summary of some unifying aspects that articles share. The idea is that articles of entities of a similar kind share the same Infobox structure. In this way, similar entities share the same

[59] http://www.mkbergman.com/1009/deconstructing-the-google-knowledge-graph

structure that should flow into the corresponding Knowledge Graph entity summaries.

But the Infoboxes can be quite extensive, often having more than 30 attributes, much more than the Knowledge Graph snippet should comprise. Choosing the "right" attributes to build the entity summary is vital for the whole system. For instance, the snippet for Barack Obama (Fig. 30) comprises 6 attributes part of the "personal details" section from the Wikipedia Infobox (derived from the Infobox Person template[60]). However, nothing really specific regarding his activity as a president is mentioned, other than the first sentence being copied from the Wikipedia article. The information chosen to display in the Knowledge Graph is quite general, common to any person be it a politician, writer, painter, actor or any other personality. In fact, the same snippet structure is provided for instance also for actor Kevin Bacon. But relevant information like the year Obama took office or which political party he belongs to, are not being included in the Graph's summary despite being present in the corresponding Infobox. It seems that for this entity, the Knowledge Graph only presents the first few attributes of the broader Infobox template the entity is associated with – in this case the *Person* Infobox template. This method obviously misses out on important information that we believe should be included in the summary. Instead finding a sweet-spot between too broad and too specific information, like for instance a subset of the Office holder template[61], seems more sensible for choosing the attributes to include in the summary.

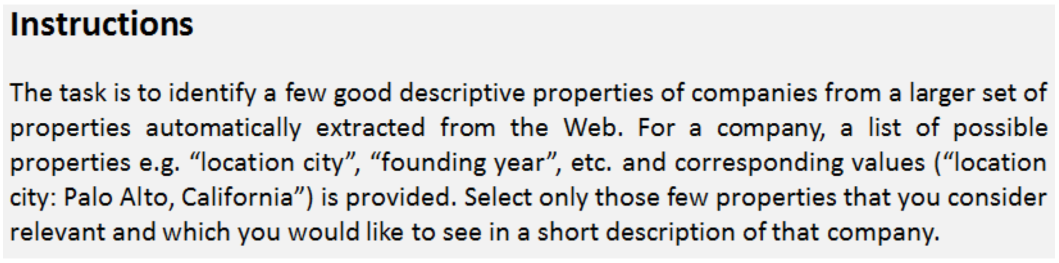
But is this only a problem of choosing the right attributes from Infoboxes, i.e. do Infoboxes include the right attributes to build entity summaries? In Section 5.2 we presented an experiment showing that the Wikipedia article richness is not always correctly captured by the uniform article structure. The reason for this behavior lies in the fact that the Wikipedia entity category system is not always as specific as needed, grouping also articles of heterogeneous entities together. On the example of companies from the S&P 500 list, our experiment showed that, with the exception of only two headings (Products and Acquisitions) there is no common article structure for this category of entities. Going a step further and inspecting the article headings and topics for companies from the same business field, the structure becomes more homogeneous. Despite articles for companies being highly heterogeneous, all their Infoboxes follow the same template (the Company Infobox template[62]), summarizing information with 41 generic attributes. Does this general structure provide suitable selections of attributes that reflect article differences?

Focusing on the structure provided by the Infoboxes we conducted an experiment to investigate two aspects: the *number of expected attributes* (i.e., how many an entity

[60] http://en.wikipedia.org/wiki/Template:Infobox_person

[61] http://en.wikipedia.org/wiki/Template:Infobox_officeholder

[62] http://en.wikipedia.org/wiki/Template:Infobox_company


Instructions

The task is to identify a few good descriptive properties of companies from a larger set of properties automatically extracted from the Web. For a company, a list of possible properties e.g. "location city", "founding year", etc. and corresponding values ("location city: Palo Alto, California") is provided. Select only those few properties that you consider relevant and which you would like to see in a short description of that company.

Fig. 32. Instructions on how to complete the task of selecting attributes suitable for entity summaries.

summary should feature) and the *suitability of generic attributes* encompassing all companies to build knowledge snippet structures reflecting important aspects of the heterogeneous Wikipedia articles. We selected 50 companies, split into 10 groups, each group corresponding to a major business field (e.g. Automotive, Energy, Financial, IT, Retail, etc.). DBpedia invested considerable effort in manually handcrafting mappings for extracting relatively clean Infobox attributes. After manually eliminating semantic duplicates (paraphrases) from the DBpedia Ontology[63] for the Infoboxes, the number of applicable attributes was reduced to 27. Each company and its corresponding attributes and values have then been presented to 25 human subjects. Since the concept of "entity summarization" may not be familiar to everybody, the task instruction was to select those few relevant properties they would like to see in a short description of the company (Fig. 32). The experiment was conducted through a crowdsourcing platform (CrowdFlower[64]) and targeted only workers from the Amazon Mechanical Turk. Being more complex than classical crowdsourcing tasks, this task required thorough understanding of the instructions. To minimize the risk of receiving workers that have low English skills, we limited the workers' country of origin to USA. In total we collected 1250 judgments. Companies were presented in random order and attributes were shuffled for each task.

The number of selected attributes over all judgments on all companies (Fig. 33) ranges from 1 to 18, with a clear focus between 3 and 7, an average of 5.3 and a standard deviation of 3.12. This behavior is consistent for all companies: Averages of the selected number of attributes per company range between 5.1 and 6.0. Also in terms of attribute relevance there is large consensus: the same few typical attributes are considered relevant by most subjects for all companies. The histogram presented in Fig. 34 shows companies from the financial sector (histograms for all other companies are all very similar). In fact, low standard deviation values for each attribute on all companies, show that subjects selected the same attributes over and over, regardless of the company. As a consequence, histogram based similarity metrics like the Minkowski distance [71] measured pairwise between all companies, can't

[63] http://wiki.dbpedia.org/Datasets#h18-11

[64] http://www.crowdflower.com/

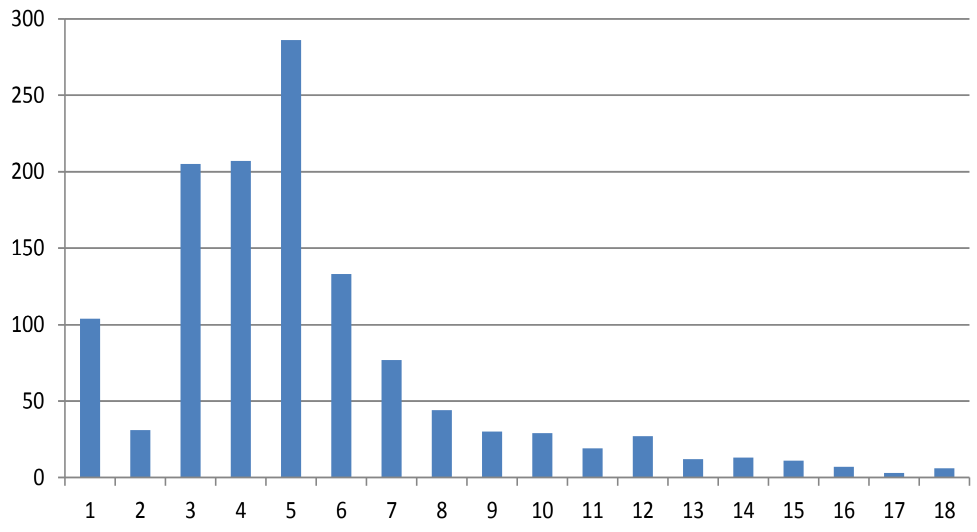


Fig. 33. Number of selected attributes (x-axis) by the number of judgments (y-axis) selecting this number of attributes.

really differentiate between the various business sectors or other semantically meaningful criteria.

Of course, since the Infobox structure has to cover all kinds of companies, the respective attributes were general. Thus, the resulting summaries are generic and provide only marginal information when compared to the rich Wikipedia article structure. But the fact that popular attributes selected from this structure are not correlated to the different topics presented in the articles suggests more sophisticated measures have to be taken when categories are heterogeneous.

Our experiments show a clear tendency regarding the number of attributes an entity summary should feature and a surprisingly high consensus about what attributes are considered important. A good entity summary structure highlights *between 3 and 7 attributes*, and focuses on *typical properties* of the entity. However, seeing the respective articles' richness, considering just generic properties may poorly reflect the real world. If the entity is part of a homogeneous category, properties are usually typical for the entire category. But, if categories are heterogeneous, good structures have to be derived in a data-driven fashion with properties typical for a more homogeneous semantic subgroup.

In summary, the schema-driven approach of Google's Knowledge Graph presents scalability issues; it has issues in choosing the right attributes to include in the entity summary; and it sometimes relies on generic structure that misses out on defining attributes that would reflect the article information richness.

Besides the famous Knowledge Graph, other systems have also been proposed to extract highly informative attributes. In the following section we give an overview of such approaches.

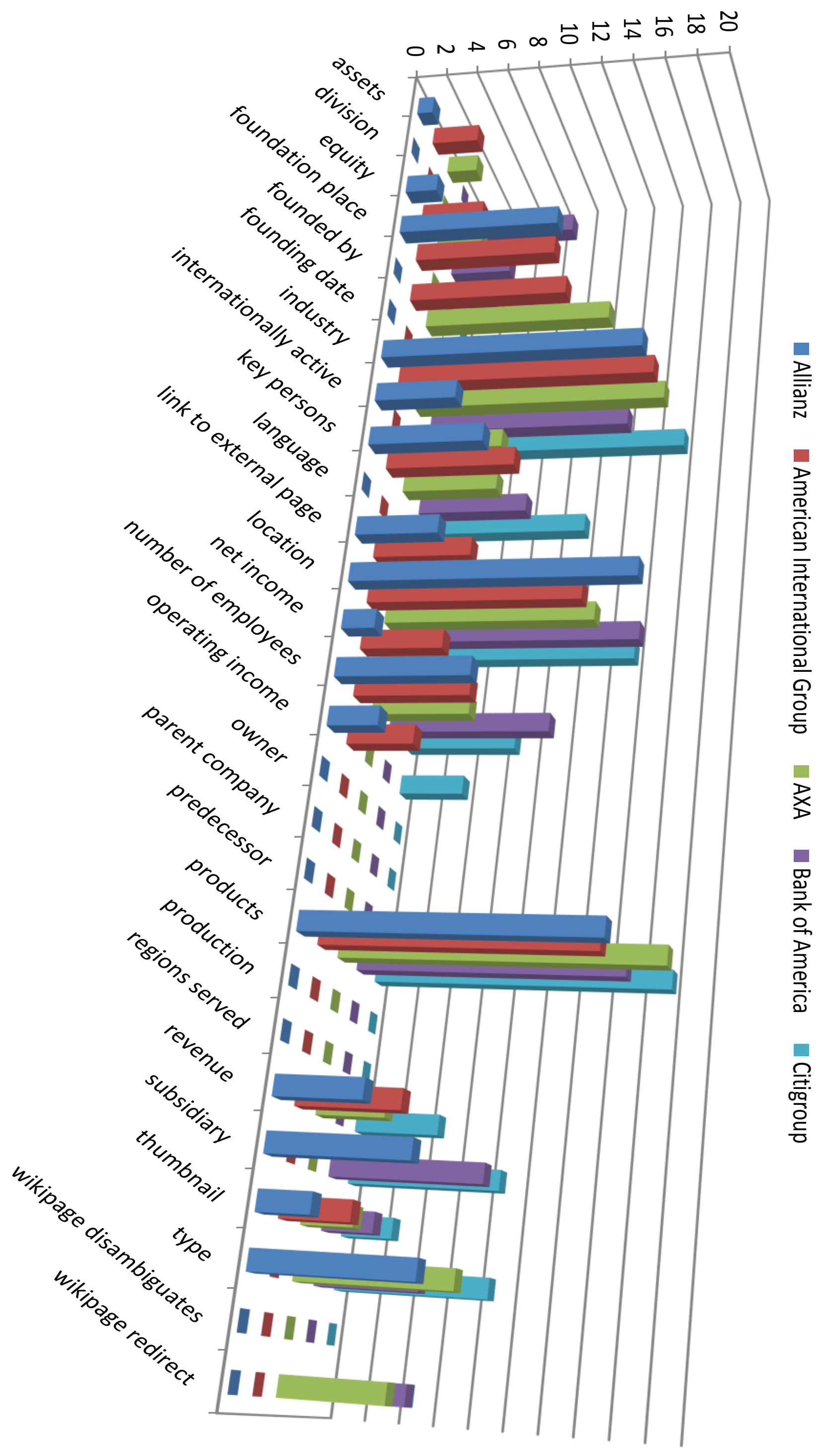


Fig. 34. The number of subjects (y-axis) that have selected an attribute (x-axis) for a certain company (z-axis). For companies from the financial sector only.

6.2. Related Work

Knowledge graphs have been used for entity summarization even before Google's system was proposed. For instance, in [117] the authors present a greedy algorithm that adapts the idea of diversification from information retrieval [1] to extract entity summaries from subject-predicate-object RDF triple stores. The authors argue that a diversity unaware system is likely to present only certain aspects of an entity. For instance in a knowledge base representing movie information, entity Tom_Cruise is connected multiple times to movies by the acted_in predicate but just once to the literal representing his birthday through the born_on_date predicate. In this case, the many movies Tom Cruise played in would be more likely to be included in the summary while other information (like personal data) that might also be interesting to the user would be ignored. To incorporate the concept of diversification into the summarization algorithm they rely on a knowledge graph that comprises edge weights. The weights should represent the "importance" of the edges. These weights are assumed to be provided as input together with the knowledge graph without further clarification. We consider that the concept of attribute typicality introduced in this work is suitable to be used as weights for the graph edges. Furthermore, we separate between predicates and values. Each predicate is considered only once. In consequence, acted_in has same chance of making it into the summary as born_on_date has. The decision which attributes to include in the summary is made entirely based on the attribute typicality, value which is influenced by the user given entity type provided in the query.

Related to our work, in [76] the authors propose a probabilistic approach to compute attribute typicality. But there is a fundamental difference: the authors ignore the difference between entities and entity types. This way for any entity type, say company, both IT company and Toyota are considered to be instances of company. This simplifying assumption doesn't consider data heterogeneity: for entity types comprising heterogeneous entities the extracted attributes only loosely represent the corresponding entities. In contrast, relying on the concept of family resemblance introduced in the previous chapter, our approach distinguishes between heterogeneous and homogeneous groups of entities. It follows a data-driven approach with attributes typical for each sufficiently homogeneous semantic subgroup.

From a broader perspective, our work is related to the field of schema matching and mapping. Such systems use various structural matching techniques as well as data properties to overcome syntactic, structural or semantic heterogeneity. But most approaches focus on data from relational databases or some already existing structure [96]. Systems like WEBTABLES [24] or OCTOPUS [23] rely on semi-structured data like html lists and tables on the Web to extract data structure. In contrast, our system may use all extractions from text without any restrictions increasing the number of supported entities. In [21] the authors propose OMNIVORE. This systems aims to create a comprehensive Web database by combining the output of several OpenIE tools over a Web crawl. For this purpose the authors propose an all-purpose entity-relation database structuring all entities in the form of attribute, value and type. On query time, a relational table comprising query "relevant attributes" is automatically generated. Unfortunately, there is no clear definition to what "relevant" actually means. It seems that the structure is given by the shared attributes. In such a case, a simple query involving a small number of entities poses real difficulties. For instance, querying for Bill Clinton and Barack Obama leads to hundreds of useless attributes. Recognizing the need to disambiguate, for the shared attributes, we calculate their contribution to the structure, based on how similar the entities sharing each attribute really are to the query.

In large corpora true and essential facts are repeated many times. The redundancy of the Web has been exploited by many applications. We leverage it to build a frequentist inference-based method which has repeatedly proven effective for extracting information highly relevant to the query [27, 82]. We consider such an approach is a perfect baseline to compare our system with.

6.3. Attribute Extraction for Entity Summarization-System Description

The task is to extract a common entity structure featuring high quality attributes, which are typical for entities of the same type directly from Web data. The query provided by the user is of the form 'Instance: Entity Type'. Our analysis on the Knowledge Graph shows that there are two main problems to be tackled: providing specific structure even for entities whose corresponding entity types group together heterogeneous entities and supporting even newer entities not having a record in one of the source knowledge bases.

Our claim is that by building on the concept of families of entities (see Section 5.3.1 Definition 9 for more details) a subgroup of homogeneous entities can be selected which in turn allows for extracting a high quality, specific structure, reflecting the essence of the provided entity. Furthermore, following a data-driven approach, of extracting facts directly from Web documents, any entity present on the Web can be supported by such a system.

At the core of this claim lays the concept of attribute typicality, providing the means to extract those attributes being highly typical for the entity in question. This subject will be discussed in more detail in the following subsection.

6.3.1. Attribute Typicality

Following on the *family resemblance* theory introduced by Wittgenstein (presented in Chapter 4, Section 5.3.1), properties that an entity shares with its family are more typical for the entity than properties that are shared with other entities. Also in the context of Web entities, similar entities can be considered to form families. For homogeneous categories, entities are all more or less similar to each other and form a family on the category level. For heterogeneous groups of entities, a family represents just one of the clusters of entities from the group. Applying the Tversky's family resemblance model (Eq. 25) enables the selection of a *most typical family member* or entity. However, our main goal is to find a common structure, i.e. a *most typical set of attributes* for some entity and its respective entity type. Hence, when talking about factual information extracted from the Web we have to restrict the notion of family resemblance based on generic properties (like characteristics, capabilities, etc.) to clear cut attributes as given by extracted predicates. Moreover, we need to find out which of the attributes occurring in a family actually are typical with respect to this family. Since the family definition relies on the measure of members' similarity, we adapt Tversky's measure as follows: Assume we can determine some family *F* consisting of *n* entities $E_1, \dots, E_n$ and a total of *k* distinct attributes given by predicates $p_1, \dots, p_k$ are observed for family *F*. Let X_i and X_j represent the respective attribute sets for two members E_i and E_j, then:

$$|X_i \cap X_j| = 1_{X_i \cap X_j}(p_1) + 1_{X_i \cap X_j}(p_2) + \cdots + 1_{X_i \cap X_j}(p_k) \qquad (26)$$

where $1_X(p) = \begin{cases} 1 \; if \; p \in X \\ 0 \; if \; p \notin X \end{cases}$ is a simple indicator function.

Now we can rewrite Tversky's shared similarity measure to make all attributes explicit:

$$S(X_i, X_j) = \frac{\sum_{l=1}^{k} 1_{X_i \cap X_j}(p_l)}{|X_i \cap X_j| + \alpha |X_i - X_j| + \beta |X_j - X_i|} \qquad (27)$$

Where the same conditions as above apply to α and β.

According to Tversky, each attribute shared by X_i and X_j contributes evenly to the similarity score between X_i and X_j. This allows us to calculate the *contribution score* of each attribute of any member of the family to the similarity of each pair of members:

Let *p* be an attribute of a member from *F*. The *contribution score* of *p* to the similarity of any two attribute sets X_i and X_j, denoted by $C_{X_i, X_j}(p)$, is:

$$C_{X_i, X_j}(p) = \frac{1_{X_i \cap X_j}(p)}{|X_i \cap X_j| + \alpha |X_i - X_j| + \beta |X_j - X_i|} \qquad (28)$$

where $\alpha = \beta \geq 0$.

The contribution of some attribute towards the similarity of two family members in this way is dependent on the degree of similarity between the two members. This is a fundamental difference to simply performing property set intersections (like in Fig. 31), where all family members are assumed to be equally similar to each other. In particular, this enables us to cope even with difficult cases where entity collections are rather heterogeneous.

Building on the contribution score we are now ready to introduce the notion of attribute typicality. Additionally further normalization could be applied to avoid small values.

Definition 10: Attribute Typicality. *Let F be a set of n entities $E_1, \dots, E_n$ of similar kind represented by their respective attribute sets $X_1, \dots, X_n$. Let U be the set of all distinct attributes of all entities from F. The typicality $T_F(p)$ of an attribute/predicate $p \in U$ w.r.t. F is the average contribution of p to the pairwise similarity of all entities in F:*

$$T_F(p) = \frac{1}{C_2^n} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{X_i,X_j}(p) \tag{29}$$

where $C_{X_i,X_j}(p)$ is the contribution score of attribute p regarding the similarity between X_i and X_j (see eq. 28) and C_2^n represents the number of possible combinations of entities from F.

6.3.2. System Architecture

The main task to be handled by the system is the computation of the attribute typicality according to Definition 10. But these values can only be computed after the family of the query entity has been extracted. Obviously, for this reason there is a large overlap between ARES and the system based on family resemblance presented in the previous chapter. Actually, ARES represents an evolutionary development of the instance-based entity search system, sharing most of its architecture (Fig. 35 shows an overview of ARES). Also here there are two main components: the Information Extraction and Query Engine. The Information Extraction did not suffer any changes and works as described in Section 5.3.2. However, for ARES the Query

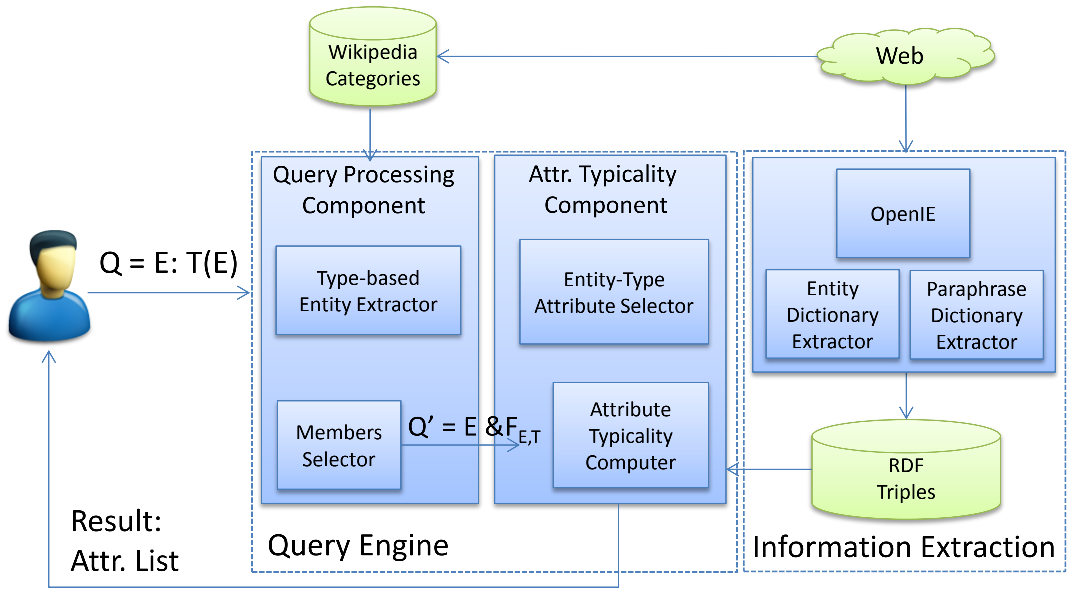

Wikipedia Categories
Web
Query Processing Component
Attr. Typicality Component
OpenIE
Q = E: T(E)
Type-based Entity Extractor
Entity-Type Attribute Selector
Entity Dictionary Extractor
Paraphrase Dictionary Extractor
Members Selector
Q' = E &F_E,T
Attribute Typicality Computer
RDF Triples
Result: Attr. List
Query Engine
Information Extraction

Fig. 35. Attribute extraction for entity summarization based on attribute typicality. System architecture.

Engine has been adapted to extract the typical attributes. The new component works as follows: starting from the provided entity type all known entities belonging to this type are extracted either directly from Web documents or from special dictionaries like Wikipedia. In a filtering step only entities being most similar to the query entity are selected to form the query's *family*. With the query rewritten from "entity plus corresponding entity type" to "entity plus corresponding family", we can now proceed to extract the attributes that are central for the entity types' structure.

Attribute Typicality:
Obviously, only attributes of members of the family should be considered for computing attribute typicality. Once the family has been selected, all attributes occurring together with entities of the family in extracted facts, are taken into account. Following the definition of *attribute typicality*, for each attribute, we calculate its contribution to the family definition of the query entity. Depending on the user needs [27], both Top-k pruning and thresholding can be applied to the typicality scores for selecting the typical attributes.

For better overview of how the quantification of attribute typicality is performed we present the pseudo-code of our system's algorithm in Algorithm 2. Since the attribute extraction has to be performed online, in the following we present an analysis of the systems' efficiency.

Runtime Analysis
For the online part, ARES requires even for broad categories with thousands of entities about 40 seconds per query. To continue with the example of diseases, given as a reference for the entity family selection algorithm, for query "hypertension", ARES needs 42.977 seconds to extract typical attributes on commodity hardware. Additionally to the 22.472 seconds needed for computing the entity family, the computation of typicality values for all 2,711 attributes (lines 2 to 25 in Algorithm 2) takes 20.505 seconds to compute (about 7.5 milliseconds per attribute). Also in this case, all tests have been performed single threaded. Since all major operations allow for parallelization, we have reason to believe that parallelization on a cluster with a few hundred CPU cores will reduce query time to less than a second.

Algorithm 2: Extraction algorithm for typical attributes.

Input: *X* - query entity, *C* - set of entities of type *T*, ϕ - attribute quality threshold, RDF triple collection

Output: *S (structure)* - set of typical attributes

```
1:  $F \leftarrow$ FAMILY($X$, $C$)
2:  $S \leftarrow \{\}$; $U \leftarrow \{\}$
3:  foreach $X$ in $F$ do
4:          $x\_attr \leftarrow$ ATTRIBUTES($X$, RDF)
            // all attributes from triples where $X$ is the subject or the object
            // stored in memory for heavy reuse (*)
5:          $U \leftarrow U \cup x\_attr$
6:  end for
7:  foreach $a$ in $U$ do
8:          $a\_typ \leftarrow 0$
9:          for $X_i \in F$ do
10:                 for $X_j \in F - \{X_k | 1 \leq k \leq i \wedge X_k \in F\}$ do
11:                         $x_i\_attr \leftarrow$ ATTRIBUTES($X_i$, RDF)  // (*)
12:                         $x_j\_attr \leftarrow$ ATTRIBUTES($X_j$, RDF)  // (*)
13:                         $contr \leftarrow 0$
14:                         if $a \in x_i\_attr \wedge a \in x_j\_attr$ then
15:                                 $contr \leftarrow \frac{1}{|x_i\_attr \cap x_j\_attr| + \alpha|x_i\_attr - x_j\_attr| + \beta|x_j\_attr - x_i\_attr|}$
                                    // contribution of $p$ to similarity between $X_i$ and $X_j$ (eq.5)
16:                         end if
17:                         $a\_typ \leftarrow a\_typ + contr$
18:                 end for
19:         end for
20:         $a\_typ \leftarrow 2 \cdot \frac{a\_typ}{|F| \cdot (|F|-1)}$          // the number of pairwise comparisons ($C_2^{|F|}$)
21:         if $a\_typ > \phi$ then
22:                 $S \leftarrow S \cup a$
23:         end if
24: end for
25: return $S$

26: function FAMILY($X$, $C$) // this function represents the functionality of the system in Chapter 4
27:         $F \leftarrow \{X\}$; $x\_attr \leftarrow$ ATTRIBUTES($X$, RDF)
28:         foreach $Y$ in $C$ do
29:                 $y\_attr \leftarrow$ ATTRIBUTES($Y$, RDF)
30:                 $sim \leftarrow$ similarity($x\_attr$, $y\_attr$)        // Tversky's similarity, eq. 25
                    // computed only once then stored in memory for later use (**)
31:                 $S \leftarrow S \cup sim$
32:         end for
33:         $\theta \leftarrow$ THRESHOLD($S$)     // same function as in Algorithm 1
34:         foreach $Y$ in $C$ do
35:                 $y\_attr \leftarrow$ ATTRIBUTES($Y$,RDF)
36:                 $sim \leftarrow$ similarity($x\_attr$, $y\_attr$)  // (**)
37:                 if $sim \geq \theta$ then
38:                         $F \leftarrow F \cup Y$
39:                 end if
40:         end for
41: return $F$
```

6.3.3. Evaluation

The task is to derive a common entity structure comprising high quality attributes. Our claim is that attributes being shared by similar entities are perfect for such a structure. Furthermore, we claim that by using the concept of families of entities introduced in the previous chapter, we can extract representative structure even in cases where the entity type gathers together heterogeneous entities. To validate our claim we assess the quality of the extracted structure with a set of experiments on both homogeneous and heterogeneous entity types. But before proceeding to the experimental subsection, in the following we introduce the basic setup.

Experimental Setup

Dataset: In previous experiments we observed that a large portion of the facts extracted from ClueWeb09 was of poor quality. To assess if there were influences based on the quality of the data, for this evaluation we also experimented on PubMedCentral. This corpus comprises about 250,000 biomedicine and life sciences research papers. Like in the case of ClueWeb09, all documents are processed offline by our IE module. We extracted about 23 million triples with 3 million entities and 1.2 million predicates. This is much more than the 15 million triples extracted from the 500 million documents from ClueWeb09. Besides the fact that research papers are usually larger than Web pages, the high quality of documents from PubMedCentral may be the reason for the large number of facts extracted from this corpus.

Queries. Entity summarization is a follow up of the evaluation on instance-based entity search. In consequence, these experiments focus on the same types of entities. We experiment with persons in the sense of American presidents which have proven to form a homogeneous group of entities and organizations in the sense of companies building a heterogeneous group of entities. Besides named entities, we also test our approach with other simple entities like medical conditions which according to [74] make for an important portion of the Web entity search. In total we experiment over 16 queries split as follows: 5 American presidents 6 companies and 5 well-known medical conditions.

Establishing Ground Truth. Diseases build a homogeneous group according to disease Wikipedia articles. In consequence, the structure for diseases is perfectly suitable to use as a ground truth. But exceptionally, the Wikipedia Infobox for diseases is nothing but a collection of links to the National Library of Medicine - Medical Subject Headings. Since the goal is to obtain data-driven structure, we consider the structure provided by the Wikipedia articles. To be specific, the content headings that were shared by the majority of diseases on Wikipedia were used. The quality of these headings as attributes is confirmed by the fact that they were approved by the standardization committee of schema.org to build the “MedicalCondition” schema (http://schema.org/MedicalCondition). The complete list of attributes is: *Associated anatomy*, *Cause*, *Diagnosis*, *Epidemiology*, *Prognosis*, *Pathophysiology*, *Possible treatment*, *Prevention*, *Risk factors*, *Signs or symptoms*.

For the case of the American presidents the relatively small number of entities (only 44) and the weaker homogeneity (shown in Fig. 27) is not as compelling as in the case of disease. For companies, the Wikipedia Infobox is definitely too general when compared to the article content. Unfortunately schema.org also doesn't provide for a better alternative. In these cases we rely on human assessment to establish ground truth. All attributes extracted (by both the attribute-typicality and the frequency-based method) were mixed together with the Google Knowledge Graph attributes and ordered alphabetically for each query. We presented the resulting lists to subjects and provided the same instructions as in Section 6.1. We selected relevant attributes based on the 'majority rule'. All assessments proved *substantial agreement* ([19]) showing Fleiss' Kappa ([12]) agreement levels between 0.71 and 0.76.

Measures. Our goal is to extract a high quality structure with limited, yet precise attributes for the entity summary. Therefore the success of all algorithms in our experiments is measured in terms of precision (also in aggregated form as mean average precision MAP). Given the small number of attributes included in an entity summary, recall is less important than precision but still relevant for our task.

The frequency-based baseline algorithm. Drawing on the literature, for the baseline we will assume that attributes frequently appearing together with either the query entity or with entities of the same type as the query entity define a good structure for their entity type. Relying on the same infrastructure as ARES we thus implemented the frequency-based baseline approach (in the following called Frequency-based Entity Summarization - short FES). Another reference system is of course Google Knowledge Graph, the attributes from the knowledge snippet to be specific.

Experiments
In Fig. 36.a) we present the top 10 attributes for the example of American presidents. The precision and recall values obtained by the systems are presented in **Table 18**. Both ARES and FES return lists of attributes ranked by their relevance values. In consequence the precision values in the table represent MAP values. For the Knowledge Graph there is no information about the relevance of attributes. All of them are considered to have the same relevance. An averaged precision value is presented in this case. Recall is also presented as the average value over all entities of a category. For the case of American presidents, ARES is with a MAP of 0.75 superior to the other systems. The Knowledge Graph focuses in this case on family and education, elements considered irrelevant by the assessors. This severely affects its recall. These precision and recall values were computed based on attributes presented by the Knowledge Graph. For ARES and FES the top10 attributes were considered.

While entity types are perfect for disambiguation, some may prove heterogeneous. Fortunately, our approach features a self-tuning resemblance measure able to automatically refine categories: all queries are focused on a *family* of entities with sufficiently homogeneous structure, while keeping the focus on the query entity. In

Table 18: Precision & recall by system and query category.

	Precision			Recall		
	ARES	FES	Knowledge Graph	ARES	FES	Knowledge Graph
American Presidents	0.75	0.33	0.37	0.5	0.3	0.2
Companies	0.73	0.44	0.49	0.6	0.3	0.25
Diseases	0.87	0.37	0.52	0.7	0.4	0.38

the previous chapter we have seen that out of the S&P 500 list of companies, for queries like “Toyota Motor Corporation”, “Renault S.A.”, or “Volkswagen A.G.”, our systems builds families with 17 to 24 entities, clearly focusing on car companies (30% - 50% of the selected family members are car makers). For queries like “Apple Inc”, “Google Inc” or “Microsoft” the results are similar, with 40% - 60% of IT companies in the selected family.

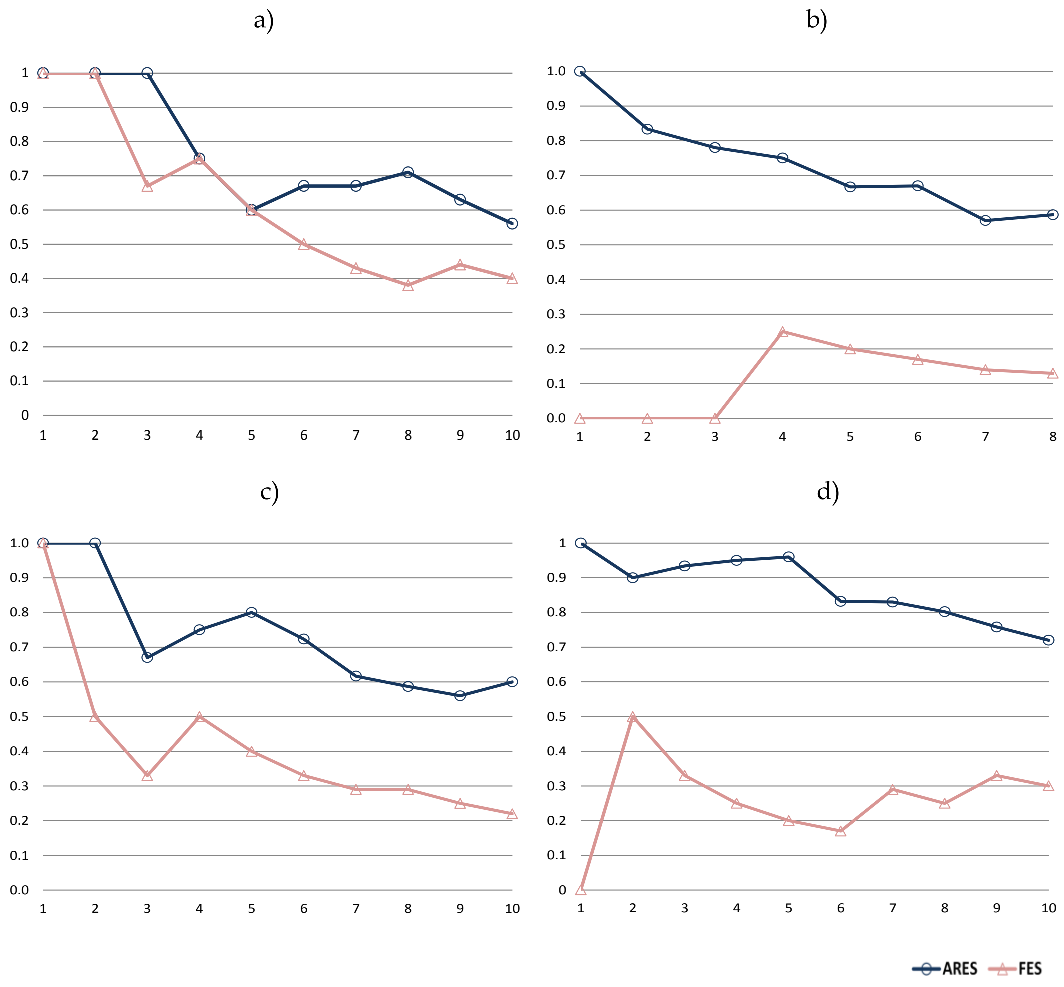


Fig. 36. Precision@k averaged over all query entities for American presidents (a), automotive companies (b), IT companies (c) and diseases (d).

Indeed the self-tuning works well also for structure extraction. In contrast to the results observed for generic attributes from Wikipedia Infoboxes (Section 6.1), where Minkowski similarity metrics (Manhattan distance) showed no particular difference between companies in different fields, for the attributes extracted by our system these differences are considerably more expressive. The average histogram distance for the attribute selection generated by the crowd starting from the company Wikipedia Infobox between the car companies is of 69.8. The same distance for the attributes selected by ARES for car companies is of 6.7. For IT companies the respective average values are 56.4 for the selection from the Wikipedia Infobox vs. 1.4 for the ARES selection. However, the average distance between different sectors stays large also for ARES: 78.16 vs. 65.3 for the selection from the Wikipedia Infoboxes and ARES respectively, for car vs. IT companies. This clearly shows that ARES is able to extract attributes particular to homogeneous entities (small histogram distances for similar companies) while keeping heterogeneous entities apart. It's interesting to notice that when presented with data-driven attributes, assessors were able to pick attributes that differentiate between business sectors.

In terms of precision, our approach achieves 0.73 MAP for all company queries, superior to both baselines. We present the results, averaged by sector, in Fig. 36.b) and Fig. 36.c). For car makers, 8 attributes proved relevant according to the majority of human assessors. There is quite some difference in the precision that the baseline is able to achieve in the two sectors. Deeper inspection lead to the conclusion that there seems to be more information about IT companies than about car makers, probably because ClueWeb09 contains many blogs and forum posts. In fact most information on automotive topics actually refers to the cars and not to the respective companies. Thus, it is understandable that a frequency based method shows such poor results.

Motivated by these findings, we repeated the above experiments on PubMedCentral with diseases as query entities. Unlike in the case of companies, here we can't recognize any particular (especially taxonomically motivated) patterns regarding the members in our automatically derived families. Cardiovascular diseases are mixed together with infectious diseases, skin conditions and forms of cancer, regardless of the query entity. We evaluated both systems in terms of precision and recall, over five well-known diseases and medical conditions ("cancer", "diabetes mellitus", "hepatitis", "hypertension" and "tuberculosis"). The results, averaged over the five queries, are presented in Fig. 36.d). Boosted by the high quality information, our method achieves an impressive MAP of 0.87. The Knowledge Graph returns some of the National Library of Medicine headings for each disease obtaining fair average precision of 0.52 over all entities. We would have expected that the frequency baseline also performs better in face of the large amount of relevant and indeed very structured information. It seems that the broad coverage of various subjects with respect to medicine and diseases leads to the frequency of attributes being spread rather evenly without notable differences in frequency. Also in terms of recall (**Table 18**) our method is consistently superior, showing its overall practical usefulness.

Taking these results into account, we consider that attribute typicality based on the principle of family resemblance indeed is a highly promising solution for automatically discovering high quality attributes for summarizing entities from the Web

6.4. Conclusions

In this chapter we presented an in-depth view of *entity summarization*. Currently, Google's Knowledge Graph represents the state of the art for this task. Relying on curated knowledge bases, this particularly excludes all new and less widely known entities. The alternative is not to rely on prearranged schemas, but to use a data-driven approach. As our experiments show, even simple, frequency-based approaches already reach similar and often even better performance than the Knowledge Graph in terms of quality. But for good user experience even more is needed: especially in the case of heterogeneous collections of entities an intelligent selection for extracting compact, yet high quality entity summaries is needed. Therefore, tuning the data-driven schema selection for entities over the vast variety of facts that can be extracted from the Web is a major contribution of the ARES approach presented in this chapter.

Since lessons learned from instance-based entity search show that by itself, an entity is not enough for proper disambiguation, as an input ARES requires a query comprising an entity and its type. Also regarding the produced entity summary, our experiments show a clear tendency with respect to the number of attributes an entity summary should feature: a good entity summary structure highlights between 3 and 7 attributes, and focuses on typical properties of the entity. If entities grouped together by the provided entity type are homogeneous, properties are usually typical for the entire group of entities. If they are heterogeneous, good structures have to be derived in a data-driven fashion with properties typical for a more homogeneous semantic subgroup.

ARES relies on the concept of family resemblance introduced by cognitive psychology and intelligently blends the homogeneity/heterogeneity of entity families with schema integration techniques in the light of all extracted facts. ARES is self-tuning in the sense that after family selection, entities within families show high intra-family similarity, while entities from heterogeneous categories show low inter-family similarity. Given the current advances in OpenIE, that allow to work directly on text, any entity being described somewhere on the Web, can thus be summarized appropriately. Our experiments on real-world entity classes representing different degrees of class homogeneity show that ARES is indeed superior to both, frequency-based statistical approaches and the Knowledge Graph, in terms of precision and recall. Moreover, also the run-time performance is already quite practical.

Chapter 7

Conclusions and Future Work

The Web has evolved to an all-purpose source of information on any topic. Want to know something about a medical condition, or a drug? Ask Doctor Google! With over 50% of all Web search queries, most of Web queries focus on entity-centric search. But while entity related information is abundant on the Web, search engines are not yet prepared to support such queries. Actually, with the exception of Google's Knowledge Graph, search engines did not evolve much with respect to entity search. Keyword-based entity search is possible but it falls on the users' shoulders to browse through the returned Web pages, to pick out relevant information or to formulate new queries that will hopefully lead to a satisfying answer.

The Knowledge Graph is a first step towards providing for better user experience for entity search. It performs entity summarization based on data from manually curated knowledge bases. But this way, the majority of entities (in particular, new or more obscure entities) not present in the manually curated Web resources used by Google's Knowledge Graph, cannot benefit from data summarization. Automatically integrating information extracted from unstructured data into a Web-scale knowledge base, the Knowledge Vault seems to be a promising solution in this respect. However, besides entity summarization, there are three other entity related queries, primarily focused on retrieving entities based on user input: searching for entities that have certain properties, searching for entities that are similar to a given entity, or searching for entities of a given type. Since the Knowledge Vault has just been proposed, no detailed information about its use for entity search is provided, and no running prototype is available, we can't really tell how well it will accommodate such queries. But for the time being, considering both, the weak overall support that search engines provide for entity related search, and the high demand for such queries, we believe that developing capabilities for performing entity-centric search on the Web represents a strategic advantage for any mainstream search engine.

The retrieval of entity-related data has always been among the core applications of database systems. In this context, entity-centric search is trivial because the underlying data is provided in structured form. Thinking along the same lines, the problems of entity-centric search would be solved if all data on the Web were available in structured form, just like in a big database. Linked data is the first major initiative for building a structured Web. The initiative encourages data providers to publish their data online. To make their life easy, the effort of integrating the published data into the existing data cloud is kept to a minimum: data providers have the flexibility of choosing (or keeping) their own data structure. Interlinking the published data as well as vocabulary reuse is desired and recommended, however, besides a few prominent examples like DBpedia or Freebase, most data sets available online are rather isolated. With today's LOD cloud interlinking and vocabulary heterogeneity, providing a holistic view of an entity, spanning over multiple data stores is impossible. Automatic instance matching and ontology alignment has often been proposed as a solution to integrate the various data sets together. But as we have shown in this thesis, the quality of the results obtained by such systems is far from being adequate.

Pushed by major search engine providers, schema.org followed a different approach: it offered a set of global schemata and encouraged Web page owners to annotate their content with these schemata using rich snippets and better ranking results as incentives. The schemata are well designed and have been developed with the help of experts in the corresponding fields. This approach seems promising as it solves the problems of schema mapping and matching which have proven problematic in the case of LOD. But since entities are not uniquely identifiable there is still a problem of entity reconciliation. However, the major problem of schema.org is its low acceptance on the Web.

Both LOD and schema.org represent cornerstones in the evolution of the Web. But given the problems encountered when trying to use them for entity-centric search, we believe that this "one size fits all" of structuring the Web requires a long evolution process until it can reliably serve the purpose of entity search. Instead, in this thesis we have shown that data-driven approaches, tailored for the different types of entity-centric queries already achieve quite practical results.

Borrowing from the field of cognitive psychology, the semiotic triangle now established also in information theory, models entity types in terms of intension and extension. Based on this, we identified three types of queries focused on retrieving entities and one for entity summarization:

Entity type query. For the moment, the contribution of schema.org for entity type based search is neglectable. Search on structured data on the Web is in our opinion limited to one data store search only and to *superordinate* and *basic* entity types. Because of the large number of possible *subordinate* entity types most of them are usually not considered by the data structure. This has sever effects on the recall and on the number of supported queries. Boosted by manual (crowd-sourced) or semi-manual effort to align types and interlink entity instances, linked data will, some day, play a major role in accessing entities from the Web. This can have a positive impact on the recall. But *subordinate* entity types, will not be properly supported by static vocabularies. Instead, we proposed a system that is able to dynamically mine new, unknown types out of Web data. Combining query expansion with a self-supervised vocabulary learning technique built on both structured and unstructured data, our approach is able to achieve a good tradeoff between precision and recall.

An interesting approach we leave to future work would be to combine the strength of linked data with the flexibility of query expansion. Such a hybrid system would benefit from the high precision that isolated data sources can deliver, while entity retrieval on query expansion on the Web could cater for better recall values and support for ad-hoc types not known to the LOD vocabularies. But in order to compile a list of resulting entities, duplicate detection is required, a problem that has yet to be mastered.

Prototype-based query. Motivated by positive examples in the field of programming, were *duck typing* has already been successfully applied, we proposed ProSWIP, a property-based system for retrieving entities from the Web. ProSWIP builds on user feedback in order to ensure that it captures correctly the user's intentions. To be specific, it asks the user if some property is, or is not relevant regarding the intended entity type, extending this way the property set for the entity type definition. However, entities have hundreds of properties. The challenge is to understand the user intentions with just a handful of questions. ProSWIP cleverly solves this problem with the help of information theory concepts, asking for user feedback on just a few properties showing the highest information gain. Our experiments show that within a maximum of four iterations the system achieves perfect quality.

For the time being, all entities not showing a certain property from the entity type definition are not included in the result set. Corroborated with the sparse nature of data extracted from the Web, this severely affects recall. This problem can be tackled in future work by using properties that have been found suitable to extend the concept definition, not as filters, but as features for entity ranking on structural similarity. Corroborated with the concept of attribute typicality introduced in this thesis, this relaxation should be applied only to properties not being typical for the intended entity type. This should increase the robustness against missing values and have a positive effect on recall.

Instance-based query. The most basic and user friendly form of instance-based query comprises one single entity. But our experiments have shown that with such a query it is hardly possible to capture the user intentions. Even for queries comprising up to five example entities the disambiguation was not satisfactory. Following on the example of standard example driven entity search tasks like REF and ELC we have proposed a minimalistic instance-based query comprising the example entity and intended entity type that is both user friendly and allows for satisfactory disambiguation. But the example entity is only valuable if it captures at least to some extent the user intentions. As a constraint, outliers have to be avoided. Assuming that the entity example is well-chosen, we build on the concept of family resemblance and provide a practical way for computing families of entities. These entities are of the type provided by the user, and together with the example entity form a homogeneous group. Such an approach achieves impressive results, being able to retrieve semantically meaningful entities even for entity types, which have proven problematic for REF and ELC.

For the time being, our instance-based query system building on family resemblance focuses on high precision. This comes at the cost of recall. Especially for entity types grouping together entities with a high degree of homogeneity, the ISODATA based thresholding method may be too selective. In this case more refined methods involving user interaction may be more suitable for nailing the user intentions.

Entity summarization query. Google's Knowledge Graph which is the state of the art for this task, relies on curated knowledge bases, and this particularly excludes all new and less widely known entities. Our suggestion is not to rely on prearranged schemas, but to use a data-driven approach. As our experiments show, even simple, frequency-based approaches already reach similar and often even better performance than the Knowledge Graph in terms of quality. But especially in the case of heterogeneous collections of entities an intelligent selection for extracting compact, yet high quality entity summaries is needed. Therefore, tuning the data-driven schema selection for entities over the vast variety of facts that can be extracted from the Web is a major contribution of our approach. ARES intelligently blends the homogeneity/heterogeneity of entity families with schema integration techniques in the light of all extracted facts. ARES is self-tuning in the sense that after family selection, entities within families show high intra-family similarity, while entities from heterogeneous categories show low inter-family similarity. Given the current advances in OpenIE, that allow to work directly on text, any entity being described somewhere on the Web, can thus be summarized appropriately. Our experiments on real-world entity classes representing different degrees of class homogeneity show that ARES is indeed superior to both, frequency-based statistical approaches and the Knowledge Graph, in terms of precision and recall.

For the moment, ARES extracts structure only. But following on the example of systems like SCAD [4], for the future, values for the selected attributes can also be extracted.

Our claim in this thesis is that by supporting these four query types, a system enables holistic entity search. Representing the main building blocks of such a system, the components that implement the functionality to support these query types, have been discussed and evaluated individually. They rely entirely on Web data, follow a data-driven approach, tailored for each query type and they obtain promising results better than the corresponding baselines. But considering that some components may require user feedback while others don't, that three components retrieve entities while one retrieves entity summaries, there are still a few open questions regarding the user interface. This is especially important considering the integrated result presentation. Entity snippets may for instance prove useful in this respect. However, broad user studies focusing on assessing the user friendliness and clarity of the user interface as well as the quality of the overall results are needed. We leave these aspects as a subject for future work.

Appendix A

Appendix: Analysis on rdf:type URIs in BTC

Table 19: Top 30 most comprising URIs for the 'book' entity type from various data stores in the BTC corpus.

URI	Nr.	%
<http://rdf.freebase.com/ns/book.book>	31,570	36.64
<http://rdf.freebase.com/ns/book.written_work>	25,896	30.05
<http://dbpedia.org/ontology/WrittenWork>	23,571	27.36
<http://schema.org/Book>	15,398	17.87
<http://dbpedia.org/ontology/Book>	15,397	17.87
<http://rdf.freebase.com/ns/fictional_universe.work_of_fiction>	10,304	11.96
<http://umbel.org/umbel/rc/Book_CW>	5,934	6.89
<http://rdf.freebase.com/ns/book.magazine>	3,160	3.67
<http://dbpedia.org/ontology/Magazine>	2,862	3.32
<http://rdf.freebase.com/ns/book.journal>	2,409	2.80
<http://rdf.freebase.com/ns/book.published_work>	2,357	2.74
<http://rdf.freebase.com/ns/book.newspaper>	2,342	2.72
<http://dbpedia.org/class/Book>	2,301	2.67
<http://data.kasabi.com/dataset/bricklink/schema/Book>	1,650	1.91
<http://swrc.ontoware.org/ontology#Book>	1,348	1.56
<http://umbel.org/umbel/rc/Magazine>	1,301	1.51
<http://rdf.freebase.com/ns/book.short_story>	1,259	1.46
<http://dbpedia.org/class/yago/AmericanNovels>	879	1.02
<http://dbpedia.org/class/yago/FilmsBasedOnNovels>	751	0.87
<http://dbpedia.org/class/yago/ScienceFictionNovels>	740	0.86
<http://rdf.freebase.com/ns/book.poem>	664	0.77
<http://dbpedia.org/ontology/Play>	474	0.55
<http://dbpedia.org/class/yago/MonthlyMagazines>	472	0.55
<http://dbpedia.org/class/yago/BritishNovels>	470	0.55
<http://dbpedia.org/class/yago/ShortStory106371999>	460	0.53
<http://dbpedia.org/class/yago/FantasyNovels>	434	0.50

<http://dbpedia.org/class/yago/AmericanFantasyNovels>	384	0.45
<http://dbpedia.org/class/yago/Children%27sNovels>	381	0.44
<http://dbpedia.org/class/yago/AmericanScienceFictionNovels>	374	0.43
<http://dbpedia.org/class/yago/HistoricalNovels>	365	0.42

Table 20: Top 30 most comprising URIs for the 'music album' entity type from various data stores in the BTC corpus.

URI	Nr.	%
<http://rdf.freebase.com/ns/music.album>	57,786	50.79
<http://dbpedia.org/ontology/MusicalWork>	56,725	49.86
<http://dbpedia.org/ontology/Album>	54,395	47.81
<http://schema.org/MusicAlbum>	54,395	47.81
<http://umbel.org/umbel/rc/Album_IBO>	5,334	4.69
<http://dbpedia.org/class/yago/DebutAlbums>	3,052	2.68
<http://dbpedia.org/class/yago/Album106591815>	2,892	2.54
<http://dbpedia.org/class/yago/2006Albums>	1,732	1.52
<http://dbpedia.org/class/yago/2005Albums>	1,646	1.45
<http://dbpedia.org/class/yago/2007Albums>	1,640	1.44
<http://dbpedia.org/class/yago/2004Albums>	1,492	1.31
<http://dbpedia.org/class/yago/2008Albums>	1,324	1.16
<http://dbpedia.org/class/yago/2003Albums>	1,297	1.14
<http://dbpedia.org/class/yago/2009Albums>	1,236	1.09
<http://dbpedia.org/class/yago/2002Albums>	1,177	1.03
<http://dbpedia.org/class/yago/2001Albums>	1,083	0.95
<http://dbpedia.org/class/yago/2000Albums>	1,039	0.91
<http://dbpedia.org/class/yago/GreatestHitsAlbums>	1,014	0.89
<http://dbpedia.org/class/yago/1999Albums>	967	0.85
<http://dbpedia.org/class/yago/1998Albums>	860	0.76
<http://dbpedia.org/class/yago/1997Albums>	800	0.70
<http://dbpedia.org/class/yago/1996Albums>	794	0.70
<http://dbpedia.org/class/yago/LiveVideoAlbums>	744	0.65
<http://dbpedia.org/class/yago/1995Albums>	666	0.59
<http://dbpedia.org/class/yago/1994Albums>	660	0.58

<http://dbpedia.org/class/yago/DouBleCompilationAlbums>	653	0.57
<http://dbpedia.org/class/yago/1993Albums>	616	0.54
<http://dbpedia.org/class/yago/English-languAgeAlbums>	570	0.50
<http://dbpedia.org/class/yago/ColumbiaRecordsAlbums>	556	0.49
<http://dbpedia.org/class/yago/DouBleLiveAlbums>	532	0.47

Appendix B

Curriculum Vitae

Born on 01.09.1982 in Bacau, Romania

Since 11/2008 — **Research Associate** at the Institute for Information Systems, Technische Universität Braunschweig

- Research in the area of Entity-centric Search, Web-Scale Information Extraction, Data Mining (supervised und unsupervised Machine Learning), and Data Integration
- Support for the preparation of lectures and seminars on the topics of Data Warehousing, Data Mining, und Multimedia Databases
- Coordination of students theses

10/2005 - 10/2008 — **Master Program in Computer Science,** Technische Universität Braunschweig

- Certificate: Master of Science (Grade 1,1)
- Master Thesis: „Adaptive deadline based real-time scheduling for wireless video streaming“
- Specializations: Software Engineering, Computer Networks, Embedded Systems

09/2001 - 06/2005 — **Computer Science,** University of Iasi (Romania)

- Certificate: University Diploma (Grade 10/10)
- Diploma Thesis: „Cluster Computing Raytracer”
- Specialization: Computer Networks

List of Figures

List of Tables

Bibliography

[1] Agrawal, R., Gollapudi, S., Halverson, A. and Ieong, S. 2009. Diversifying search results. *in Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM)* (2009), 5.

[2] Agrawal, S., Chakrabarti, K., Chaudhuri, S., Ganti, V., Konig, A.C. and Xin, D. 2009. Exploiting web search engines to search structured databases. *in Proceedings of the 18th International World Wide Web Conference (WWW)*. (2009), 501.

[3] Alfio Ferrara, D.L. 2008. Towards a Benchmark for Instance Matching. *in Proceedings of the International Workshop on Ontology Matching (OM)* (2008).

[4] Bakalov, A. and Fuxman, A. 2011. SCAD: Collective Discovery of Attribute Values Categories and Subject Descriptors. *in Procedings of the 20th International World Wide Web Conference (WWW)* (Hyderabad, India, 2011), 447–456.

[5] Ball, G. and Hall, D. 1965. *ISODATA: A novel method of data analysis and pattern classification*.

[6] Balog, K. 2011. Finding Related Entities. *in Proceedings of the International Workshop on Search and Mining Entity-Relationship Data (SMER)* (2011).

[7] Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M. and Etzioni, O. 2007. Open Information Extraction from the Web. *in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (Hyderabad, India, 2007).

[8] Barzilay, R. and Lee, L. 2003. Learning to Paraphrase : An Unsupervised Approach Using Multiple-Sequence Alignment. *in Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)* (Edmonton, Canada, 2003), 16–23.

[9] Berners-Lee, T. 2006. Linked Data. Design issues for the World Wide Web Consortium.

[10] Berners-Lee, T. 1999. *Weaving the Web*. Harper Collins.

[11] Berners-Lee, T., Fielding, R. and Masinter, L. 1998. Uniform Resource Identifiers (URI): Generic Syntax. (Aug. 1998).

[12] Bischoff, K., Firan, C.S., Nejdl, W. and Paiu, R. Can All Tags be Used for Search? Categories and Subject Descriptors. *in Proceedings of the 17th ACM Conference on Information and knowledge Management (CIKM)* 203–212.

[13] Bizer, C. and Berners-Lee, T. 2009. Linked Data - The Story So Far. *in Proceedings of the International Journal on Semantic Web and Information Systems (IJSWIS)*. (2009).

[14] Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. 2007. DBpedia : A Nucleus for a Web of Open Data. *in Proceedings of the 6th International Semantic Web Conference (ISWC)* (Busan, Korea, 2007), 722–735.

[15] Böhm, C., de Melo, G., Naumann, F. and Weikum, G. 2012. LINDA: Distributed web-of-data-scale entity matching. *in Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2012), 2104–2108.

[16] Bollacker, K., Evans, C., Paritosh, P., Sturge, T. and Taylor, J. 2008. Freebase. *in Proceedings of the ACM International Conference on Management of Data (SIGMOD)* (New York, New York, USA, Jun. 2008), 1247.

[17] Bonnefoy, L. and Bellot, P. 2011. LIA-iSmart at the TREC 2011 Entity Track: Entity List Completion Using Contextual Unsupervised Scores for Candidate Entities Ranking. *in Proceedings of the Text Retrieval Conference (TREC)* (2011).

[18] Bracha, G. and Lindstrom, G. 1992. Modularity meets inheritance. *in Proceedings of the International Conference on Computer Languages* (1992), 282–290.

[19] Braschler, M. and Peters, C. 2004. CLEF 2003 Methodology and Metrics. *Comparative Evaluation of Multilingual Information Access Systems Lecture Notes in Computer Science*. 3237, (2004), 7–20.

[20] Bron, M., Balog, K. and Rijke, M. 2013. Example Based Entity Search in the Web of Data. *in Proceedings of the European Conference on Informaiton Retrieval (ECIR)* (2013).

[21] Cafarella, M.J. 2009. Extracting and Querying a Comprehensive Web Database. *in Proceedings of the 4th Confonference on Innovative Data Systems Research (CIDR)* (Monterey, USA, 2009).

[22] Cafarella, M.J. and Etzioni, O. 2007. Navigating Extracted Data with Schema Discovery. *in Proceedings of the 10th International Workshop on Web and Databases (WebDB)* (Beijing, China, 2007).

[23] Cafarella, M.J., Halevy, A. and Khoussainova, N. 2009. Data Integration for the Relational Web. *in Proceedings of the Very Large Database Endowment (PVLDB)* (Lyon, France, 2009).

[24] Cafarella, M.J., Halevy, A., Wang, D.Z. and Wu, E. 2008. WebTables: Exploring the Power of Tables on the Web. *in Proceedings of the Very Large Database Endowment (PVLDB)* (Auckland, New Zealand, 2008), 538–549.

[25] Campinas, S., Ceccarelli, D., Perry, T.E., Delbru, R., Balog, K. and Tummarello, G. The Sindice-2011 dataset for entity-oriented search in the web of data. *in Proceedings of the First International Workshop on Entity-Oriented Search (EOS)*. 26–32.

[26] Cheng, T. and Chang, K.C. 2010. Beyond pages: supporting efficient, scalable entity search with dual-inversion index. *in Proceedings of the International Conference on Extending Database Technology (EDBT)* (2010), 15–26.

[27] Cheng, T., Yan, X. and Chang, K.C. 2007. EntityRank: Searching Entities Directly and Holistically. *in Proceedings of the 33rd International Conference on Very Large Databases. (VLDB)* (2007), 387–398.

[28] Cho, W.C. and Richards, D. 2007. Ontology construction and concept reuse with formal concept analysis for improved web document retrieval. *in Proceedings of the International Journal of Web Intelligence and Agent Systems (WIAS)*. 5, 1 (Jan. 2007), 109–126.

[29] Cimiano, P., Schultz, A., Sizov, S., Sorg, P. and Staab, S. 2009. Explicit versus latent concept models for cross-language information retrieval. *in Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. (Jul. 2009), 1513–1518.

[30] Clare, A. and King, R.D. 2001. Knowledge Discovery in Multi-label Phenotype Data. *in Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)* (2001), 42–53.

[31] Cucerzan, S. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *in Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (Prague, Czech Republic, 2007), 708–716.

[32] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*. (1990).

[33] Ding, L., Shinavier, J., Shangguan, Z. and McGuinness, Deborah, L. 2010. SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in Linked Data. *in Proceedings of the Int ernational Semantic Web Conference (ISWC)* (2010).

[34] Dong Wang, Qing Wu, Haiguang Chen, J.N. 2010. A Multiple-Stage Framework for Related Entity Finding: FDWIM at TREC 2010 Entity Track. *in Proceedings of Text REtrieval Conference (TREC)* (2010).

[35] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S. and Zhang, W. 2014. Knowledge vault. *in Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)* (New York, New York, USA, Aug. 2014), 601–610.

[36] Dragan, L., Delbru, R., Groza, T., Siegfried, H. and Decker, S. 2011. Linking Semantic Desktop Data to the Web of Data. *in Proceedings of the International Semantic Web Conference (ISWC)* (2011).

[37] Fader, A., Soderland, S. and Etzioni, O. 2011. Identifying Relations for Open Information Extraction. *in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Edinburgh, Scotland, UK, 2011), 1535–1545.

[38] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. 1999. Hypertext Transfer Protocol -- HTTP/1.1. (May 1999).

[39] Finkel, J.R., Grenager, T. and Manning, C. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *in Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)* (2005).

[40] Freund, Y. and Schapire, R.E. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *in Proceedings of the Journal of Computer and System Sciences*. 55, 1 (1997), 119–139.

[41] Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A. and Ciancarini, P. 2012. Automatic Typing of DBpedia Entities. *in Proceedings of the Internationl Semantic Web Conference (ISWC)* (2012), 65–81.

[42] Ganter, B. and Wille, R. 1997. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc.

[43] Ganti, V. 2010. Keyword++: A Framework to Improve Keyword Search Over Entity Databases. *in Proceedings of the Very Large Database Endowment (PVLDB)*. (2010), 711–722.

[44] Gerard Salton, C.B. 1988. Term-weighting approaches in automatic text retrieval. *in Proceedings of the Journal of Information Processing and Management (IJIPM)*. (1988).

[45] Ghias, A., Logan, J., Chamberlin, D. and Smith, B.C. 1995. Query by humming. *in Proceedings of the third ACM International Conference on Multimedia (MULTIMEDIA)* (New York, New York, USA, Jan. 1995), 231–236.

[46] Gil, J.Y. 2008. Whiteoak: Introducing Structural Typing into Java. *in Proceedings of the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications (OOPSLA)* (2008), 73–89.

[47] Gottron, T., Knauf, M., Scheglmann, S. and Scherp, A. 2013. A Systematic Investigation of Explicit and Implicit Schema Information on the Linked Open Data Cloud. *in Proceedings of the 10th Extended Semantic Web Conference (ESWC)* (2013).

[48] Halpin, H. and Hayes, P.J. 2010. When owl:sameas isn't the same: An analysis of identity links on the semantic web. *in Proceedings of the Linked Data on the Web Conference (LDOW)* (2010).

[49] Hancock, T., Jiang, T., Li, M. and Tromp, J. 1996. Lower Bounds on Learning Decision Lists and Trees. *in Journal for Information and Computation*. 126, 2 (1996), 114–122.

[50] Harth, A. 2012. Billion Triples Challenge data set.

[51] Hasegawa, T., Sekine, S. and Grishman, R. 2004. Discovering Relations among Named Entities from Large Corpora. *in Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics (ACL)* (Barcelona, Spain, 2004).

[52] Hindle, D. 1990. Noun classification from predicate-argument structures. *in Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL)* (1990).

[53] Homoceanu, S. and Balke, W. 2011. What Makes a Phone a Business Phone. *in Proceedings of the International Conference on Web Intelligence (WI)* (Lyon, France, 2011).

[54] Homoceanu, S. and Balke, W.-T. 2014. Querying concepts in product data by means of query expansion. *Web Intelligence and Agent Systems*. 12, 1 (Jan. 2014), 1–14.

[55] Homoceanu, S., Dechand, S. and Balke, W.-T. 2011. Review Driven Customer Segmentation for Improved E-Shopping Experience. *in Proceedings of the 3rd International Conference on Web Science (WebSci)* (Koblenz, Germany, 2011).

[56] Homoceanu, S., Geilert, F., Pek, C. and Balke, W.-T. 2014. Any Suggestions? Active Schema Support for Structuring Web Information. *in Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)* (Denpasar, Bali, 2014).

[57] Homoceanu, S., Kalo, J.-C. and Balke, W.-T. 2014. Putting Instance Matching to the Test: Is Instance Matching Ready for Reliable Data Linking? *in Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS)* (2014).

[58] Homoceanu, S., Loster, M., Lofi, C. and Balke, W.-T. 2011. Will I Like It? Providing Product Overviews Based on Opinion Excerpts. *in Proceedings of the IEEE 13th Conference on Commerce and Enterprise Computing (CEC)* (2011), 26–33.

[59] Homoceanu, S., Wille, P. and Balke, W. 2013. ProSWIP : Property-based Data Access for Semantic Web Interactive Programming. *in Proceedings of the International Semantic Web Conference (ISWC)* (Sydney, Australia, 2013).

[60] Hu, M. and Liu, B. 2004. Mining and summarizing customer reviews. *in Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)* (New York, New York, USA, Aug. 2004), 168.

[61] Isele, R., Jentzsch, A. and Bizer, C. 2010. Silk Server - Adding missing Links while consuming Linked Data. *in Proceeding of the International Workshop on Consuming Linked Data (COLD)* (2010).

[62] Jain, P., Hitzler, P., Sheth, A.P., Verma, K. and Yeh, P.Z. 2010. Ontology Alignment for Linked Open Data. *Information Retrieval*. 6496, November (2010), 402–417.

[63] Jain, P., Yeh, P.Z., Verma, K., Vasquez, R.G., Damova, M., Hitzler, P. and Sheth, A.P. 2011. Contextual ontology alignment of LOD with an upper ontology: A case study with proton. *in Proceedings of the 8th Extended Semantic Web Conference on The Semantic Web (ESWC)* (2011), 80–92.

[64] Jiménez-Ruiz, E., Cuenca Grau, B., Jim, E. and Grau, B.C. 2011. LogMap: Logic-Based and Scalable Ontology Matching. *in Proceedings of the International Semantic Web Conference (ISWC)* (2011), 273–288.

[65] Jolliffe, I. 2002. *Principal Component Analysis, (2nd Ed.)*. Springer Series in Statistics.

[66] Kato, T., Kurita, T., Otsu, N. and Hirata, K. 1992. A sketch retrieval method for full color image database-query by visual example. *in Proceedings of the 11th International Conference on Pattern Recognition (ICPR)* (1992), 530–533.

[67] Khalili, A. and Auer, S. 2013. WYSIWYM – Integrated Visualization , Exploration and Authoring of Un-structured and Semantic Content. *in Proceedings of the International Conference on Web Information System Engineering (WISE)* (2013), 1–14.

[68] Kiselyov, O., Lämmel, R. and Schupke, K. 2004. Strongly typed heterogeneous collections. *in Proceedings of the SIGPLAN Workshop on Haskell.* (2004), 96–107.

[69] Komatsu, L.K. 1992. Recent views of conceptual structure. *Psychological Bulletin*. 112, 3 (1992), 500–526.

[70] Konrath, M., Gottron, T., Staab, S. and Scherp, A. 2012. SchemEX — Efficient construction of a data catalogue by stream-based indexing of linked data. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web (JWS)*. 16, (2012), 52–58.

[71] Kruskal, J.B. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Pstchometrika*. 29, 1 (1964).

[72] Kullback, S. and Leibler, R. 1951. On information and sufficiency. *Annals of Mathematical Statistics*. 22, (1951), 49 – 86.

[73] Kumar, R. and Tomkins, A. 2010. A characterization of online browsing behavior. *in Proceedings of the International World Wide Web Conference (WWW)* (New York, New York, USA, Apr. 2010), 561.

[74] Kumar, R. and Tomkins, A. 2009. A Characterization of Online Search Behavior. *In Proceedings of the IEEE Data Engineering Bulletin*. 32, 2 (2009), 1–9.

[75] Lee, L. 1999. Measures of Distributional Similarity. *in Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)* (1999), 25–32.

[76] Lee, T., Wang, Z., Wang, H. and Hwang, S. 2013. Attribute Extraction and Scoring : A Probabilistic Approach. *Proc. of. ICDE* (2013).

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

[78] Lin, J. and Katz, B. 2003. Question answering from the web using knowledge annotation and knowledge mining techniques. *in Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2003), 116.

[79] Liu, B., Hu, M. and Cheng, J. Opinion Observer : Analyzing and Comparing Opinions on the Web. *in Proceedings of the 14th International World Wide Web Conference (WWW)* 342–351.

[80] Malouf, R. 2002. Markov models for language-independent named entity recognition. *in Proceedings of the 6th Conference on Natural Language Learning (CoNLL)* (Taipei, Taiwan, 2002).

[81] Mervis, C.B. and Rosch, E. 1981. Categorization of Natural Objects. *Annual Review of Psychology*. 32, 1 (Jan. 1981), 89–115.

[82] Metzger, S. and Schenkel, R. 2011. S3K : Seeking Statement-Supporting top-K Witnesses. *in Proceedings of the 20th Conference on Information and Knowledge Management (CIKM)* (Glasgow, Scotland, UK, 2011), 37–46.

[83] Mika, P. and Potter, T. 2012. Metadata Statistics for a Large Web Corpus. *in Proceedings of the International Workshop on Linked Data on the Web (LDOW)* (2012).

[84] Miller, G.A. 1995. WordNet: a lexical database for English. *Communications of the ACM*. 38, 11 (Nov. 1995), 39–41.

[85] Montaner, M., López, B. and Rosa, J.L. de la 2003. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*. 19, 4 (Jun. 2003), 285–330.

[86] Mühleisen, H. and Bizer, C. 2012. Web Data Commons – Extracting Structured Data from Two Large Web Corpora. *in Proceedings of the International Workshop on Linked Data on the Web (LDOW)*. (2012).

[87] Murphy, G. 2002. *The Big Book of Concepts*. The MIT Press.

[88] Nakashole, N., Weikum, G. and Suchanek, F. 2012. Discovering and Exploring Relations on the Web. *in Proceedings of the Very Large Database Endowment (PVLDB)* (Istambul, Turkey, 2012).

[89] Nguyen, K., Ichise, R. and Le, B. 2012. Interlinking Linked Data Sources Using a Domain-Independent System. *in Proceedings of the Joint International Semantic Technology Conference (JIST)* (2012).

[90] Niblack, C.W., Barber, R., Equitz, W., Flickner, M.D., Glasman, E.H., Petkovic, D., Yanker, P., Faloutsos, C. and Taubin, G. 1993. QBIC project: querying images by content, using color, texture, and shape. *in Proceedings of the Symposium on Electronic Imaging: Storage and Retrieval for Image and Video Databases (SPIE)* (Apr. 1993), 173–187.

[91] Nie, Z., Wen, J.-R. and Ma, W.-Y. 2007. Object-level Vertical Search. *in Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*. (2007), 235–246.

[92] Ogden, C.K. and Richards, I.A. *The Meaning of Meaning*. ARK Paperbacks.

[93] Ora Lassila, Ralph R. Swick, World Wide, W.C. Resource Description Framework (RDF) Model and Syntax Specification.

[94] Pan, Z. and Chen, H. 2011. TongKey at Entity Track TREC 2011: Related Entity Finding. *in Proceedings of Text REtrieval Conference (TREC)* (2011).

[95] Pukelsheim, F. 1994. The Three Sigma Rule. *The American Statistician*. 48, 2 (May 1994), 88.

[96] Qian, L., Cafarella, M.J. and Jagadish, H. V 2012. Sample-driven schema mapping. *in Proceedings of the International Conference on Management of Data (SIGMOD)* (Scottsdale, Arizona, USA, 2012).

[97] Qiu, Y. and Frei, H.-P. 1993. Concept based query expansion. *in Proceedings of the Special Interest Group on Information Retrieval (SIGIR)* (New York, New York, USA, Jul. 1993), 160–169.

[98] Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc. San Francisco.

[99] Quinlan, J.R. 1987. Simplifying decision trees. *International Journal of ManMachine Studies*. 27, 3 (1987), 221–234.

[100] Resnick, P. and Varian, H.R. 1997. Recommender systems. *Communications of the ACM*. 40, 3 (Mar. 1997), 56–58.

[101] Ricci, F., Rokach, L., Shapira, B. and Kantor, P.B. 2010. *Introduction to Recommender Systems Handbook*. Springer.

[102] Robertson, S.E. and Sparck Jones, K. 1976. Relevance Weighting of Search T erms. *Journal of the American Society for Information Science and Technology*. 27, 3 (Dec. 1976), 129–146.

[103] Rosch, E. 1975. Cognitive reference points. *Cognitive Psychology*. 7, 4 (Oct. 1975), 532–547.

[104] Rosch, E. 1975. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*. 104, 3 (1975), 192–233.

[105] Rosch, E. and Mervis, C.B. 1975. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*. 7, 4 (Oct. 1975), 573–605.

[106] Rosch, E., Mervis, C.B., Gray, W.D., Johnson, D.M. and Boyes-Braem, P. 1976. Basic objects in natural categories. *Cognitive Psychology*. 8, 3 (Jul. 1976), 382–439.

[107] Rowley, J. 2000. Product search in e-shopping: a review and research propositions. *Journal of Consumer Marketing*. 17, 1 (Jan. 2000), 20–35.

[108] Salton, G., Wong, A. and Yang, C.S. 1975. A vector space model for automatic indexing. *Communications of the ACM*. 18, 11 (Nov. 1975), 613–620.

[109] Sarkas, N., Paparizos, S. and Tsaparas, P. 2010. Structured annotations of web queries. *in Proceedings of the International Conference on Management of Data (SIGMOD)*. (2010), 771.

[110] Scheglmann, S. and Gröner, G. 2012. Property-based Typing for RDF-Access. *in Proceedings of the First Workshop on Programming the Semantic Web* (Rio de Janeiro, Brasil, 2012), 4–7.

[111] Scheglmann, S., Gröner, G., Staab, S. and Lämmel, R. 2013. Incompleteness-aware programming with RDF data. *in Proceedings of the Workshop on Data Driven Functional Programming (DDFP)*. (2013), 11.

[112] Scheglmann, S., Scherp, A. and Staab, S. 2012. Declarative Representation of Programming Access to Ontologies. *The Semantic Web: Research and Applications*. E. Simperl, P. Cimiano, A. Polleres, O. Corcho, and V. Presutti, eds. Lecture Notes in Computer Science. 659–673.

[113] Srinivasan, P. 1996. Query expansion and MEDLINE. *Information Processing & Management*. 32, 4 (Jul. 1996), 431–443.

[114] Stock, W.G. 2010. Concepts and Semantic Relations in Information Science. *Journal of the American Society for Information Science and Technology*. 61, 10 (2010), 1951–1969.

[115] Suchanek, F.M., Abiteboul, S. and Senellart, P. 2011. PARIS: probabilistic alignment of relations, instances, and schema. *in Proceedings of the Very Large Database Endowment (PVLDB)* (Nov. 2011), 157–168.

[116] Suchanek, F.M. and Weikum, G. 2007. YAGO : A Core of Semantic Knowledge Unifying WordNet and Wikipedia. *In Proc. of the 16th Int. World Wide Web Conf. (WWW)* (Banff, Canada, 2007).

[117] Sydow, M., Pikula, M. and Schenkel, R. 2010. DIVERSUM: Towards diversified summarisation of entities in knowledge graphs. *in Proceedings of the International Conference on Data Engineering Workshop (ICDEW)* (2010), 221–226.

[118] Tang, J., Li, J., Liang, B., Huang, X., Li, Y. and Wang, K. 2006. Using Bayesian decision for ontology mapping. *in Proceedings of Web Semantics: Science, Services and Agents on the World Wide Web*. 4, 4 (2006), 243–262.

[119] Tang, J., Liang, B., Li, J. and Wang, K. 2004. Risk Minimization based Ontology Mapping. *in Proceedings of the Advanced Workshop on Content Computing (AWCC)* (2004).

[120] Tesauro, G., Gondek, D.C. and Prager, J.M. 2013. Analysis of Watson's Strategies for Playing Jeopardy! *Journal of Artificial Intelligence Research (JAIR)*. 21, (2013), 205–251.

[121] Tsoumakas, G. and Katakis, I. 2007. Multi Label Classification: An Overview. *International Journal of Data Warehousing and Mining*. 3, 3 (2007), 1–13.

[122] Tversky, A. 1977. Features of similarity. *Psychological Review*. 84, 4 (1977), 327–352.

[123] Vijay V. Raghavan, G.S.J.P.B. 1989. A critical investigation of recall and precision as measures of retrieval system performance. *Journal of Transactions on Information Systems (TOIS)*. 7, (1989), 205–229.

[124] Volz, J., Bizer, C., Gaedke, M. and Kobilarov, G. 2009. Discovering and Maintaining Links on the Web of Data. *in Proceedings of the International Semantic Web Conference (ISWC)* (2009), 650–665.

[125] Voorhees, E.M. 1994. Query expansion using lexical-semantic relations. *in Proceedings of the Special Interest Group on Information Retrieval (SIGIR)* (Aug. 1994), 61–69.

[126] Wang, Z., Tang, C., Sun, X., Ouyang, H., Lan, R., Xu, W., Chen, G. and Guo, J. 2010. PRIS at TREC 2010: Related Entity Finding Task of Entity Track. *in Proceedings of Text REtrieval Conference (TREC)* (2010).

[127] Weld, D.S., Hoffmann, R. and Wu, F. 2008. Using Wikipedia to bootstrap open information extraction. *in Proceedings of the Special Interest Group on Management of Data (SIGMOD)*. 37, 4 (Mar. 2008), 62.

[128] Whitelaw, C., Kehlenbeck, A., Petrovic, N. and Ungar, L. 2008. Web-scale named entity recognition. *in Proceedings of the International Conference on Information and Knowledge Management (CIKM)* (2008), 123.

[129] Wittgenstein, L. 1953. *Philosophical investigations*. New York: The MacMillan Company.

[130] Yates, A. and Etzioni, O. 2009. Unsupervised Methods for Determining Object and Relation Synonyms on the Web. *Journal of Artificial Intelligence Research (JAIR)*. 34, (2009), 255–296.

[131] Yu, J.X. and Yu, P.S. 2006. Text classification without negative examples revisit. *Journal of Transactions on Knowledge and Data Engineering (TKDE)*. 18, 1 (2006), 6–20.

[132] Zhanyi Wang, Wenlong Lv, Heng Li, Wenyuan Zhou, Li Zhang, Xiao Mo, Liaoming Zhou, Weiran Xu, Guang Chen, J.G. 2011. PRIS at TREC 2011 Entity Track: Related Entity Finding and Entity List Completion. *In Proc.Text Retrieval Conference (TREC)* (2011).

[133] Zheng, Q., Shao, C., Li, J., Wang, Z. and Hu, L. 2013. RiMOM2013 Results for OAEI 2013. *in Proceedings of the International Workshop on Ontology Matching (OM)* (2013).

[134] Zhou, M., Wang, H. and Chang, K.C. 2013. Learning to Rank from Distant Supervision : Exploiting Noisy Redundancy for Relational Entity Search. *in Proceedings of the International Conference on Data Engineering (ICDE)* (2013).

[135] Zhou, X., Gaugaz, J., Balke, W.-T. and Nejdl, W. 2007. Query relaxation using malleable schemas. *in Proceedings of the Special Interest Group on Management Of Data (SIGMOD)* (2007).