User-centric Knowledge Extraction and Maintenance

Dissertation
zur Erlangung des Grades des
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Steffen Metzger

Saarbrücken
11. Februar 2014

Dekan und Vorsitzender des Promotionsausschusses	Prof. Dr. Mark Groves
Tag des Promotionskolloquiums	28.03.2014

Prüfungsausschuss

Vorsitzender	Prof. Dr. Dietrich Klakow
Berichterstatter	Prof. Dr. Ralf Schenkel
Berichterstatter	Prof. Dr. Gerhard Weikum
Akademischer Mitarbeiter	Dr. Klaus Berberich

Acknowledgements This work would not have been possible without the support of my supervisor, Prof. Dr. Ralf Schenkel. I would like to thank him for providing me with the freedom to explore topics that interested me and for being always accessible even when physical distances and obligations made that more difficult. I would also like to thank Assistant-Prof. Dr. Katja Hose for being a great office mate during my time at the MMCI and for always being open for a discussion and providing her valuable opinion. Similarly, I appreciate the great company Erdal Ekuzey provided in the office at the MPI. I am also thankful to Prof. Dr. Gerhard Weikum for ensuring a healthy atmosphere in the research group and being accessible whenever needed. Several other people at both institutes deserve my gratefulness, among them the secretaries and IT staff for making administrative tasks an easy obstacle. I would also like to thank Michael Stoll, Sergej Isak-Geidel, Anastasiya Goncharova and Hassan Issa as well as Shady Elbassuoni and Marcin Sydow for their commitment in our shared efforts. Further thanks go to the whole group for providing a nice working environment.
Finally, this work would not have been possible without the support of my parents, who always encouraged me in pursuing my own path.

This work has been partially funded by the BMBF (German Federal Ministry of Education and Research) through the WisNetGrid project[1].

[1] `http://www.wisnetgrid.org/`

Zusammenfassung

Eine Ontologie ist eine Wissenssammlung in maschinenlesbarer Form. Da eine große Bandbreite an Informationen nur in natürlichsprachlicher Form verfügbar ist, werden maschinenlesbare Ontologien häufig durch imperfekte automatische Verfahren erzeugt, die eine Übersetzung in eine maschinenlesbare Darstellung vornehmen. In der vorliegenden Arbeit werden Methoden zur menschlichen Unterstützung des Extraktionsprozesses und Wartung der erzeugten Wissensbasen präsentiert.

Dabei werden drei Beiträge geleistet:

1. Zum ersten wird ein interaktives Extraktionstool (LUKe) vorgestellt.
   Hierfür wird ein bestehendes Extraktionssystem um die Integration von Nutzerkorrekturen auf verschiedenen Ebenen der Extraktion erweitert und an ein beispielhaftes Szenario angepasst.

2. Zum zweiten wird ein Ansatz (S3K) zur Dokumentsuche basierend auf faktischen Aussagen beschrieben.
   Dieser erlaubt eine aussagenbasierte Suche nach Belegstellen oder weiteren Informationen im Zusammenhang mit diesen Aussagen in den Dokumentsammlungen die der Wissensbasis zugrunde liegen.

3. Zuletzt wird QBEES, eine Ähnlichkeitssuche in Ontologien, vorgestellt.
   QBEES ermöglicht die Suche bzw. Empfehlung von ähnlichen Entitäten auf Basis der semantischen Eigenschaften die sie mit einer als Beispiel angegebenen Menge von Entitäten gemein haben.

Alle einzelnen Komponenten sind zudem in eine modulare Gesamtarchitektur integriert.

Abstract

An ontology is a machine readable knowledge collection. There is an abundance of information available for human consumption. Thus, large general knowledge ontologies are typically generated tapping into this information source using imperfect automatic extraction approaches that translate human readable text into machine readable semantic knowledge. This thesis provides methods for user-driven ontology generation and maintenance. In particular, this work consists of three main contributions:

1. An interactive human-supported extraction tool: LUKe.
   The system extends an automatic extraction framework to integrate human feedback on extraction decisions and extracted information on multiple levels.

2. A document retrieval approach based on semantic statements: S3K.
   While one application is the retrieval of documents that support extracted information to verify the correctness of the piece of information, another application in combination with an extraction system is a fact based indexing of a document corpus allowing statement based document retrieval.

3. A method for similarity based ontology navigation: QBEES.
   The approach enables search by example. That is, given a set of semantic entities, it provides the most similar entities with respect to their semantic properties considering different aspects.

All three components are integrated into a modular architecture that also provides an interface for third-party components.

Contents

1. Introduction

1.1. Motivation

It is often said, that we live in an information age, where the availability of information is no longer a problem, but the extraction of crisp knowledge from the huge pile of information is. In recent years there has been much effort in aggregating human knowledge not only in human readable form (e.g. Wikipedia [wikb]), but also in a machine readable format, i.e. as *ontologies*, providing computers with a canonic information representation that can be used to boost applications with semantic information, such that computers can better assist us in knowledge intense tasks. Such ontologies are typically concerned with formulating common knowledge (of a particular domain) in the form of factual assertions, i.e. an ontology typically defines class concepts, such as "`science`", "`politician`" or "`movie`", individual entities, such as "`Barack Obama`", "`Quentin Tarantino`" or "`Pulp Fiction`", and relations like "`is a`" or "`directed`" that can be instantiated to connect entities or entities and class concepts, e.g. "`Barack Obama is a politician`" or "`Quentin Tarantino directed Pulp Fiction`". There are already many areas in which ontologies are applied, namely in named entity disambiguation [BP06, HYB+11], word sense disambiguation [CCB06], machine translation [KL94, OH94, CGN05], and document classification [IW06, BTV08, SLKL05]. It is also used in information retrieval tasks, e.g. query expansion [GSW05, LLYM04] or to enable entity- and statement driven search [BCSW07, MEHS11, CRSE07], and in question answering [VVMD03, HLN04, BCD+07] as well as question guessing, i.e. playing Jeopardy [FBCC+10]. Among further applications are semantic web-service matchmaking [KK09] and virtual reality management [KCKW11, xai]. One might also consider canonical product databases as ontological knowledge, thus including online shops like Amazon [ama], price search engines [goo, gei, pri] or public databases such as the IMDB [imd] as applications of ontologies. In short, ontologies as machine readable knowledge are paramount in an information driven digital society. However, information is often only available in human readable form not suitable for machine consumption. Thus, ontology generation can be considered a translation task, where human knowledge needs to be transformed into machine readable semantic statements.

There are typically two ways to create such knowledge collections. Either humans manually create the ontology [GL90, Vra12], i.e. providing their beliefs as ontological basis, or to employ automatic methods that try to extract knowledge from source documents written for humans. The latter in fact means for the computer to interpret human belief

representations in textual form [ABK+07, CBK+10, NWS12, EFC+11][1].

While some applications require knowledge only on a very narrow domain, several applications such as general purpose search engines, question answering systems or general artificial intelligence approaches require broad, up-to-date domain knowledge or even universal, i.e. across all domains, knowledge. Besides commercial approaches [gkg, SVT+12], there are several academic efforts to attain a general knowledge ontology [GL90, Vra12, HSBW13, ABK+07, CBK+10, NWS12, EFC+11] that range from very precision-oriented approaches focused on hand-picked canonical relations [HSBW13, CBK+10, ABK+07, GL90] via Freebase with its crowd-driven semi-structured approach [BEP+08] to Open Information Extraction approaches [EFC+11, NWS12] that try to capture any form of relation observable in text available on the world wide web.

Due to the sheer size of knowledge to be covered in general purpose common knowledge ontologies, human manufacturing is very resource intense; working on the OpenCyc project, one of the earliest common knowledge ontology projects, Douglas Lenat estimated in 1986 the effort to complete Cyc would be 350 man-years of effort [Boo86]. Hence, most modern ontologies with such a general ambition are at least to a large degree generated automatically, by methods that extract information from a human readable format, e.g. web documents, into a machine readable format [ABK+07, CBK+10, NWS12, SSW09, AG00]. With the exception of some Open Information Extraction approaches [EFC+11, EBC06, Bri99, ECD+04, EBSW08], all variants transform the information into a canonic format, i.e. disambiguating entity references and relations to canonic representations. The problem in this interpretation approach is that the system might misinterpret human formulations of their knowledge or be misguided by misleading or 'faulty' formulations, when it translates such textual representations into *factual* knowledge. Often such extraction systems are based on an iterative approach that learns from some relational examples the textual representation of a relation, thereby generates more examples which it uses again to learn more ways this information can be textually represented. Thus, mistakes multiply as they are carried on through the iterations and lead to more subsequent mistakes. Here human intervention can help to avoid interpretation mistakes. Often a particular application needs human supervision anyway to guarantee at least the best humanly possible quality of the information generated. In particular, some scenarios focus not on the generation of ontologies, but on the collaborative annotation of documents with semantic information, a shared ontology being just a side effect or the requirement providing a semantic vocabulary. Since ontology generation often is a long-term process, in this thesis we suggest to integrate human supervision with an iterative extraction approach, thus intermingling ontology generation with ontology maintenance. Our approach is tailored towards iterative extraction approaches that identify textual expressions of semantic relations and in particular aims at feedback based on the links between semantic information and their textual representation (see Chapter 3). This has benefits for both, as the extraction engine can learn from user-feedback on various levels and in an annotation scenario, the user

[1] Alternatively, computers could make their own observations, e.g. through sensors. We ignore this for ontology generation, but briefly discuss this option in Section 6 in a slightly different context.

only needs to verify annotations and amend oversights by the system instead of annotating the document completely by hand. In addition, when the system can learn from user involvement, this can reduce the initial obstacle to user engagement that extensive domain knowledge requirements to be provided at the outset often are. While there still remains the need for some start information, to some degree domain knowledge can rather be learnt during the process than be gathered completely beforehand. We also provide a modular architecture that allows to replace individual components on multiple levels in Section 3.6 and discuss a prototype implementation that has been adapted to a particular application scenario in Section 3.7.

From a user's perspective there is also the issue of credibility. An ontology is simply a collection of statements. Without prior knowledge, we cannot judge whether a particular statement is true or false, but if we have such prior knowledge, we would not need to look up the information. Encyclopedias and journalism have the same problem: aggregated information is only trustworthy if it can point out its sources. The same holds for semantic statements and ontologies. In an ontology such source information describing the heritage of information pieces is typically called *provenance*. As we are mainly concerned with information extracted from human readable sources, we focus on provenance information in the form of textual representations of a piece of information in texts written by humans. Typically, the more independent sources serve as *witnesses* that provide a *testimony* supporting a piece of information, the more likely this piece of information is considered trustworthy. Hence, when investigating statements, we are typically interested in identifying witness documents that confirm the piece of information. In this thesis, we provide methods that allow users to search for witness documents supporting one or several statements provided as a query. These methods provide an important tool for ontology maintenance, where they can be used to verify doubtful statements that may have been added by mistake. Besides ontology maintenance, these methods can also be of general value when investigating claims, for instance, to analyse the environment in which they are supported, or to observe a particular combination of statements in context, e.g. identifying sources that bring different statements in a shared context. Our approach at identifying witness documents for semantic information relies on the aforementioned provenance-tracking, user-supported information extraction approach. However, in Section 4.8 we also briefly discuss a possible variation of our methods, such that they can be applied to search for documents without a prior extraction on these particular documents. In addition, the links from semantic statements to source documents allow us not only to navigate from the realm of abstract knowledge to their concrete textual context, but also allows us to compare documents based on the contained information and thus, navigate along links based on semantic similarity between documents (see Section 4.9).

Figure 1.1.: Information shown by Google for search "Arnold Schwarzenegger";
Note: some unessential information has been removed; Source: Google.com

When humans use or maintain an ontology, ease of navigation is of importance. Traditionally an ontology is navigated via its relations, i.e. from one entity a user may reach all entities related by some semantic property. For instance, given Arnold Schwarzenegger as an entity, a user might find that there is a link - namely the fact that he was born there - that connects him with Austria. An orthogonal dimension typically not explicitly modelled is entity similarity. So given Arnold Schwarzenegger, a user might rather be interested in other action movie stars than in his place of birth. Since a while ago Google amends some search queries with such a kind of information from its knowledge graph [gkg], Figure 1.1 provides an example that is shown when searching for Arnold Schwarzenegger. If an entity is recognised, some information about the entity, like date of birth or his movies in case of an actor, is shown next to the normal document search results along with a set of entities that "people also search for". The latter are often entities of similar type. Yet these entities often only reflect the most popular *aspect* of an entity. For instance, for Arnold Schwarzenegger, Google provides action actors Sylvester Stallone, Jean-Claude Van Damme and Bruce Willis, but also Maria Shriver, his ex-wife, and Patrick Schwarzenegger, his son. However, a user might also be interested in other governors of California, actors that are also politicians, or bodybuilders from Austria etc. One can consider this as a lack of diversity with regard to the information available, i.e. mainly the 'being an actor'

aspect is represented. Since the selection seems to be based on search queries rather than the underlying knowledge graph, only rarely is an indication given how the selected entities are connected to the original entity. Of course, if a user is interested in the set of all actors that are also politicians, the semantic web provides means to define such a set of entities in very specific terms, e.g. via query languages such as SPARQL [SPA]. However, formulating a SPARQL query needs expert knowledge about the query language and the underlying ontology. A typical non-expert user would not find such an interface comfortable. In order to find a set of entities, it is much more intuitive to provide some example entities from the set, especially in cases where the user is not sure what the properties he looks for precisely are or how they would be modelled in an ontology. In this thesis we provide methods that support querying by entity examples, taking all aspects of the provided entities into account. The user can direct the selection process by iteratively refining the query when she selects additional entities (see Chapter 5).

Finally, the semantic web provides a vast amount of distributed ontological knowledge and a broad field of systems that operate on ontologies. These systems can be used to participate in the generation of semantic information, be it via information extraction, crowd sourcing or reasoning, are supporting the maintenance and storage of semantic information or provide applications that users can interact with on an intuitive level. Yet using this pool of information and in particular several types of systems in a collaborative manner is a time consuming effort. Most systems need to be locally deployed and the setup of particular tools and the adaption of published ontologies to a particular application domain, which might require assessing the trustworthiness of ontologies, merging ontologies or manually adapting ontologies, is often a redundant task that users independently repeat without benefiting from the work of others. In Chapter 6 we sketch a vision for a collaborative knowledge network that enables collaboration of *information agents* via a simple protocol based on a peer-to-peer structure that requires little user intervention.

1.2. Contributions

With this work we provide

- LUKe, an extraction system that integrates user feedback into the extraction processes, supporting the correction of extracted information, the addition of not (properly) recognised information and the integrated provision of domain knowledge such as entity names. This system is integrated into a modular framework providing a web service frontend that enables the integration of separate user-frontends ensuring the possibility to adapt to particular user-groups.

- S3K, a document retrieval model, that enables the identification of documents supporting a subset of a given set of statements and provides a language model based ranking of those documents that can be tailored either towards preferring documents providing additional insight or towards preferring documents containing testimonies strongly supporting the statements.

- QBEES, a model to find similar entities based on the shared ontological aspects by first identifying groups of maximal similarity, i.e. maximal property overlap, and then first ranking these entity groups before ranking the entities in each group.

1.3. Publications

1.3.1. User-Supported Extraction

Our Learning from User Knowledge (LUKe) feedback integration framework along with the Knowledge Workbench frontend (MIKE, see Section 3.7.2) as well as the underlying architecture have been part of demonstrations at the International Conference on Information and Knowledge Management 2012 (CIKM2012) [MSHS12] and, in an early version, at the Lernen, Wissen, Adaption (Learning, Knowledge, Adaptation) forum 2011 (LWA2011) [SHMS11].

- Steffen Metzger, Michael Stoll, Katja Hose, and Ralf Schenkel, *LUKe and MIKE: Learning from User Knowledge and Managing Interactive Knowledge Extraction*, 21st ACM International Conference on Information and Knowledge Management (CIKM 2012) (Maui, USA), 2012.

- Michael Stoll, Katja Hose, Steffen Metzger, and Ralf Schenkel, *Interaktive Wissensextraktion und Wissenssuche*, LWA 2011 : Technical Report ; Report of the symposium "Lernen, Wissen, Adaptivität 2011" of the GI special interest groups KDML, IR and WM (Magdeburg) (Myra Spiliopoulou, Andreas Nürnberger, and Rene Schult, eds.), Otto von Guericke Universität, 2011, pp. 205–206.

In a presentation at the EGI community forum 2012 (EGI2012) [JMD+12] and a follow up report we discuss the integration of LUKe into the WisNetGrid project context. This

discussion is further deepened in the paper published via the Journal of Internet Services and Applications (JISA) [DHJ+13] and the International Workshop on Applications of Semantic Technologies 2011 (AST2011) [HMS11].

- Milad Daivandy, Denis Hünich, Rene Jäkel, Steffen Metzger, Ralph Müller-Pfefferkorn, and Bernd Schuller, *Heterogeneous resource federation with a centralized security model for information extraction*, Journal of Internet Services and Applications **4** (2013), no. 1, 10

- Katja Hose, Steffen Metzger, and Ralf Schenkel, *Sharing Knowledge between Independent Grid Communities*, 6th International Workshop on Applications of Semantic Technologies (AST 2011) (Berlin, Germany) (Catherina Burghart, Stephan Grimm, Andreas Harth, and Jens Wissmann, eds.), 2011.

We have also presented the feedback system and its integration into the WisNetGrid architecture at the "WisNetGrid Ergebniskworkshop", the final project presentation workshop of the WisNetGrid project.

1.3.2. Provenance based Document Retrieval

We have first presented our statement search approach as part of a demo at the VLDB conference 2010 [EHMS10], which also included a prototype of an approach to retrieve additional source files for given statements at runtime via keyword search.

- Shady Elbassuoni, Katja Hose, Steffen Metzger, and Ralf Schenkel, *ROXXI: Reviving witness dOcuments to eXplore eXtracted Information*, Proceedings of the VLDB Endowment **3** (2010), no. 1-2, 1589–1592.

In the following year we presented the details of the S3K-framework (Seek Statement-Supporting top-K Witnesses) and an evaluation at the CIKM conference in [MEHS11].

- Steffen Metzger, Shady Elbassuoni, Katja Hose, and Ralf Schenkel, *S3K: Seeking Statement-Supporting top-K Witnesses*, 20th ACM Conference on Information and Knowledge Management (CIKM 2011) (Glasgow, UK), 2011.

Early work on our work on semantic document similarity has been presented at the LWA2011 in [IHMS11].

- Hassan Issa, Katja Hose, Steffen Metzger, and Ralf Schenkel, *Advances Towards Semantic Plagiarism Detection*, Workshop Information Retrieval at LWA 2011 (Magdeburg, Germany) (Ingo Frommholz and Claus-Peter Klas, eds.), 2011.

1.3.3. QBEES - Query By Entity ExampleS

Our work on querying an ontology by example entities has been initially published in a poster paper at the CIKM conference 2013 [MSS13]. While this publication describes the basic approach in its early version, it lacks in particular the aspect relaxation component.

- Steffen Metzger, Ralf Schenkel, and Marcin Sydow, *QBEES: Query by Entity Examples*, 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013) (San Francisco, USA), 2013.

1.3.4. Colledge - Collaborative Knowledge Networks

In [MHS12] we first introduce the notion of a collaborative knowledge network and outlined the basic architecture we propose.

- Steffen Metzger, Katja Hose, and Ralf Schenkel, *Colledge - a vision of collaborative knowledge networks*, 2nd International Workshop on Semantic Search over the Web (SSW 2012) (Istanbul, Turkey), ACM, 2012, p.

1.4. Outline

The remainder of this thesis is organised as follows. First, in Chapter 2 we will discuss the basic foundations that the main contributions are based on. Then, in Chapter 3 the integration of user-feedback into a pattern based extraction process is discussed and we present a modular framework that is also the basis for the following contributions of this work. We also discuss the WisNetGrid application scenario in 3.7. Afterwards our focus shifts from ontology generation to maintenance, and the methodology to retrieve and rank source documents in support of given ontological statements is provided in Chapter 4. We also briefly discuss a search for additional source documents for statements and a potential application that applies the links from extracted information to its sources to provide a semantic similarity measure on documents. In Chapter 5 our approach to similarity search based on example entities is presented. In Chapter 6 we envision a much broader collaboration of different knowledge agents beyond what this work can provide. Finally, the thesis is concluded in Chapter 7 with a summary and an outlook into future work.

2. Foundations

In order to maintain a machine readable knowledge collection, one first needs to define how to store knowledge and then this knowledge collection needs to be filled with information. The first task can be split into the problem to define a knowledge representation format and provide a vocabulary used to express information, oth together form an *ontology model*. In the following, we will first discuss how information can be expressed semantically in Section 2.1. Based on this we will discuss information extraction, the task to gain knowledge from human readable source documents, in Section 2.2. And finally, we will provide some basics about information retrieval and evaluation measures in Section 2.3.

2.1. Knowledge Representation

The problem to logically represent knowledge has a long tradition in Artificial Intelligence research dating back to the 1970's, when, for instance, Prolog, a first-order logic based programming language allowing to declare facts and logically derive further knowledge from such axioms, was conceived and developed by Alain Colmerauer and Philippe Roussel [Kow88, CR93]. One of the earliest larger general purpose common knowledge ontologies was started in 1984 as the Cyc project, which since then has been privatised and is still continued by the Cycorp company [ocy, GL90, MCWD06]. There are various forms to represent factual knowledge and model the underlying logic [Kow88, CR93, SS09b, KL89, FN71, Gen91, Bar77, BCM+03, Wor04a, Wor12, RN03], in recent years the resource description framework (RDF) [Wor04a], its schema RDFS [Wor04b] and the web ontology language OWL2 [Wor12] based on both, all specified by the World Wide Web Consortium (W3C), have become an accepted standard based on description logic [BCM+03] and represent the most used knowledge representation model at least within the Semantic Web and large scale ontology community. This work is based on the YAGO ontology model [Suc08], which in turn is based on RDF and RDFS. Hence, we will now first discuss some general basic concepts, then how these concepts can be represented in RDF/RDFS and finally, we will note the relevant modifications provided by YAGO.

2.1.1. Basic concepts

Many ontology models share at least the following five basic underlying concepts: *entities*, *literals*, *classes*, *relations* and *statements*.

Definition 2.1.1 (Entity)**.** *Entities* are the set of all *information objects*, i.e. "things"-existing or abstract, about which the ontology can make any kind of statement, they represent the ontology's *knowledge domain*. An *entity* can be *anything individually identifiable* including individual living beings (e.g. `Barack_Obama`, `Angela_Merkel`), non-animate things like a particular house or a value, e.g., a particular number, a weight value or a date or even abstract concepts like `science` or particular statements. We denote the set of all entities by $\mathcal{E}$.

Definition 2.1.2 (Class)**.** A *class* represents an abstract concept, such as the notion of `science`, `art` and `entertainment` or the general idea of what a `house`, a `politician` or a `number` is. Classes can be used to group entities, i.e. entities can be assigned to classes in which case we call an assigned class a *type* of the entity and say the entity is an *instance* of the class. For instance, we might consider `Barack_Obama` as an instance of the class `politician`. Typically classes are ordered in a hierarchy with more general classes *above* the classes they conceptually subsume. For instance, `Astrophysics` might be considered a sub-class of `physics` which in turn is a sub-class of `science`. Thus `Astrophysics` is also an indirect sub-class of `science`, i.e. the sub-class relationship is transitive. We denote the set of all classes by $\mathcal{C}$. Classes are entities themselves, such that it holds $\mathcal{C} \subseteq \mathcal{E}$.

Definition 2.1.3 (Literal)**.** One particular type of abstract entities are *literals*. The set of literals contains all forms of *values* such as numbers, date expressions or quantities as well as other character strings. The set of literals $\mathcal{L}$ is a subset of all entities, i.e. $\mathcal{L} \subseteq \mathcal{E}$.

So far we have only discussed information objects. Now we consider how to make relational statements about these entities. *Statements* connect entities. By linking them they imply a semantic connection between both entities. The type of the connection is provided by a label on the link, the *relation*. For instance, if we connect `Barack_Obama` with `Hawaii` via the `bornIn` relation, this represents the information that the entity Barack Obama is born in Hawaii.

Definition 2.1.4 (Relation)**.** A *relation* r is the label used to describe a semantic relationship between n entities. Depending on the number of entities a relation typically occurs with, we call it *unary*, *binary* or *n-ary* relation.

For instance, the binary relation `actedIn` may describe the relationship of 'playing a role' that holds between two arguments, an actor and a movie. If the relation would also connect the actor to the role he played in a movie, it would be a ternary relation etc. In the following we will be concerned mainly with binary relations, but we will discuss how arbitrary n-ary relations can be broken down into several binary relations. We denote the set of all relations by $\mathcal{R}$. Note that relations are also entities: $\mathcal{R} \subseteq \mathcal{E}$.

Definition 2.1.5 (Statement)**.** A *statement* or *relation instance* is a concrete instance of a relation with particular arguments, i.e. a particular combination $r(e_1, ..., e_n)$ of a relation $r \in \mathcal{R}$ and entities $e_1, ..., e_n \in \mathcal{E}$. For any entities $e_1, ..., e_n$ the relation instance $r(e_1, ..., e_n)$ *states* the information that the relation r holds with respect to the given entities. For instance, given the binary relation `actedIn` and the entities `Arnold_Schwarzenegger` (of class `actor`) and `The_Terminator` (of class `movie`), the relation instance `actedIn(Arnold_Schwarzenegger,The_Terminator)` represents the information that Arnold Schwarzenegger appeared at some point in the movie "The Terminator ". A statement in itself holds no truth, it simply represents an assertion that may be valid or may be wrong (i.e. consistent or inconsistent with our observations), such that the set of relation instances basically represents "all that can be said" about the entities in $\mathcal{E}$ with the relations from $\mathcal{R}$. We denote the set of all possible relation instances by $\mathcal{RI}$. Statements are entities as well: $\mathcal{RI} \subseteq \mathcal{E}$.

Definition 2.1.6 (Facts)**.** Statements *considered true are called facts.* The *set of all facts* is denoted as $\mathcal{F} \subseteq \mathcal{RI}$. When considering calculations, we assume statements that are true to have a value of 1 and represent a value of 0 otherwise.

Definition 2.1.7 (Domain and Range)**.** Given all statements of a particular binary relation, we call the set of all entities that appear in the first argument position as the relation's *domain* and the set of all entities that appear at the second argument position the relation's *range.* For example given a binary `actedIn` relation, its domain is the set of actors and its range the set of movies.

Definition 2.1.8 (Ontology)**.** An ontology consists of a set of entities $\mathcal{E}$, a set of literals $\mathcal{L}$, a set of classes $\mathcal{C}$, a set of relations $\mathcal{R}$ and a set of facts $\mathcal{F}$.

2.1.2. RDF/RDFS

RDF and RDFS provide a particular knowledge representation format and an ontology language.

In RDF there are three types of logical symbols to address entities, namely Uniform Resource Identifier (URI) references, literals and blank nodes.

Definition 2.1.9 (URI reference)**.** A *URI reference* is a globally unique entity identifier, such that the same URI reference always refers to the same entity, but there may be several URI references that identify the same entity. URI references consist of five parts, a scheme, an authority, an optional query and an optional fragment making up the full URI reference according to this format:

```
scheme ":" authority [ "?" query ] [ "#" fragment ]
```

For instance, a valid URI, that refers to the entity `Barack_Obama`, the President of the United States: `http://www.wikipedia.org/wiki/Barack_Obama`, and another equally valid URI referring to the same entity `http://mpii.de/yago/resource/Barack_Obama`. As you can

see, URLs are a subset of all URIs, but other forms are possible. The syntax and allowed characters for each part of the format is described in detail in [BL98].

In RDF, a common prefix may be replaced by a *namespace*. For instance, the full URI reference `http://mpii.de/yago/resource/Barack_Obama` representing the entity of US President Barack Obama can be shortened to `y:Barack_Obama` given 'y' has been defined as the namespace representing the URI prefix `http://mpii.de/yago/resource/`. In the following we assume that within a single ontology, represented by a namespace, every entity is only represented by a single URI reference. In addition, if the namespace is clear from the context or of no relevance, we will omit it in the following, just writing `Barack_Obama`, for instance, to address the entity representing the 44th US president.

Definition 2.1.10 (Literals)**.** A literal is a string that represents a value. For example the literal "7" represents the value *seven*. Literals can be typed by attaching a URI representing the literal's *datatype* as a suffix separated by '^^'. For instance, "27"^^`http://www.w3.org/2001/XMLSchema#integer` represents the *integer* number 27 and "1999-08-16"^^`xsd:date` represents the *date* 16th August in 1999.

Blank nodes are used as anonymous placeholders for an unnamed entity. We will see an example for their application later when we discuss n-ary relations.

RDF triples RDF knowledge bases are based on binary relations. All statements are modelled in the form of subject-predicate-object triples. For example, the information that US president Barack Obama was born in Hawaii, reflected by the relation instance `bornIn`(`Barack_Obama`,`Hawaii`), is represented in RDF format as the triple (`Barack_Obama`, `bornIn`, `Hawaii`) with subject `Barack_Obama`, predicate `bornIn`, and object `Hawaii`. So, a *triple* is a (binary) relation instance represented in subject-predicate-object form. However, RDF restricts the triple format such that not all forms of entities can occur in all positions. While URIs can occur on every position, literals can only occur as objects. Blank nodes however can replace URIs and therefore also occur in every position. Note that relations are also represented by URIs, but since statements are represented as triples, they cannot occur as subject, predicate or object in another triple. However, RDF provides another way to address statements in other statements called *reification*; we will come back to this issue later.

As statements in RDF strictly represent binary relations, the set of statements form the labelled edges of a graph with the entities as its nodes. Hence, an RDF ontology can also be called an RDF graph.

RDF vocabulary RDF and RDFS provide a predefined vocabulary that covers the main basic concepts, allows to define a type hierarchy and use these classes to type entities and declare the domain and range of relations. The vocabulary belongs to the name space `rdf` representing the URI reference prefix `http://www.w3.org/1999/02/22-rdf-syntax-ns#` and the namespace `rdfs` representing the prefix `http://www.w3.org/2000/01/rdf-schema#` respectively.

- First, the resources `rdf:Resource`, `rdfs:Class` and `rdf:Property` refer to the set of all entities $\mathcal{E}$, the set of all classes $\mathcal{C}$ and the set of all relations $\mathcal{R}$ respectively, while `rdf:Statement` refers to the set of all relation instances $\mathcal{RI}$.

- To declare a binary relation's domain and range the relations `rdfs:domain` and `rdfs:range` can be used. For instance, the following two triples indicate that the `actedIn` relation's subject arguments are actors and the object arguments are movies:
  (`actedIn`, `rdfs:domain`, `actor`)
  (`actedIn`, `rdfs:range`, `movie`)

Earlier we noted entities can be associated with classes. Now we can express this via a relation provided by RDF, the `rdf:type` relation.

Definition 2.1.11 (Entity Types)**.** Entity types are formalised as statements of the `rdf:type` relation where the entity is the subject and the class is the object. For instance, (`Barack_Obama`, `rdf:type`, `politician`) expresses that Barack Obama is a politician. However, entities can have multiple types, e.g. (`Arnold_Schwarzenegger`, `rdf:type`, `politician`) and (`Arnold_Schwarzenegger`, `rdf:type`, `actor`) express that the entity `Arnold_Schwarzenegger` is an instance of class `politician` and class `actor`, i.e. he is both a politician and an actor.

Similarly, RDFS provides the relation `rdfs:subclassOf` to formally define the type hierarchy.

Definition 2.1.12 (Type Hierarchy)**.** Given two classes `a` and `b`, we express that `a` is more general than `b` using the `rdfs:subclassOf`-relation: (`b`, `rdfs:subclassOf`, `a`). We call the set of all instances of the `rdfs:subclassOf` relation the *type hierarchy*.

Note that we assume the type hierarchy to be transitive, i.e. when class `a` is a sub-class of `b` and `b` is a sub-class of `c` then `a` is also an (indirect) sub-class of `c`: (`a`, `rdfs:subclassOf`, `b`) $\in \mathcal{F} \wedge$ (`b`, `subclassOf`, `c`) $\in \mathcal{F} \rightarrow$ (`a`, `subclassOf`, `c`) $\in \mathcal{F}$

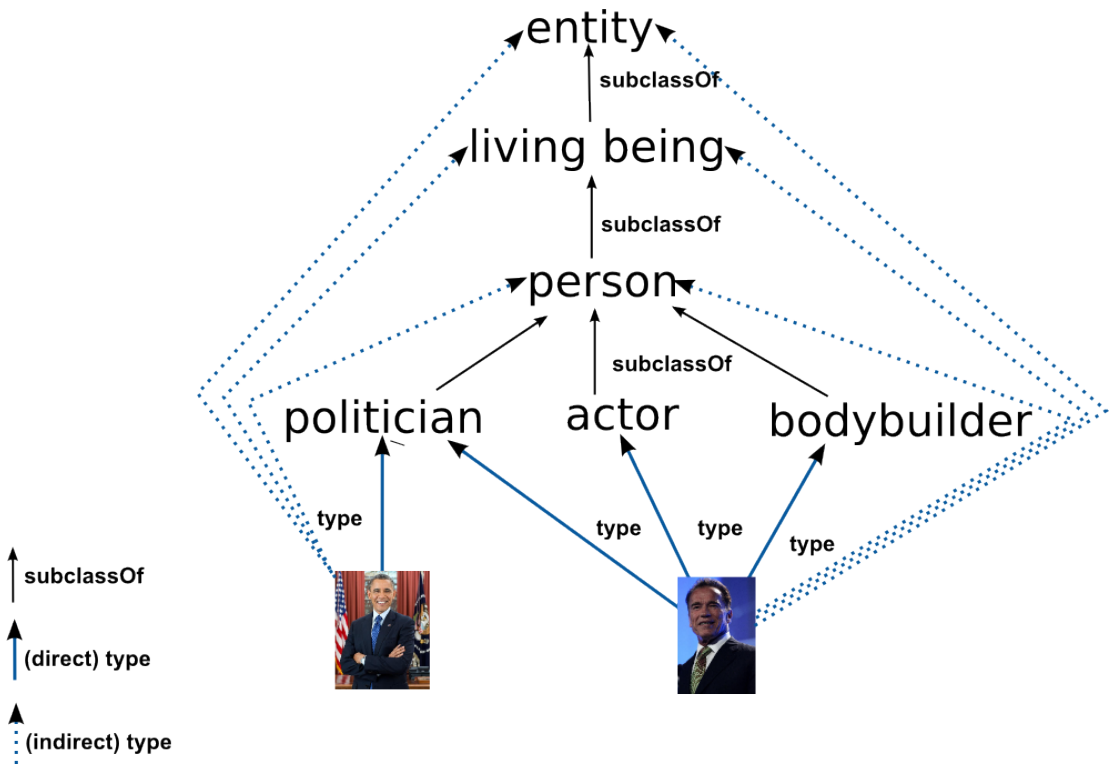


Figure 2.1.: Example: Type Hierarchy

Sources: Barack Obama from White House via Wikipedia via Flickr (public domain); Arnold Schwarzenegger from Eva Rinaldi via Flickr (commons attribution)

Beyond assuming this transitive closure of the type hierarchy, we also assume it on the `type` relation. That is, if an entity `e` is of type `a` and `a` is a sub-class of class `b` then `e` is also of type `b`: $(\texttt{e}, \texttt{rdf:type}, \texttt{a}) \in \mathcal{F} \wedge (\texttt{a}, \texttt{rdfs:subclassOf}, \texttt{b}) \in \mathcal{F} \rightarrow (\texttt{e}, \texttt{rdf:type}, \texttt{b}) \in \mathcal{F}$

Assuming `politician` to be a sub-class of `person`, this also entails `Barack_Obama` to be a `person` by transitivity. Figure 2.1 depicts the transitivity of the entity typing at our two examples. Note that we only show the necessary `subclassOf` links, although transitivity would require to also have a `subclassOf`-link from each class to the class `entity` and all classes except the `entity`-class would need to have a subclassOf link to `living being`. Note that we may still distinguish between direct and indirect entity types. Direct types are the set of classes an entity is an instance of that are the most specific. A class is more *specific* than another class if it is a sub-class of the other class. A *most specific* type of an entity, is any class a that is associated with the entity as its type while no other class being associated with that entity is a sub-class of a. In other words, the most specific and thus direct type classes are those that are leafs in the example depicted in Figure 2.1.

In the following we will typically omit to write the `rdf` and `rdfs` prefixes.

Querying an RDF Ontology A RDF ontology can conceptually be regarded as a large graph with nodes corresponding to entities and edges denoting relationships or properties, which we refer to as the ontology graph or RDF graph. Search on this kind of data is naturally expressed by means of structured graph-pattern queries using query languages such as the W3C-endorsed SPARQL [SPA] query language. While SPARQL provides a range of filtering and sorting options, within this thesis only the pattern matching capabilities are of interest. Hence, we express queries simply as a set of *parameterized statements* that are separated by a dot ('.'). For the parameterization we introduce two types of entity variables, namely *named* and *anonymous* variables, as follows.

Definition 2.1.13 (variable)**.** A *variable* is a placeholder for an entity and can replace a subject, predicate or object. We denote a variable by '`?`' (*anonymous variable*) or '`?`' followed by a variable name (*named variable*), e.g. '`?x`' represents the variable named 'x'. The set of variables is given by ϑ. Each occurrence of the anonymous variable implicitly represents a different named variable drawn from a set of infinitely many named variables different from all other named variables.

Definition 2.1.14 (parameterized statement)**.** A *parameterized statement* is a statement where some components, specifically subject, predicate, or object, have been replaced by a variable. Multiple variables can occur in the same statement and the same variable can occur multiple times. Note that the anonymous variable may occur multiple times, but represents a different named variable at each occurrence.

Definition 2.1.15 (resource assignment)**.** A *resource assignment* f is a function $\mathcal{E} \cup \vartheta \rightarrow \mathcal{E}$ that assigns each named variable $?x$ (and thus, each named variable represented by individual occurrences of an anonymous variable) an entity e and assigns each entity e' itself.

Definition 2.1.16 (ontology query)**.** An *ontology query* is a set of parameterized statements. Each variable can occur across multiple statements. We may call an ontology query simply a 'query', when it is clear that we refer to this type of query.

Definition 2.1.17 (solution)**.** Given an ontology query $q = \{t_1, ..., t_n\}$, the set of statements $\{t'_1, ..., t'_n\}$ is a *solution to* q iff there exists a resource assignment f such that it holds for each $t_i = (s_i, r_i, o_i)$ that:

1. $t'_i = (f(s_i), f(r_i), f(o_i))$
2. $t'_i \in \mathcal{F}$

So for a named variable the resource assignment replaces all occurrences of the same named variable in the query with the same entity, while for an anonymous variable each occurrence can be replaced with a different entity. Any resource assignment f that satisfies the two conditions is called *valid* with respect to q, i.e., if all statements obtained from the original set by applying the replacement induced by the assignment f are in $\mathcal{F}$. We also may call a solution to q a *query answer* or *query result* with respect to q.

In other words, a query result to a query expressed in such a way is a subgraph of the underlying knowledge graph, defined by $\mathcal{E}$ and $\mathcal{F}$, that consists of a set of statements matching the query .

For instance, the query

(?b, type, book).(?b, wasCensoredAt, ?).(?b, wasInTopTenAt, ?)

matches any entity of class book that was (to some degree) restricted in some country and was in the top-ten in some (potentially another) country.

Thus, the following would, for instance, both be a solution:

- (Harry_Potter_I, type, book).(Harry_Potter_I, wasCensoredAt, USA). (Harry_Potter_I, wasInTopTenAt, USA)
- (Harry_Potter_I, type, book).(Harry_Potter_I, wasCensoredAt, USA). (Harry_Potter_I, wasInTopTenAt, Germany)

Binary vs. n-ary relations While binary relations cover many relationships between entities, there are also some semantic relationships that naturally involve more entities. For instance, the act of giving something to someone involves three entities, the person giving, the person receiving and the object being given. Similarly, we might also want to express the role an actor played when we state that he was involved in a movie. Other relations are tightly connected to a place and date or might describe a limited time interval, e.g. a relation isMarriedTo expressing that someone is married to another person typically has a start date and potentially a termination date associated. However, we can model such relations as instances of special entities. That is, we can lift the relation instance to be expressed as an entity that has each relation argument represented by a corresponding triple. For instance, if we want to express that Arnold_Schwarzenegger played the role of the T-800 terminator in the movie The_Terminator we can create an intermediate entity about the actor's involvement in the movie that we call movieOccurrence_1 as follows.

- (movieOccurrence1, hasActor, Arnold_Schwarzenegger)
- (movieOccurrence1, hasMovie, The_Terminator)
- (movieOccurrence1, hasRole, T-800)

In addition, we could define a special class ActorRoles for this kind of n-ary statements and assign this instance to the class:

- (movieOccurrence1, type, ActorRoles)

Now, the actual entity movieOccurrence_1 is typically not of importance, as it is just a placeholder name to tie together the different individual properties[1]. This is (one reason) why RDF also allows blank nodes. If we describe the statement, we may simply replace the entity movieOccurrence_1 by such a placeholder variable:

[1]for a discussion of different ways to model n-ary relations via binary relations see also [Wor06]

- (_:movieOccurrence1, type, ActorRoles)
- (_:movieOccurrence1, hasActor, Arnold_Schwarzenegger)
- (_:movieOccurrence1, hasMovie, The_Terminator)
- (_:movieOccurrence1, hasRole, T-800)

Blank nodes start with a blank and a colon followed by a blank node name. The blank node can be used to indicate that the movieOccurrence1 entity is actually not of interest.

Reification RDF allows reification to make statements on other statements[2]. This is typically used to attach additional information to a statement, e.g. provenance information such as how it was generated or who created it. Alternatively, time constraints that limit the validity of a statement can be attached this way and it may as well be used to model n-ary relations. The latter is however discouraged, see [Wor06] for a discussion. In RDF reification requires a description of the statement to be addressed first. For instance, assume the statement (Barack_Obama, bornIn, Hawaii) is given and we would like to attach the information that this statement can be observed on the official website of the White House http://www.whitehouse.gov. In RDF this first requires to write the statement down in a reifiable form:

- (statement1, type, Statement)
- (statement1, subject, Barack_Obama)
- (statement1, predicate, bornIn)
- (statement1, object, Hawaii)

Now that we have given the statement a name, we can address it to attach further information:

- (statement1, supportedBy, ‘‘http://www.whitehouse.gov’’)

2.1.3. YAGO

The work in this thesis is based on the YAGO ontology. While the YAGO ontology model is based on RDF/RDFS, it extends the model notably by replacing triples with quadruples adding an id to each triple, which can be used as a reference to address a particular statement in another statement simplifying reification. In addition, the YAGO model allows to define relations not only as transitive but also as acyclic. This we apply to the subclassOf relation, such that the type hierarchy forms a directed acyclic graph. Furthermore we assume that there is a single root class named yago:Entity from which every other class can be reached along subclassOf links. For further details on the YAGO ontology model, see [Suc08].

[2] see also http://www.w3.org/TR/rdf-primer/#reification

YAGO Quadruples In YAGO statements are expressed via labelled triples: each triple is extended to a quadruple where the fourth component is an identifier uniquely identifying the statement[3]. For instance, the statement (`Barack_Obama`, `bornIn`, `Hawaii`) could be expressed as (`stmnt575`: `Barack_Obama`, `bornIn`, `Hawaii`), labelling it with the unique id '`stmnt575`'.

Reification By associating an id with every triple, and thus in fact using quadruples, YAGO simplifies reification. We simply write down a quadruple:

- (`stmnt1`: `Barack_Obama`, `bornIn`, `Hawaii`)

and in the following can refer to its id `stmnt1` to attach more information:

- (`stmnt2`: `stmnt1`, `supportedBy`, ‘‘`http://www.whitehouse.gov`’’)

With that in mind we further simplify our notation of reified information. Notice that both variants to realise reification make statements (or their identifier) a subset of entities on which statements can be formed, they just formalise this in different ways. With that in mind, we can simply write the statement as the subject or object entity, such that the example can be expressed in our notation as follows:

- (`(Barack_Obama, bornIn, Hawaii)`, `supportedBy`, ‘‘`http://www.whitehouse.gov`’’)

Hence, in the following, we will omit the id when denoting statements.

In addition, considering the ontology as a graph, we can also define an entity's neighbourhood as all the other nodes (entities) that are reachable via at most one edge (relation).

Definition 2.1.18 (Neighbourhood)**.** We define the neighbourhood $N(e)$ of an entity e as 1) e itself and 2) all entities that are in some relation with e, i.e. that occur in a statement with e: $N(e) = \{e\} \cup \{a \in \mathcal{E} | \exists(\mathtt{e},\ \mathtt{r},\ \mathtt{a}) \in \mathcal{F} \vee \exists(\mathtt{a},\ \mathtt{r},\ \mathtt{e}) \in \mathcal{F}\ \}$.

Relation instances vs. Triples In the following we will typically assume an ontology consisting of triple statements, i.e. $\mathcal{F}$ is a set of triples. However, occasionally we will use a relation instance based denotation, in particular when referring to statements in logical formulae. In order to simplify notation, we assume that a relation instance $\mathtt{r}(\mathtt{s}, \mathtt{o})$ represents a value of *one* or *true* iff $(\mathtt{r},\ \mathtt{s},\ \mathtt{o}) \in \mathcal{F}$.

[3]In fact, a function assigning ids is assumed [Suc08]. One way to implement this is with a hash function.

2.2. Gaining Knowledge

Statement support As discussed earlier, to gain knowledge we can only rely on observations as the only universal truth might be that we individually exist[4]. Thus, an ontology representing knowledge maintained by a computer system either has to be entirely human-made or needs to rely on "observations" made by the system maintaining it. One type of observations are written *testimonies* that indicate certain information.

Definition 2.2.1 (Testimony)**.** A testimony is an expressed belief that a particular statement is true (or false), e.g. a form of textual or verbal representation of the canonic assertion.

If a testimony indicates that a statement (`s`, `r`, `o`) holds, we say it *supports* the statement, if it negates the statement, we say it *opposes* it.

For instance, the quote *"President Barack Obama was born in Honolulu, Hawaii"* observable on Barack Obama's Wikipedia page is a (supportive) testimony for the triple (`Barack_Obama`, `bornIn`, `Hawaii`).

We can consider testimonies as a particular form of provenance information, *explaining* the belief in certain facts by referring to this form of observation. Other forms would be logical deduction or direct observation.

As there are many forms to express one's beliefs, there are many forms of testimonies. In this work we focus on two types of testimonies: 1) Textual testimonies and 2) user testimonies, modelled themselves as ontological statements. The first type is found in documents written for human consumption and the second type may be provided directly by human users that aim to improve the ontology. The understanding of type 1 testimonies requires a computer system to have some understanding of natural language or the semi-structured format used in the source documents to extract information from such sources and transform them into semantic statements. The second type requires a user interaction component that allows the user to directly communicate with the system, be it an ontology design tool or a human feedback frontend. The latter will be discussed in Chapter 3.

Information Extraction The process of automatically extracting semantic information from unstructured and semi-structured sources typically meant for human consumption is called Information Extracion (IE). Two typical types of such semantic information are canonicalized entity occurrences and relationships between entities supported by testimonies within the documents. The recognition of entity references in human readable texts has a long history [MP98, muc98, GS96, NS07, FGM05, FP10, PK00, TKS02, ope08, CMBT02a]. The named entity recognition (NER) task traditionally was defined as annotating any entity occurrence with a predefined set of (relatively rough) possible categories [GS96, Chi98], such as PERSON, ORGANIZATION or MONEY. With increasing result quality achieved in solving this task [MP98, muc98]

[4]For a computer system though, even this is doubtful as we typically deny computers the ability to think self-consciously, yet.

the categories got more complex and in recent years a trend towards identifying not the general type of entity references, but the individual entity being referenced has arisen [HYB+11, RRDA11, KSRC09, KB07, WTA10, KSRC09]. In either case, the process of resolving entity references consists of two sub-problems. First entity references need to be identified within the text (Named Entity Recognition (NER)). Second, depending on the task granularity, the entity references need to be assigned to a category or be fully *disambiguated*, i.e. the canonic entity referenced - if known to the system - needs to be identified. In our setup, we aim for an entity disambiguation to individual entities following the general framework provided by the SOFIE extraction system [SSW09].

Entity Dictionary Canonic entities usually have several non-canonical *names* that humans use to address them. For instance, the entity `Barack_Obama` can be represented by the names *"Obama"*, *"Barack Hussein Obama"*, *"the US President"*, etc.

Definition 2.2.2 (Entity Name)**.** An entity name sn is a string that addresses a named entity s. The relationship between entities and their names is an m-to-n relation, i.e., every entity can have many names and every name can refer to different entities. We address the set of entities that share the name sn by $\mathcal{E}(sn)$. RDF provides the predefined relation `rdf:label` to assign labels/names to entities, e.g. (`Barack_Obama`, `label`, *`"US President"`*). As an entity name can potentially address many entities, we may assume a given (context-independent) prior $\mathcal{E}(sn, s) \in [0, 1]$ that provides the probability that entity name sn refers to entity $s \in \mathcal{E}(sn)$. If no prior is given, then we assume the prior to be equal for all interpretations: $\mathcal{E}(sn, s) = \frac{1}{|\mathcal{E}(sn)|}$.

When an entity name is used in a textual document we say the entity is *mentioned* in the document and we call each individual occurrence of an entity name *entity reference* or *entity occurrence*.

The same entity name can in principle refer to different entities within the same document. For instance, in a biography about President Obama, the entity name *"Obama"* might in general refer to President `Barack_Obama`, but in the paragraph about his parents it might also be used to refer to his father `Barack_Obama_Sr`. However, at this point we assume that each entity name sn is uniquely addressing exactly one entity s within a given document.

Definition 2.2.3 (Entity occurrence)**.** An entity occurrence $sn_{d@pos}$ is an individual occurrence of the entity name sn in document d at position pos. The position might be given as an interval $[x, y]$ of the start (x) and stop (y) position in the character string representing the document. Given a document d, we denote all entity occurrences that appear in d by $\mathcal{E}_o(d)$. When position and document are not of importance we may abbreviate for easier reading and simply write $\tilde{s} \in \mathcal{E}_o(d)$ instead of $sn_{d@pos}$ ignoring the concrete document d and position pos.

Definition 2.2.4 (Entity mention)**.** An *entity mention*[5] sn_d is the (repeated) occurrence of an entity name sn within a document d that addresses a named entity s. Basically, an

[5] [SSW09] calls this a word-in-context (wic).

entity mention sn_d represents the fact that at least one entity occurrence with name sn appears in document d, but if there is no entity occurrence with entity name sn within d then there is also no entity mention with sn in d. Given a document d, we denote all entity mentions that appear in d by $\mathcal{E}_m(d)$.

Definition 2.2.5 (Entity Disambiguation)**.** Given an entity mention sn_d, the entity it (most likely) refers to within d is given by: $entity(sn_d)$. As there might be different candidates (the candidates being given by $\mathcal{E}(sn)$), $entity(sn_d, s) \in [0, 1]$ provides the likelihood that sn_d refers to s in d such that $(\sum_{s \in \mathcal{E}(sn)} entity(sn_d, s)) \leq 1$. The task to identify such probabilities and determine this *entity reference* function is called *entity disambiguation*. Given the assumption that the same entity name sn always refers to the same entity within a single document d, we can extend both functions easily to entity occurrences, i.e. given an entity occurrence $sn_{d@pos}$ then $entity(sn_{d@pos}) = entity(sn_d)$ and for all entity candidates s similarly $entity(sn_{d@pos}, s) = entity(sn_d, s)$.

Note that we can express an entity occurrence $sn_{d@pos}$ via statements by representing each component, the position, the document and potentially the surface string (although this is implied by the position) by respective relations `hasStartPosition`, `hasEndPosition`, `inDocument` and `hasSurfaceForm`. For instance, a pattern occurrence *"Obama"* that appears directly at the beginning of a text document $w1$ can be represented via statements as follows:

- (`entityOccurrence1`, `type`, `entity_occurrence`)
- (`entityOccurrence1`, `hasStartPosition`, `0`)
- (`entityOccurrence1`, `hasEndPosition`, `5`)
- (`entityOccurrence1`, `inDocument`, $w1$)
- (`entityOccurrence1`, `hasSurfaceString`, ‘‘`Obama`’’)

However, as the surface string and the resulting disambiguation of the entity reference is shared among all occurrences of the same entity mention in a document, the entity occurrence can be modelled by referencing to an entity mention. To this end, first we create an entity mention:

- (`entityMention1`, `type`, `entity_mention`)
- (`entityMention1`, `inDocument`, $w1$)
- (`entityMention1`, `hasSurfaceString`, ‘‘`Obama`’’)

then we can replace the information about the surface string in all associated entity occurrences by a reference to the entity mention:

- (`entityOccurrence1`, `type`, `entity_occurrence`)

- (`entityOccurrence1`, `hasStartPosition`, `0`)
- (`entityOccurrence1`, `hasEndPosition`, `5`)
- (`entityOccurrence1`, `isoccurrenceOf`, `entityMention1`)

Now, this allows us to state which entity is referenced by all occurrences of this mention with a single statement using a `disambiguatedAs` relation: (`entityMention1`, `disambiguatedAs`, `Barack_Obama`). However, for readability we will use the entity mention itself as an entity in statements. For instance, to express that the entity mention *"Obama"*$_d$ has been disambiguated as a reference to `Barack_Obama`, we may write:

(*"Obama"*$_d$, `disambiguatedAs`, `Barack_Obama`).

Note that from our assumption follows that all occurrences of `entityMention1` are also disambiguated as a reference to `Barack_Obama`. And analogously to entity mentions we simplify notation by using the entity occurrence as an entity instead of its id:

$(sn_d,$ `disambiguatedAs`, `s`$) \in \mathcal{F} \rightarrow (sn_{d@pos},$ `disambiguatedAs`, `s`$) \in \mathcal{F}$,
holds for any entity occurrence at any position *pos* in *d*.

As to how `disambiguatedAs`-statements are generated it suffices for the moment to assume that the `disambiguatedAs` relation is used to express the $entity(sn_d)$ function for all entity mentions. More details can be found in Section 3.3.5.

Relation Extraction Once entity references are identified[6], an extraction system can try to extract relational knowledge from the textual or semi-structured information available in the sources. There is a broad range of approaches, reaching from hand-crafted extraction rules potentially based on particular structure, e.g. semantic or syntactic tags or structure, via regular expressions to machine learning approaches that extract and iteratively extend their extraction capabilities semi or non-supervised [Bri99, ECD+04, CHM11, CKGS06, Sar08, SSW09, NTW11, EBSW08, LKR+08]. For this work we focus on the family of pattern based extraction approaches. By this we understand any extraction approach that is based on recognising recurring textual and/or structural patterns that reflect a certain relation. That is, for each relation r there is a set of patterns $P(r)$ that represent different surface forms of r and for each $p \in P(r)$ there is a weight $conf(p, r)$ reflecting the confidence that an arbitrary instance of p with entity references s, o actually expresses (`s`, `r`, `o`). These confidences might be fixed or can be learnt over time.

While we assume in the following examples and discussion, for simplicity, that the patterns we deal with are simple textual phrase patterns, we make in general no assumptions on the way a pattern is defined. A pattern could, for example, be represented by a textual phrase, an n-gram, or a regular expression. While many approaches limit themselves

[6] In principle relations can also be extracted based on non-canonicalized entity references, but we focus on the fully canonic case.

to binary relations, there are also approaches that try to extract more complex relations or aim to extract binary relation instances along with temporal or geospatial information [LW10, KW12, HSBW13, KKK+11]. However, for this work, we limit the relation extraction to binary relations (while many of our concepts can easily be extended to n-ary relations) and follow the extraction approach from [SSW09].

Definition 2.2.6 (pattern)**.** A pattern is a textual or structural phrase that occurs repeatedly in a document corpus together with two entity references. We write "X p Y" to denote a pattern represented by a string p that occurs with the entity references X and Y. Note that X and Y do not belong to the actual pattern string, they are placeholders for entity references. The set of all patterns is given by $\mathcal{P}$.

For instance, the textual phrase *" was born in "* corresponds to the binary pattern "X was born in Y" indicating the `bornIn` relation, such that an instance that instantiates X and Y represents the information that the entity represented by X was born in a place represented by Y.

Similar to entity occurrences, concrete occurrences of patterns at a given place in a document are called *pattern occurrences*.

Definition 2.2.7 (pattern occurrence)**.** We write $p_{d@pos}(\tilde{s}, \tilde{o})$ to refer to a pattern occurrence that appears at position *pos* in document d with entity occurrences $\tilde{s}$ and $\tilde{o}$. The position *pos* of the pattern occurrence is given by a set of intervals in the document's character string. This also allows for more complex patterns, although we only consider patterns that stretch without interruption from one entity to the other in this work. We may omit the document and position if it is irrelevant and simply use $\tilde{p}$ as an abbreviation of $p_{d@pos}$ instead. The set of all pattern occurrences is given by $\mathcal{PO}$. Given a document d all pattern occurrences in d are given by $\mathcal{PO}(d)$.

Definition 2.2.8 (pattern instance)**.** Given a pattern p and two entities s, o we say the *pattern instance* $p(s, o)$ exists iff there exists *some* pattern occurrence $\tilde{p}(\tilde{s}, \tilde{o}) \in \mathcal{PO}$ such that $\tilde{s}$ and $\tilde{o}$ have been disambiguated as s ($entity(\tilde{s}) = s$) and o ($entity(\tilde{o}) = o$) respectively. Given a document d, all pattern instances that occur in d are given by $\mathcal{PI}(d)$, $\mathcal{PI}$ is the set of all pattern instances and $\mathcal{PI}(p)$ are all pattern instances of a pattern p.

While pattern instances are representing all variants to combine a pattern with entities we call the analogous set of all combinations of patterns with entity names the set of *pattern variants*.

Definition 2.2.9 (pattern variant)**.** A combination of a pattern p with two entity names sn, on independent of any document is called *pattern variant*. The set of all pattern variants is given by $\mathcal{PV}$. If there exists a pattern occurrence $\tilde{p}(\tilde{s}, \tilde{o}) \in \mathcal{PO}(d)$ for a document d we say pattern variant $p(sn, on)$ occurs in d. Given two entities s, o and a pattern p we define $\mathcal{PV}$(p,s,o) as the set of all pattern variants of p where $s \in \mathcal{E}(sn)$ and $o \in \mathcal{E}(on)$, i.e. all possible combinations of entity names for s and o with pattern p.

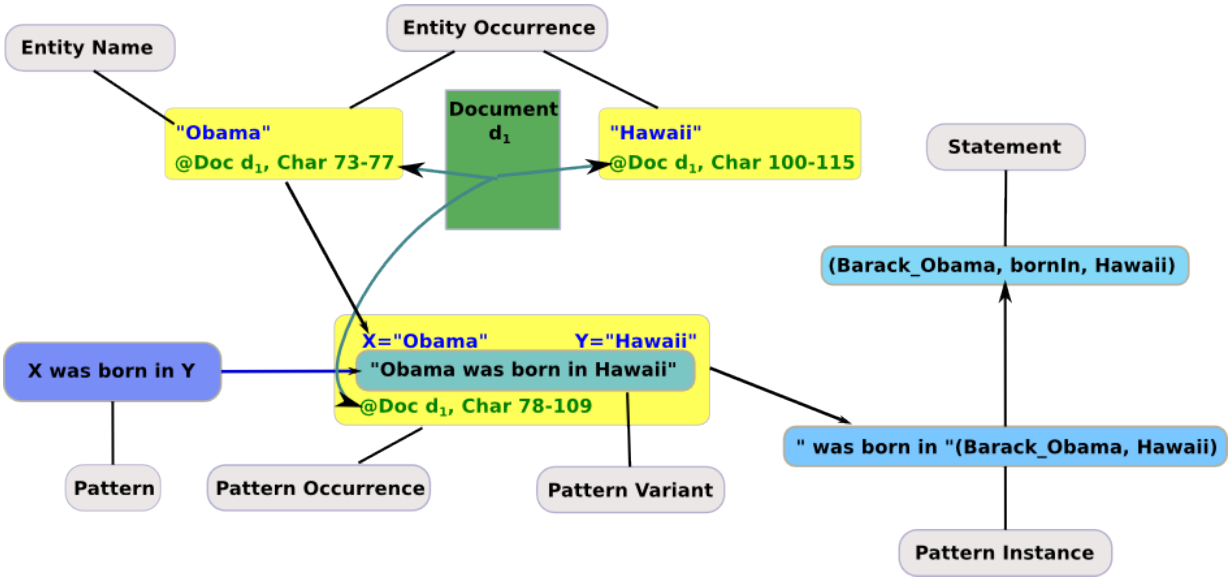


Figure 2.2.: Pattern Model

Analogously to entity mentions we also consider pattern mentions as an aggregation per document over pattern occurrences.

Definition 2.2.10 (pattern mention)**.** If a document contains any pattern occurrence $\tilde{p}(\tilde{s}, \tilde{o})$ of a pattern p we say d contains a *pattern mention* $p_d(sn_d, sn_d)$. The set of all pattern mentions is given by $\mathcal{PM}$.

Basically a pattern occurrence represents an individual appearance of a pattern at a particular position *pos* in a particular document d with two entity occurrences, a pattern mention aggregates over the pattern occurrences in a single document d, a pattern variant represents a possible combination of a pattern with 2 entity names without referencing to a particular occurrence and a pattern instance $p(s, o)$, finally, indicates that a pattern p occurs in some document with two entity occurrences $\tilde{s}$ and $\tilde{o}$ that are disambiguated as references to s and o.

Any pattern can indicate multiple relations with varying certainty. For instance, the phrase *"Obama's home country Kenya"* could refer to the fact that Barack Obama has family roots in Kenya (`Barack_Obama`, `hasAncestorsFrom`, `Kenya`) or it could indicate that he was born in Kenya (`Barack_Obama`, `bornIn`, `Kenya`).

Thus, for any relation r, that can be expressed by pattern p, a confidence value $conf(p, r)$ representing the likelihood that p expresses r is given.

Definition 2.2.11 (Pattern Confidence)**.** Assume two entity occurrences $\tilde{s}$ and $\tilde{o}$, a pattern p, and a relation r are given. The confidence $conf(p, r) \leq 1$ represents the likelihood that, given a pattern occurrence $\tilde{p}(\tilde{s}, \tilde{o})$, the author wanted to express the relation $r(entity(\tilde{s}), entity(\tilde{o}))$. We say p indicates r ($indicates(p, r)$) if and only if $conf(p, r) > \sigma$.

Figure 2.2 illustrates the pattern model on an example. Consider, for instance, the phrase *"Obama was born in Hawaii"* occurring in document d_1. In the example this would be a pattern occurrence at position 78-91 of pattern p ="X was born in Y" with X being replaced by an entity occurrence with surface string "Obama" and Y being replaced by an entity occurrence with surface string "Hawaii". Hence this would represent an occurrence of the pattern variant p(*"Obama"*,*"Hawaii"*). Given that the entity occurrences are being disambiguated as references to the entities `Barack_Obama` and `Hawaii` the pattern occurrence represents a pattern instance p(`Barack_Obama`, `Hawaii`). Assuming that the pattern indicates the relation `bornIn`, it is a supportive testimony of the statement (`Barack_Obama`, `bornIn`, `Hawaii`).

Similarly to entity occurrences, a pattern occurrence $p_{d@pos}(\tilde{s}, \tilde{o})$ can be expressed via statements by representing each component, the position, the document, the entity references involved and potentially the surface string (although this is implied by the position) by respective relations `hasStartPosition`, `hasEndPosition`, `inDocument`, `hasArgument1`, `hasArgument2` and `hasSurfaceForm`. For instance, a pattern occurrence from the phase *"Obama was born in Hawaii"* that appears directly at the beginning of a text document $w1$ can be represented via statements as follows:

- (`patternOccurrence1`, `type`, `pattern_occurrence`)
- (`patternOccurrence1`, `hasStartPosition`, `7`)
- (`patternOccurrence1`, `hasEndPosition`, `17`)
- (`patternOccurrence1`, `inDocument`, $w1$)
- (`patternOccurrence1`, `hasArgument1`, `entityOccurrence1`)
- (`patternOccurrence1`, `hasArgument2`, `entityOccurrence2`)
- (`patternOccurrence1`, `hasSurfaceString`, `"was born in"`)

where `entityOccurrence1` and `entityOccurrence2` refer to entity occurrences described via statements as described earlier. Note that, analogously to entity occurrences and entity mentions, we may denote a reference to a pattern occurrence in a statement by writing down the pattern occurrence itself instead of its identifier.

Now that we know how to identify textual testimonies of a statement, we can also identify the witnesses that provide these testimonies, which are the documents the testimonies appear in.

Definition 2.2.12 (witness)**.** For a statement t =(`s`, `r`, `o`) any document d with a pattern mention $p_d(sn_d, on_d)$ where p indicates r, ($\tilde{s}$, `disambiguatedAs`, `s`) $\in \mathcal{F}$ and ($\tilde{o}$, `disambiguatedAs`, `o`) $\in \mathcal{F}$ is called a *witness* for t. That is, a witness for statement t is a document in which a testimony for t in the form of a pattern occurrence exists. The set of all documents in the corpus is given by $\mathcal{W}$and the set of all witnesses for a statement t is given by $\mathcal{W}(t)$.

While there are several approaches that are known as pattern based approaches [SSW09, NTW11, Blo10, CL01, SIW06, ECD+04, AG00] we can also cast other extraction approaches into this model. For instance, an extraction approach based on hand-crafted or automatically learnt regular expressions like [LKR+08] can also be seen as a pattern based approach, where each regular expression is a pattern for a relation.

Given patterns for relations of interest, the actual extraction is a two-fold process. First, the system analyses documents and extracts *statement candidates* from the text by identifying testimonies in the form of pattern occurrences. Second, it needs to decide whether to accept the candidates and if so integrate any new statements derived from surface testimonies into the knowledge base. In addition, every time new statements are added, they can be used as axiomatic knowledge to derive further information through inference. Once statements accepted as ontological facts have been extracted, they are stored in a fact repository or a knowledge base.

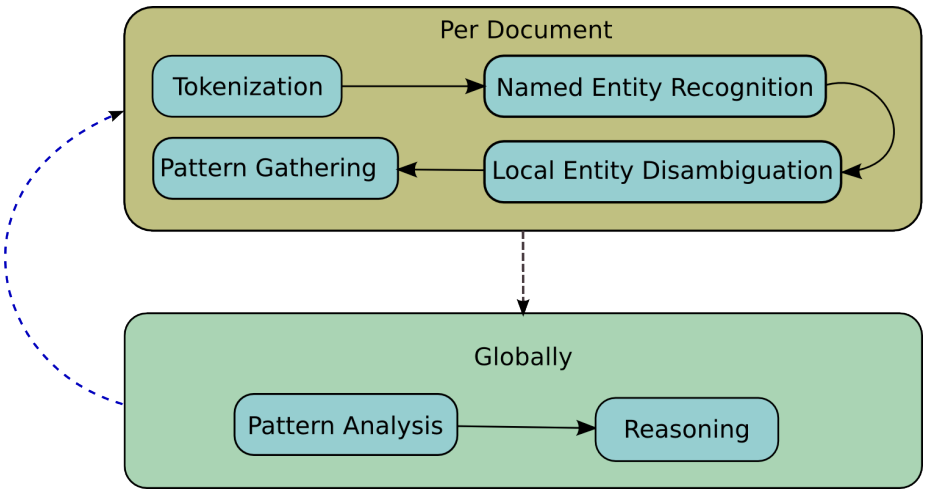


Figure 2.3.: Extraction Stages

Extraction System In our setup we employ a pattern based extraction system following the SOFIE [SSW09] and PROSPERA [NTW11] framework for extraction of binary relations. Basically it consists of six stages that can be separated into a local per document and a global part (see Figure 2.3). First, the source document is tokenized. Second, entity references need to be identified and disambiguated. In contrast to the original framework, our modified version stores all entity references with the entity assigned to each reference and the exact position within the source. Then all pairwise combinations of entities within the same sentence are considered (or rather all pairs within a token sequence where token sequences are separated by predefined separator tokens, e.g. punctuation or certain HTML tags). For each such combination the intermediate tokens make up a pattern occurrence with the two entity occurrences. All such pattern occurrences are collected. Optionally, a statistical analysis matches the resulting pattern instances against the knowledge base and tries to predict which pattern represents which relation. Finally, a reasoning step jointly

solves ambiguous entity references that could not be disambiguated clearly, decides which patterns to accept as representation for which relations and which relation instances to accept. For more details on the individual steps see Section 3.3.

Canonic Relations vs. Pattern Clusters In this work, we assume that our framework is working with named (canonic) relations originating from information extraction tools working with predefined canonic relations, i.e., the system is configured to look for patterns for a set of given relations, e.g., `bornIn`. Some systems [EBSW08, BE08, NWS12] extract "interesting" patterns from a document corpus without naming them and try to cluster semantically similar patterns. Each obtained cluster represents an anonymous relation. We can easily extend the framework to work with anonymous relations defined by clusters of similar patterns. For a cluster the confidence of each pattern can be defined by the distance to the cluster center.

2.3. Information Retrieval

Information Retrieval (IR) is a relatively broad field that deals with different types of information (e.g. text documents, semantic entities, images, videos etc.) and different problems (e.g. searching, organisation, and storage) [MRS08]. As the name says, the core task in IR is to retrieve some form of information given a query q of some form and a pool of information objects D (of documents, entities etc.) to draw from. While keyword queries are the most prevalent form of queries, other forms, like structured queries, e.g. via SQL or SPARQL, or example queries that allow to search for more information of the same kind by providing an example are possible as well. The outcome of an information retrieval task is typically a list of results ranked by their assumed *relevance* to the query.

Definition 2.3.1 (Result ranking)**.** Given a query q and an information object pool D, a *result ranking* is a list $R =< r_1, ..., r_n >$ of elements r_i from D. We say n is the length of R, and assuming no element appears multiple times, we may denote n by $|R|$. Sometimes, we may wish to make explicit that a ranking has a certain length k, then we may write R_k instead of R. For instance, we may investigate a given ranking R at various lengths, then R_k addresses the sub-ranking $< r_1, ..., r_k >$ of the first k elements of R. The i-th element r_i in R may also be addressed by $R(i)$ for convenience.

Since an IR-system can only estimate the relevance of results to the query, one of the key points of interest to judge the quality of an IR-system is the quality of the result ranking. This is typically evaluated by manually assessing the relevance of potential results to the given query and then applying an quality measure. We will discuss several such measures in the following.

Evaluation Measures We will now discuss several evaluation measures suitable for the common ranked list retrieval task. For additional background, see [MRS08, JK02]. For the following, assume a query q from a set of queries Q has been processed and a ranking R of n

result entities from D, be that semantic entities from an ontology or, e.g., documents, has been produced. Also assume there are human judgements for each result $r \in R$ asserting the relevance to the query on a graded ($rel_g(r)$) and on a binary ($rel_b(r)$) scale. Further assume that all the relevant entities (according to the binary judgements) are given by $Rel \subseteq D$.

Definition 2.3.2 (precision)**.** The *precision* $prec(k)$ of the ranking R at length k is measured by the fraction of relevant entities over the length of the ranking:

$$\frac{|R_k \cap Rel|}{|R_k|} \tag{2.1}$$

Definition 2.3.3 (recall)**.** The *recall* $rec(k)$ of a ranking R at rank k is measured by the number of relevant entities in the ranking over all relevant entities to be retrieved:

$$\frac{|R_k \cap Rel|}{|Rel|} \tag{2.2}$$

Definition 2.3.4 (average precision (ap))**.** The *average precision (ap)* $ap_q(k)$ of a result ranking R for query q at rank k is the sum over all precision values measured at any rank where a relevant entity has been returned divided by the number of overall relevant entities:

$$\frac{\sum_{i=1}^{k} prec(i) \cdot rel_b(R_k(i))}{|Rel|} \tag{2.3}$$

Definition 2.3.5 (mean average precision (MAP))**.** Based on the average-precision the mean average precision (MAP) at rank k is the average over the ap-values for all queries considered:

$$\frac{\sum_{q \in Q} ap_q(k)}{|Q|} \tag{2.4}$$

Definition 2.3.6 (recall-precision (r-prec))**.** The *recall-precision* (r-prec) is the precision at the rank that equates the number of relevant entities:

$$prec(|Rel|) \tag{2.5}$$

Definition 2.3.7 (mean satisfaction rank (msr))**.** The *satisfaction rank* (satrank) $sr_q(k)$ of a ranking R for a query q at length k is the first rank smaller or equal to k at which a user would be satisfied (or $k+1$ if no such entry $i \leq k$ exists). Typically a result is considered satisfactory, if and only if it is relevant. The *mean satisfaction rank* (msr) then is the average over all queries in Q:

$$\frac{\sum_{q \in Q} sr_q(k)}{|Q|} \tag{2.6}$$

Definition 2.3.8 (mean reciprocal rank (mrr))**.** Based on the satisfaction rank the reciprocal rank(rr) $rr_q(k)$ is the inverse, i.e. $\frac{1}{sr}$, while in cases where the ranking of size k does not contain a satisfactory entry up to k we define rr to be 0. To compute the mean-reciprocal rank (mrr), we average over the reciprocal rank (rr) of all queries considered.

$$\frac{\sum_{q \in Q} rr_q(k)}{|Q|} \tag{2.7}$$

Definition 2.3.9 (normalized discumulative gain (nDCG))**.** The discounted cumulative gain (dcg) measures the information gain for a ranking of length n by summing up the relevance rel_i of the results at each rank i before k, however, the contribution is discounted on a logarithmic scale:

$$DCG_{q,k} := rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2(i)} \tag{2.8}$$

To compute the nDCG a ranking's dcg value is normalized by the dcg of a perfect ranking (idcg):

$$nDCG_{q,k} := \frac{DCG_k}{iDCG_k} \tag{2.9}$$

The mean version of nDCG is computed by averaging over the set of queries Q:

$$mnDCG_k := \frac{\sum_{q \in Q} nDCG_{q,k}}{|Q|} \tag{2.10}$$

3. User-Supported Extraction

3.1. Motivation

While manual ontology generation and document annotation is very resource intense, the aforementioned methods to automatically generate ontologies are much more efficient and achieve relative good precision - at least on some relations [SSW09]. However, the automatic approaches are still prone to make mistakes and the more precise approaches typically lack good per-document recall. There are several challenges that automatic extraction systems need to overcome. First, while advances are made, the problem of natural language processing itself is considered *AI complete* [Sha92, Fra96] and cannot be considered solved [Bat95, JM00]. Second, many methods need basic domain data, as, for instance, the entities of the domain, their names and potentially relation examples may have to be provided. Third, a resulting ontology or document annotation based on recognised semantic information might be incomplete, e.g. due to mistakes, missing information in the sources, too strict consistency requirements and/or faulty assumptions, for instance, on properties of relations. Humans might disagree with the fact acceptance strategy employed by the extraction system. Depending on corpus and extraction strategy the resulting ontology might contain only one view on a particular topic or include rather unlikely claims, e.g. due to misunderstandings of satirical text or extractions from esoteric sources. While humans might also make mistakes or disagree, some use-cases require the ability to oversee results efficiently by a human assessor. Such a human oversight of results however, requires efficient tools that help a user assessing the legitimacy of a particular information piece that has been included into the ontology.

So, to summarise, automatic extraction methods can greatly support humans in generating huge knowledge collections, but they may need human support. Human expert resources on the other hand are scarce and thus we need to ensure a human maintainer or user of an ontology can spend her time as efficient as possible, hence she needs automatic methods that support her in verifying ontological knowledge and in navigating the ontology.

In this work, we present a framework that combines automatic extraction methods with user based extraction to improve the overall extraction quality and per-document recall and to enable a simple way to apply extraction methods to new domains by training the system interactively.

Most automatic pattern based extraction systems follow an iterative approach, in order to apply knowledge learnt from some sources in one iteration to other sources in the next

iteration and vice versa. In this work, we insert user-feedback as another learning step in-between extraction iterations. This has benefits for the user and the system. In particular, in contrast to annotating the corpus from hand, a user providing feedback, needs only to touch problematic knowledge that she finds need inspection and potentially correction, while she might expect the extraction system to learn from feedback and apply it in the next iteration. The system on the other hand can learn from any feedback and apply this new extraction knowledge in the next iteration on all source documents, thus decreasing the probability to draw too many faulty conclusions from early mistakes.

While the saying *"The customer is always right"* is often also applied to users (*"The user is always right"*), in some cases different users may hold different opinions on the same matter, which might lead to some predicament if we follow each user's opinion absolutely. One example where this becomes apparent in the supposedly neutral field of knowledge are Wikipedia *edit wars*[1]. Hence, we cannot always assume a user's feedback as the absolute truth, but should consider the extracted ontology merely as a collection of substantiated beliefs that might include contradictions keeping track of supporting testimonies, textual or by users, for statements included.

3.1.1. Use-Cases

From a user's perspective, we consider three different use-case scenarios in how she is interacting with the extraction system.

Task-oriented In this setting, a user is interested in extracting all information from a given set of source documents. For example, the user might want to analyse and annotate the documents or she might consider the fact extraction process as a form of indexing the documents, e.g., to generate a fact based summary of the documents. This might, for instance, be used for journalistic purposes, in order to support an article backed by a large document corpus, like the Wikileaks publications. Another application scenario are the social sciences where large corpora of text are annotated, for instance, for comparative investigation of the documents themselves or to support researchers in finding sources to specific topics. In each case, the user would provide a set of documents and would then want to make sure that the extractions from all documents are correct.

Explorative In an explorative setting, an ontology has already been built and the user is exploring the ontology or the document corpus it is based on. The user might either be interested in improving the ontology, verify certain extracted facts or be actually using the ontology for other purposes and stumble across mistakes while doing so. In either case, she might correct such mistakes then individually. The difference to a task oriented approach is that the corrections happen as a side-effect or at least without a particular focus on a fixed document set.

[1] for a humorous overview see `http://en.wikipedia.org/wiki/Wikipedia:Lamest_edit_wars` and for an analysis of the most controversial topics based on edit wars see [YSGK13]

System-Driven In this setting the user is merely an oracle for the extraction system assisting it in decisions that seem to difficult to decide for the extraction system. In contrast to the other two cases the user has no interest other than supporting the extraction system to maximise the quality improvement his assistance has on the overall ontology. Such an approach would require a model estimating the impact of corrections to guide the user to those pieces of knowledge that provide the extraction system with the most additional insight or otherwise have the most impact on the ontology or its particular application, e.g. in an online ontology that is used by a large user group, those entries that are used the most often might be more important than those that never are looked at. Note that this work does not provide such a feedback impact model, but there is existing work in this area [DSJS10].

Our work, and this chapter in particular, is mainly concerned with the task-oriented and explorative (see also Chapter 4 and 5) use-cases. One of the difficulties when aiming to extract information from a document corpus of a particular domain is the requirement of prior domain knowledge the extraction system needs to function. A system such as SOFIE [SSW09] needs to know all entities of the domain, their names, their types (along with the type class hierarchy) and example relation instances for all relations to be extracted.We will show that some of these requirements can be loosened by collecting this information through user feedback. This lowers the initial effort to setup an extraction system and thus, we argue, makes it more user-friendly as users tend to be more likely to provide information of it is directly related to their main task of interest. While we are mainly interested in the task-oriented and explorative use-cases, in all three cases there are the same types of feedback a user can provide. We will discuss these types of feedback in Section 3.4 and how we can integrate each kind of feedback into the extraction process in Section 3.5, but before that we shall discuss the extraction system in a bit more detail in Section 3.3. In Section 3.7 we will discuss a particular task-oriented application scenario that arose from the WisNetGrid project. This also contains the presentation of a user frontend system based on our feedback integrating extraction system in Section 3.7.2. First however, related work is discussed in the next section.

3.1.2. Contributions

We extend an iterative extraction framework that is general enough to cope with a family of extraction approaches by a user feedback integration step that seamlessly integrates with the extraction iterations and integrates the user feedback in the form of ontological information into the extracted ontology. Thus, usage of these user testimonies is left to the extraction system. However, we also provide an adaption of an existing information system to integrate this user feedback into its approach in order to support the correction of extracted information, the addition of not (properly) identified information and the integrated learning of necessary domain knowledge in a way that engages the user by directly involving her in the process such that the initial effort to employ the system is lowered.

3.2. Related Work

There exists a broad range of approaches to visualize and edit semantic information [TNFM11, KPS+05, owl, kno, LD08, FL04, SMMS02, MFRW00a, MFRW00b, FFR97, vit, LCC+11, OVBD06]. However, traditional tools to model ontologies, e.g. WebProtégé [TNFM11, Dom98], lack the capability to link information extracted from documents back to their sources. Thus, human input is only directed directly at the ontology itself, without a) support for the user in assessing the validity of a relation instance, and b) without integration of the human feedback process with an extraction system. Similarly, resulting ontologies created by human efforts do often not have any source or other provenance information associated with single facts [GL90, ocy, NP01]. Most large scale ontologies based on automatic extraction methods, however, either explicitly track the sources of their facts and make them available (in newer versions) or at least can identify them if needed [HSBW13, ABK+07, CBK+10, EFC+11], but they typically also do not explicitly harvest user-feedback taking the sources into account. Some of these editors are, however, are web-based [vit, LCC+11, KPS+05, kno, FFR97], and some allow collaboration, either by providing methods to access and search shared ontology repositories [CC06] or to work online together on the same ontology [FFR97, LD08]. However, these works are all concerned with human driven ontology generation and maintenance, but not with extraction processes.

On the other hand there are various extraction tools and frameworks [Bri99, ECD+04, CHM11, FL04, CKGS06, Sar08, SSW09, NTW11, EBSW08, LKR+08, CMBT02b, Ade98, LRNdS02] and some user oriented frontends that allow to influence the extration process [CMBT02b, Ade98, LRNdS02], e.g. by intervention at a particular extraction step using GATE [CMBT02b] allowing user interaction by modeling the extraction process as a workflow where each step produces some sort of document representation, e.g. a token stream, and this representation can potentially be modified before being fed into the next step of an extraction pipeline. While manually achievable, an iterative loop seems not to be explicitly addressed by these frameworks. Also, those are typically desktop-applications not supporting to manage a shared extraction-driven ontology project. In order to collaborate, one typically needs to export an ontology, then give it to a collaborating partner, and import the ontology in the local tool.

Independently, since their first installment in 1995 by Ward Cunningham[2], *wikis* [LC01] drove a move to human collaboration in (human oriented) knowledge collection and management with Wikipedia [wikb] as the most prominent example of a very general knowledge collection. In recent years, wikis have been extended with semantic annotation techniques in order to allow a human-driven collection of machine-readable facts alongside the human oriented knowledge representation prevalent in wikis [VK06, Sch06, BGE+08, SS09a, MTS+06, SMMC09, OVBD06]. Semantic Wikis have a stronger focus on user driven data generation, yet by themselves, they typically lack information extraction capabalities to support users in the annotation process. While they allow to annotate documents as sources

[2] http://www.wiki.org/wiki.cgi?WhatIsWiki

for factual knowledge when writing them, it is limited to documents in such wiki-formats and requires annotation by the human author(s) of the document. Still, there has been work on integrating existing ontologies and reasoning capabilities as a background model into semantic wiki systems[VK06] and on allowing distributed collaboration on the same wiki-based ontology based on a peer-to-peer network [SMMC09]. In principle a semantic wiki could be used as a frontend for collecting user-feedback in collaboration with our feedback integration approach.

In an approach more oriented towards collecting general facts a sister project to Wikipedia called *Wikidata* has been founded in October,2012. It aims at creating a large general purpose ontology "that can be read by humans and machines alike", but also allows to provide source information for semantic statements added [wika, VKV+06]. In the spirit of Wikipedia, which requires citations for claims in its articles[3],the project aims to annotate ontological statements with source documents, yet not with the actual phrases that support the statement. While it is likely that some of the 15 million semantic items created within about a year until the 30th September 2013 are automatically generated from Wikipedia's data, to the best of our knowledge, there is no general extraction system integrated to support users in providing these links.

[EHL+08] also suggests an interactive extraction system, but the interaction is limited to selecting a topic which the extraction system uses to identify a set of documents it will then analyse for potential textual relations, i.e. patterns with several instances, that are presented to the user which may in turn provide a new topic, the extraction itself is unsupervised, so the extraction system is not learning from the user, but training itself for the user specified topic. The WALU annotation tool presented in [WR07] allows to review extraction results from a NER system, yet entity references seem only be assigned to a set of categories rather than to individual entities. In addition it seems the interaction frontend does not support statement annotations (the paper cites however a co-occurrence and context analysis[4] which can be corrected by users via a wiki). While [DSJS10] aims at the system-driven feedback use-case by providing an analysis method to identify those facts in an ontology that have the most impact when changed, the most similar vision of an iterative learning process is probably shared by the KiWi project[5]. It *envisions* an integration of information extraction and knowledge management with Wiki technology [SS09a].

[3]How strictly this policy is and can be enforced is another question and often debated:`http://www.welt.de/wirtschaft/webwelt/article2743585/Warum-Heilmanns-Wikipedia-Kritik-berechtigt-ist.html`

[4]"Hierfür werden ... Named Entities ... erkannt und klassifiziert... werden diese Entitäten mit semi-automatischen Methoden zu semantischen Netzen verknüpft, indem ihre Kookkurenzen und ... Kontexte ... analysiert werden." [WR07]

[5]http://www.kiwi-project.eu/

3.3. Extraction System Stages

Remember the general extraction stages discussed in Section 2.2 summarized in Figure 2.3. In this section, additional details on each stage are provided. Based on this we can discuss how feedback can be integrated in Section 3.5. In general, the extraction system we employ is based on the framework suggested in [SSW09] with an additional component similar to the statistical analysis extension presented in [NTW11]. Given a document, first separate tokens and entity references are identified, then these references are disambiguated to individual known entities (if possible) and finally the entity references are used to generate phrase pattern candidates. This provides the basis for a statistical analysis of the textual patterns, which in turn generates suggestions linking phrase patterns with relations. Based on this a reasoning step decides which patterns to accept as representations of which relations, which entity disambiguations to accept as correct and which relation instances to consider supported strongly enough to incorporate into the ontological fact corpus. Once the system has iterated through one such cycle it will re-evaluate its statistical pattern analysis also taking into account the statements extracted in the last iteration. Thus, more patterns will be recognized as expressions of a relation and more statements can be extracted. This cycle will be repeated for a fixed number of iterations or until no further statements can be extracted. In the following we discuss each step.

3.3.1. Entity Recognition

The tokenization and entity recognition are mostly identical with the methods used in the system portrayed in [SSW09]. Tokenization is based on a set of string matching rules, that filter markup tags like HTML tags and identify several types of literals, like dates and measurements based on a number and date parser[6]. Named entities are recognized based on an upper-case heuristic, i.e. words starting in upper-case are considered as an entity and, to filter out false positives, are matched against a list of common words.

3.3.2. Entity Disambiguation

Any recognized named entity mention sn_d is in [SSW09] disambiguated using the global prior for the name-entity interpretation and a context-dependent bag of words model [MRS08], i.e. the document is considered as a bag-of-words which is matched against the bag of words based on the semantic neighbourhood of a candidate entity $s \in \mathcal{E}(sn)$, this results in $entity_{bag}(sn_d, s)$, providing a likelihood that the entity s is the one referenced by mention sn_d. In particular, consider $T(d)$ to be the list of all the words of a given document d and let $NT(s)$ be the set of words representing the semantic context of the entity, that is all the words that occur in the neighbourhood $N(s)$ (see Definition 2.1.18) of an entity s. In particular we generate $NT(s)$ by collecting all entities from $N(s)$ and then from each entity's URI collect all individual words, where we ignore any possible prefix and

[6] these parser implementations can be found in the Javatools package from the YAGO-NAGA project available here: `http://www.mpi-inf.mpg.de/yago-naga/javatools/`

stopwords. For instance, assume that `yago:United_States_of_America` $\in N(s)$ then $NT(s)$ would contain the words "united", "states" and "america". As we assume the ontology to provide entity names, we may alternatively use a neighbouring entity's names instead of or in addition to its URI to generate its contribution of context words. In YAGO2 [HSBW13] a rich resource for context information are the type classes, in particular the relatively specific wikicategory based classes, which have no names associated. Hence, our tendency to use the URI.

An alternative approach applied by disambiguation systems (e.g. [HYB+11]) is to use a precomputed set of context keywords or context phrases generated from documents concerned mainly with the entity, e.g. its Wikipedia article, in order to directly associate a keyword context with each entity instead of considering the neighbouring entities. Yet, this requires that such a set of keywords is available for all entities, which is an additional difficulty when adapting to a new domain where entities until then unknown to the ontology might be added.

Given $T(d)$ and $NT(s)$, for each candidate entity s we first compute the intersection of both sets $T(d) \cap NT(s)$ and then normalize this over all candidates, such that $entity_{bag}(sn_d, s)$ is given by:

$$entity_{bag}(sn_d, s) := \frac{|T(d) \cap NT(s)|}{\sum_{s' \in \mathcal{E}(sn_d)} |T(d) \cap NT(s')|} \tag{3.1}$$

We use this bag-of-words model as a local component based just on the entity candidates and the document ignoring all other extraction decisions, but extend the original method by an iterative approach where the bag of words based likelihood $entity_{bag}(sn_d, s)$ is used as a fixed component and the coherence between all entity references $entity^i_{coh}(sn_d, s)$ is used as a second iteratively recomputed component that also takes into account the disambiguation decisions of the other entity occurrences in the document. That is, we define

$$entity^0(s) := \mathcal{E}(sn, s) \cdot entity_{bag}(sn_d, s) \tag{3.2}$$

and for all following iterations $i > 0$, $entity^i(s)$ is recomputed as

$$entity^i(s) := \alpha \cdot entity^0(s) + (1 - \alpha) entity^i_{coh}(sn_d, s) \tag{3.3}$$

The coherence component is aimed at estimating the likelihood of a particular interpretation of an entity reference based on the coherence with the interpretation of the other entity references in the same document, thus we define

$$entity^i_{coh}(sn_d, s) := \frac{\sum_{o \in N(s)} occ^i(o, d)}{|\mathcal{E}_o(d)|} \tag{3.4}$$

where $occ^i(o, d)$ refers to the potential occurrences of entity o in document d each weighed by the confidence in the last iteration $i - 1$:

$$occ^i(o, d) = \sum_{on_{d@pos} \in \mathcal{E}_o(d)} entity^{i-1}(o) \tag{3.5}$$

In-between each iteration we fix all disambiguations that reach a predefined threshold t_{minfix}, i.e. once $entity^i(s) > t_{minfix}$ for some candidate resolution s, we fix this mention to s, i.e. assume from now on $entity^j(s) = 1$ for s and $entity^j(s') = 0$ for all $s' \neq s$ and any $j > i$. Remember that the neighbourhood $N(s)$ of an entity s also includes s. Thus, not only related entities of s make it a more likely candidate, but also other (potential) references to s within the same document. After a fixed number of iterations (or when all entity mentions are fully resolved) the disambiguation is stopped and mentions not fully resolved are kept for the reasoning step, which takes into consideration not only local entity disambiguation decisions, but all extraction decisions across all documents (see Section 3.3.5).

The inclusion of a coherence component serves two purposes. First, it introduces a general quality improvement as the disambiguation task is seen as a combined problem where all occurrences in a document are solved jointly instead of independently. Second, it strengthens the effects from feedback on entity occurrences. If some entity occurrences are "corrected" by human users, a coherence measure can use this feedback to also affect other disambiguations, while the bag-of-words model would be ignorant to this. For more on advanced disambiguation methods, see [HYB+11].

3.3.3. Pattern Collection

Given the token list representing the document, pattern occurrences are collected by considering all pairwise combinations of entity references within each sentence (or otherwise separated token sequence). For each pair of entity references, the tokens between both references are considered as a textual phrase pattern. For instance, consider the sentence *"President Obama, who was born in Hawaii, married Michelle LaVaughn Robinson"*. Now assume *"President Obama"* has been identified as an entity reference to the entity `Barack_Obama`, *"Hawaii"* has been identified as `Hawaii` and *"Michelle LaVaughn Robinson"* as reference to `Michelle_Obama`. Then all three pairwise combinations will generate a pattern candidate, i.e. *"President Obama"* and *"Hawaii"* will generate a candidate for the pattern "X , who was born in Y", while *"Hawaii"* and *"Michelle LaVaughn Robinson"* will generate a candidate of the pattern "X , married Y" and finally *"President Obama"* and *"Michelle LaVaughn Robinson"* will generate a candidate of the pattern "X , who was born in Hawaii, married Y". Note that for simplicity we assume this simple pattern generation model. In practice, we may replace entity references like *"Hawaii"* by a typed placeholder, e.g. providing a pattern "X , who was born in [entity], married Y" or "X , who was born in [place], married Y" or apply heuristics to generalize the pattern ("X , ... , married Y") or reduce/relate it to a similar pattern ("X married Y"); for work in this area see, for instance, [NTW11] or [NWS12].

3.3.4. Pattern Analysis

This step provides a statistical analysis of patterns found in the document corpus, by weighing the number of pattern instances matching instances of any candidate relation the pattern might express against a) all pattern instances (*accuracy*) and b) all relation instances of that relation (*coverage*). The accuracy provides a probability estimation that any instance of a pattern p is representing an instance of relation r while the coverage provides an estimate for the probability that a given statement is represented somewhere by pattern p. In the absence of a good estimation how many occurrences of a pattern are "normal" the coverage provides a means to "normalize" the accuracy. This allows 1) to estimate the value a pattern has for the extraction system, i.e. the more facts it supports the more valuable is the pattern in terms of provenance it provides and the more likely that more facts can be extracted with the pattern, and 2) to prevent accepting patterns that are very rare and match a very small number of facts mainly by accident. Consider, for instance, the phrase *"... presented Barack Obama with a jar of sand from the beach of Hawaii as a symbolic gift to..."*. The pattern "X with a jar of sand from the beach of Y" might not occur anywhere else, such that it will appear as perfectly accurate to express the `bornIn` relation in case (`Barack_Obama`, `bornIn`, `Hawaii`) $\in \mathcal{F}$. For the extraction this is not a problem as as long as it appears nowhere else. However, if used to guide which testimonies to provide as provenance information (see Chapter 4) this is a problem, as the user will first be shown those testimonies based on patterns with high confidence. In addition, the more mistakes the extraction system makes early on, e.g. by accepting patterns that appear good enough at the time, the more likely it will fail later, as mistakenly accepted statements will weaken the basis on which patterns are evaluated. Therefore the less instances of a pattern we see the more likely it is that the accuracy will change when we see additional instances and the more careful should we be before accepting the pattern. Of course, a threshold on the pattern instances would be an alternative method to solve this problem, yet the coverage can adapt to relations that are rare and those that are more common.

Definition 3.3.1 (pattern accuracy)**.**

$$acc(p, r) := \frac{|\{p(s, o) \in \mathcal{PI}(p) | r(s, o) \in \mathcal{F}\}|}{|\mathcal{PI}(p)|} \tag{3.6}$$

Definition 3.3.2 (pattern coverage)**.**

$$cov(p, r) := \frac{|\{p(s, o) \in \mathcal{PI}(p) | r(s, o) \in \mathcal{F}\}|}{|\mathcal{F}(r)|} \tag{3.7}$$

Definition 3.3.3 (pattern confidence)**.**

$$conf(p, r) := \beta \cdot acc(p, r) + (1 - \beta) cov(p, r) \tag{3.8}$$

Remember from Definition 2.2.11 that based on a σ-threshold on the confidence $conf(p, r)$ that a pattern p represents relation r patterns are considered as potential representations of relations, e.g. (`‘‘X , who was born in Y’’`, `indicates`, `wasBornIn`) is added to

$\mathcal{F}$. This procedure is similar to the work presented in [NTW11] though we do not employ any pattern similarity measures or n-gram based unification here, but in principle this could be done.

3.3.5. Reasoning

The framework discussed so-far allows to identify entity occurrences and match patterns to relations and thus pattern occurrences to statements. However, in order to also consider logical constraints across different documents the SOFIE extraction framework [SSW09] introduced a reasoning component into the extraction process. This reasoning component is based on a set of logical rules that integrate the entity disambiguation with the pattern interpretation and other logical constraints such as the functionality of relations.

Based on a set of weighed rules the extraction task is cast into a weighed Maximum-Satisfiability (MAX-SAT) problem [BLM06, SSW09], thus, the aim is to accept a set of new statements that maximizes the weighed sum over the number of satisfied rules. A solution for this MAX-SAT problem is computed using the algorithm from [SSW09]. The basic rules cover entity disambiguation and pattern-relation association, but also model Ockham's razor by setting a base threshold with a low-weight rule negating any statement, such that enough other rules need to fire to overrule this general disbelief. A rule is a propositional first order logic formula over logic literals (instantiable variables that represent an entity; not to be confused with (RDF) literal values), i.e. simple *if-then* assertions that consist of a left-hand side of several parameterized relation instances that if satisfied with a fixed variable instantiation imply a right-hand side of logical expressions. This means the variables are implicitly universally quantified. For instance, $\texttt{bornIn(?x,Hawaii)} \rightarrow \neg\texttt{bornIn(?x,Kenya)}$ expresses that no-one can be born in Hawaii and in Kenya (only at most one of both is typically considered possible). Note that in general the relations used in these formulas refer to statements, i.e. they are true if the corresponding statement exists in $\mathcal{F}$. However, some have a special meaning such as the `different` relation that asserts that two entities are different, i.e. $\texttt{different(s,o)}$ is true when a) s and o are different URIs (blank nodes), b) s and o are different literals or c) either s or o is a literal and the other one is not. In the following we shall present and briefly discuss the rules introduced by [SSW09].

Let us start with the generalisation of the example given above: functional relations, i.e. relations r that given a domain argument a always have only one range argument b s.t. $\texttt{r(a,b)}$ holds, i.e. if $(\texttt{a},\ \texttt{r},\ \texttt{b}) \in \mathcal{F}$ then $(\texttt{a},\ \texttt{r},\ \texttt{c}) \notin \mathcal{F}$ for all $c \neq b$. As a rule this can be expressed as follows:

Rule 1 (Functions)

$$\texttt{?r(?x,?y)}$$
$$\wedge\texttt{type(?r,function)different(?y,?z)}$$
$$\rightarrow \neg\texttt{?r(?x,?z)}$$

In Section 3.3.2 it is mentioned that entity mentions that cannot be clearly disambiguated, i.e. for no candidate interpretation s of a mention sn_d the threshold t_{minfix} is

reached by $entity^i(s)$, are kept for the reasoning step to decide. In order to formulate a rule, we need to introduce a relation to model the set of entity candidates. We consider all entities that have a chance of being referenced by an entity mention (and thus any occurrence of this mention) as a candidate interpretation:

Definition 3.3.4 (candidates)**.**

$$\texttt{candidate}(\texttt{x},\texttt{y}) \Leftrightarrow entity(x,y) > 0 \tag{3.9}$$

To model a disambiguation decision we use the (functional) `disambiguatedAs` relation, such that we can formulate the following rule to consider all candidates for disambiguation.

Rule 2 (Disambiguation)

`candidate(?xo,?x)`

$\rightarrow$ `disambiguatedAs(?xo,?x)`

In Section 2.2 we stated that for each entity mention sn_d, a `disambiguatedAs`-statement reflects the most likely entity determined by $entity(sn_d)$. When employing the reasoning component, we reduce this to all the entities that are decided with certainty, i.e. where $entity^i(s)$ reached t_{minfix}during the local disambiguation. So, for all disambiguations decided earlier on (where $entity^i(s)$ reached t_{minfix}), `disambiguatedAs`(sn_d,`s`) holds.

In order to learn which patterns express which relations, pattern mentions are matched with relation instances in a rule taking the disambiguation into account. We introduce a `patternMention` relation to represent the existence of a pattern mention, i.e. if `patternMention(p,xo,yo)` holds there is a pattern mention with pattern `p` and entities `xo` and `yo`.

Rule 3 (Pattern Learning)

`patternMention(?p,?xo,?yo)`

$\wedge$`disambiguatedAs(?xo,?x)disambiguatedAs(?yo,?y)`

$\wedge$`?r(?x,?y)`

$\rightarrow$ `expresses(?p,?r)`

Note that in these rules the instances of relation `p` represent any pattern mention, for simplicity we spare the document annotations as they are of no relevance.

However, remember that the pattern analysis (if applied) suggests patterns as *indicating* relations (see Section 3.3.4). To include this statistical analysis we adapt the original rule to only consider suggested patterns for relations. However, in order to consider the disambiguation decisions that are yet undecided, the pattern matching remains part of the rule:

Rule 4 (Pattern Learning with Pattern Analysis)

`patternMention(?p,?xo,?yo)`

$\wedge$`diambiguatedAs(?xo,?x)disambiguatedAs(?yo,?y)`

$\wedge$`?r(?x,?y)indicates(?p,r)`

$\rightarrow$ `expresses(?p,?r)`

Now, the knowledge about patterns expressing certain relations can be used to generate new instances:

Rule 5 (Relation Learning)

$$\texttt{patternMention(?p,?xo,?yo)}$$
$$\wedge\texttt{disambiguatedAs(?xo,?x)disambiguatedAs(?yo,?y)}$$
$$\wedge\texttt{?r(?x,?y)domain(?r,?dom)}$$
$$\wedge\texttt{type(?x,?dom)range(?r,?ran)}$$
$$\wedge\texttt{type(?y,?ran)expresses(?p,?r)}$$
$$\rightarrow \texttt{?r(?x,?y)}$$

Finally, as a way to ensure random noise, e.g. entity misinterpretations, SPAM in webpages or will-fully wrong testimonies in comments is not too easily introducing faulty knowledge, each relation instance that can be generated is protected by a general counterweight that aims to model Ockham's razor[Ari76]:

Rule 6 (Ockham's razor)

$$\emptyset \rightarrow \neg\texttt{?r(?x,?y)}$$

Weighing All rules are weighed to reflect their importance. In particular, the disambiguation and pattern rules may potentially be violated, as not all candidates should result in a disambiguation and patterns should not simply be regarded as expressing a relation due to a single matching instance. Similarly, a single pattern occurrence may not be deemed enough to accept a new statement. Thus, those rules are assigned a low weight w. However, for the disambiguation, the local disambiguation likelihood can be taken into account and similarly when a pattern analysis step is included, the pattern confidences can also be used to modify the corresponding rule weights. We do so by multiplying the rule weight with the entity disambiguation likelihood ($entity(sn_d, s)$) and the pattern confidence $conf(p, r)$. The other rules are considered more strict and thus assigned a weight W that is significantly larger than w. Note that, given enough evidence, i.e. enough instances of pattern based rules this can still lead to inconsistencies, e.g. overriding the functionality constraint. It is merely a logical heuristic filtering out noise, but allowing to include competing beliefs - with the evidence that led to their inclusion.

3.4. Types of Feedback

There are several types of feedback a user can provide that basically refer to the different stages and decisions the extraction system may make. In principle, a user may either agree with the extraction system, disapprove the system's decisions or be neutral, e.g. for information that is not in the user's domain of expertise. Alternatively she may also approve an alternative solution the system did reject, e.g. a different interpretation of an entity reference. We consider only positive or negative feedback as relevant and represent user feedback as instances of a `supports` and `opposes` relation respectively. Both relations take a user identifier as first argument and another statement as a second argument. Note that this can be modelled in triple form using reification as explained in Section 2.1.3. To express that user u disagrees with statement (`x`, `r`, `y`) we write (`u`, `opposes`, `(x, r, y)`) and vice versa we write (`u`, `supports`, `(x, r, y)`) to signal that user u supports (`x`, `r`, `y`). Philosophically we can consider such user feedback as a claim by the user to have observed (`x`, `r`, `y`) or the opposite.

3.4.1. Entity Occurrences

First let us consider entity occurrences. A human user might support or disagree with the disambiguation decision of an extraction system. For instance, given the appearance of *"Obama"* in a text d the extraction system might have concluded that this refers to the President Barack Obama: (*"Obama"*$_d$, `disambiguatedAs`, `Barack_Obama`). However, if d is the Wikipedia page about Barack Obama Sr., i.e. the president's father, then it is more likely that the entity referenced is `Barack_Obama,Sr.`. If a user u wants to correct the system's understanding, this can be modelled as the following two feedback instances:

- (`u`, `opposes`, (*"Obama"*$_d$, `disambiguatedAs, Barack_Obama`))
- (`u`, `supports`, (*"Obama"*$_d$, `disambiguatedAs, Barack_Obama,Sr.`))

However, not only the interpretation of the entity reference might be arguable, but also the entity reference recognition task, i.e. to identify which tokens form an entity reference, can in some cases have multiple possibly correct solutions, aside from the fact that the system might be faulty. Consider for instance the phrase *"The German philosopher Nietzsche and the French Descartes were ..."*. It is arguable whether *"German"* and *"Nietzsche"* are two separate references (referencing the German nationality or country and Nietzsche the philosopher) or *"German philosopher Nietzsche"* is a single reference to the philosopher. Also, while it is clear that the reference to Nietzsche is separate from the reference to Descartes, whereas the phrase *"Procter and Gamble Co."* is typically considered a single reference to a particular company (although one can argue about that). So, even though entity recognition methods achieve in general relatively good quality results [muc98, HYB+11, MP98] there might not be a perfect solution as understandings of how to separate entity references can differ. In addition, the difficulty rises when extraction methods are applied to new languages, language variants or texts that do not

properly adhere to standard grammar and writing style. Hence, users should also be able to provide suggestions on the entity recognition task. Thus, the text stretch covered by an existing entity occurrence $\tilde{s}$ can be corrected by providing feedback with regard to the entity occurrence's position interval:

- (u, `opposes`, (`$\tilde{s}$`, `hasStartPosition`, `posX`))
- (u, `supports`, (`$\tilde{s}$`, `hasEndPosition`, `posY`))

Similarly a separate entity occurrence can be described by generating support instances for the corresponding statements describing it.

3.4.2. Testimonies

Similarly as with entity occurrences there are two decisions that the extraction system needs to make to assign testimonies to statements on top of the entity recognition and disambiguation of the entities involved. First it needs to identify patterns and then it needs to interpret what relation each pattern represents. While the association of patterns with relations holds for all pattern instances (and occurrences) there might still be differences in the interpretation of different instances, as depending on the actual entities and their types not all relations fit a particular pattern instance. In addition, users might also feel that the system did miss a particular testimony, e.g., due to the pattern generation approach when a symbol that is typically a sequence separator, like a period, is used in a different context. Hence, feedback is also collected based on pattern occurrences.

Consider a pattern occurrence $po_1 = \tilde{p}(\tilde{s}, \tilde{o}) \in \mathcal{PO}(d)$ from document d that represents the phrase *"Tarantino's newest movie Django Unchained"* where *sn* =*"Tarantino"* and ($\tilde{s}$, `disambiguatedAs`, `Quentin_Tarantino`) as well as *on* =*"Django Unchained"*$\in \mathcal{F}$ and ($\tilde{o}$, `disambiguatedAs`, `Django_Unchained_(film)`)$\in \mathcal{F}$. If (`p`, `expresses`, `actedIn`) $\in \mathcal{F}$ such that the pattern occurrence is interpreted as a testimony for (`Quentin_Tarantino`, `actedIn`, `Django_Unchained_(film)`), the user might want to rectify the interpretation of this particular occurrence :
(u, `opposes`, (`$\tilde{p}$`, `expresses`, `actedIn`)),
but confirm or add that the text states that Tarantino directed the movie:
(u, `supports`, (`$\tilde{p}$`, `expresses`, `directed`)).

Of course, we could also model feedback on pattern occurrences as feedback on the general level of pattern interpretation, e.g., as: (u, `opposes`, (`p`, `expresses`, `actedIn`)) However, when a user looks at a certain phrase in a given document and provides feedback on that basis, it is not clear, whether she opposes the pattern interpretation in general, or just in this particular occurrence, for reasons that may potentially not be captured by the extraction system's approach. Also there are patterns that are inherently ambiguous, such that information on particular instances can help identify the cases (e.g. types of entities) in which it can be interpreted in the one or the other way. Secondly, even if the user were asked to judge the pattern in general, this is a much more difficult choice to be made, as the user might not be able to foresee all cases in which the pattern can occur (especially if

patterns are more complicated, e.g. based on regular expressions or actually representing a set of phrases etc.).

Similarly, a user might consider a larger part of the sentence to be a better indicator and thus adjust the pattern position (and thereby change the pattern). For instance, assume the full sentence in which the phrase appears is *"Director Tarantino's newest movie Django Unchained is likely to be a huge hit."*. A user might consider it more indicative to include the "Director" bit thus extending the pattern accordingly. Ignoring for a moment that the extraction system only deals with simple patterns between the two entities concerned, our model allows to arbitrarily define the pattern text intervals that make up the pattern. Assuming that this is the first sentence of document d a user could express her point by adding an additional text interval to pattern occurrence $\tilde{p}$ ($\tilde{s}$,$\tilde{o}$):

- (u, `supports`, ($\tilde{p}$, `hasStartPoint, 0`))
- (u, `supports`, ($\tilde{p}$, `hasStopPoint, 8`))

3.4.3. General Statements

So far, we were only considered with observations based on documents that could differ for a user from the findings of the extraction system, where the user is mainly trying to help the system to understand the documents. However, we also might want to consider the user as a direct information source, i.e. instead of providing feedback on testimonies in documents, the user himself can provide factual statements. For instance, a user might provide his own preferred movies, or his opinion on where Barack Obama is born:

- (`smetzger`, `supports`, (`smetzger, likes, Django_Unchained_(film)`))
- (`smetzger`, `supports`, (`smetzger, bornIn, Earth_(planet)`))

3.4.4. Other Forms of feedback

There are other forms of feedback that user's might want to provide. For instance, a user might decide not to trust information from certain pages in general or might find the system cannot recognize certain forms of literal values, e.g. ISBN numbers or the like. We are not considering such forms of feedback, yet, but argue that many types of feedback could be modelled similarly and included into the ontology as user observations. Modelling data formats as ontological knowledge, for instance, would decouple this general knowledge about the format from a particular extraction system implementation and make it more easy to employ different extraction systems in a comparable way. In addition, some forms of feedback can also be modelled already with the provided vocabulary. For instance, a user's mistrust in certain sources could be handled by a frontend by providing opposing feedback for all statements from these pages.

3.5. Feedback-Integration

Now let us consider how LUKe can *Learn from User Knowledge* by integrating user feedback of the kind introduced in Section 3.4 into the extraction process.

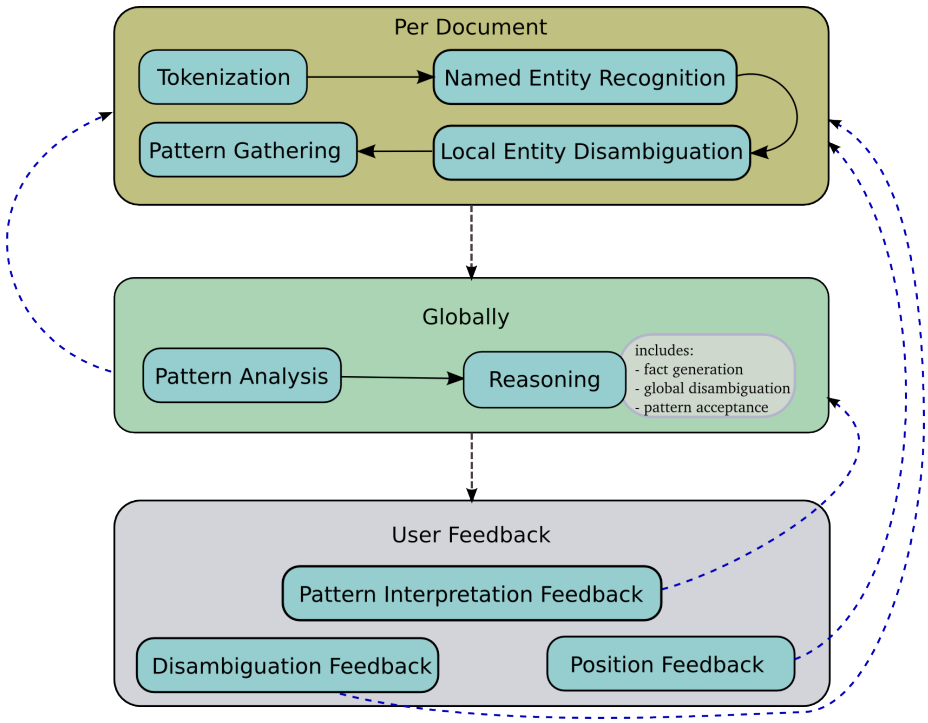


Figure 3.1.: Extraction Stages with Feedback integration

Figure 3.1 depicts an adapted extraction process with an additional feedback collection stage. Depending on the type of the feedback provided the system can either directly reiterate over its global tasks (pattern analysis and reasoning) or may need to go back to the document level if user opinions are to be considered. In particular, when position feedback is used to adapt what part of text is considered as an entity or a pattern, this can have effects on the context, such that other extractions might be affected. For instance, if an entity occurrence's stretch is changed this may affect a pattern in which it is involved in, changing the actual pattern. Feedback on pattern interpretation can be dealt with globally, assuming that all possible pattern occurrences are generated and stored, no matter whether they seem to express a relation. When it comes to the disambiguation of entity references however, user opinions on the correct interpretation that change the entity for at least one reference also may have effects on other entity reference interpretations via the coherence measure and when new entity-name combinations are learnt (see Rule 7, page 49) this may affect any document in which the name appears.

Figure 3.2 provides an overview which feedback is used at which extraction stages and to which effect. We shall discuss this in the following for each type of feedback.

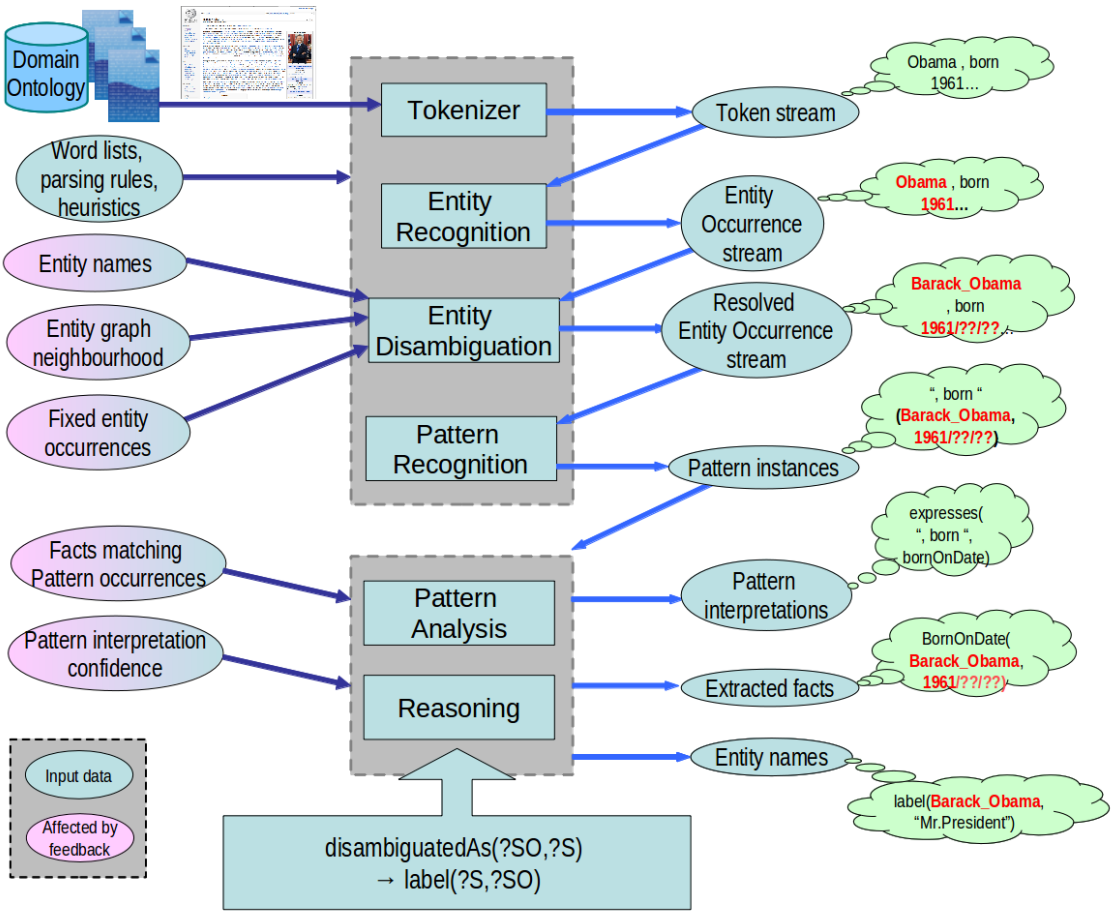


Figure 3.2.: Feedback Integration effects

3.5.1. Disambiguation Feedback

First we consider users' support or opposition to entity disambiguations. Reconsider the example where a user u aims to correct the interpretation of a reference using the entity name *"Obama"* that is interpreted as referencing President Obama (`Barack_Obama`) while the user thinks it actually refers to his father (`Barack_Obama,Sr.`).

- (`u`, `opposes`, (*"Obama"*$_d$, `disambiguatedAs`, `Barack_Obama`))
- (`u`, `supports`, (*"Obama"*$_d$, `disambiguatedAs`, `Barack_Obama,Sr.`))

We redefine the initial disambiguation step by adding a third user-feedback based component as:

$$entity^0(s) := \mu_{eo} \cdot usup(sn_d, s) + (1 - \mu_{eo}) \cdot (\mathcal{E}(sn, s) \cdot entity_{bag}(sn_d, s)) \tag{3.10}$$

and analogously modify the iterative component

$$entity^i(s) := \mu_{eo} \cdot usup(sn_d, s) + \alpha \cdot (\mathcal{E}(sn, s) \cdot entity_{bag}(sn_d, s)) + (1 - \alpha - \mu_{eo}) entity^i_{coh}(sn_d, s) \tag{3.11}$$

and we define the user-feedback component by:

$$usup(sn_d, s) := \frac{\sum_{u,pos} \texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s))}{\begin{array}{c}\sum_{u,pos} \texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s)) + \\ \sum_{u,pos} \texttt{opposes}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s))\end{array}} \tag{3.12}$$

while this includes all feedback directly related to the entity occurrence, we typically consider the disambiguation to be a functional problem, i.e. only one entity can be correct. Thus, a user supporting a particular interpretation is also implicitly opposing any other entity candidate. To reflect this, the denominator can be extended by the sum over all support for any other entity $s' \neq s$':

$$usup(sn_d, s) := \frac{\sum_{u,pos} \texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s))}{\begin{array}{c}\sum_{u,pos} \texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s)) + \\ \sum_{u,pos} \texttt{opposes}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s)) + \\ \sum_{u,pos} \texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s'))\end{array}} \tag{3.13}$$

This means, we consider positive as well as direct or indirect negative feedback. To handle cases where there is no feedback, we assign $\mu_{eo} = 0$ if the denominator is 0, falling back to the original model. By the choice of μ_{eo} the system can be adapted to either strictly adhere to user feedback, i.e. considering feedback more as ultimate corrections (as long as the users agree) than additional input, or - with a lower setting of μ_{eo} as an opinion that is on an equal level with other context information. A setting of $\mu_{eo} = 1$, for instance, will only use user-feedback for all mentions where there is feedback for any occurrence in the document.

A user may also choose an entity not considered a candidate by the system at all, since it did not know that the entity name could refer to that particular entity. In that case, the entity needs to be added to the set of candidate entities, and when it is accepted, we may learn the entity-name relationship. To this end, an additional rule is added:

Rule 7 (Names)

$\texttt{disambiguatedAs}(sn_{d@pos},\texttt{?s})$

$\rightarrow \texttt{label}(\texttt{?s},sn)$

3.5.2. Pattern Interpretation Feedback

Similarly as the entity disambiguation, the computation of the pattern confidence from Equation (3.8) (Definition 3.3.3) is redefined including a user-feedback component:

$$conf(p,r) := \mu_p usup(p,r) + \beta acc(p,r) + (1-\beta-\mu_p)cov(p,r) \tag{3.14}$$

$$usup(p,r) := \frac{\sum\limits_{u,d,pos} \texttt{supports}(u,\texttt{expresses}(p_{d@pos},\texttt{r}))}{\sum\limits_{u,d,pos} \texttt{supports}(u,\texttt{expresses}(p_{d@pos},\texttt{r})) + \sum_u \texttt{opposes}(u,\texttt{expresses}(p_{d@pos},\texttt{r})))} \tag{3.15}$$

Note that we sum over all occurrences that belong to the pattern of interest, as we assume no direct feedback for a pattern, but feedback on single occurrences. Again, in case there is no feedback, such that the denominator becomes 0, we assign 0 to μ_p falling back to the normal confidence measure. Note that patterns are assumed to be able to express multiple relations, thus support of other interpretations are not considered as counter-evidence.

In addition when given user support or opposition to a pattern occurrence's interpretation, we adjust the confidence we have in the pattern representing the respective relation for this particular occurrence, as there might be other factors than the general pattern reliability at play, e.g. the general trustworthiness of the document or an ironic intention that is not captured by the pattern:

$$conf(\tilde{p},r) := \mu_{po} \cdot usup(\tilde{p},r) + (1-\mu_{po})conf(p,r) \tag{3.16}$$

where

$$usup(\tilde{p},r) := \frac{\sum_u \texttt{supports}(u,\texttt{expresses}(\tilde{p},\texttt{r}))}{\sum_u \texttt{supports}(u,\texttt{expresses}(\tilde{p},\texttt{r})) + \sum_u \texttt{opposes}(u,\texttt{expresses}(\tilde{p},\texttt{r})))} \tag{3.17}$$

This allows us to provide an estimate for particular pattern occurrences on how supportive the testimony is for an associated statement. This in turn can be used by a user frontend to guide a user to occurrences that are doubtful and on the other hand a user sees which occurrences have already been considered by a user as they will have a higher confidence.

3.5.3. Fact Feedback

So far, we have only considered user feedback based on observations in documents. In addition, users may also directly state their own beliefs, i.e. provide direct testimonies on general statements, such as `bornIn` or `actedIn` instances. In the current framework this can be included via a rule to include new facts based on user feedback:

Rule 8 (Feed in)

`supports(?u,?r(?x,?o))`
$\rightarrow$ `?r(?s,?o)`

and a rule to exclude information users consider false:

Rule 9 (Delete)

`opposes(?u,?r(?x,?o))`
$\rightarrow \neg$`?r(?s,?o)`

Remark 1. *Also note that the extraction framework does currently not remove information already present, i.e. input to the extraction provided as original domain knowledge as well as statements generated by earlier extraction iterations. The reasoning step's decisions are only concerned with information candidates generated by the extraction machinery in the current iteration. However, a separate cleaning process could take care of this.*

Note that such feedback, if it leads to additional statements being included or others being prevented from inclusion, may also affect the pattern analysis and further extraction of other statements. In principle, this is the traditional way to provide an extraction system with additional examples for an existing relation (or even examples of a new relation). However, by modeling this as user feedback we ensure 1) the inclusion of provenance, as the `support`-statement needs to be present as well and 2) it allows personalization based on user trust (see Section 3.5.5).

3.5.4. Localization Feedback

Both the entity recognition component and the pattern generation component are fixed heuristics, the first being based on identifying upper-case words the second always considering all tokens between a pair of entity references. Thus, localization feedback from users, i.e. which tokens make up an entity reference (see Section 3.4.1) and which parts of a phrase contribute to expressing an associated relation (see Section 3.4.2), are not used to adapt these steps so far. Such user observations are, however, used to override the extraction system's findings, thus allowing for interleaved human processing that will also keep sustained through further extraction iterations. This allows a human control and correction of documents, although not tapping into the learning potential to apply such feedback to extractions elsewhere. However, the extraction process is modular and all user observations are kept within the ontology. This leaves the possibility to alternatively

employ tokenization and entity recognition methods that adjust to special cases as well as pattern generation techniques that learn over time which words to include and which can be ignored.

3.5.5. Personalization

Remark 2. *In contrast to the remainder of this chapter, this section is only theoretical and has not been implemented in the prototype framework.*

While in the current model all users are considered equal, we may trust some users' testimonies more than others. For instance, a user of our own working group may better know how to apply the group's guidelines to annotating entities and resolve entity references than a crowd worker hired for a small amount of money per work hour. To deal with such cases a trust function can be included and coupled with all feedback statements to weigh them. However, the trust put into different users may also vary from user to user, such that the trust function needs to be personalizable as well, implying that a trust function per user is required. Given such a trust function the disambiguation support could, for instance, be adapted like this:

$$usup(u', sn_d, s) := \frac{\sum_{u,pos} trust(u', u)\texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s))}{\begin{array}{l}\sum_{u,pos} trust(u', u)\texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s))+ \\ \sum_{u,pos} trust(u', u)\texttt{opposes}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s))+ \\ \sum_{u,pos} trust(u', u)\texttt{supports}(u, \texttt{disambiguatedAs}(sn_{d@pos}, s'))\end{array}} \tag{3.18}$$

Similarly the pattern support can be adjusted by including the trust function:

$$usup(u', p, r) := \frac{\sum_{u,pos} trust(u', u)\texttt{supports}(u, \texttt{expresses}(p_{d@pos}, \texttt{r}))}{\begin{array}{l}\sum_{u,d,pos} trust(u', u)\texttt{supports}(u, \texttt{expresses}(p_{d@pos}, \texttt{r}))+ \\ \sum_{u,d,pos} trust(u', u)\texttt{opposes}(u, \texttt{expresses}(p_{d@pos}, \texttt{r}))\end{array}} \tag{3.19}$$

and the rules from Section 3.5.3 could be weighed by $trust(u', u)$.

The simple approach to use such a personalization, where only the second variable in the trust function is used, i.e. we ignore who is using the ontology and only consider different users that provide feedback with varying trust, results in a non-personalized ontology. A typical application for this would be an extraction project that relies to some part on external users, e.g. via crowd sourcing. Such users might have a different degree of expertise and thus their feedback might be valued less than domain experts managing the project. Similarly, a Wikipedia-style approach might want to manage different types of users and limit their influence. If we consider both variables in the extraction process, such that depending on who is using the extraction system, existing user-feedback is weighed differently, then this can be used to a) generate different personal ontologies by aggregating over a shared ontology of feedback but keeping the information extracted for each user

separate or b) to generate one ontology that contains the union over the extraction results of all users. In the latter case, any access to the ontology should then also take into consideration the trust function when visualizing the ontology. This could be helpful if multiple parties work on the same data set with different perspectives.

3.6. Architecture

In this section the underlying architecture is presented. The framework also integrates components implementing the methods that will be discussed in chapters 4 and 5 and provides a Representational state transfer (REST) [FT02, Fie00] based web-service interface for external frontend components.

The design of the framework was driven by four main goals:

1. to enable collaboration in user-supported ontology generation and maintenance amongst different users or user groups sharing the same ontology,
2. to provide a system that is easily usable, in particular with low setup effort for end-users,
3. to allow domain-specific adaptations in the processes, and
4. to allow for supporting different types of users with tools that match their needs

This led to a modular web-based framework that provides a web-service interface for front-ends, but also decouples the ontology storage method from the remaining system.

Outline In the next section (3.6.1) we provide an overview over the general architecture with its three different layers. Each layer is then discussed separately in the following three sections (Section 3.6.2, 3.6.3 and Section 3.6.5).

3.6.1. Overview

Our framework consists of three layers of components: a *core components* layer that contains the ontological data and the extraction system, a *backend* layer that provides higher level functionality, e.g. extraction control, witness document retrieval, ontology navigation etc., and finally a *frontend* layer that comprises of any user frontend that interacts with a non-expert end-user. A REST [FT02, Fie00] based web-service interface handles the communication between any frontend and the backend layer. Figure 3.3 depicts this framework architecture. Each layer is an interaction point that potentially allows distribution of the components. That is, in principle a frontend, the backend component and the extraction system could run on separate machines. For instance, a desktop frontend could communicate with the backend server, while the extraction actually takes place in a distributed fashion on a cluster of compute machines. In the following we will discuss each layer.

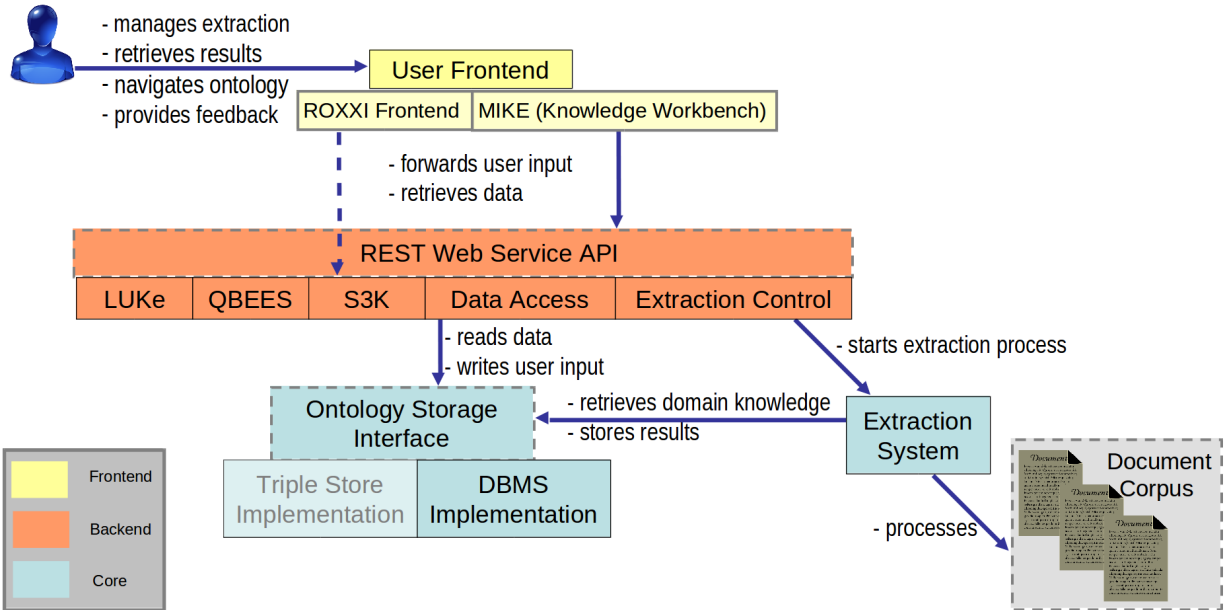


Figure 3.3.: Architecture

3.6.2. Core Components

The core components are the two essential pieces that all the other methods and applications are based on. That is first the ontology in a broad sense, i.e. the actual knowledge of interest as well as the meta-knowledge about the extraction process, and the actual data structure managing this information in particular, and second the automatic ontology generation methods, i.e. the extraction system.

3.6.2.1. Storage Engine

The storage engine provides an interface for access to ontological facts in quadruple format, i.e. triples with associated id. Separately various meta information, most importantly entity occurrences, patterns, pattern occurrences, document information and user-feedback statements are stored. For each such data structure methods to store, remove and access instances are provided explicitly. This allows an implementation of the storage engine interface to realize the data storage individually if this promises a more efficient solution. Within this work a storage engine implementation based on traditional SQL databases (explicitly supporting MySQL [mys] and PostgreSQL [pos]) has been developed. It manages each data structure type used in the extraction process, i.e. patterns, entity occurrences etc., in a separate table and all other statements in a quadruple table of facts.

In hindsight of different application scenarios two versions of the database based storage engine have been implemented: the *multi-ontology* version that supports to manage multiple ontologies in parallel, and the default *single-ontology* version that assumes only one large ontology exists that contains all knowledge. The multi-ontology variant allows for users to work on different *projects*, separating the extracted knowledge (see Figure 3.4 for

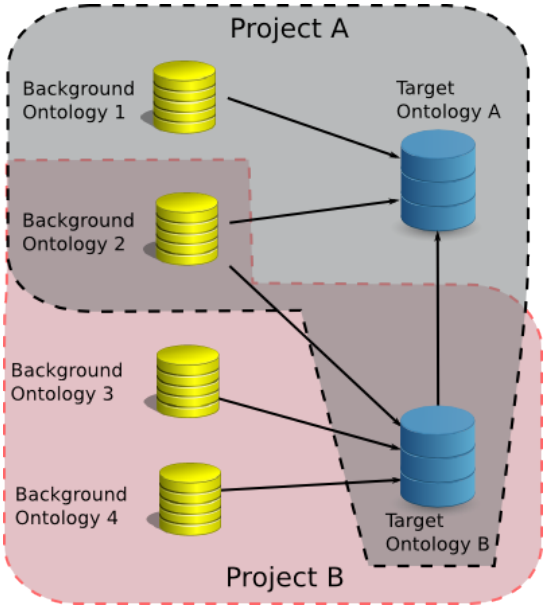


Figure 3.4.: Ontology Management example

an illustration). This can be used, for instance, to keep different domains of source information separated or to allow different user-groups with potentially varying belief systems to maintain their own ontology without having to deal with inconsistencies. To this end we distinguish between *background* and *target* ontologies. Each project consists of exactly one target ontology that represents the project and arbitrary many background ontologies, all of them need to be stored within the same storage engine. With regard to retrieving and storing ontological knowledge, background ontologies are considered stable while extraction results or otherwise added ontological facts are stored in the target ontology. In particular, all read actions on the project are answered by virtually creating the union over all ontologies, while all updates are only executed at the target ontology, i.e. no system may alter a background ontology when accessing the target ontology. However, background ontologies may be accessed directly and then be modified. This also allows to modularize ontological background knowledge essential to different domains, such as entity name dictionaries and class hierarchies etc. An extension of this variant (not implemented) would be to allow references to external ontologies in the same fashion as practiced within the Linked Open Data [**?**] (LOD) ontology community. The drawback of the multi-ontology variant is lower efficiency, as different ontologies are kept in separate tables, which allows space efficient storage no matter in how many projects ontologies serve as background knowledge, yet this requires to join the ontologies at run-time.

3.6.2.2. Extraction System

The extraction system follows the general outline discussed in Section 3.3. Not only is the extraction system as a whole component replaceable, but also its main stages (see

Section 3.3) can be replaced by different implementations. In addition, the framework supports in principle the use of different types of patterns, although depending on the nature of the different patterns, this might suggest adjustments at the pattern analysis stage, e.g., some patterns might be given additional weight by their very nature. In order to exchange the extraction system as a whole, one needs to adapt the class calling the extraction system's main stages. A separate implementation can in principle also place remote calls to a different machine or an extraction system written in another language than the rest of the system. In order to integrate into the framework the extraction system needs to use the storage engine interface to access and in particular to store the information it extracts in the form of entity occurrences and pattern occurrences. Finally, if the pattern analysis and reasoning component are also exchanged, then the extraction system needs to produce the corresponding statements, i.e. statements that indicate which patterns express which relation and the statements it sees enough support for in the pattern occurrences found.

3.6.3. Backend Components

On top of the core layer the backend components provide more complex methods that are based on the ontological data and the extraction system. This includes the methods discussed in the Chapters 3, 4 and 5, but any other module can use the core components to provide additional functionality. Each such backend functionality module is made accessible via a REST [FT02, Fie00] web service API[7]. The implementation of the API is based on the Jersey library [jer, Dub10]. Replacements of components or extensions can adapt the existing API implementation or simply extend the resource name space with their own resources and operations.

In particular, the *Extraction Control* backend component allows to start an extraction process on a given set of source documents and provides status information on the progress of the extraction process. The feedback integration as described in Chapter 3 is handled by the *LUKe Feedback handler* component, while the *S3K Witness Finder* provides the provenance oriented document retrieval methods described in Chapter 4. Finally, the user similarity methods from Chapter 5 are provided by the *QBEES Entity Finder* component. Note that each component brings its own portion of the REST interface and thus the actual methodology might be replaced without requiring any change of components that access it through the web service interface.

3.6.4. Web-Service Resource Hierarchy

A REST-API is resource based, i.e. it provides stateless (with regard to the client's state) access to resources and operations on resources. Each resource is represented by a URL. Operations on a resource are typically based on HTTP, but often for non-standard operations it is also common to represent operations by attaching the operations name to the URL. We

[7] application programming interface

follow the latter approach. Assume for instance a fact being represented by the URL `http://www.mpi-inf.mpg.de/facts/fact3888`. To apply a hypothetical `delete`-operation, one would need to access the URL `http://www.mpi-inf.mpg.de/facts/fact3888/delete`. Typically accessing the resource should provide information representing the resource, such as a web-page for human consumption or a machine readable representation such as XML or JSON. The exact format is negotiated based on a request's header and the servers data representation options. Our API implementation supports both JSON and XML for all resources and resource operation results, and in most cases also provides a simple HTML based summary of resource information for human consumption. When retrieved via JSON or XML all data structures are represented as 'objects' with a set of attributes. For instance, each statement is represented as a *fact* object that has the attributes *id, subject, predicate* and *object*. Our API is based on a three level resource hierarchy. First, there are different ontologies, each a collection of general facts and potentially extraction knowledge. The extraction knowledge makes up the second and third resource level. First, any document d having been parsed to link its entity occurrences to entities and its statements to facts of an ontology o is considered a *witness* resource of ontology o. Each such witness may contain several entity occurrence or statement occurrence resources. In addition, each ontology maintains extraction *run* resources that bundle a set of document URIs to be processed by the extraction tool. Each such run can be considered an independent extraction assignment. Then there are the general knowledge *facts* contained in the ontology, they are represented by fact resources. In the following we shall briefly discuss these resources and the main operations available on each such resource.

3.6.4.1. Ontology Resources

Each ontology represents a fact collection. Different projects, e.g. representing different knowledge domains or different analysis tasks, can be kept separate by maintaining separate *target ontologies*. Each *target ontology* is associated with a possibly empty set of *background ontologies*. While any extracted information, either entity occurrences or statement occurrences found in parsed documents or new relation instances generated by examining the statement occurrences, is stored in the target ontology, for all extraction and querying purposes the union over all ontologies is considered as the knowledge base. That is, the *background* ontologies provide additional domain knowledge for the extraction task (or any user querying the ontology). Each ontology can have multiple background ontologies and each ontology can be a background ontology for multiple target ontologies. Given the resource `id` of an ontology, i.e. the ontology name, the `ontology/{id}/background` resource URI will provide all background ontologies, and the operations available via `ontology/{id}/background/add` and `ontology/{id}/background/remove` can be used to add or remove ontologies from the background ontology collection. The components of Chapter 4 and Chapter 5 each provide an operation representing their main functionality described in the respective chapter: There is a `ontology/{id}/s3k` operation that allows to provide a set of facts and retrieve a list of witness documents for that list (implementing the witness retrieval as it will be discussed in Chapter 4) and another `ontology/{id}/qbee`

operation that provides similar entities given a set of query entities (implementing the functionality described in Chapter 5).

In addition the following operations are available:

`/integrate`	Integrates another ontology given by its ontology id.
`/materialize`	Materializes the ontology in a file for download.
`/store`	Allows to store the ontology at a provided URI (either local path on the server or (limited support) Web-DAV path).
`/state`	Provides the state of the ontology (whether busy with an operation or available).
`/remove`	Deletes the ontology and all data contained from the storage.

Table 3.1.: Operations on ontology resource identified by `ontology/{id}`

In addition there are operations to list all available ontologies (`/ontologies/all`), to create a new ontology (`/ontologies/create`), to load an ontology from a file, locally or via Web-DAV (`/ontologies/load`), and to import an ontology from a file upload (`ontologies/upload`).

3.6.4.2. Run Resources

Each extraction *run* is a collection of document URIs and represents an extraction task on these documents. In order to start an extraction task a new run can be created via the operation `/ontology/{id}/run/create`. The current status - queued, being processed, completed and failed in case of a problem - of each such extraction task can be retrieved via `/ontology/{id}/run/status`. Once the extraction is completed, results of a particular run in the form of entity and statement occurrences found can be retrieved together via `/ontology/{id}/run/results`. Alternatively all such results are also available via the individual witness results (see below).

3.6.4.3. Witness Resources

Each document that has been processed by the extraction tool is represented by a witness resource. Each such witness resource is accessible under `/ontology/{ontologyId}/witness/{witnessId}`. If the witness id is not known but its original URI, the resource `/ontology/{ontologyId}/witness/byURI` redirects to the canonic resource URI of the witness given its original URI. For each witness its original URI, a potentially extracted title and a timestamp of the last parsing by an extractor is stored and provided via its resource URI. All entity and statement occurrences are accessible via respective operations. A summary of all operations on a witness resource is provided in Table 3.2.

/cachedcontent	Provides the witness content as a string from the storage engine - if it has been stored earlier (depends on extraction tool).
/reset	Use this to get rid of all extraction information obtained from this witness.
/occurrences	Use this to retrieve all extraction information found in this witness.
/entityOccurrences	Use this to retrieve all entity occurrences found in this witness.
/statementOccurrences	Use this to retrieve all statement occurrences found in this witness.

Table 3.2.: Operations on any witness resource identified by ontology/{ontologyid}/witness/{witnessId}

3.6.4.4. Entity Occurrence Resources

An entity occurrence's main properties are its position (given by start and stop position), the string from the document represented by this occurrence, and the entity candidates that are potentially referenced, which potentially is only one entity if the disambiguation has been completely resolved (see Section 3.3). The entity occurrence resource operations basically realize the feedback capabilities discussed in Section 3.4 and some additions to create (/ontology/{ontologyId}/witness/{witnessId}/entityOccurrences/create) or remove an entity occurrence. The operations on an existing entity occurrence are summarized in Table 3.3.

/editPosition	Adapts the scope of the entity occurrence in the text.
/editReferencedEntity	Sets support for a particular entity interpretation of the occurrence.
/verify	Confirms the (automatically) selected entity as the one being referenced.
/remove	Removes the entity occurrence.

Table 3.3.: Operations on any entity occurrence resource identified by ontology/{ontologyid}/witness/{witnessId}/entityOccurrence/{entityOccurrenceId}

3.6.4.5. Statement Occurrence Resources

Statement occurrence resources realize the statement feedback capabilities described in Section 3.4. The main operations are listed in Table 3.4.

In addition /ontology/{ontologyId}/witness/{witnessId}/statementOccurrences/create allows to manually create a new statement. Note that this will - with the current extraction backend - generate a human defined pattern that is not included in the pattern analysis.

/supportMeaning	Allows to state a user's support for a particular relation interpretation of the statement.
/refuteMeaning	Allows to state a user's opposition for a particular relation interpretation of the statement.
/refuteMeanings	Convenience Operation, allows to state a user's opposition to all automatically detetected relation associations.
/resetSupport	Resets a user's support or opposition for all interpretations.
/resetSupportForRelation	Resets a user's support or opposition for a particular relation interpretation of the statement.
/remove	Suggests to remove this statement occurrence.

Table 3.4.: Operations on any witness resource identified by `ontology/{ontologyid}/witness/{witnessId}`

3.6.4.6. Fact Resources

Fact resources are sub-resources of ontologies representing general ontological facts. They are identified by the fact id, i.e. accessible via `/ontology/{ontologyId}/fact/{factId}` and besides retrieval of the fact support the two feedback operations described in Section 3.4.3:

/support	States that a user supports the statement.
/oppose	States that a user opposes the statement.

Table 3.5.: Operations on any fact resource identified by `ontology/{ontologyid}/fact/{factId}`

3.6.5. Frontend Components

The frontend components provide user-level functionality based on the backend components. They might either use backend components directly, which requires both to run on the same server, or through the REST web-service interface, which allows remote calls. For both variants some basic user frontend interface has been implemented. First, the S3K frontend allowing to define statement search queries, which will be discussed in Section 4.7, directly accesses the storage engine. The *workbench*, which allows to control an extraction process and provide feedback in a task-oriented setting (see Section 3.7.2), however, uses the web-service frontend to start extraction processes, retrieve result data and feed user corrections and additions back into the system.

3.7. WisNetGrid

The work discussed in this chapter was driven to a large degree by the WisNetGrid project [wng] funded by the BMBF[8] (Bundesministerium für Bildung und Forschung/Federal Ministry of Education and Research). Among others the main relevant goals of the WisNetGrid project were the following. First, to provide unified access to distributed Grid resources, in particular storage space and computation power, including a single-sign-on (SSO) user authentication system. Second, to develop integrated extraction tools allowing user-interaction to annotate documents with entity occurrences and contained factual statements and extract this information into ontological knowledge that are easy to deploy and setup for individual users while allowing to maintain different shared projects. This led to the development of the *knowledge workbench* frontend (see Section 3.7.2), influenced the architecture design (see Section 3.6) and drove optional support of the WisNetGrid SSO authentication architecture and unified Grid computation power interface (see Section 3.7.1), as well as a relation extraction approach for German texts (see Section 3.7.3).

3.7.1. Unified Resource Access

The WisNetGrid architecture, see Figure 3.5, provides a federation layer aggregating access management for distributed resources. Its main service at the federation layer is unified access to distributed data via a WebDAV interface. A Security-Federator component provides user management allowing users to register and provide their credentials for various grid resources authenticating the user to those distributed systems. The WisNetGrid SSO server then manages these credentials to authenticate the user to distributed systems once the user is authenticated to the WisNetGrid SSO server.

While the WisNetGrid architecture provides a user portal allowing users to login and browse their distributed data, the SSO-server can be used to login the user via any frontend that supports the SSO principle (the *knowledge workbench*, Section 3.7.2, does). Once a user is logged in, a session id is used to authenticate the user and applications acting on his behalf to the WisNetGrid grid resource access layer via the resource federator (see Figure 3.5). For instance, the extraction tool is using the session id to access information sources available via the WebDAV based grid interface and frontends that want to present the source documents need to do this as well. In order to avoid third parties eavesdropping on the connection and sniffing the session id or the user account credentials, all such connections are typically SSL encrypted. In addition as you can see in Figure 3.5, the extraction tool can optionally make use of the distributed computation resources available to the user (or the project hosting the extraction tools depending on setup). To this end a generic two-fold interface has been designed. First, on the side of the WisNetGrid resource layer, an extraction adaptor provides an interface to submit extraction jobs to the grid and monitoring their progress. This interface is realized as a REST [FT02, Fie00] based resource, thus all communication is subject to the same authentication and encryption

[8] http://www.bmbf.de

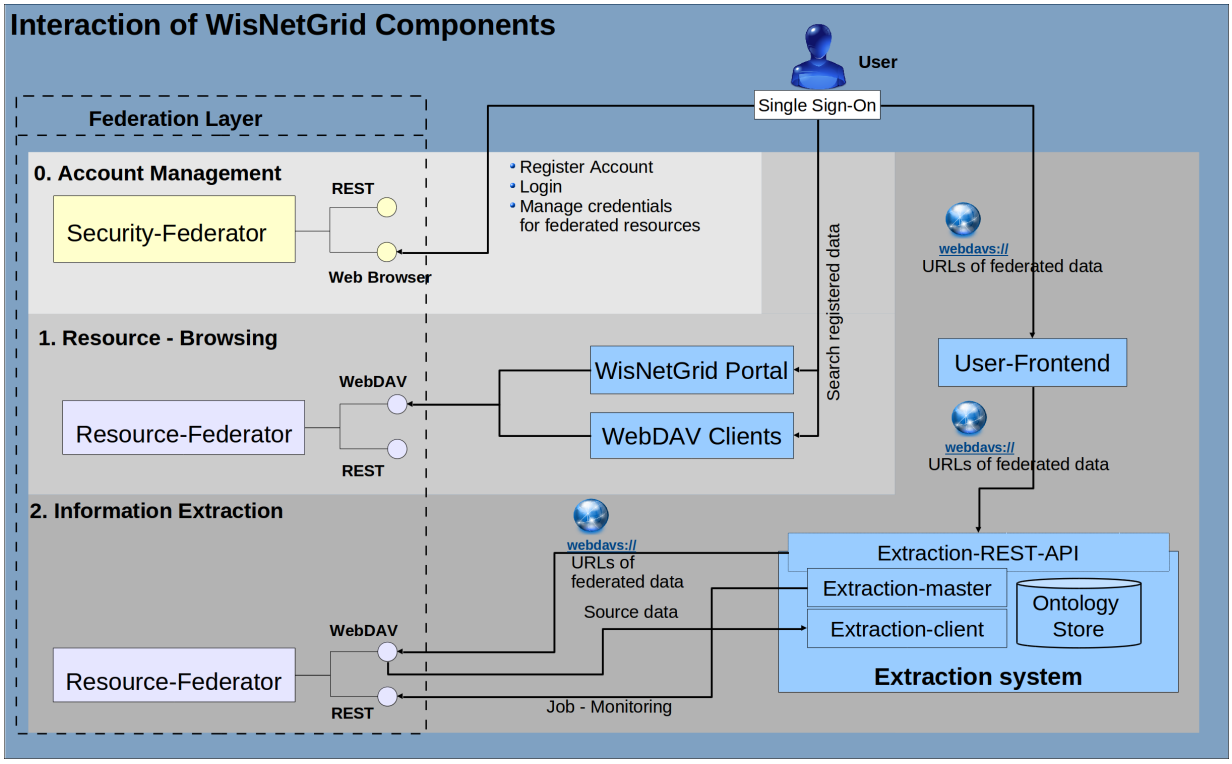


Figure 3.5.: WisNetGrid Federation Architecture

methods as any other resource access, i.e. an SSO session or username and password credentials as well as SSL encryption. This computation resource interface integrated into the Resource Federator has been designed in collaboration with and implemented solely by Bernd Schuller from Forschungszentrum Jülich GmbH. On the other hand, the central extraction system backend uses an internal (Java) interface to start each extraction stage (see Section 3.3), as stages can work independently. We provide distributed implementations for the entity recognition and disambiguation as well as the pattern collection and pattern analysis stages while the reasoning step is always done centrally; there is work though that allows the distribution of the reasoning task with minimal information loss as well [NTW11]. So the extraction control backend component (called Extraction-master in Figure 3.5 as it implements a client-master architecture) is starting the distributed extraction system stages, observes the execution of these Extraction-client stages and then collects the results (via the storage engine). Note that for distributed execution, the client component - in its current form - needs to be installed first on the grid cluster to be used. While this may arguably require more initial effort, it also means the central extraction components need not be aware of the actual distribution architecture, e.g. which programming languages are supported, which resources are available etc. This reduces central control, but increases adaptability, e.g. additional grid systems can be supported without requiring adaptation at the central components.

3.7.2. Knowledge Workbench

Within the WisNetGrid project we worked with two partners representing potential user-groups of an extraction system integrated into the WisNetGrid architecture. They represented quite different use-cases of semantic information. While one group from the humanities was aiming to support an annotation task, mainly oriented towards identifying entity occurrences in partially historical literature, the other group was interested in extracting factual information from a broad range of document types and bringing this information together. They both had in common that their documents were from special domains, thus requiring a domain specific entity and relation corpus, and that at least a substantial part of their documents were in German. For the latter point see Section 3.7.3.

The use-case of the first user group was to annotate documents with - amongst others - the entities referenced. This would then be used, for instance, to investigate different versions of the same story, bring together information and text passages about the same entity in different text documents etc. Traditionally this annotation of documents was a manual task and it was paramount that each document is annotated completely and as correct as humanly possible. Hence, a human would have to counter-check automatic results in any case.

While the use-case in the other user-group was a different one, namely to bring together relational information from different sources, this task was done manually so far as well. However, while the focus here was less on a high per document quality, at least those bits of information used in the end would have to be reviewed. Hence, the second use case could benefit from our work in Chapter 4, in order to verify particular pieces of information. Together with representers for both user groups, we first designed a workflow that could be applied to projects in both use-cases alike. Based on this workflow the *knowledge workbench* has been designed and implemented as a tool to provide information extraction support in an interactive setting.

3.7.2.1. Workflow

The workflow we designed is illustrated in Figure 3.6.

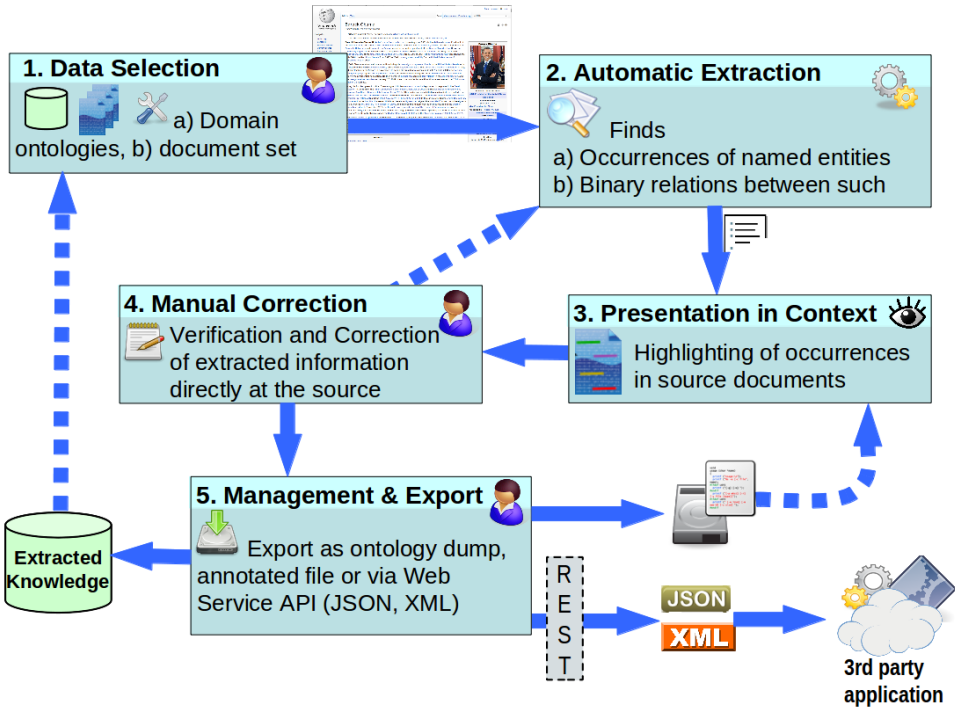


Figure 3.6.: Interactive Extraction Workflow

First (step 1 - data selection) the system needs to be setup with basic information, i.e. an ontology with a type hierarchy, a basic set of entities etc. This needs only be done once in principle. However, a user needs also to select the document corpus to work on, and this can be an incremental task working on small sets of documents at a time. In the second step, the automatic extraction system finds entity occurrences and pattern occurrences. The user than can review the documents and see all entity occurrences and pattern occurrences in their original context (step 3). Based on this a human user should be able to correct wrong entity reference disambiguations or to assign relations to pattern occurrences that have the wrong relations or none at all (step 4). Once a user has done corrections she can send the document corpus back to the extraction system in order for it to apply the new information provided by the user to the whole set of documents. Again a user might revisit the results and at some point might stop his work, e.g. when all is correctly identified. She then has the possibility to export her results, e.g. as an annotated document to feed it into another application or work process. In addition the semantic information can also be retrieved separately from the documents (but on a per document basis) via the REST interface (see Section 3.6.4) or be aggregated into an exported ontology (step 5). The workflow allows this information to be fed back into the process in two ways: Exported

ontologies can be reused in other projects (see Section 3.6) and annotated documents can be re-imported into the annotation process.

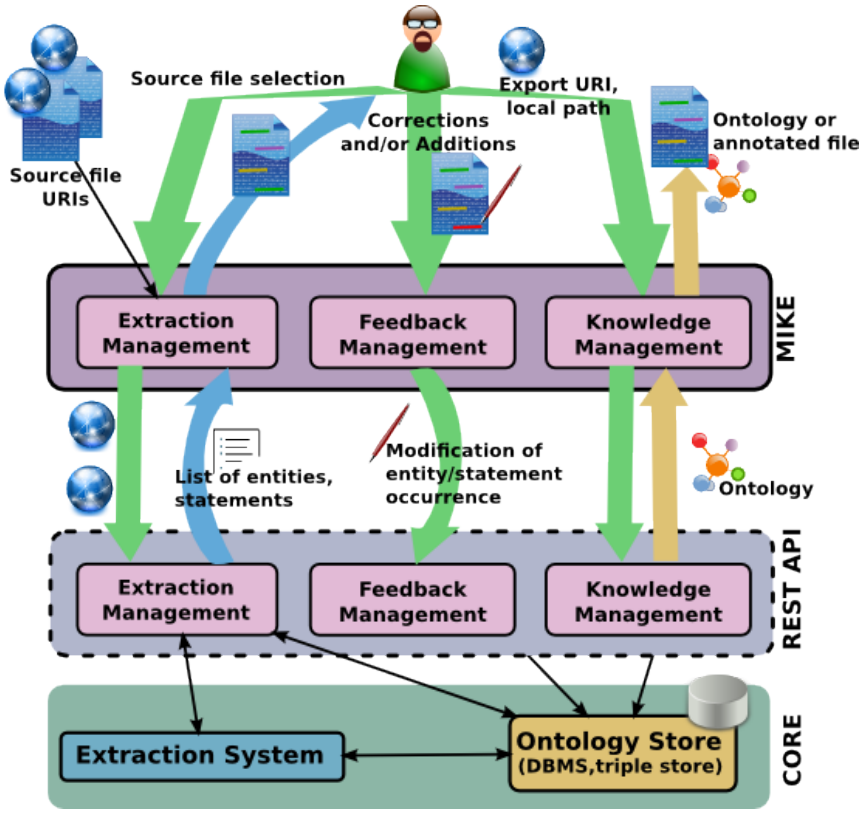


Figure 3.7.: MIKE architecture

3.7.2.2. Architecture

The knowledge workbench realizes this workflow with the help of the feedback capabilities discussed in Section 3.5, the system architecture discussed in Section 3.6 and a special frontend implementation called MIKE. This frontend takes care of user interaction to select documents, start the extraction, navigate and correct the results and the import and export of annotated documents. Information is received from and feedback provided to the backend components via the REST interface discussed in Section 3.6.4. However, MIKE also maintains its own data structures to manage, once received, the extraction information for a set of documents currently processed and it takes care export and import of annotated documents from and into the workflow process respectively. Note that the MIKE frontend was designed in collaboration with and implemented by Michael Stoll as part of his master's thesis [Sto12]. The interactions of the frontend with the remainder of the system are illustrated in Figure 3.7.

3.7.2.3. User experience

A typical user workflow consists of two main iteratively repeated steps: 1) Information Extraction, and 2) Evaluation and Modification of extracted information. After each extraction phase, the user can provide feedback, such that in the next extraction iteration the system can learn from user corrections and apply any new insights on the whole corpus.

Initially a user first has to select source files by providing their URIs (e.g. web urls or WebDAV directories), which will be added to a list of project files to work on. Next, the user may configure the extraction run including the choice of domain knowledge used and which extraction steps (entity recognition, relation recognition) shall be performed. Once the extraction process is started the *MIKE* knowledge workbench displays status information on the extraction progress. When the extraction process has finished, results are retrieved from the extraction system and provided as a listed overview, which may be grouped and sorted by files, entities or statements (Figure 3.8, area 1). In case of grouping by files, the extracted entities and the extracted statements within each file are indicated. In both the other cases, for each entity/statement the files where it is mentioned are listed.

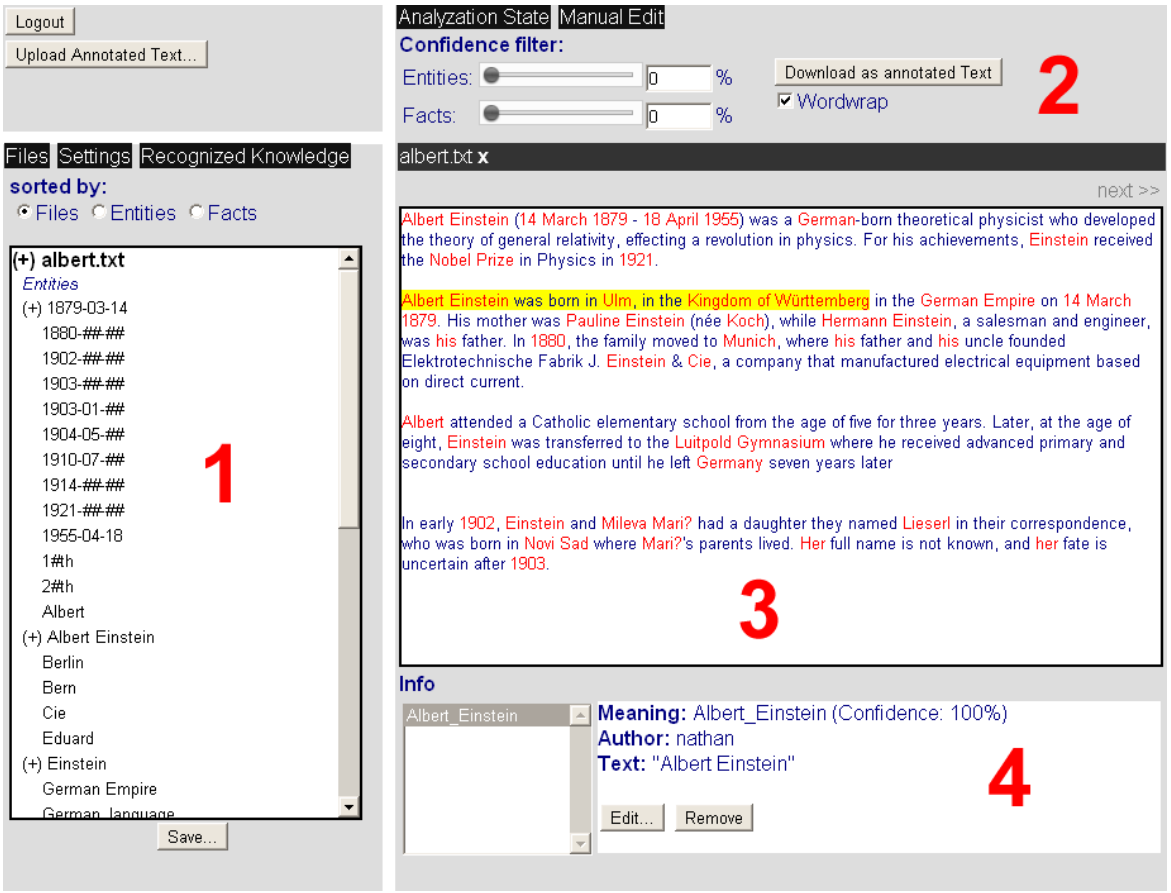


Figure 3.8.: Overview with opened Document

A click on a list item opens the corresponding document within an editor component.

There, its textual content is displayed enriched with highlighted entity and statement occurrences (Figure 3.8, area 3). This way, a user directly sees the extracted information in context allowing her to judge correctness and completeness of the extraction results. Both entity and statement occurrences can be filtered by a confidence threshold based on the disambiguation confidence and the pattern (occurrence) confidence (area 2). Occurrences with a confidence value below the chosen threshold are displayed in a paler color. A click on an entity or pattern occurrence provides further information, such as its author (the user or system name that found or last edited the mention), the recognition confidence and its context (area 4). In case of an entity occurrence also the referenced entity is shown, while in case of a statement all relations recognized between the included entities (at that occurrence) are listed. Entity occurrences may be corrected by changing the referenced entity and/or by adjusting their bounds. For a pattern occurrence a user may support/refute a given relation and/or add a new one. Both types of occurrences can be added by selecting a text part and providing an entity reference or at least one relation, respectively. For adding a statement, however, the selected text already must contain two or more entity occurrences. After giving feedback a user may restart the extraction on all or a subset of documents and the system can re-evaluate all findings based on the given feedback. Finally, the frontend allows for export of edited files in an annotated XML format or of all extracted information in a single dump retrieved from the backend.

3.7.3. German Extraction

As our project partners work (also) with German texts, a point of interest was extraction from German texts. The heuristics adopted from the implementation of [SSW09] to recognize entities and generate patterns work well on English texts. However, although they are relatively simple and not based on a particular language parser both are more problematic when it comes to German language texts. First, entity recognition is mainly based on which words are written in upper-case. While this separates named entities from other words quite well in the English language, in German every known is written in upper-case. Thus, we might still recognize all named entities, but at the prize of trying to disambiguate large numbers of words that are no entity references. While we are aware, that there exist approaches for named entity recognition in German [FP10, ZS09], we are unaware of a German oriented general relation extraction approach, there is however a broad field of work looking into German sentence parsing [THK+04, SSVW09]. In applying the existing framework however, there is another problem. In English it often works out to consider the sentence part between two entities when looking for the main semantic connection between these entities, as the verb typically is placed between subject and object in English. In German however, verbs in many tenses are often constructed in such a way, that only the auxiliary verb is placed between two entities with the main verb expressing their relationship being placed at the other end of the sentence. Consider, for instance, the simple sentence *"Obama was born in Kenya"*. Translated to German this is would be *"Obama wurde in Kenya geboren"*. If we only look at the words in-between both entity references, namely *"Obama"* and *"Kenya"* we will generate an instance of the pattern "X

wurde in Y", which is pretty meaningless as the same pattern would be found in the sentence *"Obama wurde in Kenya zum Essen eingeladen"* or *"Obama wurde in Kenya niemals gesehen"* meaning *"Obama was invited for dinner in Kenya"* and *"Obama was never seen in Kenya"* respectively. While these examples seem regular, the problems increase with more complex sentences, since several other words and subordinate clauses can increase the complexity making it more difficult to identify the "right" words to take into account. In order not to deal with such language specifics we delegate the language parsing to a dependency parser. Similar to the approach described in [SIW06] patterns can then be generated based on the dependency graph generated by the language specific parser. In addition we can restrict entity references to those nouns that are identified as subjects or objects in a sentence and exclude those that are normal noun words. In his master's thesis Sergej Isak-Geidel implemented such an approach in such a way that it can be plugged into the general framework by replacing the first three stages, i.e. entity recognition, disambiguation and pattern generation [IG12]. It is based on [ZS09] used for stemming and to identify noun words and on the dependency parser ParZu [SSVW09] used to generate dependency trees from sentences that then are used to generate patterns.

3.8. Summary and Future Work

In this chapter we have extended an extraction framework with the capabilities to learn from user-feedback, as a side-effect reducing the domain knowledge initially needed to set up the extraction system for a new domain, all the while the ontological modeling of user-feedback allows to provide user feedback as provenance for facts contained in the ontology.

Due to the modularity of the framework, single components can be replaced at different levels in order to adapt to particular domains or users. For instance, for user groups working with other languages than English, the extraction system - or at least the tokenization, entity recognition and pattern collection stages - would need to be replaced, but the remaining framework could be used. Similarly, different expert-levels of users could use different domain specific front-ends developed for their particular domain task on top of the existing backend components. As the existing frontend components and the web-service interface to the backend components suggest (but do not restrict to) web based frontends, the initial setup cost for average end-users is reduced compared to a custom desktop based solution. However, to adapt the framework to particular domains, experts are needed when components need to be replaced or new frontends need being developed. This work provides the general framework, prototype implementations of the methodology as well as user frontends to use them in a particular setting.

While in our solution all relations need to be defined a priori, there is work that could be employed to elevate this last requirement as well by following a more open extraction approach, generating a pattern hierarchy first which then may be translated into relations, see [NWS12] for work in this direction.

Our method to model user testimonies as ontological relations suggests to keep this

information even after the extraction is completed, to provide users curious about particular pieces of ontological information with provenance information in order to allow verification, e.g. by talking to the user who provided the supportive testimonies for a doubted piece of information. However, to this end, it might be beneficial to store additional meta-information, e.g. about the time the testimony was made, from where the user was and potentially general user statistics including his proficiency with the topic at hand etc. This could then be used to automatically provide a basic trust function for user personalization as discussed in Section 3.5.5. In addition, information quality might vary between different sources, be it concrete source URLs or general domains, in both senses, a) web-domains, and b) topical domains. Hence, a trust estimation based on sources might improve the overall quality. After all, this would also fit well with our understanding that both users and source documents are simply sources of information testimonies, that can in both cases be (occasionally) wrong.

4. Provenance based Document Retrieval

4.1. Motivation

Information retrieval (IR) is traditionally based on keyword queries and aims at identifying documents containing a set of keywords. This provides a simple human understandable user interface. Semantically aware techniques, as for instance used in information extraction, can be used to enhance this basic paradigm [PIW10, BZ10, BIPM97, QF93]. Query expansion, for instance, may consider the semantics of the given keywords [QF93]. Thus, the system is looking not only for exact matches of the given keywords but also for semantically similar keywords. Other approaches use information extraction techniques as a filter to rerank retrieved results [BIPM97] or to generate keyword queries using an IE framework [PIW10].

Still, these systems have their shortcomings. Assume, for instance, a user is looking for documents that verify if Barack Obama was born in Kenya using "Obama" and "Kenya" as keywords. Although keyword search might consider techniques such as query expansion and distance based measures, in the end documents are still considered as bag-of-words, such that the system might also output documents mentioning that Obama visited Kenya on a diplomatic mission. Although the search result might be improved by adding keywords such as "birthplace" or "born", query results still potentially contain many false positives and documents using alternative formulations are not found. The search can be improved by making use of the full potential that information extraction offers. If the document corpus has been processed by an extraction tool as discussed in Chapter 3 this provides us with an index over entity occurrences and, in particular, pattern occurrences of patterns that are thought to express the `bornIn` relation. That way, testimonies supporting the statement (`Barack_Obama`, `bornIn`, `Kenya`) can easily be identified. Alternatively, a user might already know or learn from the ontology that there are is another belief that Barack Obama was born in Hawaii which is inconsistent with him being born in Kenya. Whether a user wants to verify a particular statement or is aware of competing statements and wants to estimate which is correct, her intentions are to find testimonies that are strongly supportive of the statement(s) of interest and potentially provide additional information, such that she can decide which statement to accept and which to reject. However, the document corpus may be huge such that a simple lookup of all potential testimonies is an inefficient approach.

Thus, we propose a document retrieval framework that takes a set of statements in the form of RDF triples and retrieves the most relevant witness documents supporting them using the information generated by the extraction tool to 1) identify the witness documents and 2) rank them according to their supportiveness and general relevance.

This allows us to find documents expressing the statement(s) of interest in various ways using different representations of the entities and different textual expressions for the relationships between them. For example, the statement (`Barack_Obama`, `bornIn`, `Kenya`) can be textually expressed in many different forms: *"Barack Obama was born in Kenya"*, *"Obama's birthplace is Kenya"*, *"Kenya, the birthplace of the first African-American president,..."*, etc. In analogy to the term *keyword search*, we refer to this concept as *statement search*. The user input is an arbitrary set of RDF statements that is selected by the user or corresponds to the result of a SPARQL query evaluated over an RDF knowledge base. We also suggest an extension translating a natural language user query into RDF triples using information extraction methods. For instance, given the phrase query *"Barack Obama was born in Kenya?"*, information extraction tools could identify the contained statement in canonic form as the RDF triple (`Barack_Obama`, `bornIn`, `Kenya`).

While statement search on its own can be a valuable tool, for instance, to analyze a large data set based on the statements it contains (e.g. analyzing data releases such as the Wikileaks diplomatic cables or gaining insights form a news collection etc.), it may be valuable in particular when maintaining a large general knowledge ontology as a tool to verify facts that have been extracted automatically.

While many statements would require expert knowledge in their field to be assessed on their own, the problem can be simplified if the task is reduced to verifying whether statements are actually supported by convincing testimonies in witness documents.

To make this statement based document retrieval work, our framework employs the extraction system to provide a) the mapping from statements to textual phrases in the form of pattern-relation combinations and entity name dictionaries and b) the maintenance of entity and pattern occurrence indices, linking concrete pattern instances to their occurrences and c) to provide estimations which pattern occurrences most likely are the most supportive testimonies for a statement.

In particular, an appropriate ranking model needs to consider how well the statement is expressed in the document. For example, a document containing the sentence *"Barack Obama was born in Kenya"* should be preferred over a document stating *"Obama spent his childhood in Kenya"*. This can be achieved by exploiting the confidences associated with the patterns.

There are some other aspects that can indicate more relevant documents.

Consider, for example, again our example statement (`Barack_Obama`, `bornIn`, `Kenya`). Documents containing this statement multiple times should be ranked higher than documents that only casually mention it. This is similar in spirit to the term frequency in traditional IR models.

Another important aspect that an appropriate ranking model may need to consider is the *on-topicness* of a document, as the user might be interested in context information. Biographical pages about Barack Obama, for instance, are more likely to be relevant than a

news article that simply mentions the claim that he was born in Kenya as a side note. Furthermore, authoritative pages such as `http://www.whitehouse.gov/` should be preferred over documents such as blog entries. This is similar to authority-ranking in traditional IR.

Similar to keyword search engines, our statement search approach consists of two main stages. First a document corpus needs to be indexed, then users can perform queries and navigate the corpus. The difference is, that in our approach the information extraction system is generating an index on entity and pattern occurrences rather than on keyword occurrences. While internally queries are sets of statements, the translation between surface strings and semantic statements can be a two-way process allowing users to provide queries in natural language. However, in this work we focus on the document retrieval part and assume a query is given as a set of statements.

In the absense of a sufficiently large document index based on an information extraction system, pattern variants may be used to mimic our approach via a keyword engine. Instead of using the pattern occurrence index we employ, such an approach could use patterns matching the relations in the statement query and entity names matching the entities to generate keyword based queries. With a plain keyword search that considers the words independently each such query will contain some noise, i.e. documents that contain the words but not in a meaningful connection, but when pattern variants combining patterns with entities are formed and the phrases they represent are used to retrieve documents with a phrase search supporting keyword search engine this can mimic a pattern occurrence index. However, such an approach is in principle inherently less efficient, since it is a priori unknown which pattern variants are present in the corpus, i.e. each variant needs to be formed and a lookup in the phrase index for this variant needs to be performed. In addition, such an approach only works with patterns that can be easily represented as a phrase. More complex patterns that apply to tables, are based on deeper sentence analysis and may contain parts lifted to part-of-speech tags or stretch over several sentences are unlikely to be properly represented in a keyword query.

Contributions We propose a *statement search* framework that enables semantic-aware document retrieval based on queries consisting of a set of semantic statements and given such a query retrieves witness documents expressing these statements in different textual forms. We provide an appropriate ranking model based on statistical language-models that considers the following criteria: 1) statement frequency, 2) pattern confidence, 3) entity occurrences, and 4) page authority. Our framework is modular in the sense that it can be applied outside of the framework described in Section 3.6 and is independent of the particular extraction system used, as long as another extraction system provides the required information, namely (weighed) entity names, entity occurrences, (weighed) patterns for (binary) relations and pattern occurrences. We also provide an evaluation that compares our approach against typical keyword based search representations of the queries.

Outline The rest of this chapter is organized as follows: First related work is discussed in the next section. Then, in section 4.3 we define our understanding of a statement search. While Section 4.4 introduces the framework implementing a statement search, Section 4.5 presents the ranking model that the framework relies on. Section 4.6 presents evaluation results. In Section 4.7 a prototype frontend application is presented. And finally, in Section 4.8 we discuss an approach to estimate witness documents in the absense of a sufficient extraction based index via keyword queries to confirm (or reject) semantic statements.

4.2. Related Work

Utilizing information extraction (IE) techniques to support keyword-based document retrieval has been studied in the literature [BIPM97, Hea92, QF93]. While applying such techniques can improve the performance of a keyword-based document search engine by providing some understanding of the underlying information need of a user, these approaches usually only use additional techniques extending existing information retrieval (IR) algorithms.

Recently published approaches try to integrate ontological knowledge databases with document retrieval techniques. Pound et al. [PIW10] propose a framework using descriptive structured queries with keywords as basic operators. A query is assumed to describe a set of entities by a set of constraints. After resolving the entities that satisfy the query constraints using an ontological knowledge base, the framework retrieves documents containing references to those entities. However, the retrieved documents do not necessarily contain the relations given as constraints in the query. Hence, looking for a US President born in Hawaii, President Obama would probably satisfy the statement and the retrieved documents would include documents containing Obama and a reference to Hawaii, but the documents are not necessarily talking about the claim that he was born in Hawaii. In contrast, our approach also ensures the presence of the relation - applied to the entities - in retrieved documents. While the proposed framework is similar to ours to some extent, the authors focused on the query language and used an ad hoc tf-idf approach for the ranking whereas our work focuses on the ranking of result documents. Still, the proposed query language could be combined with our ranking mechanism.

Bear et al [BIPM97] couple an NLP information extraction tool with an information retrieval engine by using it as a re-ranking filter for result lists obtained from the IR engine. They report only small gains by their technique compared to the results of the IR engine alone. A similar combination is proposed by Hearst [Hea92] for the particular domain of topic based sentiment analysis.

Similarly, Blanco, and Zaragoza [BZ10] examine ranking of support sentences for entities. Given a keyword query, they aim at finding entities in context answering the query, i.e., support sentences that explain why an entity is an answer to the query. The paper compares a number of different ranking approaches for the support sentences. In contrast to our approach, the sentences are ranked based on entities, not on statements.

Fagin et al. [FKL+10] investigate how to interpret keyword queries, given an auxiliary database providing semantic concepts for entities and relations. Given a keyword query, their technique can produce various interpretations that (partially) match the given keywords to semantic concepts or relations. For instance, it is not clear whether the keyword query "buzz price" refers to documents containing those words explicitly, documents about the Disney affiliated economist Harrison "Buzz" Price, or documents stating the price of the "Buzz Lightyear" toy. These *parses* of a keyword query – and documents containing expressions of such a parse – can be ranked by their specificity. Using this terminology, our approach is given a specific parse (or a set of parses) as input. Instead of considering different interpretations of the query, the problem we focus on is to rank the documents

containing textual manifestations of the given (set of) parse(s). Similarly, other approaches that try to interpret natural language queries, e.g. for question answering, could be employed as a frontend to translate user queries [Kat97, Sga82, HBR14, UBL+12].

[LDH06] investigages the other way around, how to turn a structured query into a keyword query by selecting the right entity and relation labels. We use such a translation in the evaluation to formulate keyword queries in order to compare our method with a keyword based document retrieval. However, as our queries are relatively small, we think it natural to fully translate the query without a selection of important labels. In our case we also can use the patterns as special already weighed labels (see Section 4.6.

Sereno et al. [SSM05] as well as Uren et al. [USBL06] propose a set of tools to annotate documents, especially research documents, with ontological expressions (claims) that form a claim network that can be browsed and searched. While claims can be seen as statements, their annotating approach is semi-automatic while our approach tries to automatically annotate documents with contained statements. This kind of search analysis focuses on the claim network, while we investigate document based rankings.

Our ranking model for witness documents is based on language models. Due to their good empirical performance and sound theoretical foundation, approaches based on language models have received much attention in the IR community. Many variations of the basic language model approach have been proposed and studied. Language models have been applied to various retrieval tasks such as cross-lingual retrieval [XWN01], expert finding [BAdR09], XML retrieval [Hie03]. In particular, recently Elbassuoni et al. [ERS+09] proposed a language-model-based ranking for structured queries over RDF graphs. Also, Nie et al. [NMS+07] used language models to retrieve and rank entities given a keyword query. However, our retrieval model is the first to adopt language models for statement search over documents.

An overview of semantic search for textual documents is given in [Man07], which surveyed 22 systems, none of which providing any document ranking; the authors actually point out the necessity of research in that direction. The Semantic Desktop [SSS08] is a semantic search engine that combines fact retrieval with semantic document retrieval using a triple-based algorithm and graph traversal. Given a natural language query, the approach tries to infer a structured query and retrieve matching triples. If there is no perfect answer, the ontology that the queries are evaluated against is used to expand the user query, and the expanded query is evaluated on a document collection to retrieve matching documents. RankIE [BBH+09] is a system for entity-based document search that can exploit structural background knowledge such as ontologies. It includes the ExplainIE tool [BBM09], which was developed to debug IE results in business intelligence applications, mainly by visualizing how entities and documents are related.

Semantic Web search engines try to find RDF content on the Web based on keywords or URIs. To rank the sources of relevant content, they often use variations of the the PageRank algorithm [DFJ+04, HHD+07] or the tf-idf measure [ODC+08]. Semantic Web search engines focus on locating RDF sources and do not consider provenance of RDF data. An important aspect in this context is how to present results to a user, the problem of snippet generation for facts has been studied a lot [BDT08, PWTY08].

4.3. Statement Search

As explained in the introduction, *statement search* aims at retrieving a set of relevant documents to a set of given factual statements. This type of search can be motivated by two slightly different information needs. First, a user might want to verify whether a certain statement is true. For instance, a user might wonder whether Barack Obama was born in Kenya or Hawaii and seek documents confirming either statement. Second, a user might want to investigate a certain statement, i.e., learn more details about it. For example, the user might hear that Heath Ledger acted in the movie The Imaginarium of Doctor Parnassus but wants to find more details about his involvement (which was shortly before his death). In both cases, the user is interested in documents referencing the statement in some way. However, with respect to these two user goals, a document is relevant to a statement query if it either verifies the statement *(persuasiveness)* or provides further information about the statement topic *(on-topicness)*. These two main properties might compete with each other in determining the relevance of a given witness document. There might be documents that very clearly and convincingly support the statement (*persuasive*) but only casually mention the statement and are mainly concerned with another topic (not *on-topic*). For instance, the Wikipedia article about the Kapiolani Medical Center in Honolulu, Hawaii, might clearly state that President Obama was born there, but then would provide no further information about his childhood. A blog post discussing Barack Obama's childhood on the other hand could provide more information about the American president's youth but might be a less convincing source for the verification of his birthplace. The quality of a witness for a given query therefore depends on the user's preference on these two aspects. Our ranking model incorporates different features estimating a document's *on-topicness* and its *persuasiveness* in a way that allows to adapt the ranking according to a user's preferences. Still, a perfect witness would satisfy both properties to the full, being informative on the topic and persuasive for the statement(s) in the query. Thus, at the core, our ranking model aims at finding the best documents based on the probability that they are a perfect witness for a given set of statements. In our model, the quality of a witness is therefore based on the following aspects:

1. Given a set of statements g, a witness can only be perfect if it covers all statements in g. This affects both properties, as a statement cannot be verified with a document that does not contain the statement, and it is also unlikely that additional information related to the statements can be found there.

2. A document is only informative to a user if she can learn more about the statement(s) of interest. Therefore, the document needs to focus on the topic given by the statement(s). This is the main aspect of on-topicness: that more information on the topic needs to be present.

3. Statements can be expressed in various textual forms. Some of these statement representations are more directly associated with the abstract statement than others. A witness can only be used to verify a statement if the statement is expressed clearly

within the document's content. A perfect witness will formulate the statement(s) in a way that leaves no room for other interpretations. This is the main aspect of persuasiveness, if the expression of the statement is vague it will not convince a user.

4. In general, we trust information obtained from some documents more than information contained in others. For instance, if the White House website issues a birth certificate for President Obama stating that he was born in Hawaii while a conspiracy blog page states that he was instead born in Kenya, ignoring other factors, most people would rather believe the White House. Both, the statement(s) of interest expressed in the documents (persuasiveness) as well as further information *(on-topicness)* are subjects to trustworthiness.

Our ranking model, introduced in detail in Section 4.5, takes all these considerations into account. Before we dwell into the details of our ranking model and how it achieves the aforementioned goals, we describe the system framework in which it is employed in the next section.

4.4. Architecture

In principle our system is a document retrieval system that employs the links between factual statements and the text sources linked to them by an information extraction (IE) tool to enable a *statement search*. That is, we assume a user either formulates a phrasal query which is translated or she uses an ontology based frontend to formulate a statement query. Given a query consisting of a set of statements our system retrieves the most relevant witness documents supporting them and displays the results to the user. Figure 4.1 gives an overview of the architecture and internal workflow of our system.

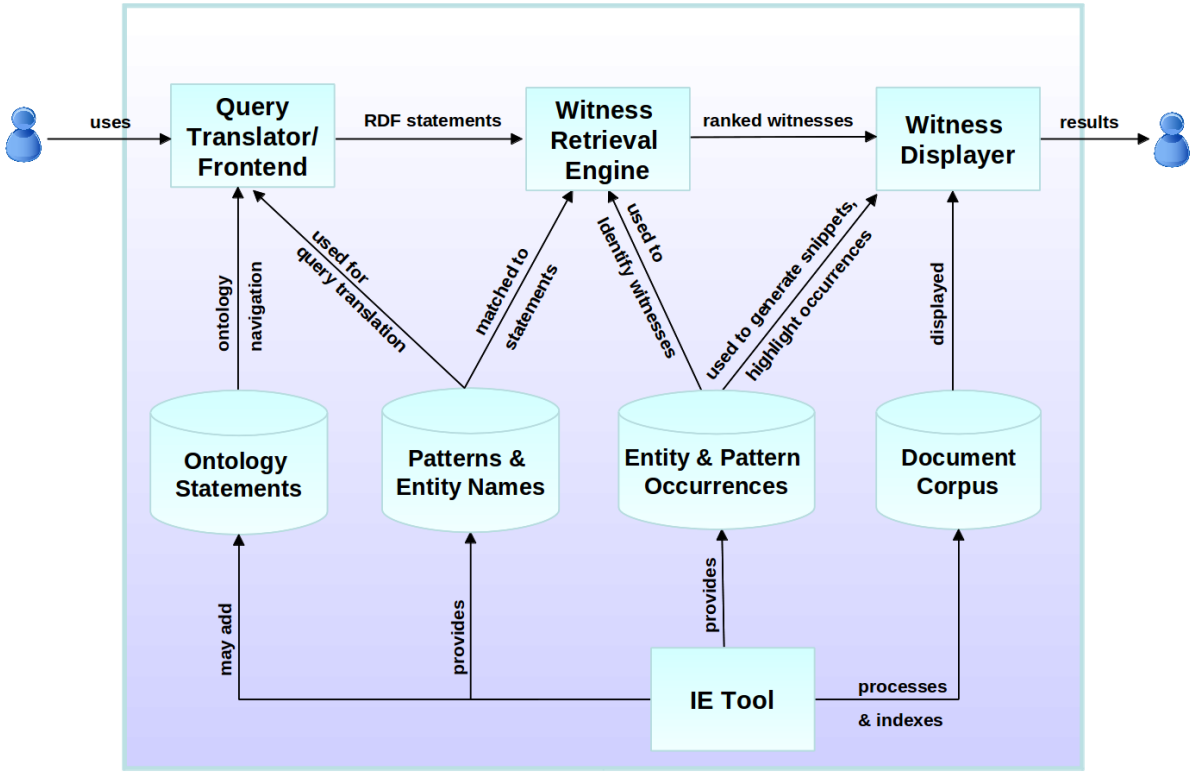


Figure 4.1.: Architecture

In a nutshell, our system works as follows:

1. The document corpus is processed by an information extraction (IE) tool. This means in particular
   - entity and pattern occurrences are identified in each document
   - pattern confidences are computed
   - potentially new statements are added to an existing ontology.

2. A user formulates her query, either with the help of a translation tool that uses entity names and relation patterns to translate a natural language query or by means of an ontology frontend which allows the user, for instance, to navigate the ontology and select statements of interest. Either way, this results in a set of statements g referring to entities and relations contained in our ontology.

3. The system retrieves the documents that match these statements and ranks them based on how likely they are to be relevant witnesses to the given statements.

4.4.1. IE Tool

The information extraction tool is responsible for processing the documents in our corpus and providing and maintaining entity names, patterns, confidences linking patterns to relations and indices linking patterns and entities to their occurrences in documents of the corpus. Generally speaking, we can use any information extraction tool, if it provides us with the required data. In our setup however, the modified version of the SOFIE [SSW09] framework described in Section 3.3 (without user feedback) was used.

We call the combined output needed from the extraction system, i.e. dictionaries and indices, a *statement index*, as it allows to locate textual expressions of statements.

4.4.2. User Frontend

The user frontend is typically responsible for generating an RDF statement query, a set of RDF statements that our system can process, given input from the user. The original input could either be a text query that is then translated, the user could browse an ontology in order to improve or explore it and select facts of interest or she could, for instance, use a SPARQL query engine to retrieve triples.

For a translation, the frontend can utilize the entity names and pattern relation mapping available through the IE tool or simply apply the IE tool to perform the translation of the query into a set of statements is used to identify statements and entities in the document corpus. Assume, for instance, a user wants to verify that Obama was born in Kenya, she might provide the phrase query *"Obama was really born in Kenya?"*. It would then be translated into the statement triple (`Barack_Obama`, `bornIn`, `Kenya`) and the system would output witnesses containing the statement. We do not provide or suggest a concrete query translation tool; such a query translation can be in its own a challenging problem related to natural language (query) understanding [FKL+10, Kat97, Sga82, UBL+12].

However, we do provide a frontend that allows a user to browse the existing ontology via a graph visualization, issue SPARQL queries or search for entities and then select an arbitrary set of facts to form a query. Alternatively, queries can be given directly as a set of RDF triples. This way our system can, for instance, be used as a verification tool for automatically extracted ontologies. Assume, for instance, a movie knowledge base generated by an information extraction tool. Then, a user might consider it to be a mistake if she sees four different statements claiming that the role of the character Tony

in the movie "The Imaginarium of Dr. Parnassus" was played by four different actors. Using our framework, she could issue the set of all four statements to see which of them is more likely to be true. This might lead her to a document containing expressions of all four statements, which explains that three actors replaced Heath Ledger as he died in the midst of the production such that they all played the same character at different points in the movie. This frontend enabling our statement search module to be used as a tool for ontology maintenance is described in Section 4.7.

4.4.3. Witness Retrieval Engine

The witness retrieval engine is responsible for retrieving *any document that matches the given query* (i.e. at least one statement of the query is indicated in the document) and rank this set of *witnesses* according to the ranking model described in Section 4.5. It first identifies the pattern mentions supporting any of the statements in the query. Assume a query $g = \{t_1, ..., t_n\}$ where $t_i = (s_i, r_i, o_i)$ is given. Now, in this setting we typically assume all entity occurrences are completely disambiguated, hence we identify all relevant pattern mentions $\mathcal{PM}(t)$ for a statement t by:

$$\mathcal{PM}(t) = \{p(sn_d, on_d) \in \mathcal{PM} | \texttt{expresses}(\texttt{p}, \texttt{r}) \wedge entity(sn_d) = s \wedge entity(on_d) = o\} \quad (4.1)$$

Having the relevant pattern mentions for each triple, we can use them to identify the the set of all witnesses for the statement: $\mathcal{W}(t) = \{d \in \mathcal{W} | \exists p(sn_d, on_d) \in \mathcal{PM}(t)\}$. The set of witnesses for g then is the union over the witness sets of all the statements in g:

$$\mathcal{W}(g) = \bigcup_{t \in g} \mathcal{W}(t) \quad (4.2)$$

Once the witnesses are identified, any witness for g is considered as a candidate and ranked according to the ranking criteria described in Section 4.3. The details of the ranking approach are discussed in Section 4.5

4.4.4. Witness Displayer

Once witnesses are ranked, they are passed to the witness displayer which is responsible for displaying the ranked witnesses to the user. In principle, the witness displayer is decoupled from the retrieval of witnesses and the ranknig model. Thus the witness displayer can utilize a variety of techniques to display the witnesses to the user in response to its original query. A typical setup in document retrieval setting would be to present first the list of witness documents, presenting for each its title, a link to the full document and a portion of the content, typically called *snippet*, representing the most relevant piece of the document for the query as an explanation for the listing of the document and brief insight into its content. In order to allow such kinds of witness display solutions, our witness retrieval engine provides the following meta-data for each witness: 1) the witness title, 2) the witness URI, 3) a text snippet constructed from statement occurrences in the document. The snippet construction is discussed in Section 4.5.4. Our witness display solution is discussed in Section 4.7.

4.5. Witness Retrieval and Ranking

As explained in the previous section, our ranking model is concerned with ranking a set of witnesses that are relevant to a given set of RDF statements. Note that we decouple the problem of transforming the user query into RDF statements from the ranking of witnesses problem since they are basically independent problems. We thus assume that the transformation of queries into RDF statements is deterministic. Our ranking model can easily be extended to handle the case where this transformation process involves inaccuracies. However, our model aims on the problem of ranking the witnesses given a set of RDF statements.

Our ranking model is based on statistical language models (LMs) [PC98, Hie00]. The witness documents are ranked based on the probability of being relevant to the query statements $\{t_1, t_2, ..., t_n\}$, which we denote as $P(d|t_1, t_2, ..., t_n)$ (here, d is a witness). Applying Bayes' rule, we have:

$$P(d|t_1, t_2, ..., t_n) = \frac{P(t_1, t_2, ..., t_n|d)P(d)}{P(t_1, t_2, ..., t_n)} \tag{4.3}$$

Since $P(t_1, t_2, ..., t_n)$ does not depend on the witnesses, we can ignore it during the ranking. This implies that

$$P(d|t_1, t_2, ..., t_n) \propto P(t_1, t_2, ..., t_n|d)P(d) \tag{4.4}$$

$P(d)$ is the prior probability that witness d is relevant to any statement. This probability can be estimated in various ways, and in our case we estimate it using the static authority of the page or pagerank [BP98]. We do this in order to take into consideration the trustworthiness of the witnesses (i.e., give higher weight to witnesses that are authoritative).

As for the probability $P(t_1, t_2, ..., t_n|d)$, we assume independence between the query statements for computational tractability (in-line with most traditional keyword ranking models). Thus,

$$P(t_1, t_2, ..., t_n|d) = \prod_{i=1}^{n} P(t_i|d) \tag{4.5}$$

To avoid over-fitting and to ensure that the witnesses that do not contain all the query statements do not have a zero probability, we use Jelinek-Mercer [ZL01, JM80] smoothing:

$$P(t_1, t_2, ..., t_n|d) = \prod_{i=1}^{n} [\alpha P(t_i|d) + (1 - \alpha)P(t_i|Col)] \tag{4.6}$$

The first component is the the probability of the statement t_i being generated by document d and the second component is the probability of generating the statement t_i by the whole collection Col.

We rely on two different methods to estimate the probability $P(t_i|X)$ of generating statement t_i using X, where $X \in \{d, Col\}$, an unbiased estimator only based on pattern instances representing the statements (see Section 4.5.1) and an entity-biased estimator that also takes into consideration entity occurrences independent of pattern occurrences to

estimate the likelihood that the document focusses on the entities of interest (Section 4.5.2). While we typically assume entities to be completely disambiguated, we also provide a third variant that takes uncertain disambiguations into account (Section 4.5.3).

4.5.1. Unbiased Estimator

Since statements can appear in different forms in witnesses, we need to first fold a statement into all corresponding indicative patterns instances that can be used to represent the statement in order to estimate the probability $P(t_i|X)$. This is similar to translation models [Zha08], when the query is expressed in one language, and the documents retrieved are in a different language. Let the set $Y = \{y_1, y_2, ..., y_m\}$ be the set of all pattern mentions $\mathcal{PM}(t_i)$ for t_i. The probability $P(t_i|X)$ is then computed as follows:

$$P(t_i|X) = \Sigma_{j=1}^{m} P(t_i|y_j)P(y_j|X) \tag{4.7}$$

The first component $P(t_i|y_j)$ in Equation 4.7 is the probability of representing the statement t_i using pattern mention y_j, which is computed as a function of the confidence of representing the relation of t_i using the pattern of y_j as we assume the entity references to be disambiguated with absolute certainty. Assuming that the relation of t_i is r_i and the pattern of y_j is p_j, we have: $P(t_i|y_j) = conf(p_j, r_i)$. Note that $P(t_i|y_j)$ would be *zero* if r_i cannot be expressed using pattern p_j.

The second component $P(y_j|X)$ in Equation 4.7 is the probability of generating pattern mention y_j given X where $X \in \{d, Col\}$. This is estimated using a maximum-likelihood estimator as follows:

$$P(y_j|X) = \frac{c(y_j; X)}{\Sigma_{y \in Y} c(y; X)} \tag{4.8}$$

Here, $c(y; X)$ denotes how often the pattern mention y occurs in X.

4.5.2. Entity-Biased Estimator

The unbiased estimator only takes into consideration the occurrence of the statements in the witnesses. It does not take into consideration what other statements occur in the witness and whether or not they are related to the query statements. For instance, assume the user query consisted of the statement t =(`Barack_Obama`, `bornIn`, `Kenya`).

Also assume that witnesses d_1 and d_2 both contain x statements, and that the query statement appears in each witness only once. Thus, $P(t|d_1) = P(t|d_2)$. However, assuming that d_2 contains more references to Obama than d_1 which is a news directory for instance, we would like to rank d_2 higher than d_1. To this end, let the statement t_i be of the form: *(s, r, o)* with subject s, predicate r, and object o. The probability $P(t_i|d)$ is now defined as follows:

$$P(t_i|d) = \beta_s P_e(s|d) + \beta_o P_e(o|d) + (1 - \beta_s - \beta_o) P_t(t_i|d) \tag{4.9}$$

where $P_e(s|d)$ and $P_e(o|d)$ are the probabilities of generating the subject s and object o using witness d respectively, and $P_t(t_i|d)$ is the probability of generating the whole statement t using d. The parameters β_s and β_o control the influence of each component.

The probability $P_t(t_i|d)$ is estimated using the unbiased estimator from Equation 4.7. The probabilities $P_e(s|d)$ and $P_e(o|d)$ are in turn estimated using the frequency of occurrence of s and o in d, respectively. For instance, $P_e(s|d)$ is computed as follows:

$$P_e(s|d) = \frac{\{\tilde{s} \in \mathcal{E}_o(d) | entity(\tilde{s}) = s\}}{|\mathcal{E}_o(d)|} \tag{4.10}$$

4.5.3. Uncertain Entity Mapping

So far we only considered the confidence the extraction system places in a pattern. However, as with patterns, the disambiguation mapping from entity mentions in the text to the canonical entities might be uncertain. Assume that instead of considering the functional entity mapping given by $entity(sn_d)$, we use the probabilistic mapping given by $entity(sn_d, s)$ that for each entity mention sn_d and each possible entity s provides the probability that sn_d refers to s in d. Our model can easily be adapted to deal with this additional uncertainty by redefining two components.

1. $P(t_i|y_j)$, the likelihood that a statement t_i =(s_i, r_i, o_i) is generated by a pattern mention $y_j = p_d(sn_d, on_d)$ now has to be computed based not only on the pattern confidence, but also needs to take into account the confidence that both entity mentions sn_d,on_d of y_j are correctly mapped to s_i and o_i. If we assume independence between all three components, this can be done by computing $P(t_i|y_j)$ as the product over the likelihood of all three parts, i.e.

$$P(t_i|y_j) = entity(sn_d, s_i) \cdot conf(p, r_i) \cdot entity(on_d, o_i) \tag{4.11}$$

2. In addition, the biased estimator needs to adapt its estimation on entity occurrences by taking into account the disambiguation confidence, i.e. $P_e(s|d)$ is computed by:

$$P_e(s|d) = \sum_{sn_{d@pos} \in \mathcal{E}_o(d)} \frac{entity(sn_d, s)}{|\mathcal{E}_o(d)|} \tag{4.12}$$

4.5.4. Snippet Generation

Once the source documents have been ranked, we present to the user the $top-k$ documents. For each of these documents, a short snippet that contains pattern occurrences with some additional context for the triples the user selected can be viewed. Assuming we can only present m pattern occurrences in the snippet, we retrieve these m pattern occurrences according to the following strategy:

- First, we rank the pattern occurrences for each of the n triples t_i present in document d according to the confidence values of the respective patterns to retrieve a ranked lists $L_1, L_2, ..., L_n$.
- We go around in round-robin fashion and retrieve the top pattern occurrence from each list L_i for $i = 1$ to n and remove them from the corresponding lists.
- If we still have not retrieved all m pattern occurrences, we repeat the previous step until we retrieve m patterns or none are left.

Of course, we can also revise the previous strategy to give priority to lists with the highest confidence value. That is, during each round, we retrieve the pattern with the highest confidence value first, rather than retrieving the pattern from L_1 then L_2, etc. And in case entity name weights are present the pattern occurrences are not only ranked by the pattern confidence, but also by the entity name prior.

Note that, so far, we do not employ any means of diversification or duplicate avoidance. This would be a natural extension and could, for instance, be achieved by re-ranking L_i each time a pattern instance is picked to be included in the final snippet, such that the ranks of all other instances of the same (or a similar) pattern are increased, i.e. pushed back within the ranked list.

Consider for instance a search for the statements t_1 =(`Barack_Obama`, `bornIn`, `Hawaii`) and t_2 =(`Barack_Obama`, `bornIn`, `Kenya`). Now assume three patterns p_1 ="X 's official place of birth, Y", p_2 ="X ' birthplace is claimed to be Y" and p_3 ="X went to school in Y". Hopefully we can agree that the patterns express a `bornIn` relation with decreasing persuasiveness, let us assume the extraction system's pattern analysis came to the same conclusion such that $conf(p_1, \texttt{bornIn}) < conf(p_2, \texttt{bornIn}) < conf(p_3, \texttt{bornIn})$. Given a document d that contains an occurrence of patterns p_1 and p_3 that correspond to t_1 and for p_2 an occurrence that corresponds to t_2. Let $m \geq 3$. Then first the pattern occurrence of p_1 will be used in the snipped, then that of p_2 and finally the one of p_3. As context around the pattern occurrences is included, we could end up with a snippet like this *"While Obama's official place of birth, Hawaii, is often disputed ... in these circles the president's birthplace is claimed to be Kenya, the home country ... not bothered. Obama went to school in Hawaii and completed his undergraduate ...."*. Note, that even if both patterns that support t_1 had a higher confidence, the snippet would provide alternating support for the statements present, thus the cites included in snippets can be in different order than they appear in the document.

4.6. Evaluation

In this section, we discuss a four-fold evaluation of our document retrieval model. The general setup of our evaluation environment is outlined in Section 4.6.1. Then, we first discuss what effect the parameters in general have (Section 4.6.2). Afterwards, the performance of our ranking model is evaluated based on several concrete model settings tailored towards different use cases. We use a naive approach that only finds documents matching the statement query, i.e. documents with matching patterns, and then applies an arbitrary ranking, as a simple baseline (Section 4.6.3). Third, our retrieval approach is compared to a purely keyword-based retrieval approach, that is not aware of any statement indication found by the extraction engine (Section 4.6.4). This shows the result quality our approach can provide to a user that otherwise would use a traditional keyword search. Finally, we investigate a combined approach, that uses our statement document index and then applies a keyword search only in the prefiltered set of witnesses, in one variant also making use of statement patterns to generate the keyword queries given a statement query. With this approach we try to give an intuition in addition to the naive ranking approach in how much influence the filter and the ranking have and how much a keyword search can gain from the statement index based prefiltering. This approach we also compare against our model and discuss how both relate to each other (Section 4.6.5).

4.6.1. Setup

Dataset & Document Pooling We use the English part of the ClueWeb 09 document corpus[1], a standard IR benchmark collection containing about 500 million English language documents harvested from the Web. The dataset comes with PageRank values[2] which we used as page trust estimation. To limit the size of the corpus, we constructed a document topic subset by selecting 43 well-known entities, i.e., politicians (e.g., Barack Obama), actors/directors (e.g., Clint Eastwood), soccer players and clubs (e.g., David Beckham), and scientists (e.g., Albert Einstein), to retrieve the top-1000 documents based on a BM25-based score [RZ09] for different names and spelling variants of these entities taken from the general purpose knowledge base YAGO [SKW07]. We also filtered out SPAM pages using the Waterloo Spam rankings[3]. This resulted in a pool of 182,139 documents, on which a modified version of the SOFIE/PROSPERA [SSW09, NTW11] extraction system was run that keeps track of fact provenance information as described in Chapter 3. The iterative components (pattern analysis and reasoning) were repeated eight times.

Queries We defined 56 queries in the form of statement sets. For instance, one query with a single statement would be the set {(`Barack_Obama`, `isLeaderOf`, `USA`)}, while one with multiple statements focusing on JFK's assassination would be given by {(`JFK`, `diedOn`, `1963-11-22`), (`JFK`, `diedIn`, `Dallas`)}. We only considered statements that were supported

[1] `http://boston.lti.cs.cmu.edu/Data/clueweb09/`
[2] `http://boston.lti.cs.cmu.edu/clueweb09/wiki/tiki-index.php?page=PageRank`
[3] http://durum0.uwaterloo.ca/clueweb09spam/

by at least 10 documents (usually more). The full set of queries used can be found in the appendix, Section A.1.1. For each query, a set of potentially relevant documents was generated by pooling the top-50 rankings using various settings of our ranking mechanism. Remember that this includes an implicit filtering step (only witnesses considered) such that only documents containing an indication for at least one statement of the query were considered for the ranking. For the system comparison (see Section 4.6.4), additional documents retrieved by the (statement indication unaware) runs based on keyword search were added to each pool. This resulted in witness pools of 80 witnesses per query on average with a minimum of 11 documents and a maximum of 435 documents for a single query. From those 56 queries, 40 contain a single fact and 16 contain multiple (two to four) statements.

Human Assessments As outlined earlier, our framework aims at two different use cases. First, a user might aim to learn more about specific statements; second, she might be interested in verifying the validity of some statements. Reflecting these two use-cases our system aims to support, all documents in the pool were assessed 1) as a whole document on their on-topicness (does it contain more information relevant to the query than the actual statement indication) and 2) separately for each statement on their persuasiveness (how strong is its support for this statement). For both assessment dimensions a graded relevance scale with three grades (*non-relevant(0), somewhat relevant(1), highly relevant(2)*) was used. This means, documents providing some more information would be rated *somewhat relevant* on the on-topicness scale, while those actually focusing on a fact of interest or one of the entities would be considered *highly relevant*. Similarly the grades for the persuasiveness reflect whether the respective fact is clearly stated, ambiguously indicated or not present at all. For instance, the phrase "Germany sends 500 additional troops into Afghanistan's North" would be clearly stating that Germany is militarily involved in the current war in Afghanistan, meaning the document would be rated as clearly supporting the fact. Another document containing no other reference to Germany and Afghanistan than the phrase "Germany may increase its commitment in Afghanistan" would be only ambiguously indicating the fact that Germany participates in the Afghan war to the unknowing user. For binary measures such as MAP we project the graded assessments to a binary relevance scale, such that a statement or document is considered *relevant* in the binary scale if it is *highly relevant* in the graded scale.

Measures We apply different measures for result quality that show 1) the general quality of resulting rankings and 2) the effort a user needs to spend until finding convincing evidence for all statements of her query. First, we compute the mean average precision (MAP) and the normalized discounted cumulative gain (nDCG) for the top 5,10, and 20 results, averaged over all queries. Second, we compute the mean satisfaction rank (msr), i.e., the average first rank at which at least one convincing (*highly relevant*) indication has been seen for each statement in the query and thus the average rank at which a user at least potentially might be satisfied. Analogously, we also compute the mean reciprocal

rank (mrr) based on the first rank where for each statement of the query at least one convincing indication has been seen.

4.6.2. Parameter Analysis

relevance type	measure	grading scheme	β_s						
			0	0.1	0.3	0.5	0.6	0.8	1
persuasiveness	MAP	`binary`	0.206	0.176	0.163	0.157	0.156	0.154	0.154
persuasiveness	nDCG	`graded`	0.86	0.81	0.79	0.78	0.78	0.77	0.76
persuasiveness	mrr	`binary`	0.89	0.80	0.76	0.75	0.74	0.74	0.74
on-topicness	MAP	`binary`	0.111	0.139	0.156	0.160	0.161	0.162	0.162
on-topicness	nDCG	`graded`	0.568	0.662	0.687	0.694	0.7	0.7	0.7

Table 4.1.: The effects of β_s with $\beta_o = 0$, quadratic pattern confidence influence, pattern weighing on, $\alpha = 0.5$ and pagerank on, measured at k=10

Our model provides several possibilities to adjust it to a user's needs. The entity bias can be controlled by the β_s and β_o variables (see Equation 4.9). Similarly the influence of the smoothing versus the main ranking function can be influenced by setting the α parameter (Equation 4.6). Additionally our implementation allows to switch the influence of document trustworthiness (i.e., pagerank in our implementation) on or off (when switched off, we simply assign a constant value to $P(d)$, i.e. 1, for all documents d). Similarly, the impact of the pattern confidence for computing $P(t_i|X)$ in Equation 4.7 can be adjusted. It can either have no influence, a linear influence or quadratic influence.

We have investigated the effects of different settings of these parameters. The pagerank values, which we use as an approximation of a document's probability to contain accurate and extensive information, do not seem to have a significant impact. This might have several reasons: 1) Our query set did not contain many disputed statements where we saw "untrustworthy" witnesses. 2) SPAM pages were already sorted out using the Waterloo Spam rankings. 3) We use pagerank only as a page trust approximation. Pagerank might not always be a good indicator that the information provided is of high quality. A more accurate representation of document trustworthiness could be generated by using user feedback either directly provided or indirectly harvested from click-rates or apparent user satisfaction. Also, as a bootstrapping method a mapping of document's domains (in case of a web dataset) to owner of the domain could provide a basic distinction between government, non-government organization, commercial and private web-pages where each group could have its own base trustworthiness assigned. Similarly machine learning mechanisms could be taught to distinguish between user-content driven pages, e.g. commentary sections, privately provided information, e.g. blog posts, and information from "official" web-sites. Also a more fine-grained approach to detect SPAM paragraphs or categorize separate paragraphs of a document could help to judge the usefulness of a pattern occurrence.

However, we did not further investigate such an approach.

relevance type	measure	grading scheme	on	off
persuasiveness	MAP	`binary`	0.187	0.125
persuasiveness	nDCG	`graded`	0.824	0.75
persuasiveness	mrr	`binary`	0.823	0.752
persuasiveness	msr	`binary`	1.824	2.241
on-topicness	MAP	`binary`	0.102	0.099
on-topicness	nDCG	`graded`	0.58	0.597

Table 4.2.: The effects (averaged) of Pattern Frequency Weighing when $\beta_s = 0 = \beta_o$

The document trustworthiness estimated by pagerank aside, our model control parameters, in general, provide the effects one would expect. For instance, increasing the β values (see Equation (4.9)) tends to improve results on the on-topicness aspect, but might decrease the results with respect to the persuasiveness (see Table 4.1), while a higher confidence influence tends to have the inverse effect. However, those parameters are quite interdependent, so that there is no definite behavior that applies to each setting, e.g., increasing one β value means decreasing either the pattern confidence influence or the other β parameter. If one of the β weights or both together reach 1 they completely dominate the ranking formula which might provoke a stronger negative (or sometimes positive) effect depending on the other settings. Similarly, the initial step from both β values being set to 0 towards having at least one of them at 0.1 has a much larger impact as similar increases on a setting where $\beta_s + \beta_o > 0$ already holds. Table 4.1 shows one of the more extreme cases clearly outlining the influence of the β_s parameter on the ranking performance.

We also investigated the impact that the pattern frequency based maximum-likelihood estimator used to estimate $P(y_j|X)$ in Equation (4.8) has by comparing the results when using this estimator versus the results when the estimator is replaced by a constant, i.e. 1, thus assigning each pattern occurring in the document d (for $P(y_j|X)$) or collection Col (for $P(y_j|Col)$) a likelihood of 1. Note that this method to "switch off" the frequency based estimator also strongly affects the smoothing: while not totally disabling it, i.e. we still avoid assigning a probability of 0 to documents that do not have pattern instances for all statements, the smoothing affects each pattern equally no matter how often it occurs in the collection or document.

Table 4.2 shows nDCG, MAP, msr, and mrr values in persuasiveness and on-topicness (where available) for pattern frequency weighing turned on versus being turned off over 18 different parameter setting combinations, while all these settings have in common that β_s, β_o both are 0. When the entity occurrence weights (β_s, β_o) are increased the influence diminishes as the pattern based portion of the ranking function loses its influence and the pattern frequency is partially captured by the entity occurrence frequency. For on-topicness there is no significant effect of using pattern frequency weighing. This changes when we look at the persuasiveness. Here, considering the pattern frequency provides a

clear benefit over ignoring the frequency.

For changes of the α value we could not find a noteworthy overall impact on average. However, we did not look into extreme values (e.g., $\alpha = 1$ or 0). Still, a high value of α can lead to over-fitting as the smoothing loses influence.

4.6.3. Ranking Evaluation

In this section, we analyze the performance of our ranking method, given a set of documents identified as witnesses for (part of) the query, in terms of *on-topicness* and *persuasiveness*. We compare the proposed ranking method to a naive ranking (*naive*) which ranks the witness set randomly and investigate the results achieved with different exemplary configurations of our ranking model.

Name	α	β_s	β_o	confidence	pagerank
persuade	0.5	0	0	quadratic	on
topic	0.9	0.5	0.5	no	off
mix	0.9	0.1	0.3	quadratic	off
per-lin	0.7	0	0	linear	on
pr	0.5	0	0	no	on
per-cf	0.9	0	0	quadratic	off

Table 4.3.: The ranking model configurations

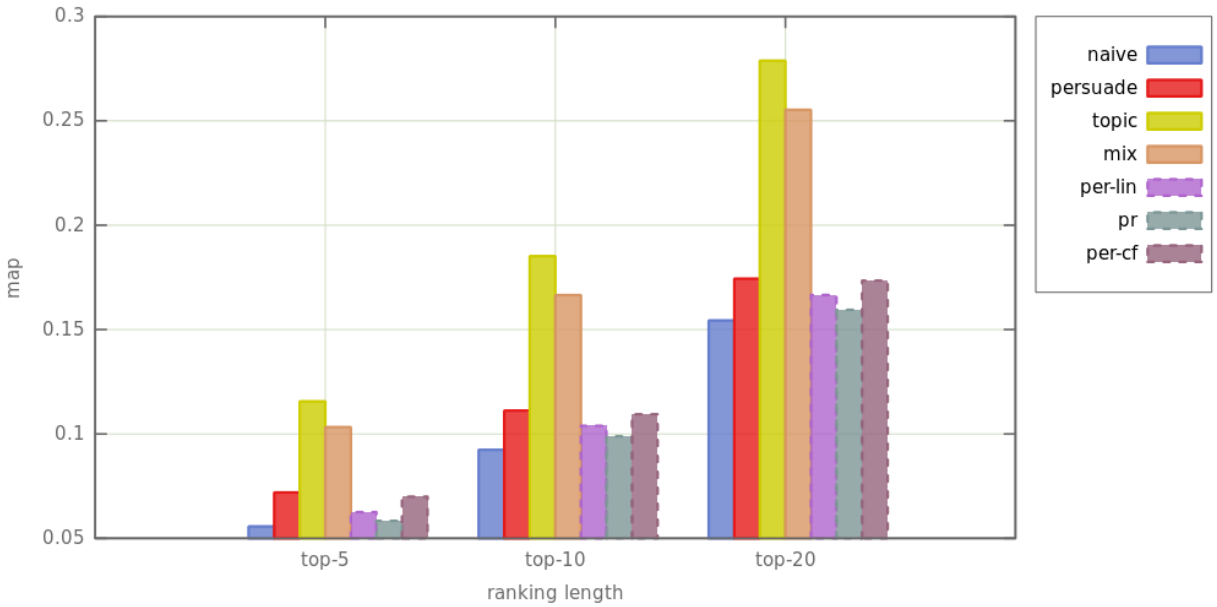


Figure 4.2.: Comparison of *on topic* performance based MAP for all queries

We make use of three main example configurations of our ranking model representing different use cases (for details see Table 4.3). One setting (*persuade*) aims at high persuasiveness by ignoring any entity occurrences, but rather relying on the pattern confidence values. Another one (*topic*) mimics an entity search, aiming at on-topicness by focusing totally on the entity occurrences ($\beta_s = \beta_o = 0.5$, thus in a maximal combination), ignoring any other influence like page rank or pattern confidence. A third configuration tries to achieve a good balanced mix of both goals by applying some entity bias while still considering pattern confidences and applying pattern frequency smoothing (*mix*). As a variation of the *persuade* configuration, we include a similar configuration where the pattern confidence only has linear impact (*per-lin*). Additionally, settings that mainly rely on pattern confidence with high α and without any pagerank support (*per-cf*) and respectively a configuration mainly relying on pagerank (*pr*) are investigated as well as.

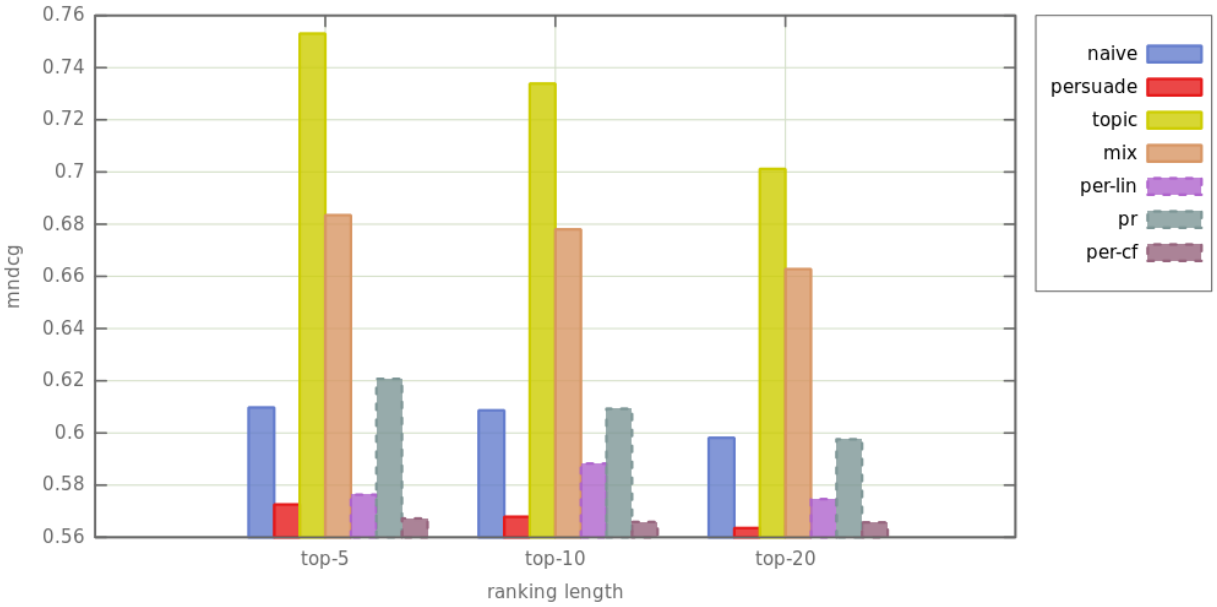


Figure 4.3.: Comparison of *on topic* performance *nDCG* for all queries

Figure 4.2 and Figure 4.3 show that our topic oriented approach (*topic*) clearly meets the expectation of performing well in the *on-topic* evaluation in terms of nDCG and MAP compared to our other approaches. Somewhat behind the *mix* approach follows, while all the persuasiveness-oriented approaches (*persuade*, *per-cf*, *per-lin*) along with the page-rank driven version (*pr*) are roughly on par with the naive baseline. In fact, they are partially even worse than the random ranking applied in the naive approach, which is due to their strict focus on convincing documents rather than documents providing much additional information on the involved entities.

The picture changes when we look at the persuasiveness of statement indications in the documents provided. The Figures 4.4 and 4.5 show the nDCG and MAP values with respect to persuasiveness for our ranking variations. Under the persuasiveness-perspective, the statement-oriented rankings (*persuade*, *per-lin*, *per-cf*) are notably better than the

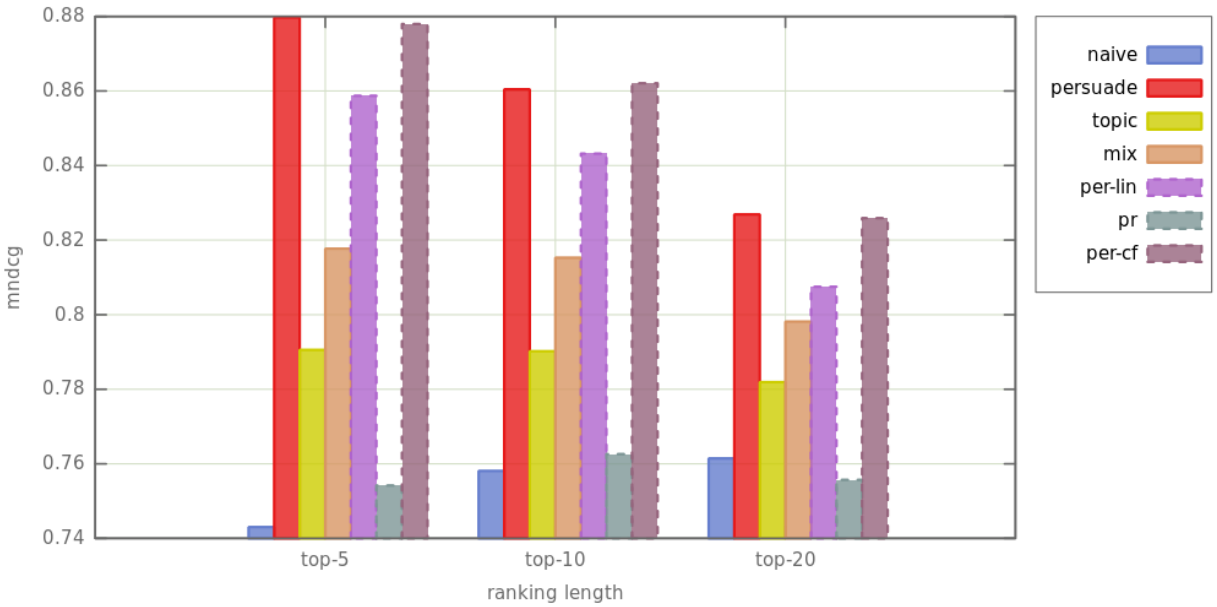


Figure 4.4.: Comparison of persuasiveness performance based on *nDCG* for all queries

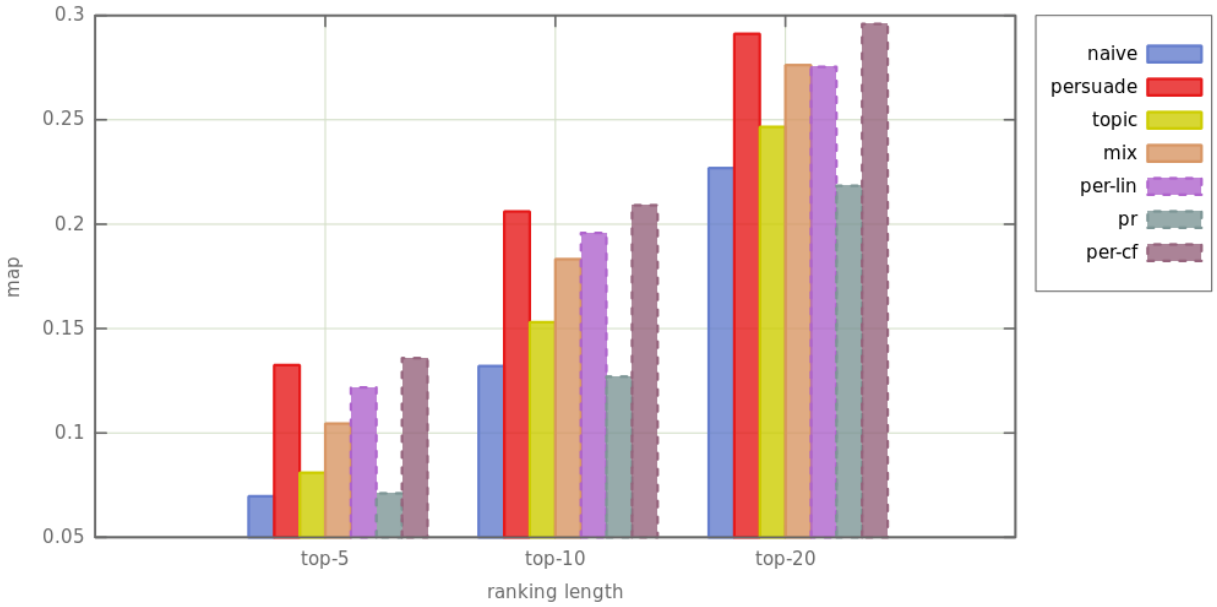


Figure 4.5.: Comparison of persuasiveness performance based on *MAP* for all queries

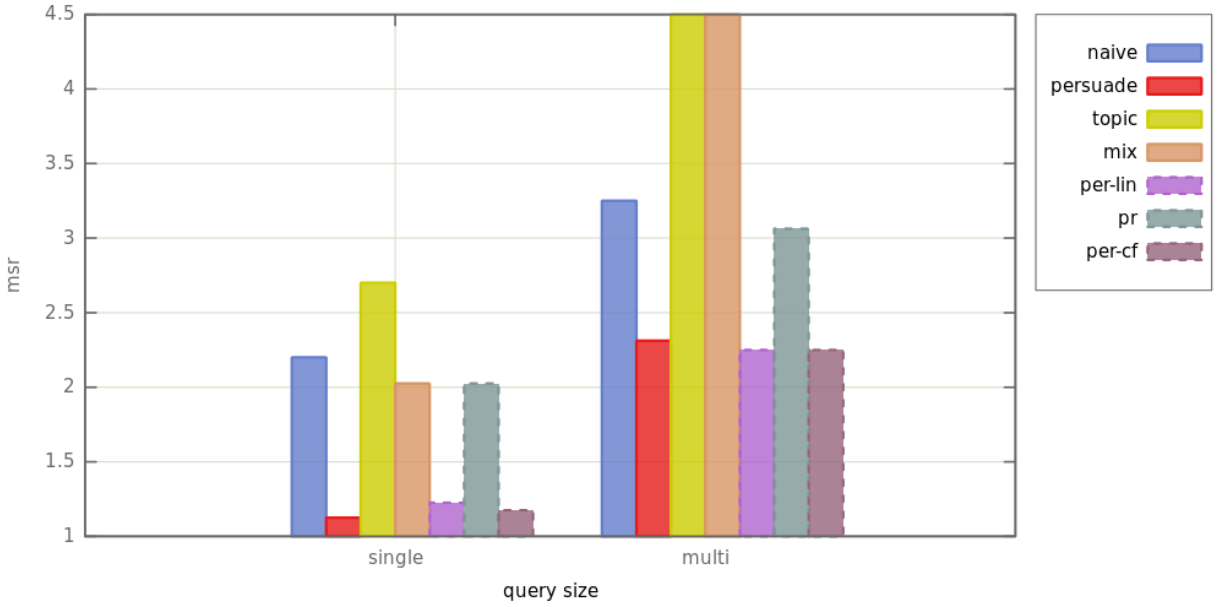


Figure 4.6.: Comparison based on *msr* for top-20 single- and multi-statement queries

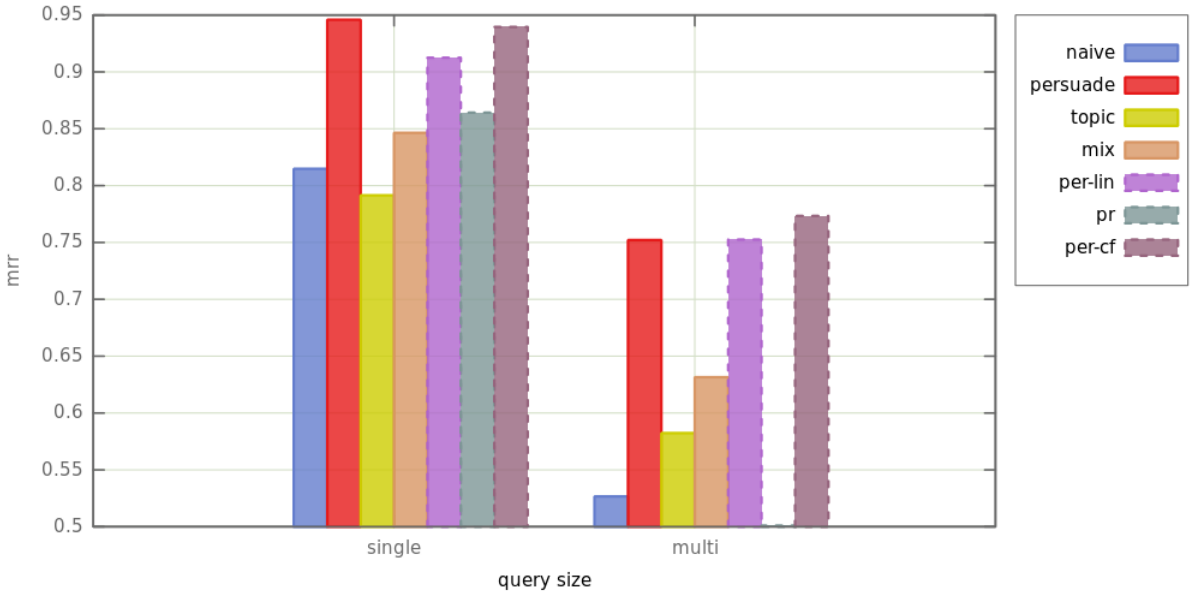


Figure 4.7.: Comparison based on *mrr* for top-20 single- and multi-statement queries

ranking approaches with an entity bias. However, with exception of the purely pagerank driven approach, all our ranking model configurations are clearly providing better results than the naive baseline. Also note that the mixed configuration (*mix*) typically follows our intuition in providing a result quality in-between the persuasion and the topic-oriented variations.

In order to get a better grasp on how the result quality differences would actually influence the user experience, have a look at the msr values in Figure 4.6. For queries consisting only of a single statement, a user would on average have to look only at the first result to be convinced that the statement she is interested in holds, if the document retrieval were based on the *persuade*, *per-lin* or *per-cf* ranking model setting, while with the other variations that number nearly doubles. We can also see that if more than one statement is involved the number of documents in the result ranking a user needs to look at as a minimum increases, but the relationship between the different configurations stays very similar.

The better performance of statement-oriented approaches is reinforced by the mrr values in Figure 4.7, where *persuade*, *per-cf* and *per-lin* dominate while pagerank-only ranking is surprisingly good for single-statement queries but falls behind on multi-statement queries. The reason for the latter might be that those documents with a large pagerank in our corpus are often from Wikipedia. So for most single-statement queries the Wikipedia page of the main entity will be in the top results and would usually contain what we are looking for. The good performance of *pr* breaks down on multi-statement queries or when we look at the nDCG values in Figure 4.4, as a single good document is not enough for a high nDCG value. While *pr* and *naive* have about the same msr value there is some distance on the mrr scale. This can be explained by the fact that the mrr is not the inverse of the msr, i.e. it is the mean over the sum of all satisfaction ranks, thus very good and very bad results have a stronger positive/negative impact than values in the middle.

4.6.4. System Evaluation

In this section the actual user experience in a statement search scenario compared to a traditional keyword search is investigated. Thus, we compare the result quality of documents retrieved by the main configurations of our model (*persuade*, *topic*, *mix*) against those retrieved with a keyword search retrieval system without any prior knowledge about statements contained in the documents. To this end, the keyword search engine *Apache Lucene*[4] Version 3.03 has been used with its default parameter setting to index the initial set of 182,139 documents retrieved from ClueWeb. Based on this index three variations of keyword queries have been generated. First, each of our queries was translated into a keyword query based only on the involved entities (lucene:ee) to provide a topic-oriented keyword based competitor ranking to our topic-oriented configuration. Second, each statement query was fully translated into a keyword query with the relations being translated by a manual mapping, e.g., translating the relation `wasBornIn` to "was born in" (lucene:ere).

[4] `http://lucene.apache.org/`

Finally, the third variation is closest to our own approach by using the pattern confidence provided by the extraction system to translate the statement relations. In this third variation, each query was translated fully again, but the relations were translated based on the best (highest confidence) pattern known to the extraction system for that particular relation (lucene:epe). For each of these variations, we gathered the top-20 documents and assessed them to add them into the document pools for the corresponding queries.

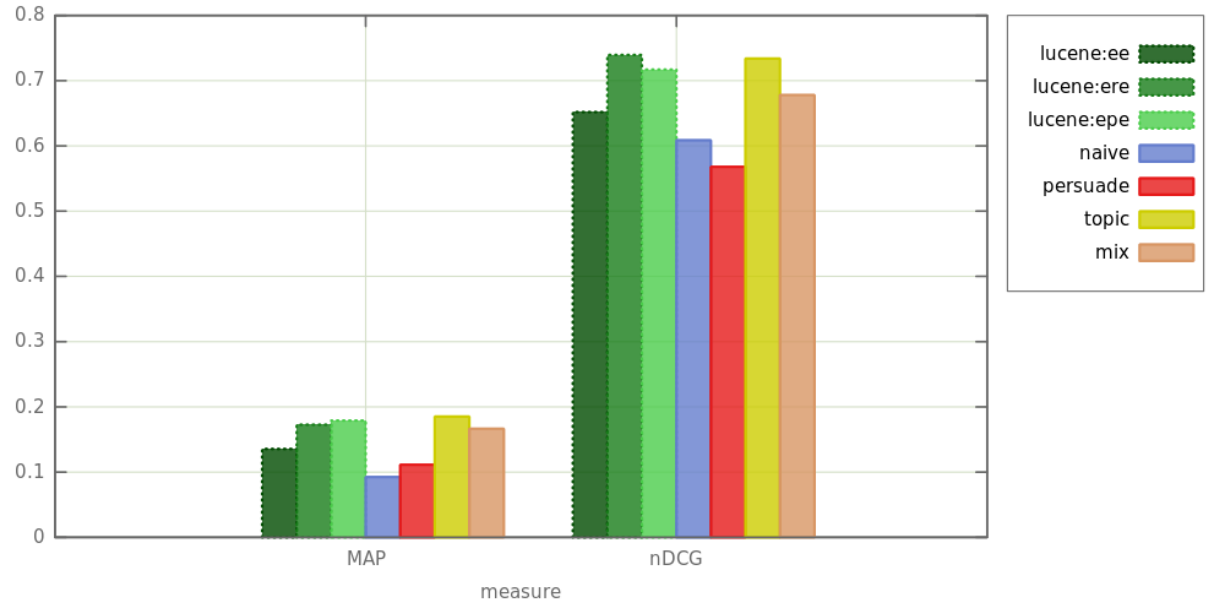


Figure 4.8.: Comparison against pure lucene based approaches with respect to *on-topicness* based on the top-10

Figure 4.8 provides an overview over the results for rankings of length 10. As expected, the keyword-based search achieves good results for on-topicness that are competitive to our approach, albeit our topic-oriented configuration *topic* provides better or at least on par result quality while *mix* is at least on the same level as the keyword based approaches. However, the keyword based approaches clearly provide better on-topicness results than our approach focused on persuasiveness, *persuade* which over-fits too much to provide the same level of topic-centered documents.

When we consider precision, provided in Figure 4.9 we can see that our topic-oriented approach provides good documents especially in the very first results, but then falls behind. This may be due to the fact that the keyword based approach can pull in more documents that are on the general topic, while our filtering to those documents that are identified as witnesses might mean we have more quickly documents in the results set that mention the statement in passing, but are only slightly concerned with the topic.

Considering persuasiveness however, as shown in Figure 4.10 for rankings of length 20, our *persuade* configuration stands out prominently as the most efficient approach, followed by *mix* and our remaining configurations. Even our naive approach outperforms the keyword based results. This shows two main points, namely that the filtering using the

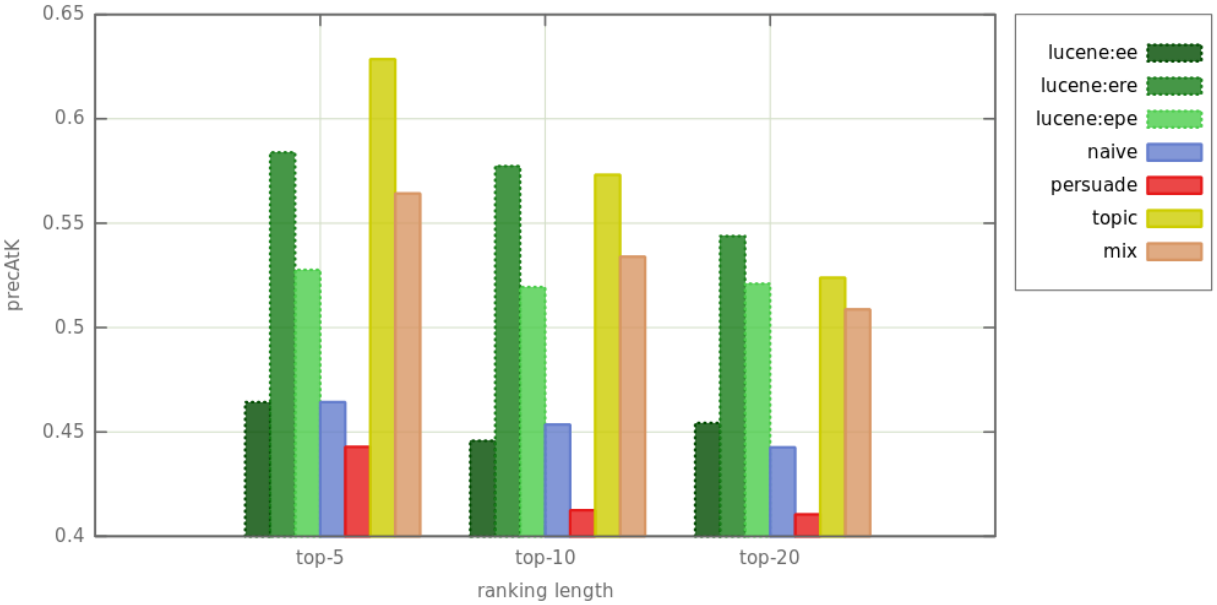


Figure 4.9.: Comparison against pure lucene based approaches with respect to *on-topicness* on different ranking sizes considering precision

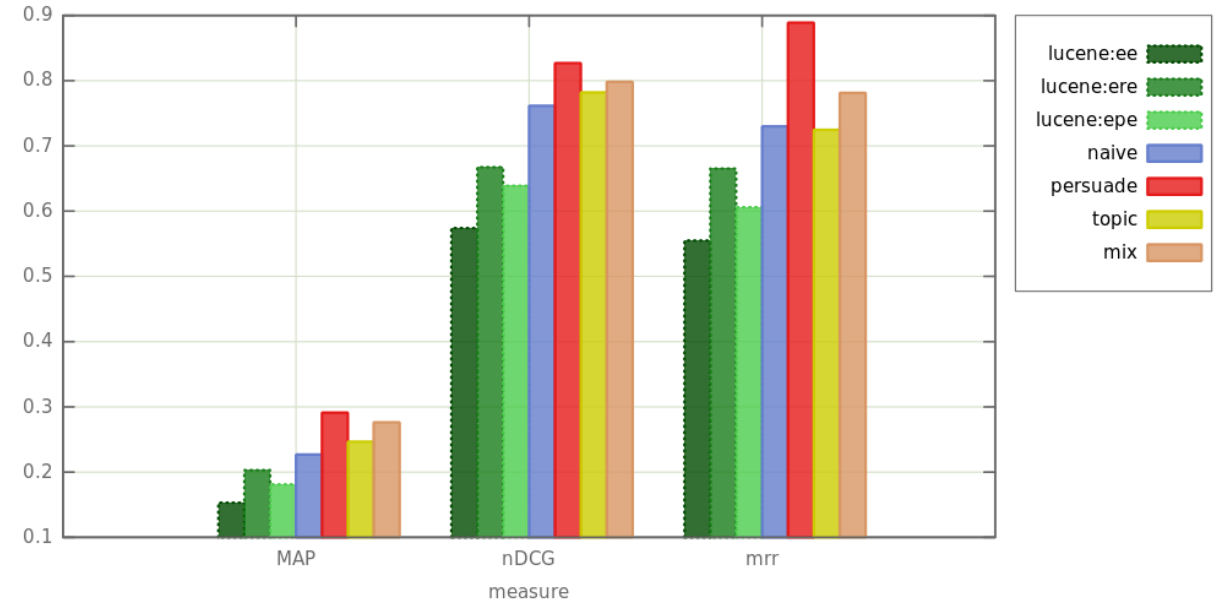


Figure 4.10.: Comparison against pure lucene based approaches with respect to *persuasiveness* based on the top-20

statement index is one important component, but the ranking is providing another sizable improvement on top of this.

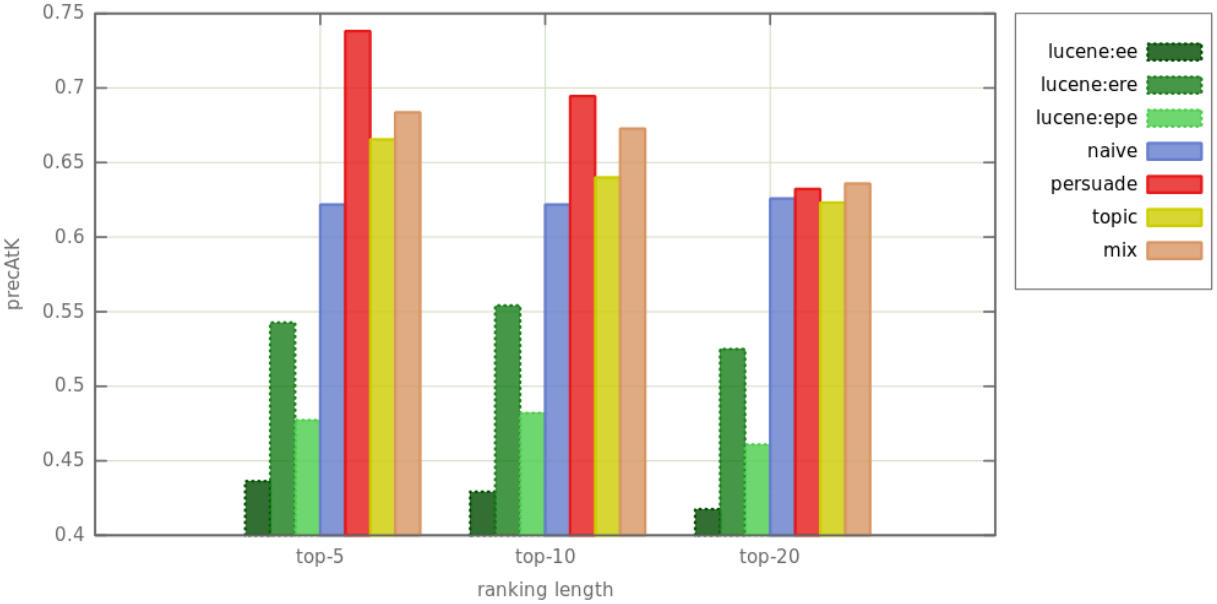


Figure 4.11.: Comparison against pure lucene based approaches with respect to *persuasiveness* on different ranking sizes considering precision

If we look at the precision, depicted in Figure 4.11, we see a quality dampening at ranking size 20, which indicates that the ranking algorithm runs out of documents correctly considered relevant and thus the ranking component becomes less efficient, such that all ranking approaches provide similar precision when considering such longer sets of result documents.

Also, Figures 4.12, 4.13 and 4.14, which provide results split up by query size, show that for multi-statement queries the benefit by the statement based document filtering along is not sufficient anymore to outperform the keyword index based approaches, thus the correct ranking gets more important in such cases.

Similarly clear as these three measures are the msr values shown in Figure 4.15 and they also provide a more intuitively understandable estimation of the actual user experience: using a keyword-based approach, a user would need to look on average at ∼4.5 results *(lucene:ee)*, ∼3.8 results *(lucene:ere)*, and ∼3.7 results *(lucene:epe)* compared to ∼2.9 results with the *mix* configuration or ∼1.5 results with the *persuade* setting.

Figure 4.15 also shows that there is some room for further tuning improvements at the *mix* configuration, in particular when it comes to multi-statement queries. The fact that the msr value is the same as for the *topic* approach indicates a dominance of the entity bias. Part of the reason might also be that the weight added for a document based on the entity bias is decoupled from the occurrence of a matching pattern instance for the same query statement, i.e. entities from different statements of a multi-statement query might occur in the document, but not in the relational connection aspired for by the query statements,

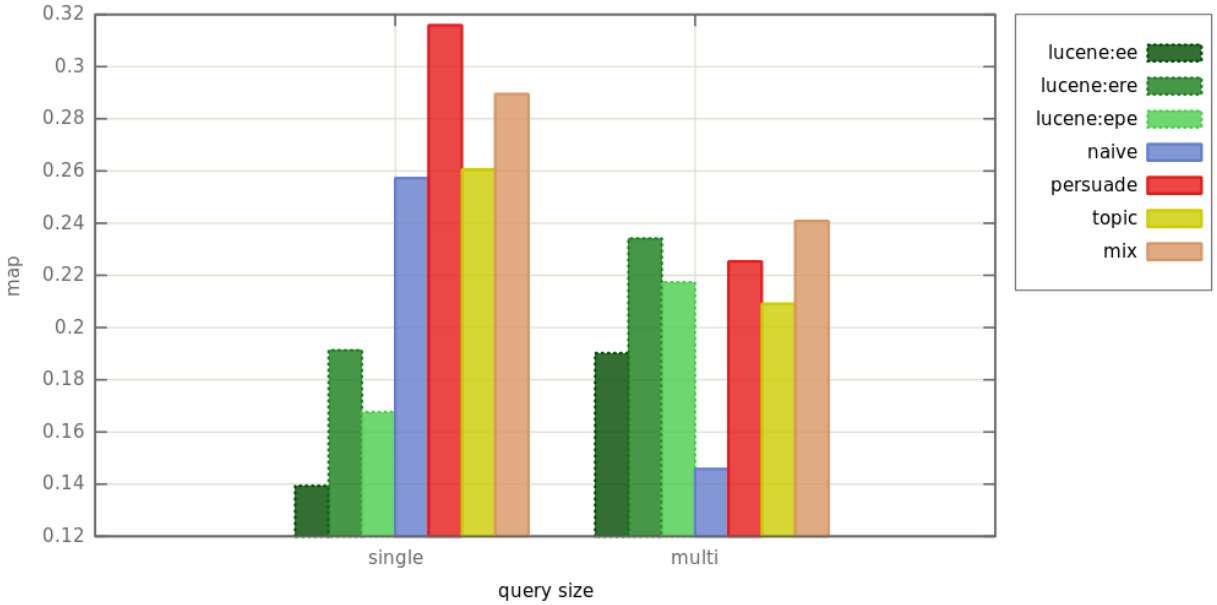


Figure 4.12.: Comparison against pure lucene based approaches with respect to *persuasiveness* based on the top-20 results considering MAP

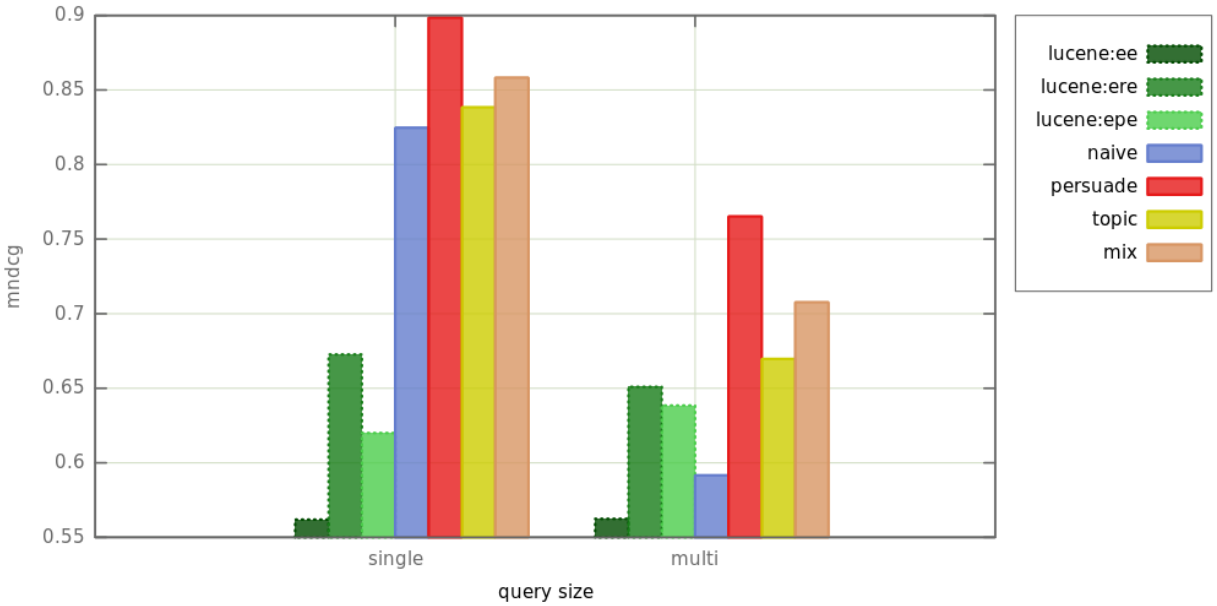


Figure 4.13.: Comparison against pure lucene based approaches with respect to *persuasiveness* based on the top-10 results considering nDCG

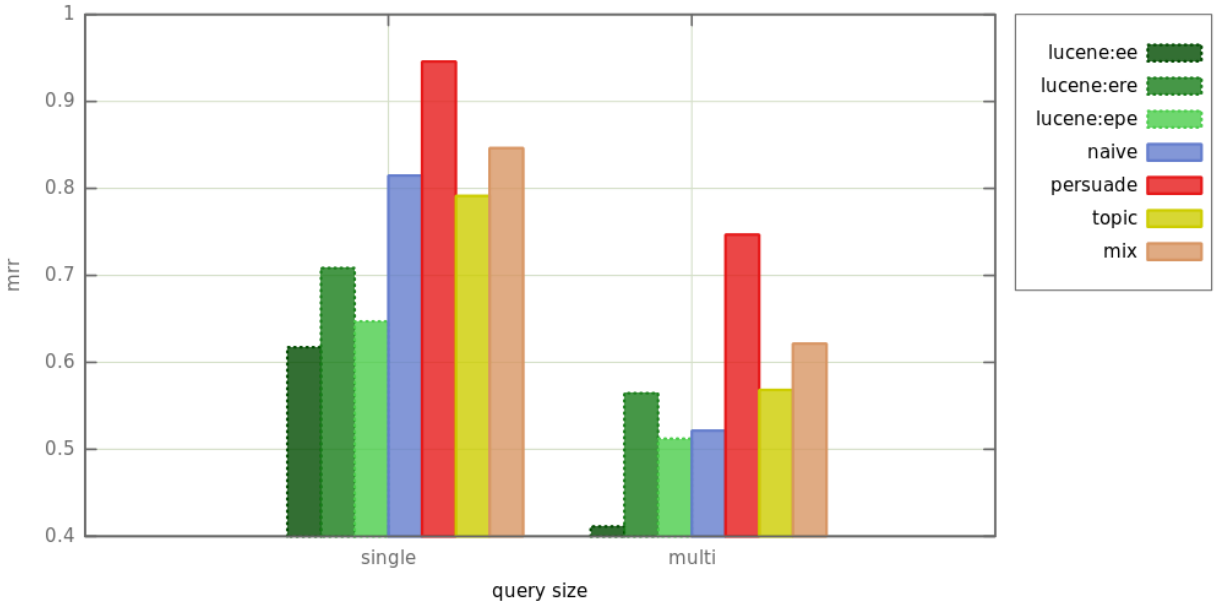


Figure 4.14.: Comparison against pure lucene based approaches with respect to *persuasiveness* based on the top-20 results considering mrr judgements

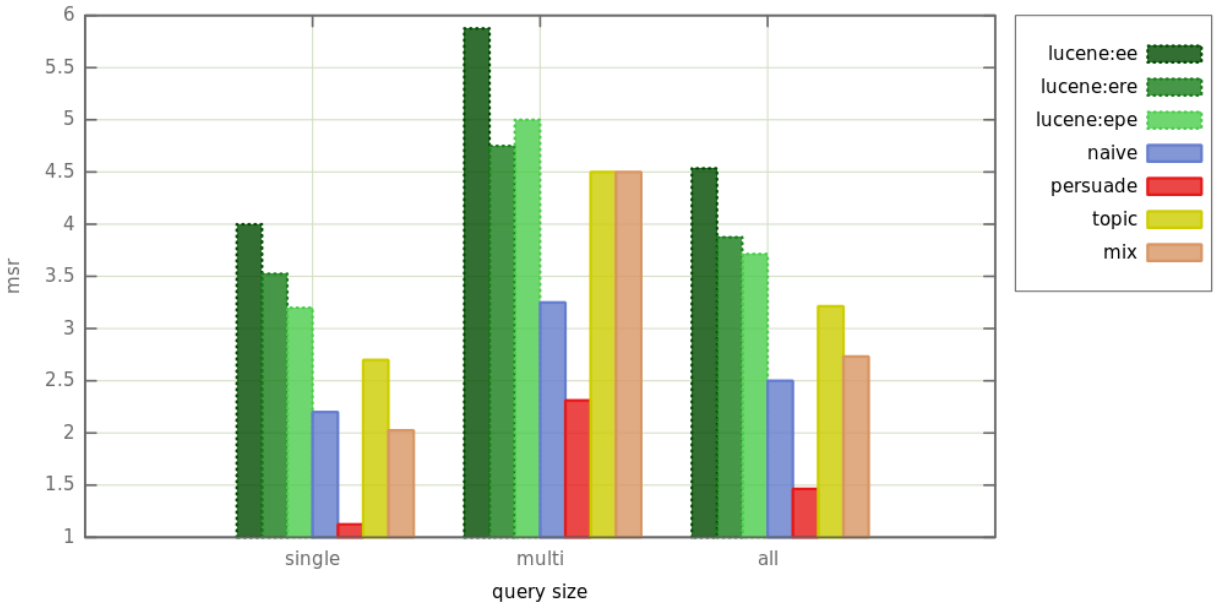


Figure 4.15.: Comparison against pure lucene based approaches with respect to *persuasiveness* based on the top-20 results considering msr judgements

thus the more statements occur in a query the more entities can trigger a (stronger) entity bias.

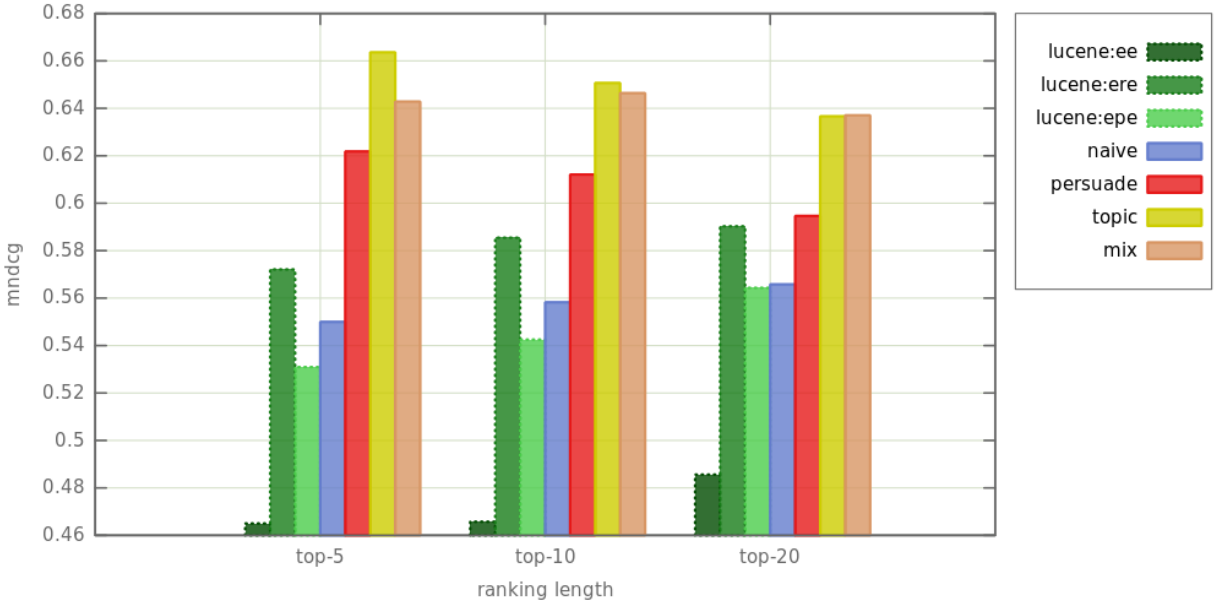


Figure 4.16.: Comparison by query size based on *nDCG* in a combined evaluation setup

Finally, in Figure 4.16 we measure the overall gain a user with an unclear interest may experience when browsing a list of returned documents. To this end, relevance is measured in a fashion that combines both relevance dimensions considered so far, i.e. on-topicness and persuasiveness. Within the combined relevance measure a document is considered

a) *relevant* (1), if it is *highly relevant* on exactly one relevance dimension

b) *fully relevant* (2), if it is *highly relevant* on both relevance dimensions

c) *non-relevant* (0) otherwise.

For instance, a document being *somewhat relevant* in the on-topicness dimension alone is *non-relevant* in the combined relevance scale, a document being *highly relevant* in the on-topicness dimension alone is *relevant* and a document being *highly relevant* in the on-topicness and persuasiveness aspect is *fully relevant*.

we can see that the statement-oriented and mixed settings perform quite well, but *topic* providing even better results under this combined perspective with *mix* being especially more adept for the long tail. Anyhow, all three variations perform clearly better than the naive baseline and the *pr* variation. The good values for *topic* might in part be due to the larger number of topic-wise *highly relevant* documents for each query than there are *highly relevant* documents with respect to persuasiveness. While the keyword based competitors can gain ground compared to *persuade* in particular, our variations apparently still provide an overall better result mix. Also note that in our setting the number of relevant documents

is limited for some queries, such that our variants run out of candidates early, which is unlikely to be the case in a real world scenario, and, at least for the setting we consider, we assume it to be unrealistic to expect a user to go through more than 20 documents. Still, the fact remains that there might be more documents that focus in general on the topic than documents that explicitly express the query statements.

Also note that in general the effectiveness of our pattern index based ranking strongly depends on the quality of the extraction system. A keyword search approach has the general advantage that it is independent of the extraction system, but vice versa does not gain from the extraction system's knowledge about statement indications.

After all, by relying mainly on entity occurrences (*topic*), we can mimic the performance of a state-of-the-art keyword-based ranking system, and this gives good results in terms of on-topicness. Considering the persuasiveness, however, even our *naive* approach as well as the topic-oriented approach can outperform the investigated keyword based variants, while bringing in the relations (*persuade*) will increase the persuasiveness (at the cost of decreased on-topicness) clearly providing better results than the keyword based variants we compared against. *mix* also indicates that finding a good balance is possible, although a more thorough tuning is certainly possible, that will especially in the long run provide more generally helpful results.

4.6.5. Hybrid Approach

The two basic components of our statement search document retrieval approach are 1) a statement indication based document filter and 2) the language model based ranking approach using the statement-document links created by the extraction tool in the form of pattern instances and the corresponding meta-information, i.e. pattern confidences. In the last section we compared our approach with a keyword engine based document retrieval approach. In this evaluation we provide an additional intuition about how much the filtering of documents to actual witnesses and how much the ranking contribute to our result quality. Looking at it from another point of view, this experiment also shows how close a keyword search base approach can get with an index that has no information about actual patterns, yet connects statements to documents and can filter a query to the set of documents that are known as supportive for the statement(s). To this end, we shall discuss a combination of both approaches using the filtering part of our approach together with a keyword search in the prefiltered set of documents. So, given a query, we identify all candidate documents that contain at least for one statement of the query a matching pattern occurrence. The resulting set of documents is then indexed using Lucene in its default configuration. Afterwards the statement query is translated in a keyword query and executed to retrieve the ranked list of result documents.

Analogously to the three keyword-based variations from the section before, we generate three types of keyword queries. First, we generate keyword queries solely based on the entities contained in the statement(s) of the query (fluc:ee). The relations are therefore not represented in these query translations.

Then, analogously to lucene:ere, each statement query is fully translated by replacing

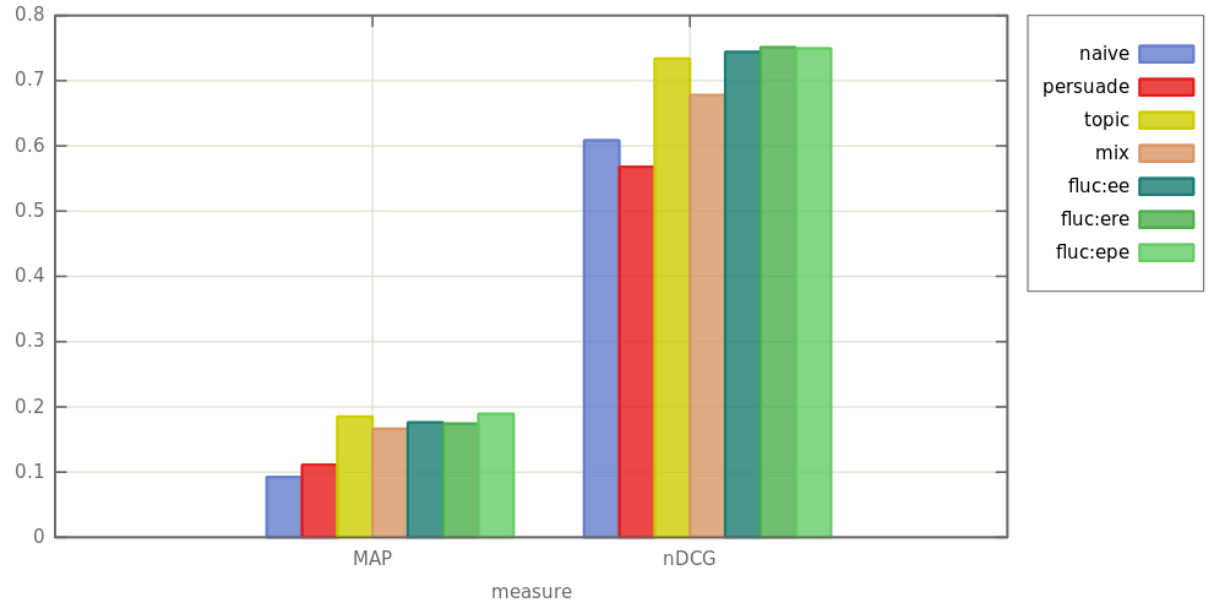


Figure 4.17.: Comparison against filtered lucene based approaches with respect to *on-topicness* based on the top-10

the relation by a canonic manually defined representation (fluc:ere). Finally, the third variation, analogously to lucene:epe, also makes use of the pattern confidences provided by the extraction tool by replacing the relation with the most likely (highest confidence) pattern (fluc:epe). Note that by using more and more information in the queries we get closer and closer to simulating our ranking model with the keyword search engine to a certain degree.

Figure 4.17 shows that all three keyword based variants achieve a very close quality level and clearly meet the expectation of performing well in the *on-topic* evaluation in terms of nDCG and MAP. Only *topic*, our parameter setting optimized for achieving high on-topicness, can compete with them. Note, however, that the mixed configuration *mix* is not too far behind.

Still, when considering the persuasiveness results shown in Figure 4.18 our *persuade* approach still dominates the result quality in all measures albeit very narrowly. The typical trend when using keyword based queries however is also clear, the closer we get to our ranking model the better the result quality.

Summary Since the keyword search paradigm is designed towards finding documents that are about the topic described by the keywords, it is clear that it performs well with respect to on-topicness. The semantic advantage an extraction tool might provide, e.g. the knowledge about different names, can easily be integrated into a keyword search engine as well, e.g. by adding potential alternative names for an entity by query rewriting. However, identifying documents that express a certain relation is a task the keyword search approach is not particularly suited for, as it is typically only concerned with single words. While

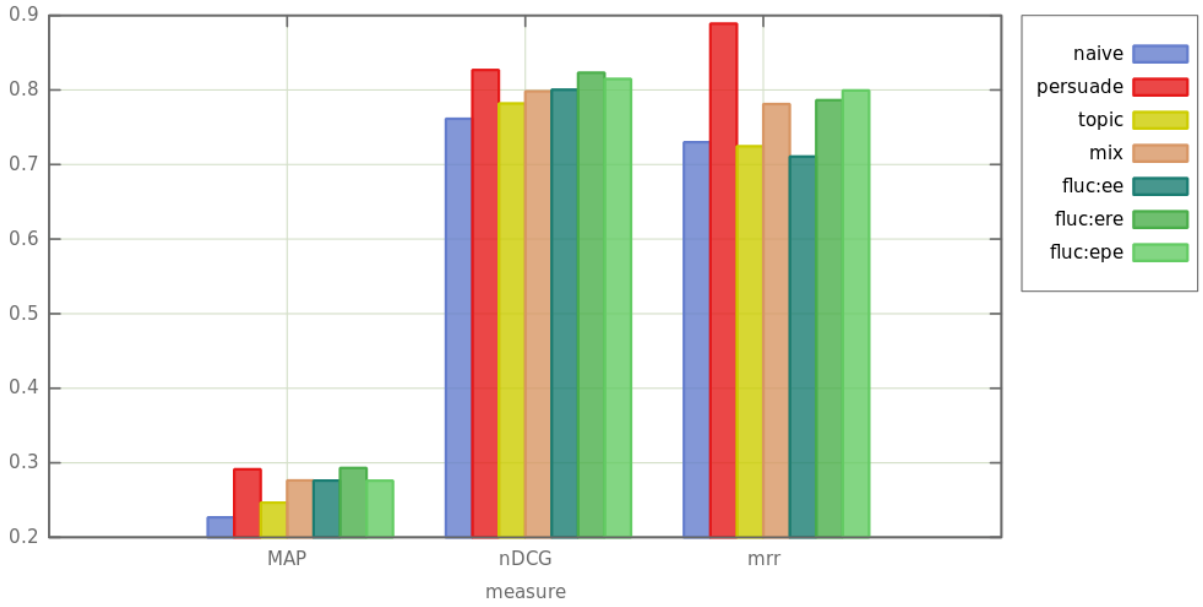


Figure 4.18.: Comparison against filtered lucene based approaches with respect to *persuasiveness* based on the top-10

n-gram indices and phrase matching can overcome some shortcomings, such an approach of trying to identifying pattern occurrences based on such a generic index becomes increasingly inefficient the more complex the respective patterns become. Imagine that a pattern for a statement might span several sentences and in principle might also include more than two entities, e.g. describing an event that happened at a certain time at a certain location or the act of providing a gift to somebody, thus the statement would include at least the present, the receiver, and the generous person handing it out. In addition, patterns might also have a structural form typically not indexed by a keyword search engine, like a table or a list structure, or they could be given as a regular expression or incorporate sentence structures like dependency trees. Such information is hard to capture with a statement agnostic keyword search engine; by adding more and more sophisticated methods to index regularly appearing phrases or sentence structures, the "keyword engine" would more and more become very similar to our approach. However, our model can deal with patterns in any surface form, as long as the extraction tool provides the necessary dictionaries and confidences. Extending the model for statements with more than two entities (or less for that matter) is also relatively straight forward.

While we could show that in a statement search setting our ranking model 1) can compete with traditional keyword based search methods when the user aims for documents that are generally on the topic given by her statement query and 2) provides clearly more convincing documents earlier in the ranking thus providing better quality when the user's aim is to verify the statements from her query. In addition our model is adaptable to the particular user interest. Similar to keyword search engines, which require a keyword index to be created upfront, our approach requires a statement and entity occurrence index

to be generated by an information extraction tool beforehand. While we showed how to incorporate keyword search as a ranking component into our approach, our approach an still be more efficient with regard to persuasiveness using just the statement and entity occurrence index than the hybrid approach. While the hybrid approach could be further extended to match our ranking method more closely, e.g. by using multiple patters for keyword queries, to bridge the remaining quality gap, this would still require to maintain both indices, a keyword index as well as a statement and entity occurrence index. Still, the hybrid approach at the moment shows a slightly better quality balance than our *mix* configuration, providing good results for on-topicness and persuasiveness. However, this may be a matter of further refinement.

Remark 3. *Note that the MAP values presented in [MEHS11] deviate in absolute numbers from the the results presented in this thesis. This is due to the fact that in the paper the denominator was limited to (at most) the ranking size; that is, even if there were more than k relevant documents in the document corpus the denominator was at most k when considering a ranking of length k. This was rectified for the evaluation presented in this chapter. This correction results in lower absolute MAP values for all compared methods, but the general outcome and the conclusions that can be drawn from the results are essentially unchanged.*

4.7. Frontend

As indicated in Section 4.4.2 we provide a user frontend that enables ontology navigation and statement based document retrieval, e.g. for fact verification or source exploration. This system integrates the user frontend to formulate a query with a frontend to present the resulting witness list, i.e. the Witness Displayer.

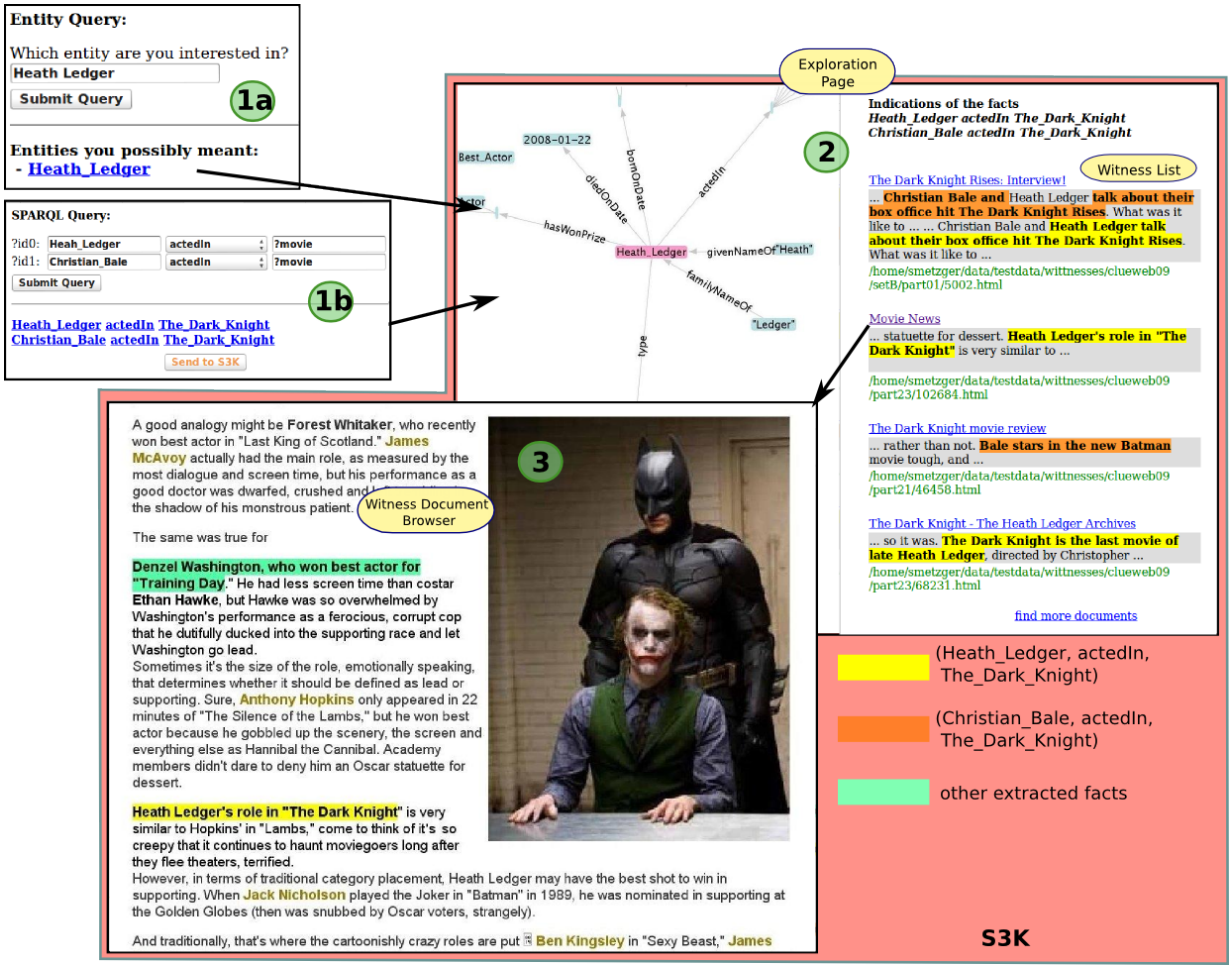


Figure 4.19.: Example Workflow

To make use of the witness retrieval, a user first identifies a set of statements she is interested in. The S3K frontend offers two ways to enter the ontology, one for users acquainted with SPARQL and the other for unexperienced users. The first allows users to retrieve RDF subgraphs via SPARQL queries whereas the second allows them to search for entities in the knowledge base. A SPARQL query and a corresponding result for the example about Heath Ledger and Christian Bale acting together in a movie are shown as step 1b in Figure 4.19. Step 1a in Figure 4.19 shows the results when performing an entity

search for "Heath Ledger". Once the user clicks on either "Send to S3K" below a result in the SPARQL interface or on an entity, the *Exploration Page* opens.

The *Exploration Page* offers two interconnected ways to learn more about a certain RDF triple or subgraph, a graphical *Ontology Browser* and the *Witness List*, the latter being a textual interface that lists the witnesses for a chosen RDF subgraph. Our ontology browser is based on the Prefuse visualization tool [5].

The *Ontology Browser* renders a hyperbolic visualization of the RDF graph and allows users to easily navigate through the RDF graph by clicking on entities. The user can define the RDF subgraph to be used as a statement search query by selecting or deselecting facts while browsing the graph. The *Witness List* shows a ranked list of witnesses containing (some of) the knowledge expressed by the chosen subgraph. For each listed document, a snippet from the document's content is shown. The snippet generated shows occurrences for some of the facts in the RDF subgraph inside the document (see also Section 4.5.4). Step 2 in Figure 4.19 illustrates this for the RDF subgraph representing the facts that Christian Bale and Heath Ledger appeared in the movie The Dark Knight (see highlighted text in *Witness List* frame in Figure 4.19).

Once witnesses are listed, the user can view an annotated version of the document through the *Witness Browser* by clicking on an entry's title. Depending on the system configuration, the annotated version is either based on a local copy of the document taken at extraction time or on the current version of the document retrieved from its original URI at the time of the user request. Alternatively, the current version of the document can be viewed by following the URI shown below the snippet. When exploring the local copy, word combinations which the facts of interest have been extracted from are highlighted using the same colors as used in the *Witness List*. Furthermore, word combinations associated with any other fact are highlighted in a different color (green) as shown in step 3 in Figure 4.19.

Should the witness list not provide enough (or even no) witnesses for a statement query, the frontend allows the user to search for more potential witnesses. To this end, the system automatically formulates several keyword queries based on the pattern variants with the highest confidence. We discuss how such an approach could be used for automatic fact verification and extraction in Section 4.8.

[5] http://prefuse.org

4.8. Extraction on Demand

As discussed in the previous sections we can retrieve documents supportive of a given statement also using keyword queries. This is helpful especially in case we do not have a full statement index yet. For instance, when we look for information not yet backed by documents, e.g. manually added, or not accepted into the ontology at all, yet. In the latter case the ontology is simply ignorant of this piece of information having no indication for it being true. However, a user (or external system) might be interested in knowledge beyond what an ontology captures at a certain point. If we assume a lifelong learning approach, where the goal is to eventually parse all documents on the web and incorporate their knowledge, and assuming the extraction system works flawlessly, at some point this piece of information is either discovered or all of the web is processed. In the latter case one might conclude that, what has not been extracted has at least not been stated in any source document and thus cannot be observed by this means. A user however, might not be so patient as to wait for one of both events to happen. For such cases an *on-demand knowledge look-up* that tries to determine whether a particular piece of information can be considered true by specifically searching for documents supporting it would be a preferable solution. The simplest case of such an ad-hoc search and extract task would be to verify a particular statement by looking for supportive documents, e.g. a user might want to verify the statement (`Barack_Obama`, `bornIn`, `Hawaii`). The system would then look for documents stating that President Obama has been born in Hawaii and either present these documents to the user or determine based on the results whether to accept the statement as observed/true. To generate keyword queries the entity names known to represent `Barack_Obama` and `Hawaii` would be combined with the phrases known to express the `bornIn` relation. For example, *"President Obama was born in Hawaii"* would be a potential query. In particular, all potential keyword queries $Q((s,r,o))$ for supporting documents of a statement (s,r,o) are given by $\mathcal{PV}(p,s,o)$:

$$Q((s,r,o)) = \mathcal{PV}(p,s,o) = \{p(sn,on) | s \in \mathcal{E}(sn) \wedge o \in \mathcal{E}(on) \wedge (\texttt{p},\ \texttt{expresses},\ \texttt{r})\} \quad (4.13)$$

If we are only interested in whether there is support for (s,r,o) (or how much support there is) and the keyword engine supports *phrase search* we may spare the actual processing of any document and simply aggregate the counts of matching documents. Still typically $Q((s,r,o))$ would contain a sizeable number of queries, while a small set of documents matching well chosen pattern variants typically would suffice to convince us that the relation is *observable* (not necessarily universally true) in the document corpus. If we consider the pattern confidence and the confidence we have in the global (document independent) name-entity association as independent probabilities, we can model the likelihood that a pattern variant $p(sn,on)$ supports a statement $(\texttt{s},\ \texttt{r},\ \texttt{o})$ by the formula given in Equation (4.14) and guide the query generation using this estimation, starting with the most promising pattern variants.

$$conf(p(sn,on),(\texttt{s},\ \texttt{r},\ \texttt{o})) := conf(p,r) \cdot \mathcal{E}(sn,s) \cdot \mathcal{E}(on,o) \quad (4.14)$$

Now, as long as we are not parsing every document that a pattern based keyword query retrieves, we cannot include the context in the entity disambiguation to get more exact values. Hence the confidence for each occurrence is essentially the same. We ignore a potential document based trust component here, which might (in case of a url based measure like pagerank) or might not (in case of a content based method) be available. However, typically when we manually try to verify a certain statement or misuse a search-engine as a grammar or spell checker, we get swayed by the raw appearance numbers. Similarly, a confidence that the pattern variant can be trusted to be broadly reported can be based on the number of matching occurrences that we get from the keyword engine's meta-information via the number of results for our full-quote query. To this end a normalising threshold ζ that represents the number of matches convincing us that the pattern variant is broadly present can be used to define a confidence component. Assume $R(p(sn, on))$ is the set of documents returned by the keyword engine for the query represented by $p(sn, on) \in Q((s, r, o))$, then we can represent the probability that a statement (`s`, `r`, `o`) is true(ly observable) by considering the probability that not all pattern variants are wrongly observed or matched incorrectly:

$$conf((\texttt{s},\ \texttt{r},\ \texttt{o})) := 1 - \prod_{p(sn,on)\in Q((s,r,o))} (1 - conf(p(sn, on), (\texttt{s},\ \texttt{r},\ \texttt{o}))) \cdot \max(1, \frac{|R(p(sn, on))|}{\zeta})) \tag{4.15}$$

Using this statement confidence function we can use Equation (4.14) to pick the most promising pattern variant and query the keyword engine until

- either a confidence threshold is reached at which we accept (`s`, `r`, `o`) as a fact
- or the remaining pattern variants cannot reach the threshold

If the extraction system has covered a large corpus of data already, there might also be information available as to the frequency with which patterns occur (with particular entities). If such information is available, we could integrate this into the query selection formula (Equation (4.14)) and the estimation how probable it is whether the remaining pattern variants can retrieve enough witness documents to beat the fact confidence threshold.

However, if we aim at deciding the actual "truth" of such an information piece, this can get more involved. First, we may consider whether we trust a given source, thus incorporating a trust function on document sources similar to our approach in Chapter 4. This would however require to consider every occurrence of a pattern variant separately - or at least every *mention*, i.e. all occurrences in the same document can be considered to form a mention analogously to entity mentions.

However, one may argue that this notion of source trust on a document falls short in measuring whether we can trust a particular pattern occurrence, as it refers not to the actual author, e.g. a textual statement from a blog or news page might either be part of the main news story (and such carry the authority of the main author) or stem from

a comment potentially generated by a SPAM bot. In addition, there are controversial statements that are even debated hotly between humans (e.g. where is Obama born, are Apple products the best etc.). Still, we may at least verify the consistency of any such claim with other accepted facts, or at least that subset of facts that the current user most likely accepts as axioms, i.e. those that represent his personal belief system. One simple example where logic can help to refute inconsistent facts are functional relations. For instance, if it is accepted that President Obama is born in Hawaii it cannot be true that he is born in Kenya as we consider being born (`bornIn`) as a function. Hence, if there are several candidates for the same subject we can compare their support and accept or present the one with the stronger support. However, since user opinions are also modelled as ontological knowledge, this allows a reasoning approach to also consider rules based on user beliefs, such as that no-one born outside the US could become US president, making it less likely that President Obama is born in Kenya, or that all "blacks" are born in Africa, which would make Kenya more likely. However, reasoning might also reveal other gaps in the ontology and result in faulty conclusions if they are not filled. For instance, to decide for the best candidate of a functional relation, first all potential candidates need to be known. Similarly a user might be interested in all movies of an actor. If all known instances of the `actedIn` relation found so far are provided, there might be instances missing that have not yet been observed in documents. This second type of ad-hoc extraction problem requires an additional step, namely to identify potential statement candidates, which then can then be verified. There are several options to generate such candidates. First, relations are typed, such that the type can be used to provide candidates. Since these types are relatively broad this will generate a large number of candidates and is therefore only feasible if access to the keyword engine is cheap, e.g. if it is locally available. As an alternative, we can generate partial patterns by instantiating the patterns with the one known entity leaving the second variable blank. For instance, given `Quentin_Tarantino` we might use the pattern "X 's newest movie Y" to generate the query *"Quentin Tarantino's newest movie "*. Using again phrase (full quote) search we will find additional candidates when parsing the document (or simply the text snippet provided by the search engine). Alternatively, when we have full access to the phrase index of the keyword engine, we can simply retrieve all possible candidates by searching for all phrases that start with any such open pattern.

Ranking If we need to actually touch the documents instead of just their count, either to perform extraction or just to present them to a user, we need to rank them. Our model from 4.5 is not applicable here, as before any extraction is applied no pattern occurrences are known. However, if only a single query is used, we can rely on the keyword engine's relevance ranking. If multiple queries are issued however, those can be aggregated. The frontend demo discussed in Section 4.7 follows such an approach. For a statement (or a set of statements) the top-k pattern variants are used to generate keyword queries. The results of those queries are then merged into a ranked list that is then presented to the user as potential additional witnesses (albeit without statement highlighting as this would require an extraction parse beforehand). The merging of several retrieved rankings for

different pattern variants is based on the document's original ranking. For each query we obtain only the first k results as rankings for the individual pattern variant queries. Each result that occurs in a ranking at position r is assigned $k - r$ points. The points are aggregated over all rankings, and finally the documents retrieved from all queries are ranked in descending order of their points and presented to the user.

Related Work From a different point of view these tasks can be seen as specific tasks to automatically fill the ontology [MC08]. Similar, further generalised tasks would be to retrieve all statements that connect to particular entities, all statements that consider a particular entity or all instances of a particular relation. While all these tasks can be guided using entity names or patterns to find instances directly, the broader the task the less efficient will such an approach be compared to a typical web crawl. This approach also ventures into the area of question answering [RH02].

[LDH06] investigates how to turn a structured query into a keyword query by selecting the right entity and relation labels. While we are mainly interested in retrieving documents for particular triples this is not so much of an issue in our setting, as the one triple needs to be fully translated - and in our case we can use the patterns as special already weighed labels.

The SQoUT project [JDG08, JIDG09, JIG08] explores the integration of information extraction into an SQL query execution database engine. Notably, it defines a cost model to determine when an extraction is a reasonable approach to answer a query. Such a cost model could be adopted to suit our approach. In a similar setup [EHFI12] suggest to model extraction by extraction views to integrate them seamlessly into a relational database. Such an approach could as well be adopted by a SPARQL engine.

4.9. Witness Similarity

While we discussed in this chapter how a user can navigate from statements to their sources, statement testimonies discovered in text documents can also be used to navigate from a document to related documents that discuss similar topics. This can help an ontology maintainer from a community where the focus lies more on annotating a set of documents correctly with semantic information than with generating an ontology. If documents are linked by their semantic content, human experts can navigate documents within their domain of expertise by exploring these links. There are also other applications where semantic document similarity could replace or amend existing keyword based models like the standard vector space model [MRS08] or user opinion based models [CM08] that try to capture document similarity. Such applications include document recommendation systems [stu, simb, sima], that suggest similar pages while browsing the web, or document search engines that take a document as input instead of a short keyword query, for instance, to help finding related work given a paper draft. Another application could be research in information flow, i.e. how ideas develop and are transformed, or plagiarism detection, especially in cases where the line of arguments is copied not the exact wording.

There is already a range of (commercial) tools available that try to identify plagiarism. In Germany, HTW Berlin regularly publishes a test report that compares state-of-the-art end user plagiarism detection systems[6], as for instance PlagAware[7] and Turnitin[8], based on their test corpus [WW10], and there is also an international competition [PBCE+10]. As Potthast et al. [PBCE+10] point out, most of the existing systems share a common algorithmic layout at an abstract level and are based on keywords or n-grams. There are, however, also approaches that incorporate other information. [GMB11], for instance, presents an approach based on citations. The evaluation results indicate that this approach results in a better recognition in cases of strong paraphrasing, translation, and idea plagiarism.

To our best knowledge however, no approach employs generic (binary) relation extraction methods, yet.In the following we suggest a general framework to combine different levels of semantic information to measure document similarity and discuss some methods for each level. In particular, we assume there are three levels of textual similarity. At the lowest level, there is the plain surface similarity, based on the words used. Second, there are the entities that are referred to, potentially using different wording in different documents. And finally, there are statements that represent relations connecting these entities. We suggest to combine all three in a document similarity measure in order to capture the different aspects, i.e. similar writing style, similar general topic identified by the entities adressed and similar arguments or points made in the text which can be identified by the statements:

$$sim(d_1, d_2) = \phi_w wordsim(d_1, d_2) + \phi_e entitysim(d_1, d_2) + \phi_f factsim(d_1, d_2) \tag{4.16}$$

[6] http://plagiat.htw-berlin.de/software/
[7] http://www.plagaware.de
[8] http://www.turnitin.com

where $wordsim(d_1, d_2)$ represents the similarity based on the words used, $entitysim(d_1, d_2)$ represents the similarity on the entity level, and $factsim(d_1, d_2)$ represents the similarity on the level of factual statements. With the extensive work available in keyword based document retrieval [MRS08] any of the standard methods available, e.g. the vector space model, can be used to rerpesent $wordsim(d_1, d_2)$.

For the entity similarity ($entitysim(d_1, d_2)$) one could apply a standard similarity measure like the vector space model or Jaccard similarity on the entities instead of the keywords. However, an ontology provides additional information on the entities, such that we can incorporate a more general similarity model that also accounts for entities that are very much alike although not the same:

$$entitysim(d_1, d_2) = \frac{\sum\limits_{\substack{e_1 \in \mathcal{E}(d_1) \\ e_2 \in \mathcal{E}(d_2)}} esim(e_1, e_2)}{|\mathcal{E}(d_1)| \cdot |\mathcal{E}(d_2)|} \tag{4.17}$$

$$entitysim(d_1, d_2) = \frac{\max\limits_{e_1 \in \mathcal{E}(d_1), e_2 \in \mathcal{E}(d_2)} esim(e_1, e_2)}{\min\{|\mathcal{E}(d_1)|, |\mathcal{E}(d_2)|\}} \tag{4.18}$$

This allows to also consider documents as similar that are talking about similar concepts but with totally different words and without refereing the same entities. For instance, given a document with talking about several medical drugs, another documents with a medical brackground can be considered more similar than one that discusses a current box office hit movie.

Finally the similarity on the third level is based on ontological facts that are considered to be stated in the documents in question. For the overall fact based document similarity we aggregate over all fact pairs.

$$factsim(d_1, d_2) = \sum_{\substack{f_1 \in d_1 \\ f_2 \in d_2}} fsim(f_1, f_2) \tag{4.19}$$

Alternatively, facts can be paired and only paired with only paired facts being compared. This can be considered as an assignment problem or an optimisation problem (assign fact pairs such that the overall similarity is maximized). In his master's thesis, Hassan Issa explores an approach that assigns fact pairs based on a heuristic employing a modified Hungarian algorithm [Iss12].

To compare two facts, we break them down into their pieces, two entities(s, o) and a relation r and recursively consider the similarity between the entities and the relations. In order to cover cases where relations have a different orientation, we consider both ways to match the two relation instances $f_1 = (s_1, r_1, o_1)$ and $f_2 = (s_2, r_2, o_2)$. That is, on the one hand, we compare subject entity s_1 with subject entity s_2 and object entity o_1 with o_2 and both relations in their native orientation. On the other hand we also consider the flipped case where subject s_1 is compared with object o_2, object o_1 is compared with subject s_2

and relation r_1 is compared with the inverted relation $\overleftarrow{r_2}$. From both variants we take the one that achieves the higher similarity value.

Note that $\overleftarrow{r}$ indicates that we consider r as if range and domain were exchanged, affecting the range and domain type and the actual instances.

$$fsim(f_1, f_2) = \max \left\{ \begin{array}{l} \lambda(esim(s_1, s_2) + esim(o_1, o_2)) + (1 - \lambda) rsim(r_1, r_2), \\ \lambda(esim(s_1, o_2) + esim(o_1, s_2)) + (1 - \lambda) rsim(r_1, \overleftarrow{r_2}) \end{array} \right\} \tag{4.20}$$

In the following, we discuss several options to compute the similarity components, i.e. entity similarity ($esim(e_1, e_2)$) and relation similarity ($rsim(r_1, r_2)$).

4.9.1. Entity Similarity

The literature proposes many approaches considering ontological concept similarity. In the following, we discuss two such approaches to compute entity similarities based on the type hierarchy given by the background ontology.

4.9.1.1. Type Hierarchy

The literature proposes many approaches considering ontological concept similarity based on the type hierarchy of such concepts, typically involving either the closest common ancestor or the minimal distance between two concepts in the concept hierarchy. [EHSH04] presents an approach combining the length of a minimal path and the depth of the closest common parent to determine concept similarity in an ontology:

$$esim_{th}(e_1, e_2) = \left\{ \begin{array}{ll} 1, & e_1 = e_2 \\ e^{-\alpha l} \left(\frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \right), & otherwise \end{array} \right. \tag{4.21}$$

Formula 4.21 defines the type hierarchy similarity (sim_{th}); l denotes the shortest path between any two types of the two entities (e_1 and e_2), h denotes the depth of the common parent in the hierarchy. α and β are parameters to weight the effect of l and h.

Although this approachs combines two common measures to provide a more balanced measure covering more special cases than one alone, this still is ignorant to the fact that often entities share several branches of types, while others only share one. For instance, Angela Merkel is a `woman` and a `politician` (amongst others). Suppose we compare her with Winston Churchill, Margareth Thatcher and Claudia Schiffer. Assuming for the moment that both `woman` and `politician` are her only types and are at the same level in the domain hierarchy, all three entities would be considered equally similar to her, if we assume that Winston Churchill is a `politician`, Claudia Schiffer is a `woman` and Margareth Thatcher is both. As it seems more fitting to consider Margareth Thatcher as a powerful politically engaged woman more similar to Angela Merkel than the other two persons, we suggest to avoid methods that only consider a single connection between two types of the entities. A solution could be to aggregate over several type combinations, but there is also a simpler approach.

4.9.1.2. Type Intersection

As an alternative we can simply take all types shared into account. However, there may also be types of varying importance. In particular, general like `person` or `product` might be less important than specific types like `SwissAstroPhysician`. To this end each type concept is weighed according to the number of the entities that share the type (see Equation (4.22)), which is used then in order to compute a weighed Jaccard similarity (see Equation (4.23)).

$$w_t = 1 - \frac{n_t}{n} \tag{4.22}$$

$$esim_{ti}(e_1, e_2) = \frac{\sum_{t \in T_1 \cap T_2} w_t}{\sum_{t \in T_1 \cup T_2} w_t} \tag{4.23}$$

t denotes an entity type, w_t is the weight of type t, n_t is the number of entities having t as a type, and n is the total number of known entities. T_1 and T_2 are the sets consisting of all the type entities of e_1 and e_2, respectively.

4.9.2. Relation Similarity

There are again many options to compute relation to relation similarity. While in principle relations are also entities, the ontology provides additional information such that they can be treated differently. In fact, the type hierarchy would not provide insightful information in general anyway, as relations are often simply of type `relation` and potentially of type `function`. However, in the following two variants for relation similarity are provided and their limitations discussed.

4.9.2.1. Domain and Range Proximity

As every relation comes with a domain and a range, the similarity between two relations can be computed by reducing the problem to computing the similarity between their domain and range. This can be done using one of the standard measures mentioned in Section 4.9.1.1 and in particular we can apply the entity similarity measure defined in Equation (4.21) assuming that the types of a type t is given by the set containing only t. This is examplified in Equation (4.24), also taking into account potentially different orientations of the relations.

$$rsim_{dr}(p_1, p_2) = \max \left\{ \begin{array}{l} \frac{1}{2}(sim_{th}(d_1, d_2) + sim_{th}(r_1, r_2)), \\ \frac{1}{2}(sim_{th}(d_1, r_2) + sim_{th}(r_1, d_2)) \end{array} \right\} \tag{4.24}$$

where p_1 and p_2 are two relations having d_1,d_2 as their domains and r_1,r_2 as their ranges respectively.

While this approach provides a *potential* similarity based on the domain and range types of a relation, it says nothing about the actual relation instances and the relations' actual semantics (other than that the relation applies to the same or similar types of entities). For

instance, consider the three relations `divorced`, `marriedTo` and `killed`. They may all three be defined as having a domain and range of `persons`, and thus be considered equally similar. However, one might argue that semantically there is a much larger similarity between two people who were at some point married and two people who at some point got divorced than the killing of one person by another, because the first two both imply that the two persons had a marriage where the mariage of those that got divorced has clearly ended, we can consider the set of person pairs that had a divorce as a subset of those that were married at some point. While there might also be an overlap of people who got killed by their spouse with both both married and divorced people there are probably much more killings unrelated to a marriage.

4.9.2.2. Common Instances

In order to capture such similarities we can employ an instance based similarity measure rather than one based solely on the relation's' domain and range typing. Equation (4.25) provides a similarity measure based on instances, that compares instances on a per entity level.

$$rsim_{ci}(p_1, p_2) = \max \left\{ \begin{array}{l} \frac{1}{2}\left(\frac{|(S_1 \cap S_2)|}{|(S_1 \cup S_2)|} + \frac{|(O_1 \cap O_2)|}{|(O_1 \cup O_2)|}\right), \\ \frac{1}{2}\left(\frac{|(S_1 \cap O_2)|}{|(O_1 \cup S_2)|} + \frac{|(S_1 \cap O_2)|}{|(O_1 \cup S_2)|}\right) \end{array} \right\} \tag{4.25}$$

where S_i denotes the set of subjects of facts with relation p_i and O_i denotes the set consisting of objects facts with relation p_i.

While this considers the subject and object instances individually, i.e. entities from different relation instances are compared, we can also be more restrictive and only consider full instance matches, where both arguments match pairwise:

$$rsim_{ci}(p_1, p_2) = \max \left\{ \begin{array}{l} \frac{\{(s_1,o_1)|p_1(s_1,o_1) \in \mathcal{F}\} \cap \{(s_2,o_2)|p_2(s_2,o_2) \in \mathcal{F}\}}{\{(s_1,o_1)|p_1(s_1,o_1) \in \mathcal{F}\} \cup \{(s_2,o_2)|p_2(s_2,o_2) \in \mathcal{F}\}}, \\ \frac{\{(s_1,o_1)|p_1(s_1,o_1) \in \mathcal{F}\} \cap \{(o_2,s_2)|p_2(s_2,o_2) \in \mathcal{F}\}}{\{(s_1,o_1)|p_1(s_1,o_1) \in \mathcal{F}\} \cup \{(o_2,s_2)|p_2(s_2,o_2) \in \mathcal{F}\}} \end{array} \right\} \tag{4.26}$$

There still remains a general lack of semantic understanding that cannot be captured without additional information. Consider, for instance, the relations `bornIn` and `bornAt` concerning the birth date and birth place of a person. None of our relation similarity measures can identify the underlying semantic relationship between both relations, i.e. the fact that they both concern a person's birth. A relation like `livedIn` indicating the places a person lived in at some point, for instance, will be consdered more similar to `bornIn` than `bornAt` that refers to the same event.

Evaluation In his thesis [Iss12] Hassan Issa provides an evaluation of the semantic similarity approach based solely on statements supported in documents against a keyword based vector space measure using cosine similarity. The evaluation is based on a set of 50 short news related documents that are provided along with human judgement based similarity scores [LPW05]. However, most current information extraction approaches are

mostly rather tailored towards precision and large corpora where the same statement occurs often enough that not every appearance needs to be recognized for a good recall in terms of globally extracted statements. This leads typically to a low per-document recall. Hence, statements were annotated manually in the evaluation. The results show that the semantic based similarity can provide a competitive quality when compared to a keyword based model. However, we would in general suggest a hybrid model, as semantics and raw word similarity represent two different aspects in which documents can be similar. For further details we refer the interested reader to [Iss12], as a detailed discussion is out of the scope of this work.

4.10. Summary and Future Work

In this chapter we discussed the notion of statement search and presented a framework implementing it. Given a statement query (a set of statements), the proposed framework retrieves documents containing textual expressions supporting the queried statement(s). We propose a configurable ranking model based on language models for ranking witnesses, which can be tuned for favoring either documents with strong statement support or further information related to the given statement(s). Our evaluation results show that our ranking model outperforms term-based document ranking in finding strongly supportive documents and can compete with term-based document ranking methods in finding documents that discuss the topic of interest. Our framework proposes a user frontend supporting query formulation, either by translating queries from a user-friendly query format, like natural language or keyword queries, or by allowing the user to explore the ontology and formulate a query by selecting a subgraph from the ontology. We provide such an example frontend that integrates ontology exploration, query formulation and the display of result documents for a statement query.

There are a number of possible directions for future work. An important extension of the ranking model would be to automatically derive optimal tuning parameters based on user preferences. Another aspect in this context is diversification, i.e., given multiple statements and a set of documents that only contain subsets of the statements, adapt the ranking of the results in a way that documents in the top ranks contain different subsets of the statements. In the current model there is no mechanism to ensure that indications for different facts are mixed in the top results. For instance, if a query of two facts f_1, f_2 is given where for f_1 many witnesses with strong patterns and a high trust value are given while for f_2 only weaker patterns appear on less trustworthy pages, the results for f_1 will dominate the ranking and the top-k documents might not contain any indication for f_2.

In addition, considering user feedback might help to improve result ranking, e.g., by exploiting indirect feedback counting user clicks to measure the importance of a witness document. This might provide us with an actual trust function on documents. While our evaluation indicates that pagerank seems not to be a good estimator for the trustworthiness of a document, this indifference might also be due to our document corpus. A more detailed analysis and if necessary another trust estimation for documents could be helpful extensions of this work.

In this work, we focused on pattern based knowledge extraction in the form of RDF triples. However, the techniques presented in this work can be adapted and applied to other application scenarios such as opinion mining, as long as the mining approach can recognize some sort of pattern and provide it with a confidence to express given relations. Assume, for instance, an opinion miner harvests opinions on whether the iPhone is better than a particular Android phone. If a user wants to form her own opinion about whether to buy an iPhone, she will probably be interested in the actual reasons why people prefer one phone over the other. Accessing the original document provides such information. Now there might be thousands of pages where people state their opinion of phones, but the interesting ones are those that prefer one over the other and go on to discuss why. So

again ranking the witness documents in a meaningful way based on the confidence we have in the opinion statement and how much more information can be learnt by looking at the witness would be needed.

As information extraction is an expensive process a globally near-complete and always up-to-date index will remain a considerable challenge at least for the foreseeable future. Thus, the ability to retrieve documents for statements that were hitherto unknown to the statement index can for some use-cases be crucial. To improve recall in cases of emerging statements, potential witnesses can be retrieved for user inspection or to apply information extraction on demand. That is, when a user requests information that is not yet contained in the knowledge base, relevant documents can be searched on the Web and be used to judge the validity (or at least observability) of the statement searched for. We have proposed such an approach based on pattern variants where the queries to be generated are guided by the confidence information available from an extraction system, as our evaluation shows that such information can provide meaningful improvement in identifying convincing witnesses.

5. Query by Entity Examples Search

5.1. Motivation

A typical way to navigate an ontology is by entity relatedness, thus exploring the ontology along the edges defined by the relations between entities. For instance, given the entity `Arnold Schwarzenegger`, all the statements known about the actor could be listed - or just those deemed important, e.g. all his movies, his political positions and affiliations as well as any bodybuilding awards he won. A user could explore the neighbouring entities by choosing one of the related entities next, e.g. a user-frontend could offer links on all such related entities to get to an overview of their properties next. Thus, from `Arnold Schwarzenegger` a user might follow an `actedIn`-link to the movie `The Terminator` and go on from there to other actors or the director of the movie. This navigation approach can be implemented relatively easy, as it directly mirrors an ontology's structure and provides an intuitive way to explore the ontology, that often serves a user's interests well. However, in many cases, we are less interested in *related* entities, but rather in *similar* entities, i.e. entities that share certain properties with a given entity - or with a set of entities. For instance, we might be interested in other actors that acted in the same movies as `Arnold Schwarzenegger` or that also had a career in movies and politics; or directors like `Clint Eastwood` and `Quentin Tarantino` that are also actors; or powerful female politicians like `Angela Merkel` and `Margaret Thatcher`. If a user were to answer these *similarity queries* manually using a relational navigation approach this became an inefficient undertaking as some entities have quite many relations. In addition, in an explorative setting, a user might not always have a clear idea for which entities she is looking, but rather, out of interest, look for the most similar entities. Especially in these cases a justification for why entities might be deemed *similar* is essential.

Another aspect from a user's point of view is convenience and ease of use. Consider, for instance, the task to replace a particular worker of your workforce by another person, or search for possible replacements of a particular part in your production process, e.g. to optimize your production costs. Instead of specifying all the abilities of your worker or all the properties of the part, the natural choice would be to provide a system with the pointer to its (already existing) semantic description. A similar application case would be recommendation systems, especially in cases of ad-hoc recommendations in the absence of long term user ratings or as a supplementary module. For instance, assume browsing a shopping page looking for a new smart-phone or computer, you could either provide the old one you did like, but which is not sold anymore, as a query, or the shop could determine

suggestions based on the items you look at.

Note that in all cases we assume that the user expects similar entities to have at least roughly the same type as the examples, i.e. providing a person like Arnold Schwarzenegger we assume returned entities should be persons as well and not a dog named Arnold or a car that has the same height as the actor.

A central problem in the general underlying setting is the inherent ambiguity of examples. For instance, assume that a user provides `Arnold Schwarzenegger` as an example inquiring for more persons similar to him. The central question given such a query is the actual interest of the user (or external system) asking the query. A user might, for instance, be interested in other Austrian (ex-)body-builders, governors of California, actors that appeared in `The Expendables` or men who cheated on their wife. The user might not even know herself what group of people she is actually looking for. Our approach tries to capture these different *branches* of similar entities and to provide the most similar entities along each such possible query interpretation.

In principle, a search by example in a semantic dataset can be considered a faceted search [Tun09] with no direct control over the facets. A holistic approach to entity similarity, like using a random walk or vector model to compute pairwise similarity values is by definition agnostic of the different possible facets of a query by example. In this paper we propose a model that captures all possible facets in so called aspects of the query. Basic aspects represent single facets while our model explores the combinatorial facet space generating compound aspects to find the most similar entities under different points of view by identifying all entities that share a *maximal* set of semantic properties with the query. Thus, in the example from above, our system might identify the property combination of "being an Austrian bodybuilder who won the Mr. Olympics title" as a maximal compound aspect if some other entities share this property combination - and none of them share any additional property with Arnold on top. There can of course be multiple such maximal aspects, e.g. another one might represent "being an actor and governor of California". Entities that satisfy any such maximal aspect will be considered as very similar to the query by our system. Thus, by design our model also can make the facet(s) responsible for the inclusion of any returned entity transparent to the user. In principle, our approach to this point is very similar to a skyline search. However, in addition, we can extend the search-space ad-hoc by relaxing aspect constraints, if more candidate entities are needed, e.g. in offline applications or when user-interactivity is low. Our model guides this relaxation by ranking the maximal aspects. By relaxing, we may not only look at "Austrian body-builders that won Mr.Olympics", but also in general for Austrian bodybuilders or, for instance, Austrian sportsmen or body-builders of any nationality as well.

Our model is general enough to be applied to the family of Query By Entity ExampleS (QBEES) use-cases, i.e. the requirement being that a set of semantic entities is given as example query while the goal is to find other entities of the same kind that share properties with the examples provided. Possible applications could, for instance, cover the following scenarios: 1) interactive ontology navigation along similar entities instead of related entities, 2) offline set-completion, where a number of example entities of a general set of entities is provided for the system which should complete the set, 3) interactive

set-completion that allows users to pick entities from a set of similar suggestions until they have enough similar entities, 4) substitute-search, e.g. suggesting a set of optimal replacements for a given entity. However, since offline set completion is a well-defined IR task with standard evaluation test-data available, we focus on this application for the evaluation of our approach in Section 5.6.

Set completion comprises finding a set of entities with particular properties, given example entities that belong to the same set. A user might, for instance, look for movie directors that were at some point in their career also active actors. Thus providing Quentin Tarantino and Clint Eastwood the user might expect to find Sylvester Stallone and Peter Jackson. Formally the task is defined as follows: Given a set of *query* entities Q as initial *examples*, the goal is to retrieve a set of entities that share (at least some of) the shared properties of the entities in Q. We consider two application scenarios of this use-case. First, the application might be interactive, i.e. a user consecutively selects entities from a list of suggested entities, which are then added to the current entity set and can be used to refine the list of suggested entities, i.e. each selection by the user refines the original query as each entity is added to Q. On the other hand, the application might be non-interactive, e.g. used in an automatic setup without a human user involved at this stage, then the query cannot be refined and instead of allowing easy refinement, the first answer needs to be as good as possible. We will address both application scenarios in our evaluation in Section 5.6.

Contributions Within this work we

1. introduce an aspect-based model for entity search by example that at the core applies the skyline search approach to entity search by example.

2. extend this basic model by ad-hoc aspect relaxation to increase the search space when needed.

3. evaluate our QBEES approach against prior state of the art work on existing set-completion benchmarks.

Outline The remainder of this chapter is organized as follows. Section 5.3 introduces entity aspects that will be the basis of our model. The actual approach to compute similar entities is discussed in Section 5.4. Section 5.5 presents the ad-hoc aspect relaxation increasing the search-space. The evaluation results are discussed in Section 5.6 and in Section 5.7 we provide a summary and an outlook into future work.

5.2. Related Work

Entity search has been considered extensively in the literature, often with a focus on unstructured or semi-structured data. The entity tracks at TREC [BSdV11] and INEX[DIdV09] introduced mainly two different retrieval tasks: finding related entities (with a textual description of the relationship and a target category), and entity list completion (with a list of example entities and a textual description). While the majority of test collections has been built based on unstructured text and semi-structured XML documents, with the rise of larger general entity ontologies, like YAGO[HSBW13], Freebase[BTPC07] and DBpedia[ABK+07] and the Linked Open Data (LOD) cloud recent developments such as the Semantic Search Challenge[1] have extended this to semantic (RDF) data that forms a data graph with entities as nodes, the same scenario considered in this paper.

Core ingredients of many entity search systems are similarity measures for entities. A large body of work exists that exploits the graph structure for determining how similar two entities are. One of the earliest approaches was SimRank [JW02] which considers two entities as similar if their context is similar.

Albertoni and De Marino suggest a pair-wise asymmetric entity similarity [AM08] that is based on the degree features of a candidate resource contain selected features of a query resource combined with typical ontology concept similarity methods for entity types. Our system is in so far similar, that we consider similarity also in an asymmetric way and that our model shares the containment approach to some degree based on the aspects we identify. However, their system expects domain experts explicitly determine the features that are mostly responsible for entity similarity (optionally in an interactive way), while our approach always considers all features, but users can guide it towards the intended set of entities by choosing additional entities of the target group and thus interactively refining the query without particular knowledge about the underlying system or features. [RE04] proposes an asymmetric similarity measure called Matching-Distance Similarity Measure (MDSM). It is tailored for geospatial entity classes and their particular features. Similarity is computed by taking the generalization level of a geospatial reference into account and applying a context-based weighing of an entity's features. Also with a focus on the geospatial domain, Janowicz et al. [JKS+07] introduce a description logic(DL) based theory called SIM-DL to model similarity in OWL ontologies and extended the DIG standard with a matching language extension.

Pirró and Euzenat [PE10] combine the main ideas from a typical ontology oriented concept similarity with a feature model from the information theory domain resulting in their own feature and information theoretic (FaITH) measure based on the ratio of intrinsic information content from two concepts' closest common ancestor to the sum of their unique information content.

A more recent line of work uses random walks with restart to compute similarities of one entity or a group of entities to all other entities, such as Personalized Pagerank [Hav03], with a focus on relational data graphs [ACA06, MC10].

[1] `http://semsearch.yahoo.com/`

Another group of approaches uses features extracted from the context of entities to determine their similarity, including textual features (terms in the entity's URI or appearing in documents close to the entity) and structural features (categories or types of the entity). Balog et al. [BBdR11] propose to use language models that include terms and categories. Bron et al. [BBdR13], which is closest to our work, combines a term-based language model assuming a textual description of target entities as a query component with a simple structural model including uniformly weighted facts about the entity. In contrast, our query model does not include a keyword component and in particular does not assume a textual description of the target entities, our set of structural features in the aspects is more general, and our model allows to give different weights to different features. We experimentally compare our model to their model in Section 5.6.

Yu et al. [YSN+12] solve a slightly different problem where entities similar to a single query entity are computed, exploiting a small number of example results. Focusing on heterogeneous similarity aspects, they propose to use features based on so-called meta paths between entities and several path-based similarity measures, and apply learning-to-rank methods for which they require labelled test data. Wang and Cohen [WC07] present a set completion system retrieving candidate documents via keyword queries based on the entity examples. Using an extraction system additional entities are then extracted from semi-structured elements, like HTML-formatted lists.

A related field is semantic service matchmaking, the search for semantic web services given a service description as a query[KK09, DHM+04]. In principle it is a similarity search for a particular type of entity (a service) where general concept similarity is typically a sub-problem to be solved when comparing the expected service in- or output. State of the art matchmakers combine logical methods with typical concept and text similarity measures[KK09].

5.3. Entity Aspects

5.3.1. Aspect Based Entity Characterization

Remember that an ontology basically consists of entities $\mathcal{E}$and facts $\mathcal{F}$ connecting entities with their type classes, literals and other entities. When considering an entity, we might see different "sides" of that entity. For instance, consider `Arnold_Schwarzenegger`. If you are young, you might associate him mainly with current Hollywood movies like `The_Expendables` or `Escape_Plan` or know him as a politician, governing California for several years. If you are a bit older, you might rather remember his older career defining movies `Conan` or `The_Terminator`, or as a sports fan rather than a movie-goer you might perhaps remember his days as a price-winning bodybuilder. Our model tries to capture the different details we remember, e.g. `Arnold_Schwarzenegger` having appeared in this or that movie, as aspects of an entity and compositions of such details that combine into the different "views " we may hold about the entity as compound aspects.

We distinguish three kinds of basic aspects, namely *type aspects*, *relational aspects*, and *factual aspects*, that describe a given entity forming three levels of specificity encircling the entity. In the following each aspect type shall be defined.

Let us start with an example. Given an entity $q \in \mathcal{E}$ such as `Arnold_Schwarzenegger`, consider all statements that are incident with q in $\mathcal{F}$. These statements can either represent a type assertion of the entity q (e.g. (`Arnold_Schwarzenegger`, `type`, `actor`)) or other facts concerning q (e.g. (`Arnold_Schwarzenegger`, `bornIn`, `Austria`)). For any entity q, each such arc represents some "atomic property" of this entity (e.g. birthplace, type, occupation); the entity is characterized by the set of all "atomic properties".

By replacing the particular entity q in such a statement with a variable represented by '?x' we obtain a *parameterized statement* as discussed in Definition 2.1.14 with one named variable. For example, a factual statement (`Arnold_Schwarzenegger`, `bornIn`, `Austria`) and a type assignment (`Arnold_Schwarzenegger`, `type`, `actor`) naturally induce predicates of the form (`?x`, `bornIn`, `Austria`) and (`?x`, `type`, `actor`) that represent the "basic properties" of this entity of "being born in Austria" and "being an actor", respectively. If we consider the underlying ontology, then each such parameterized statement yields a query pattern. For instance, (`?x`, `bornIn`, `Austria`) would match all entities that occur as subject with relation `bornIn` and object `Austria`, i.e. all entities born in Austria. We call such predicates *factual aspect* and *type aspect* of the entity, respectively.

Definition 5.3.1 (factual aspect)**.** Given an entity e, for each fact (`e`, `r`, `e'`) $\in \mathcal{F}$ and each fact (`e'`, `r`, `e`) $\in \mathcal{F}$ there is a *factual aspect* of e represented by the parameterized statement (`?x`, `r`, `e'`) and (`e'`, `r`, `?x`) respectively.

Definition 5.3.2 (type aspect)**.** Any factual aspect that is based on the `type` relation is considered as a *type aspect*. We exclude the set of type aspects from the set of factual aspects.

In addition we also consider *relational aspects* that capture the frequency in which an entity is involved in instances of a relation (e.g. the information that a person acted in 20

movies and directed 1 movie may signal that they are rather an active actor than an active director). We represent this information by replacing the remaining argument of a factual aspect by an anonymous variable ('?') and introducing an instance constraint k such that (?x, actedIn, ?)[k] indicates that the entity acted in at least k movies.

Definition 5.3.3 (relational aspect)**.** A relational aspect (?x, r, ?)[k] represents all families of variable assignments where each family contains at least k variable assignments that solve (?x, r, ?) such that each assignments assigns the same entity e to $?x$, but a different entity to ?. We will use the shortcut notation (?x, relation, ?) whenever $k = 1$. Thus, (?x, bornIn, ?) represents the aspect of "being born somewhere" and (?x, actedIn, ?)[3] represents all entities that acted in at least three different movies.

To wrap up, we consider three types of aspects that form a 3-level entity characterization hierarchy,

1. **type aspects** (level 1) of the form (?x, type, class) where the set of all type aspects of an entity e reflects the set of classes $\mathcal{C}(e) \subseteq \mathcal{C}$ that e is an instance of: $\mathcal{C}(e) = \{c \in \mathcal{C} | (\texttt{e}, \texttt{ type}, \texttt{ c}) \in \mathcal{F}\}$.

2. **relational aspects** (level 2) of the form (?x, actedIn, ?)[k] reflecting the multi-set of relations from $\mathcal{R}$ that appear in $\mathcal{F}$ with e.

3. **factual aspects** (level 3) of the form (?x, bornIn, Austria) representing the set $\mathcal{F}(e) \subseteq \mathcal{F}$ of facts involving e, i.e. $\mathcal{F}(e) = \{(\texttt{arg1}, \texttt{ r}, \texttt{ arg2}) \in \mathcal{F} | arg1 = e \vee arg2 = e\}$.

Notice that these three levels of entity characterization form a natural hierarchy. Each next level of characterization provides information concerning the entity that is more specific than the previous. Thus, the type aspects on level 1 provide the most general information and restrict the possible set of relations that may apply to the entity (level 2) and the particular selection of relations that occur with the entity puts constraints on possible related entities (level 3).

Definition 5.3.4 (basic aspects)**.** We name the union of the three kinds of aspects for an entity q the set of *basic aspects* of an entity and denote it as $\mathcal{A}(q)$. Note that we will later on (Section 5.4.3.2 and 5.4.3.3) discuss variations of $\mathcal{A}(q)$ that only include a subset of the type aspects and the relational aspects.

Now, with each basic aspect we associate its set of entities, where each entity satisfies the property represented by the aspect.

Definition 5.3.5 (entity set)**.** For a basic aspect a we define the *entity set* $\mathcal{E}(a)$ as the set of all entities $e \in \mathcal{E}$ that contain a in their set of basic aspects $\mathcal{A}(e)$.

For example, for the aspects (?x, type, actor), (?x, bornIn, ?) and (?x, bornIn, Austria) their entity sets consist of all entities that are born in Austria, born somewhere, and are actors, respectively.

Definition 5.3.6 (compound aspect)**.** A *compound aspect* of entity q is any nonempty subset A of $\mathcal{A}(q)$.

For example, for two basic aspects $a_1 =$ (`?x`, `bornIn`, `Austria`), $a_2 =$ (`?x`, `type`, `actor`) $\in \mathcal{A}(q)$ the set $A = \{a_1, a_2\}$ represents the compound aspect of "being an actor born in Austria".

We naturally extend the definition of entity set to compound aspects: $\mathcal{E}(A)$ is the set of all entities that share *all* basic aspects in A (in the former example: all the entities in $\mathcal{E}$ that are both actors and are born in Austria).

Definition 5.3.7 (subsumption)**.** For any two compound aspects A and B, where $A \subset B$, A is "more general ", as it is potentially shared by more entities. Thus, we say a (compound) aspect A *subsumes* a (compound) aspect B iff $A \subset B$.

5.3.2. Similarity by Maximal Aspects

Now let us consider how to identify similar entities given a query entity q. By definition, for any compound aspect A of entity q, any entity $e \in \mathcal{E}(A)$ has all the basic aspects represented by the compound aspect A. Furthermore, the more aspects from $\mathcal{A}(q)$ an entity $e \in \mathcal{E}$ shares with q, the more *similar* it is to q. The entities that share *all* the aspects with a given entity q would be extremely similar to q, but often only q itself has this property, since many basic aspects are very specific, and $\mathcal{A}(q)$ often characterizes the entity uniquely. Thus, to look for entities most similar to q, we have to relax $\mathcal{A}(q)$ by dropping as few basic aspects from it as possible.

Definition 5.3.8 (maximal aspect)**.** We say that a compound aspect A of entity q is a *maximal aspect* of q iff it satisfies the following two conditions:

1. $\mathcal{E}(A)$ contains at least one entity other than q
2. A is maximal wrt inclusion (i.e., extending this set of basic aspects with any more basic aspect of q would violate the first condition).

Notice that if A is a maximal aspect of q, all entities $e \in \mathcal{E}(A) \setminus \{q\}$ are "maximally" similar to q wrt to a specific set of basic aspects.

We denote the *family of all maximal aspects of entity* q as $M(q)$.

Finally, we naturally extend all the concepts related to aspects of a single entity $q \in \mathcal{E}$ for the case of a *set of entities* $Q \subseteq \mathcal{E}$. Thus, $\mathcal{A}(Q)$ will denote the set of all basic aspects *shared by all* entities in Q, i.e. $\mathcal{A}(Q) = \cap_{q \in Q} \mathcal{A}(q)$. Similarly, we define the family of maximal aspects of a *set of entities* $Q \subseteq \mathcal{E}$, $M(Q)$ by appropriately modifying the first condition in the definition of $M(q)$ as follows: the entity set of a maximal aspect of an entity set Q must contain *at least one entity other than those in* Q.

Lemma 5.3.1. *Given two compound aspects A and B where $A \subset B$, it cannot be that both A and B are a maximal aspect.*

Proof. Assume there is an aspect $A \subset B$ then A would is in violation of requirement 2 for maximality (we could add $B - A$ to A). □

Assume a query is given by $Q = \{\texttt{Arnold\_Schwarzenegger}, \texttt{Sylvester\_Stallone}\}$. Then, consider the compound aspects $A_1 = \{(\texttt{?x}, \texttt{type}, \texttt{ActionMovieActor}), (\texttt{?x}, \texttt{livesIn}, \texttt{USA})\}$, i.e. all action movie actors from the U.S., and $A_2 = \{(\texttt{?x}, \texttt{type}, \texttt{ActionMovieActor}), (\texttt{?x}, \texttt{livesIn}, \texttt{USA}), (\texttt{?x}, \texttt{type}, \texttt{director})\}$, i.e. all action movie actors from the U.S. that are also directors. If we just consider these two aspects, only A_2 would be a maximal aspect, providing, for instance, Clint Eastwood as a similar entity, since A_1 subsumes A_2.

Lemma 5.3.2. *Each entity $e \in \mathcal{E}$ can only be in the entity set of one maximal aspect of a query Q.*

Proof. Assume there are two maximal aspects A and B such that $e \in \mathcal{E}(A) \wedge e \in \mathcal{E}(B)$. Then it also holds that $e \in A \cup B$ and thus, A and B cannot be maximal as both subsume $A \cup B$ and $e \in A \cup B$ implies that $A \cup B$ is a maximal aspect. □

5.4. Aspect-based Similar Entity Search

5.4.1. Overview

We now discuss how our aspect model is used to retrieve entities given examples. Formally the task is defined as follows: Given a set of *query entities* Q, we want to retrieve a ranked list of the k most similar entities R, where similarity is primarily defined by aspects shared with the entities in Q. To solve this task, we select k entities with the following procedure:

1. Compute the family of maximal aspects $M(Q)$ of Q.
2. filter the maximal aspects by *typical types* of the entities in Q,
3. rank the maximal aspects,
4. pick the entity e not in $Q \cup R$ with the largest popularity $pop(e)$ (see Section 5.4.2) from the top aspect A, add e to R, and update the aspect's rank or remove A if $\mathcal{E}(A)$ is *empty with respect to* R,
5. repeat step 4 until k entities are picked, i.e. $|R| = k$, or no aspects are left.

We will now give a short overview for each step, a more detailed explanation follows in the next subsections.

1. Maximal Aspects Given a set of entities Q, we first compute for each query entity $q \in Q$ the set of its basic aspects $\mathcal{A}(q)$. We then identify the aspects $\mathcal{A}(Q)$ shared by all query entities by intersecting their aspect sets: $\mathcal{A}(Q) = \cap_{q \in Q} \mathcal{A}(q)$. If this intersection is empty, the query entities must be of very different types (such as a person and a location), and we do not retrieve any similar entities (In practice this constraint can be loosened to deal with faulty data (see Section 5.4.4)). From $\mathcal{A}(Q)$, we can compute the corresponding family of maximal aspects $M(Q)$. Each such maximal aspect $A \in M(Q)$ represents the set of entities $\mathcal{E}(A)$ that share a maximal set of properties with *all* query entities, but there may be several such maximal aspects. We discuss our algorithm to identify maximal aspects in Section 5.4.3.

2. Typical Types It is a natural assumption that the selected entities should be of similar type as the query entities. It would, for example, usually be intended that, given a city, the result should be other cities and not, e.g., a country, even if they share the same river passing through. Thus, for each query Q we determine a set of *typical types* $\mathcal{T}(Q)$ and consider only maximal aspects that contain at least one such typical type (or a more specific sub-type) as a basic aspect. We discuss how to compute $\mathcal{T}(Q)$ in Section 5.4.4.

3. Aspect Ranking The resulting maximal aspects are of different specificity and thus quality. For instance, a maximal aspect for Arnold Schwarzenegger might consist of (`?x`, `type`, `person`) and (`?x`, `hasBirthplace`, `Austria`) while another one might consist of (`?x`, `type`, `GovernorOfCalifornia`). Hence, we rank the maximal aspects and give priority to entities from aspects which are more likely to be of interest (see Section 5.4.5).

4. Picking an entity Similarly to aspects, the entities in the entity set of an aspect may have different likelihoods of importance to a user, especially for relatively broad aspects. We use two different entity importance measures that provide an estimated importance or popularity $pop(e)$ for an entity e, a graph-based and a click-based estimator; both are discussed in Section 5.4.2. Once an entity has been picked from an aspect A's entity set $\mathcal{E}(A)$, we check whether the aspect can contribute more entities to the result R. If $\mathcal{E}(A) - (R \cup Q) = \emptyset$, we call the aspect *empty with respect to* R and remove it; we will simply say that an aspect is *empty* when it is clear to which set of entities we refer. Note that by Lemma 5.3.2 no other maximal aspect of Q is affected as e is only in A's entity set.

5.4.2. Entity Popularity

Some entities are in general more popular than others of the same kind and are thus more likely to be the ones a user has in mind and would expect as a result. For instance, if you consider all actors then it is more likely that a random person would name Arnold Schwarzenegger or Johnny Depp as an example for the set of all actors than Ralf Möller[2] or Eric Hennig[3].

Hence, when selecting an entity in step 4 of our algorithm, we prefer entities with high *popularity*. In our model, the popularity $pop(e)$ is a numerical value in the interval $[0, 1]$, where a higher value represents a higher estimated popularity, and $\sum_{e \in \mathcal{E}} pop(e) = 1$.

We currently provide two different estimators for the popularity $pop(e)$ of an entity e, a graph-based estimator and a click-based estimator. The graph-based estimator uses the stationary probability of an entity in a global random walk on the knowledge graph to estimate its popularity. Similar node importance measures inspired by PageRank [BP98] have been applied in many graph applications.

As a big advantage, it does not use any information outside the graph and can therefore be applied to any knowledge graph. We consider the undirected version of the knowledge graph spanned by $\mathcal{F}$, i.e., a graph (V, E) where the vertices are equal to the set of entities $\mathcal{E}$ and the edge set E contains an edge $\{u, v\}$ whenever a statement from $\mathcal{F}$ connects u to v or v to u. We chose this undirected representation since each relation in the ontology has a corresponding natural inverse relation (e.g., a `bornIn` relation has the natural inverse `isBirthplaceOf`) that may not be explicitly modeled in the ontology. The

[2] A German-American actor of some minor international relevance
[3] A young American actor who played only in minor roles so far

recursive definition of our graph-based popularity measure $pop_G(e)$ is then

$$pop_G(e) = \frac{\epsilon}{|V|} + (1 - \epsilon) \sum_{f:\{e,f\}\in E} \frac{pop_G(f)}{deg(f)}$$

where $deg(f)$ is the degree of vertex f. We iteratively solve this equation through a power iteration implemented in Apache Giraph[4], setting $\epsilon = 0.15$.

Popularity measures that take user behavior into account can often better represent true popularity of entities, but such information is not always available. For the YAGO knowledge base used in our experiments (see Section 5.6), we can exploit that it was created from Wikipedia and a mapping from YAGO entities to Wikipedia articles is straightforward. For Wikipedia, precise information on hourly click frequencies for each page are available[5], from which we can derive click frequencies $c(e)$ for each entity e in YAGO, setting $c(e) = 1$ for entities that cannot be mapped to Wikipedia. The click-based popularity $pop_C(e)$ of entity e can then be computed as

$$pop_C(e) = \frac{c(e)}{\sum_{e'\in\mathcal{E}} c(e')}$$

In our experiments, we aggregated click statistics for August 2012, December 2012, and May 2013.

Our entity popularity estimators do not take the query into account, hence the popularities are independent of the query and the aspect. In some cases we may therefore be led astray picking a globally popular entity from an aspect unrelated to the reasons why this particular entity is famous. For instance, we may pick Angela Merkel before Marie Curie from an aspect about female scientists since our popularity estimate is larger for Angela Merkel than for Marie Curie. However, Angela Merkel is usually considered popular as German chancellor and not for her earlier career as a physicist, while Marie Curie would much more likely be associated with her scientific achievements. While such an aspect-dependent popularity measure is an interesting topic for future work, in most cases a global popularity estimate is a good basis for selecting relevant entities; for instance, it has been a well proven component in the form of PageRank in traditional web search.

Note that we exploit entity popularity not only for ranking entities, but also for ranking aspects where we prefer, in some variants of our ranking method, aspects that contain many popular entities (see Section 5.4.5). This is driven by the assumption that a user might rather have aspects in mind that contain on average more well-known entities.

[4] `http://giraph.apache.org/`

[5] `http://dumps.wikimedia.org/other/pagecounts-raw/`

5.4.3. Finding maximal aspects

5.4.3.1. Entity-centric Algorithm

The definition of maximal aspects suggests a naive algorithm exploring the aspect space starting from the full aspect set $\mathcal{A}(Q)$ as initial candidate. The algorithm would relax each candidate aspect not being maximal by one after another dropping its basic aspects and continuing with the generated more general aspects until along each path a maximal aspect is reached. Such a naive algorithm is sketched below in Algorithm 1 where $relax(S, a) = S \setminus \{a\}$. It starts with A (line1) and checks if relaxing any of A's basic aspects results in a maximal aspect S' (lines 5-12); if S' is not maximal, it is considered for further relaxation (line 10). This process continues until no non-maximal aspects are left for further exploration (line 3). Note that the initial candidate set is typically not maximal since the associated entity set $\mathcal{E}(A)$ will only contain the query entities Q. Should it initially also cover other entities, then those entities are maximally similar.

Algorithm 1 Naive approach to find maximal aspects

Input: A: set of basic aspects, Q: set of query entities
Output: $\mathcal{M}$: set of maximal aspects

```
1: G ← {A} //candidate set
2: M ← ∅
3: while G ≠ ∅ do
4:     for all S ∈ G do
5:         for all a ∈ S do
6:             S' ← relax(S, a)
7:             if S' maximal aspect for Q then
8:                 M ← M ∪ {S'}
9:             else
10:                G ← G ∪ {S'}
11:            end if
12:        end for
13:        G ← G \ {S}
14:    end for
15: end while
```

However, the search space for such a naive algorithm is quite large as its size is bounded by the number of compound aspects that are subsets of $\mathcal{A}(Q)$ and thus potential candidates. In the worst case, where only each basic aspect is individually representing a maximal aspect (or when there are no maximal aspects at all), the whole search space will need to be explored. Let $n = |\mathcal{A}(Q)|$, then the runtime is bounded by 2^n, the size of the power set of the basic aspects of Q. Note that there is an analogous naive approach that explores the search space in a bottom-up fashion, starting from the individual basic aspects and then adding basic aspects until maximal aspects are found.

Let us have a closer look at the runtime constraints. Consider the naive algorithm again. It would in the worst case need to check every potential compound aspect for maximality according to Definition 5.3.8.

Given a candidate A, let us first consider the first condition, namely that $\mathcal{E}(A)$ contains an entity not in Q. To compute $\mathcal{E}(A)$, we have to retrieve the intersection over the entity sets of each basic aspect $a \in A$ from the ontology. This can be done by an appropriate query that takes care of the intersection or we load (and potentially cache) the entity sets of the individual basic aspects and then do the intersection.

Either way, assuming that $k = \max_{a \in \mathcal{A}(Q)} |\mathcal{E}(a)|$ the operation is possible in $O(n \cdot k)$, e.g. by intersecting the at most n entity sets $\mathcal{E}(a)$ (where $a \in A$) that each have at most a size of k one after another with each other (assuming lookup and removal are constant operations on the set data structure). Considering the size of Q as a constant, the final step of the test is completed with a removal of Q from the intersection and a size check on the remaining set in constant time.

As for the second condition, we need to verify that there is no other aspect $B \subseteq \mathcal{A}(Q)$ which is maximal and subsumed by A, otherwise A would violate the second condition for maximality. Naively implemented, this means exploring the search space 'upwards ' and checking for all other aspects whether they contain A, an operation in $O(2^n \cdot n)$, as there are at most 2^n aspects of at most size n. Note in particular that checking the direct parents is not sufficient in the naive approach as aspects can be constructed along different paths, i.e. have many parents not all of which need to be maximal if one is. So this would mean in the worst case the runtime of a very naive approach is in $O(2^n) \cdot (O(n \cdot k) + O(2^n \cdot n)) = O((2^n)^2 \cdot n + 2^n \cdot (n \cdot k))$.

Of course this can be done more efficiently. Note that there are two critical components in the runtime. First it is the large number of nodes to consider, i.e. the potential 2^n aspects. Second, the number of entities for basic aspects can be relatively large, i.e. k is typically much larger than n. Thus the combination of both due to the repeated entity intersections is particularly crucial. In the following we discuss an algorithm that precomputes all the compound aspects based on $\mathcal{A}(Q)$ that contain any other entity than those in Q.

Using the naive approach this would reduce the first maximality condition check to a lookup in this set of precomputed compound aspects, which is possible in constant time, while we will see that the precomputation is possible in $O(n \cdot k)$. In addition we will describe an alternative algorithm to ensure the second condition. While it does not reduce the asymptotic runtime, in practice the constants will be small enough as tests provided in Section 5.6.3.1 confirm.

Given the set of query entities Q, we first identify all basic aspects $\mathcal{A}(q)$ of each query entity $q \in Q$ and intersect them: $\mathcal{A}(Q) := \bigcap_{q \in Q} \mathcal{A}(q)$. The set $\mathcal{A}(Q)$ then contains the shared basic aspects of the entities in Q. Now, for each basic aspect $a \in \mathcal{A}(Q)$ we compute its entity set $\mathcal{E}(a)$. Then for each basic aspect a either $|\mathcal{E}(a)| = |Q|$ or $|\mathcal{E}(a)| > |Q|$; $|\mathcal{E}(a)| < |Q|$ cannot occur as each entity $q \in Q$ is in $\mathcal{E}(a)$ by construction. All basic aspects a with $|\mathcal{E}(a)| = |Q|$ are specific to the query entities, i.e. no other entity shares any such

aspect, they can therefore not be member of a maximal aspect and we ignore them.

Now we consider the union $U(\mathcal{A}(Q))$ of all entity sets of all basic aspects $a \in \mathcal{A}(Q)$, i.e. $U(\mathcal{A}(Q)) = \cup_{a \in A(Q)} \mathcal{E}(a)$, the set of all entities that share at least one basic aspect with all entities in Q. Given $U(\mathcal{A}(Q))$, we compute for each entity e in $U(\mathcal{A}(Q)) \setminus Q$ all basic aspects from $\mathcal{A}(Q)$ that are also basic aspects of e, which we denote by $\mathcal{A}_Q(e) := \mathcal{A}(e) \cap \mathcal{A}(Q)$, i.e. $\mathcal{A}_Q(e)$ is the set of all basic aspects that e shares with all entities in Q.

Now, each entity generates a candidate for a maximal aspect given by $\mathcal{A}_Q(e)$. Let $I_{\mathcal{A}(Q)}$ be the set of all these candidates: $I_A(Q) = \{A_Q(e) | e \in U(A(Q)) \setminus Q\}$.

$I_{\mathcal{A}(Q)}$ contains now all the compound aspects that 1) can be generated from $\mathcal{A}(Q)$ and 2) contain another entity than those in Q. Note that it was generated by retrieving all entities for each basic aspect $a \in \mathcal{A}(Q)$, an operation possible in $O(n \cdot k)$. Then for each entity we collect all basic aspects to which it belongs. Using a hash map, this is again possible in $O(n \cdot k)$: For each basic aspect $a \in \mathcal{A}(Q)$ and each entity e of its entity set $\mathcal{E}(a)$ we place a in a set maintained for e, which gives us $I_A(Q)$. So we could use $I_A(Q)$ now as a lookup set for the test of the first condition for maximal aspects, integrate it in the naive algorithm and thus achieve an asymptotic runtime of $O((2^n)^2 \cdot n + 2^n + (n \cdot k))$.

However, using $I_A(Q)$ we can also spare the exploration of the search space, as we only need to look at these candidates and identify the subset which satisfies the first condition.

Thus, once $I_A(Q)$ is computed, we remove all subsuming aspects from $I_{\mathcal{A}(Q)}$, i.e. we remove all $A \in I_{\mathcal{A}(Q)}$ for which $\exists B \in I_{\mathcal{A}(Q)}$ such that $B \supset A$. Given an aspect B, we need to check B against all aspects $A \in I_A(Q)$ and then remove any A that subsumes B. Let $m = |I_A(Q)|$. As A and B have at most size n we need at most $O(n \cdot m)$ operations to compare B against all $A \in I_A(Q)$. The remove operation is constant in the number of aspects being removed. However, note that in the course of the whole procedure we will at most remove all of $I_A(Q)$. Thus this is in $O(m)$ which means for the complete subsumption check and removal a runtime bound in $O(m) \cdot O(n \cdot m) + O(m) = O(n \cdot m^2 + m)$ in addition to the cost to compute the candidate aspects of $O(n \cdot k)$.

So, overall $O(n \cdot m^2 + m) + O(n \cdot k) = O(n \cdot m^2 + m + n \cdot k)$. Note that m is bound by the number of entities, but also by the number of potential aspects $A \in \mathcal{A}(Q)$, such that we could replace m by k or 2^n, yet in practice m is much smaller than k or 2^m. In fact, since the ontology is stored in a database and the computations are done in memory, the crucial part now is the initial computation of the candidate set and in particular the retrieval of entities for basic aspects.

We will provide some details on the implementation in Section 5.4.6 and you can find an evaluation comparing the time needed to retrieve basic aspects with the time it takes to compute maximal aspects in Section 5.6.3.1.

Theorem 5.4.1. *This approach provides us with the maximal aspects:* $I_{\mathcal{A}(Q)} = M(Q)$.

Proof.

1. $I_{\mathcal{A}(Q)} \subseteq M(Q)$: Assume it exists an aspect $A \in I_{\mathcal{A}(Q)}$ such that $A \notin M(Q)$. Then:
   - either $\forall e \in \mathcal{E}(A)$ it holds $e \in Q$; yet, it holds that for each $\mathcal{A}_Q(e) \in I_{\mathcal{A}(Q)}$ there is some $e \in \mathcal{E}(A)$ that generated A with $e \notin Q$ by construction.
   - or $\exists B \in M(Q)$ such that $A \subset B$. Then it needs to hold by maximality of B that $\exists e \in \mathcal{E}(B)$ such that $e \notin Q$. Then e shares all basic aspects $B \subseteq \mathcal{A}(Q)$ and it cannot share any other basic aspect a with the entities of Q, otherwise B were not maximal as a could be added. But then by construction of $I_{\mathcal{A}(Q)}$ it needs to contain B before the subsumption based removal step. In this case however, A would have been removed during the subsumption based removal step as it subsumes B.
2. $I_{\mathcal{A}(Q)} \supseteq M(Q)$: Assume that there is a maximal aspect B such that $B \notin I_{\mathcal{A}(Q)}$. Then $\exists e \in B$ such that $e \notin Q$ (otherwise it is not a maximal aspect) and $B \subset \mathcal{A}(Q)$. Hence, $e \in U(\mathcal{A}(Q))$ and thus, e generates a candidate aspect $C := \mathcal{A}_Q(e)$. Then either
   a) $C = B$, then $B \in I_{\mathcal{A}(Q)}$,
   b) or $C \supset B$, then B is not a maximal aspect, as it subsumes C,
   c) $C \subseteq B$ cannot be, as $e \in B$, thus, e satisfies all basic aspects of B and hence $B \subset C$ as C is generated by e.

□

If $I_{\mathcal{A}(Q)}$ is empty, there are no maximal aspects, because no entity not in Q shares any property combination with all entities in Q, this follows by completeness of $I_{\mathcal{A}(Q)}$. In such a case we cannot provide results (see also Section 5.4.4). Some of the cases where this can happen are prevented by allowing type aspects not shared by all entities to be in $\mathcal{A}(Q)$ (see Section 5.4.4).

5.4.3.2. Specific Types

Typically, we consider only the most specific types when generating maximal aspects, as this reduces the number of candidate aspects and entities. For a query Q whose entities share the basic aspects (`?x`, `type`, `scientist`), (`?x`, `type`, `politician`), and (`?x`, `type`, `person`), and where (`scientist`, `subclassOf`, `person`) $\in \mathcal{F}$, (`politician`, `subclassOf`, `person`) $\in \mathcal{F}$, i.e. the class hierarchy indicates both `scientist` and `politician` are sub-classes of `person`, we therefore ignore (`?x`, `type`, `person`) for the maximal aspects. As a consequence, we might occasionally miss some maximal aspects that combine the more general type with basic aspects that the entities in Q exclusively combine with the more specific types. For instance, the two basic aspects (`?x`, `type`, `person`) and (`?x`, `wasBornOnDate`, `1947-07-30`)

might make up a maximal aspect, if neither another politician nor scientist share the same day of birth. We compare the effects of filtering out non-specific type aspects in Section 5.6.2.2. Note that, even when including non-specific type aspects, we will filter out 24 very general type aspects, like (`?x`, `type`, `entity`) for performance reasons. These types are typically so broad, and thus, unselective, that they would not add any helpful information anyway. In the following, unless explicitly stated otherwise, non-specific type aspects are not considered for maximal aspect generation, i.e. there exist no $a, b \in \mathcal{A}(Q)$ s.t. $a =$(`?x`, `type`, `T1`) and $b =$ (`?x`, `type`, `T2`) where (`T1`, `subclassOf`, `T2`).

5.4.3.3. Relation Aspects

Remember that $\mathcal{A}(Q)$ also contains relation aspects. For instance, $\mathcal{A}(Q)$ might contain (`?x`, `actedIn`, `?`)$[k]$ representing all entities that acted in at least k movies, though the movies can be different amongst the entities.

We investigate four different granularity variants of how such relation aspects are generated. Assume that an entity in Q participates in n different instances of a relation r, with $n \geq 1$, e.g. it might be an actor that appeared in a movie trilogy, thus it has three `actedIn` instances. When computing $\mathcal{A}(q)$ we generate relational aspects in four variants, such that for each the following holds respectively:

all $\forall k \in \{1, \ldots, n\}$.(`?x`, `r`, `?`)$[k] \in \mathcal{A}(q)$ (full multi-set)

exts $\forall k \in \{1, n\}$.(`?x`, `r`, `?`)$[k] \in \mathcal{A}(q)$

any (`?x`, `r`, `?`)$[1] \in \mathcal{A}(q)$, this represents $\mathcal{R}(e)$ as a set

none no relation aspects of any kind are included in $\mathcal{A}(q)$

So considering the movie trilogy example, there are three possible relation aspects, (1)(`?x`, `actedIn`, `?`)[1], (2)(`?x`, `actedIn`, `?`)[2] and (3)(`?x`, `actedIn`, `?`)[3]. The *all* variant would generate all three, the *exts* variant would generate (1) and (3), the *any* variant would generate (1) and the *none* variant none.

After $\mathcal{A}(q)$ is computed the results are intersected into $\mathcal{A}(Q)$. Hence, if all entities share some equal relation aspects, potentially based on totally different concrete instances, those relation aspects are carried on into $\mathcal{A}(Q)$.

While it is clear that more basic aspects will increase the aspect search space, we will investigate the qualitative effects of each variant in the evaluation.

5.4.4. Typical Types

In order to restrict suggested entities to those that have roughly the same type as the query entities, we try to identify a set of general types that would typically be associated with the query entities and use these types as constraints for result entities. Given a set Q of query entities, we determine the set $\mathcal{T}(Q)$ of typical types for result entities. The set $\mathcal{T}(Q)$ consists of all types shared by all entities in Q that fall into a particular cut $\mathcal{T}$ of the class hierarchy. The classes in this cut are a parameter to affect the granularity expected of returned results, i.e. they determine how far any entity returned as result can deviate from the types of the query entity, i.e. any returned entity will have to share at least one type of the types in $\mathcal{T}$. The set $\mathcal{T}$ can be determined individually per use-case and ontology setting. For our YAGO based setting, we selected classes like `person`, `politician` or `actor` but avoided classes like `entity` or `object` for being too general and classes like `Americans_of_Austrian_descent` for being too specific. The full list of classes we used can be found in the appendix, Appendix A.2.1. Note that concepts within the allowed cut can be sub-classes of another, this is intentional to allow the more specific case if it matches. When computing $\mathcal{T}(Q)$, we first identify the types that are shared by all entities in Q that are also in $\mathcal{T}$, i.e. we intersect $\mathcal{T}$ with $\mathcal{C}(q)$ of all $q \in Q$. Then the most specific types in this set are identified by filtering out all classes that are a super class of another class in the intersection. This yields $\mathcal{T}(Q)$. However, if the result is an empty set, e.g. because the entities do not share any type within $\mathcal{T}$, most likely due to faulty data or a fuzzy input, then we intersect $\mathcal{T}$ with the union of $\mathcal{C}(q)$ over $q \in Q$. In such a case the corresponding basic aspects will also be added into $\mathcal{A}(Q)$, but their effect in at least some of the ranking approaches is reduced, see Section 5.4.5 for this. As an example, consider the query $Q = \{$`Schwarzenegger`, `Stallone`$\}$, then the two entities might share the types `actor`, `AmericanActor` and `entity`, but an intersection with $\mathcal{T}$ would only leave `actor`, thus, enforcing resulting entities to be some kind of actor.

5.4.5. Aspect Ranking

Given a set of query entities Q, we investigate four different ranking approaches based on three general intuitions that we discuss in the following.

First, given an aspect A we might consider its likely use to the user by the popularity it represents, i.e. how likely the entity set is popular amongst users, thus more likely to satisfy her needs. Given an estimator for the popularity of an entity, $pop(e)$, the popularity of an aspect can be estimated as the aggregated popularity of its entities, i.e., $pop(A) = \sum_{e \in \mathcal{E}(A)} pop(e)$. As for estimating the entity popularity, remember the different approaches discussed in Section 5.4.2. We normalize the aggregated popularity of an aspect by its specificity, i.e. how many entities it represents and thus how specific the property combination it represents is, to yield our *simple popularity based ranker*:

$$spop(A) := \frac{1}{|\mathcal{E}(A)|} \times pop(A) \tag{5.1}$$

A basic aspect b might be considered for its value $v(b)$ to be generated. We estimate the value of a basic aspect by its selectivity, i.e. $v(b) = \frac{1}{|\mathcal{E}(b)|}$ (so aspects with few entities are preferred) or its "inverse" $v^{-1}(b) = 1 - \frac{1}{|\mathcal{E}(b)|}$ (which prefers aspects with many entities). Based on this we can estimate how much 'worth' a compound aspect A represents by $val(A) = \sum_{a \in A} v^{-1}(a)$. We normalize this by the overall worth as follows

$$nval(A) = \frac{val(A)}{\sum_{B \in M(Q)} val(B)} \tag{5.2}$$

This will prefer aspects with many general properties, but typically we are especially interested in those aspects that represent a very special combination of properties not occurring too often, thus we combine the normalized value with the aspects specificity to yield another ranker:

$$cost(A) := \frac{1}{|\mathcal{E}(A)|} \times nval(A) \tag{5.3}$$

Finally, we consider how close an aspect resembles the complete set of Q's basic aspects by looking at the value ratio $ratio(A, B)$ between two (compound) aspects A, B where $A \subseteq B$

$$ratio(A, B) = \frac{\sum_{a \in A \cap B} v(a)}{\sum_{b \in A \cup B} v(b)} \overset{A \subseteq B}{=} \frac{\sum_{a \in A} v(a)}{\sum_{b \in B} v(b)} \tag{5.4}$$

This provides us with two ranker variations, one pure version only based on this ratio 'distance' (*distp*) and one that combines the ratio with the popularity represented by the aspect (*dist*).

$$\begin{aligned} dist(A) &:= ratio(A, \mathcal{A}(Q)) \times pop(A) \\ distp(A) &:= ratio(A, \mathcal{A}(Q)) \end{aligned} \tag{5.5}$$

As discussed in Section 5.4.4, in some cases we allow type aspects to be included in $\mathcal{A}(Q)$ even when they are not shared by all entities in Q. For each such type t the value $v(b)$ of the corresponding basic aspect b is weighed by the ratio of entities in Q that share the type aspect b, i.e. $v(b) = \frac{1}{|\mathcal{E}(b)|} \frac{|\{q \in Q | t \in C(q)\}|}{|Q|}$. The same holds for $v^{-1}(b)$ analogously.

In summary, we consider the following four ranking functions:

spop	$\frac{1}{\lvert\mathcal{E}(A)\rvert} \times pop(A)$
cost	$\frac{1}{\lvert\mathcal{E}(A)\rvert} \times nval(A)$
dist	$ratio(A, \mathcal{A}(Q)) \times pop(A)$
distp	$ratio(A, \mathcal{A}(Q))$

Table 5.1.: Ranking variants

In addition a random ranking *random* is used in the evaluation as a lower bound for the ranking component. It simply picks a random entity from all available maximal aspects.

5.4.6. Implementation

The implementation of our system and the evaluation setup (see Section 5.6) is based on the YAGO2 ontology stored in a database. To improve the retrieval of entity sets that correspond to basic aspects we maintain a most-recently-used cache for entity sets of basic aspects that stores the entity sets of up to χ_{basic} basic aspects. Similarly a most-recently-used cache is maintained storing the popularity values for up to χ_{pop} entities. We also make sure to load entity popularity values in batches rather than individually. And finally, when ranking aspects based on their aggregated popularity, we sample the popularity from a sample of size χ_{sample}. Once we pick an entity from an aspect we need to retrieve the popularity for the remaining entities, such that we can update the aspects popularity and rerank if necessary. In addition, we make sure that for every (compound) aspect the entity set is only computed once during the same query execution, i.e. the entity set is kept in main memory until the query is fully answered. The system is written in Java and the database server runs PostgreSQL 9.1.

5.5. Relaxing Aspects

Our algorithm from Section 5.4.1 removes a maximal aspect as soon as all its entities are picked. Such a behavior is desired when we aim at only retrieving the most similar entities considering different points of view reflected by the different maximal aspects. However, if only the now empty maximal aspect reflects the user's general information need, entities from any other maximal aspect would likely not be relevant. Consider, for example, a now empty maximal aspect $A_1 = \{$(`?x`, `type`, `actor`), (`?x`, `bornIn`, `Austria`), (`?x`, `actedIn`, `The Terminator`)$\}$ corresponding to "Actors born in Austria that appeared in Terminator". By design of maximal aspects, there can neither exist a maximal aspect $A_2 = \{$(`?x`, `type`, `actor`), (`?x`, `actedIn`, `The Terminator`)$\}$ ("Actors who appeared in Terminator") nor $A_3 = \{$(`?x`, `type`, `actor`), (`?x`, `bornIn`, `Austria`)$\}$ ("Actors born in Austria"), because then we could in each case add an additional basic aspect ending up with A_3 which would violate the definition of a maximal aspect. Instead of retrieving entities from other unrelated maximal aspects (such as "people born in Steiermark"), it is often better to relax the constraints of the empty maximal aspect, retrieving entities that are 'similar' to those of the empty maximal aspect and belong to the same general 'kind' of entities. In the example, we could consider actors that appeared in Terminator, or actors born in Austria. Thus, we want to *relax* the now empty maximal aspect's constraints to cover more entities of the same general 'kind'.

To this end we extend the notion of maximal aspects $M(Q)$ to $M(B, Q')$, which allows us to restrict the set of basic aspects that can be included in a maximal aspect.

Definition 5.5.1 (generalized maximal aspect)**.** Given a set of basic aspects B and a set of entities Q', we define $M(B, Q')$ as the set of aspects such that for any aspect $A \in M(B, Q')$ it holds:

1. $\mathcal{E}(A)$ contains at least one entity not in Q'
2. A is maximal with respect to inclusion
3. $A \subseteq B$.

So basically this means we can restrict the set of maximal aspects to be generated from a particular set of basic aspects B. Thus, the original definition of maximal aspects $M(Q)$ can be seen as a short-form for $M(\mathcal{A}(Q), Q)$. We call any aspect $A \in M(B, Q')$ a *maximal aspect with respect to B and Q'*. We may omit to name B or Q' when it is clear what is the set of basic aspects and the set of entities the maximal aspect is based on.

Now, to relax an empty maximal aspect A we consider not only the original query entities Q but also all entities in R (which are already picked as result) and compute $M(A, Q \cup R)$, which can be done with the algorithm from Section 5.4.3. Note that when computing $M(A, Q \cup R)$, only aspects that are subsets of A are considered, such that R does not influence the choice of basic aspects, but only which entities are acceptable to determine the maximality of a considered aspect.

Since A can not be a maximal aspect with regard to $Q \cup R$, as it is empty with respect to R (it does not contain entities outside $Q \cup R$), this will yield a new set of additional maximal aspects which are all subsets of A that can be added to the set of ranked aspects given they satisfy the typical type constraints. We call this approach `recursive plain` or short `rec`. Note that, with this approach at any time all aspects considered are maximal with respect to $Q \cup R$ and $\mathcal{A}(Q)$. However, remember that typically we do only consider the most specific types and then consider the case where A represents "Austrian actors that appeared in Terminator" represented by {(`?x`, `type`, `AustrianActor`), (`?x`, `actedIn`, `Terminator`)}. In this case we could only find {(`?x`, `type`, `AustrianActor`)} or {(`?x`, `actedIn`, `Terminator`)} as new maximal aspects, but not {(`?x`, `type`, `actor`), (`?x`, `actedIn`, `Terminator`)}. Here we gain more options when we take all type aspects into account right from the beginning as discussed in Section 5.4.3.2, in which case A would need to also contain (`?x`, `type`, `actor`) assuming that (`AustrianActor`, `subclassOf`, `actor`)$\in \mathcal{F}$, otherwise A would not be maximal. Note however, that adding all type aspects has other side-effects, basically increasing the aspect search space, thus adding additional maximal aspects and increasing the run-time. If combined with the full set of type aspects, we refer to this relaxation strategy as `recursive full` or short `recf`.

In an attempt to include type relaxation while also sticking with the specific types and keeping the runtime cost of relaxing an aspect low, we suggest a heuristic that only considers a single relaxation step looking for new maximal aspects, but also considering type relaxation in an ad-hoc fashion. In principle, this *one-step* relaxation follows the naive maximal aspect identification approach described by Algorithm 1 in Section 5.4.3.1 using A and $Q \cup R$ as input, with two exceptions: 1) It does only relax one step without entering recursion, i.e. dropping line 10. 2) It uses a redefined $relax(S, a)$ as described in Algorithm 2. Basically, for every type aspect that shall be relaxed, we look for its parent concept. If there is one we replace the original type from a with its parent type instead of dropping a completely, thus relaxing type aspects along the type hierarchy only when needed. In case a type t has more than one direct parent t', we generate all variations where t is replaced by its direct parent. This approach we denote as `1-step`.

Algorithm 2 Aspect relaxation in case of one-step heuristic

Input: S: aspect, a: basic aspect where $a \in S$
Output: $\mathcal{S}'$: aspect based on S where a has been relaxed

1: **if** a=(`?x`, `type`, `t`) $\wedge\ \exists t'$ s.t. (`t`, `subclassOf`, `t'`)$\in \mathcal{F}$ **then**
2: choose most specific t' s.t. (`t`, `subclassOf`, `t'`)$\in \mathcal{F}$
3: $S' \leftarrow S \setminus \{a\} \cup \{$(`?x`, `type`, `t'`)$\}$
4: **else**
5: $S' \leftarrow S \setminus \{a\}$
6: **end if**

Figure 5.1 illustrates the three different variations at an example, however, for the sake of simplicity we left out some arcs that are not of interest for the following discussion. Assume a maximal aspect MA_1 consisting of the basic aspects p_1 =(`?x`, `actedIn`, `The Terminator`),

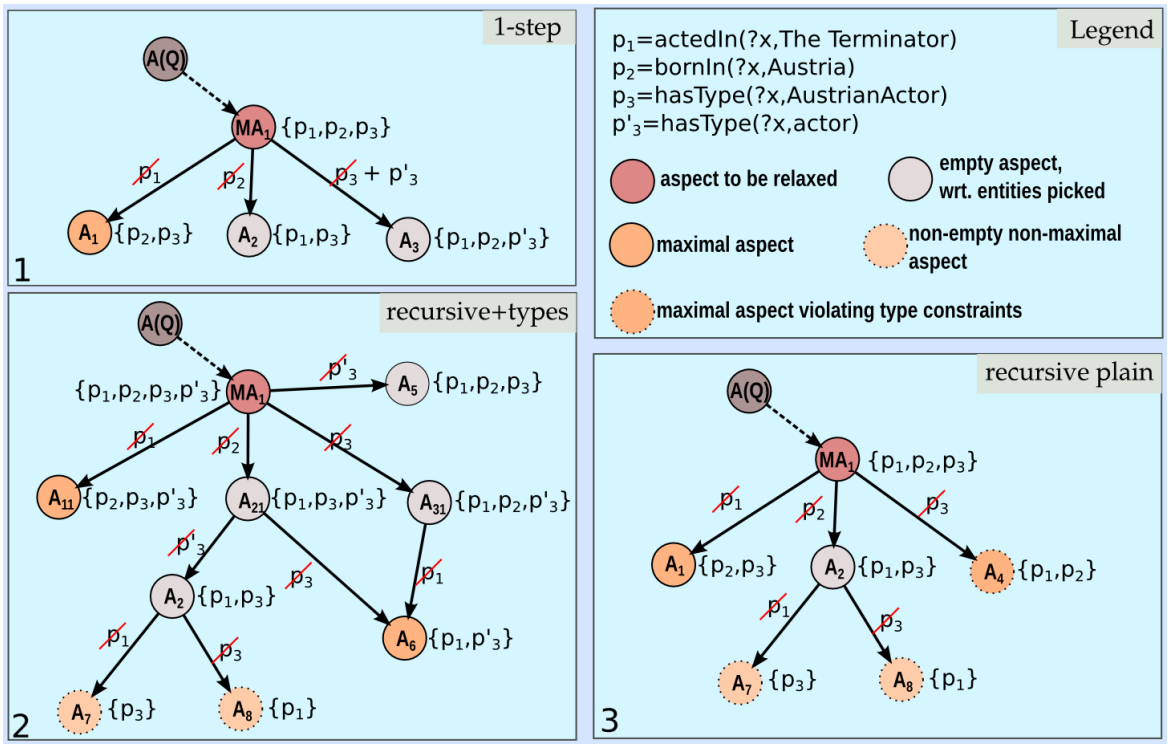


Figure 5.1.: Relaxing Aspects variations

p_2 =(?x, bornIn, Austria), p_3 =(?x, type, AustrianActor) is to be relaxed. Also assume p'_3 =(?x, type, actor). Subfigure 1 in Figure 5.1 shows the local one-step approach. Thus, when relaxing p_1 or p_2 they are simply dropped while p_3 is relaxed to p'_3 along the type hierarchy (assuming (AustrianActor, subclassOf, actor)$\in \mathcal{F}$). In this example we assume that $\mathcal{E}(A_2)$ and $\mathcal{E}(A_3)$ are empty with respect to R, such that one new maximal aspect, A_1, is found.Note that, no matter whether A_1, A_2 or A_3 are maximal aspects, the relaxation stops after this one step. However, if any aspect is added as a maximal aspect and at a later point emptied, the one-step relaxation would apply another relaxation step starting from this aspect. Thus, with A_1 being added to the set of ranked maximal aspects in a later relaxation of A_1, this path of the aspect graph can later be explored further, while the other two paths will be neglected. Subfigure 2 and 3 illustrate the aspects the recursive approach would investigate and find in the example setting, although the actual algorithm would not explore the graph along relaxation steps but follow the procedure explained in Section 5.4.3. That is, instead of $\mathcal{A}(Q)$ we start with the basic aspects from MA_1 and compute $M(MA_1, Q \cup R)$. Thus, only the non-empty aspects are considered at all and checked for maximality. As pointed out, only when all types are added from the beginning, type relaxation by dropping the more specific type aspect can occur, thus only in subfigure 2 A_6 representing actors (of any nationality) that acted in The Terminator is considered and identified as a maximal aspect. While the example may suggest that the recursive approach - even without full type aspects - is, in general, superior to the one-step heuristic, this is not

the case. Consider the possibility that A_3 is non-empty. Then the one-step solution would investigate A_3 and possibly relax that at a later stage into A_6, thus covering this branch while the plain recursive approach would never consider it. However, in Section 5.6.2.6 we will evaluate the qualitative difference between the different approaches experimentally.

5.6. Evaluation

5.6.1. Setup

As the underlying ontology for the following evaluation we use the "core" variant of the Wikipedia-based general knowledge base YAGO2 [HSBW13]. Note that we ignore the following meta-relations: `hasGloss`, `hasPreferredMeaning`, `hasPreferredName`, `hasWikipediaUrl`, `isCalled`, `means`, `subclassOf`, `subpropertyOf`. The classes for the cut $\mathcal{T}$ in the type hierarchy have been selected by a human assessor identifying those types that human users would typically use to roughly describe entities in everyday language, i.e. the goal was to pick classes that are specific enough to describe concrete entities yet not too specific excluding similar entities if we would enforce it upon all results. Thus, for instance classes like `entertainer`, `athlete`, `person`, `politician` or `institution` were chosen, while omitting classes like `entity`, `object`, `organism` for being too general and classes like `People_from_Graz` or `American_people_of_German-Jewish_descent` for being too specific. We selected 200 classes out of YAGO's more than 300000 concepts. The full set of selected classes can be found in Appendix A.2.1. Depending on the concept hierarchy layout an automatic approach to determine the cut through the concept hierarchy could be based on a minimal and maximal depth or the number of instances, or look at the local difference in the instance number when determining which concepts are becoming too specific to achieve a similar quality; this is an interesting area for future work. With regard to caching, we chose a cache size of 50 for χ_{basic} (entity sets of basic aspects) and of 250.000 for χ_{pop} (entity popularity). We restricted sampling of aspect popularity to 1000 entities (χ_{sample}).

Dataset Bron, Balog and de Rijke [BBdR13] introduced three datasets, two of which are based on data from the INEX 2007 and INEX 2008 entity-ranking track respectively, while the third consists of topics from the search challenge of the Semantic Search Workshop 2011 (SemSearch'11). Each dataset consists of several topics, to which entities have been assigned with graded relevance assessments. We have adopted all three datasets, mapping the original entities (given as DBpedia or LOD URIs) to YAGO where possible and removing them otherwise. We use the dataset from INEX 2007 (inex2007) to analyze our model variations and the datasets from INEX 2008 (inex2008) and the Semantic Search Workshop (semsearch2011) as test data to compare our approach against a random walk approach and the approach introduced by Bron et al. [BBdR13].

Queries We randomly generate example entity queries of different query size, i.e. number of example entities, for each topic from the relevant entities for this topic. In particular, we generate up to ten distinct queries per query size for sizes 1 to 5, if possible, i.e. as long as there remains at least one other relevant entity. Table 5.2 shows for each dataset the number of topics, the overall number of queries and the number of queries per query size. Note that the number of topics shrinks with increasing size of the queries, as not all topics provide enough relevant entities for larger query sizes. Thus, evaluation results between

query sizes are not directly comparable, as the topic base changes. Also note that for some queries there are less than 5 relevant entities. This is why we have slightly less queries of size 1 than of size 2, as in such cases queries of more than one entity can generate more variations as queries.

dataset	#topics	#queries	#size 1	#size 2	#size 3	#size 4	#size 5
inex2007	23	862	163	201	188	170	140
inex2008	48	1729	366	375	343	325	320
semsearch2011	39	1449	284	317	302	280	266

Table 5.2.: Datasets

Measures As evaluation measures, mean average-precision (MAP) and mean normalized discounted cumulative gain (nDCG) for rankings of length 10 and 100 have been computed. In some cases we also report precision, recall, recall-precision(r-prec) and mean reciprocal rank (mrr). The satisfaction rank to determine the mrr is based on the first rank r that provides a relevant entity. While the assessments are graded, with values from 0 to 2, for binary measures like MAP and mrr a relevance threshold of 1 was assumed, i.e. any assessment other than 0 was considered relevant. In some cases we also consider the stricter setting with the threshold being set at relevance 2. In such cases the threshold is explicitly written in brackets next to the measure, e.g. mrr (2) or mrr (1).

5.6.2. Model Analysis

First we will investigate the effects of several model variations using the inex2007 dataset.

5.6.2.1. Entity Importance Estimation.

We first evaluate the impact of the two entity importance estimators from Section 5.4.2. As clearly visible in Figure 5.2, the Wikipedia click based importance estimation (`+wi` versions) is always more effective in supporting the rankers than the knowledge graph based random walk estimator (`+rw` versions). Hence, we use the Wikipedia page clicks for entity importance estimation in the following experiments.

5.6.2.2. Specific Types vs. all Types

As discussed earlier in Section 5.4.3.2, typically we drop type aspects that subsume other type aspects with respect to the class hierarchy. Figure 5.3 shows the effects of using all types (`_allt`) vs. using only the most specific types (`_sp`) for the basic ranking approaches. We can see that, without relaxation at least, using all types has a minor mixed effect, i.e. no version is clearly better. Since using all types however, is more costly with respect to runtime and resource usage, in the following we use only the most specific types if not

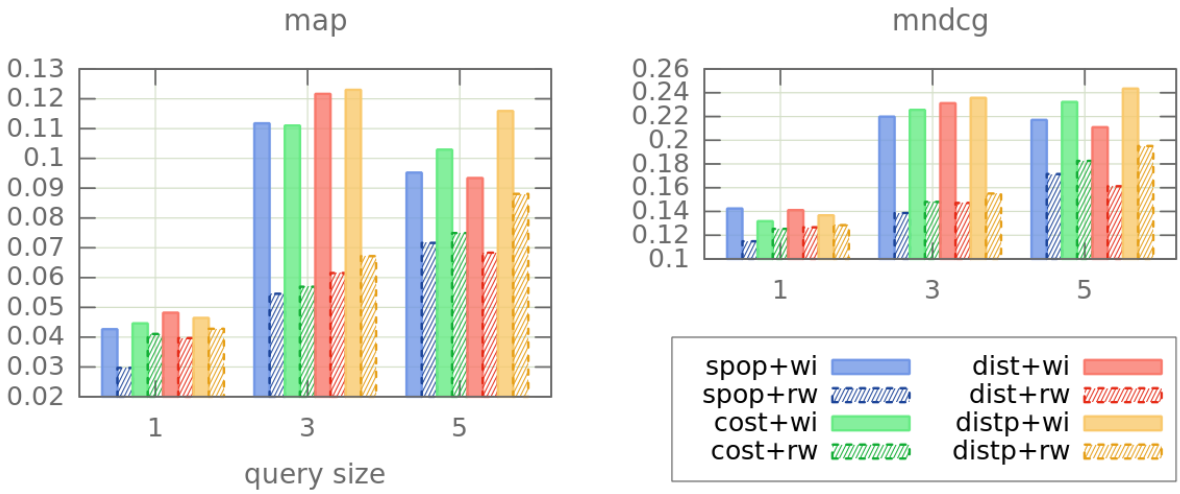


Figure 5.2.: Importance Estimators Top10 - INEX2007

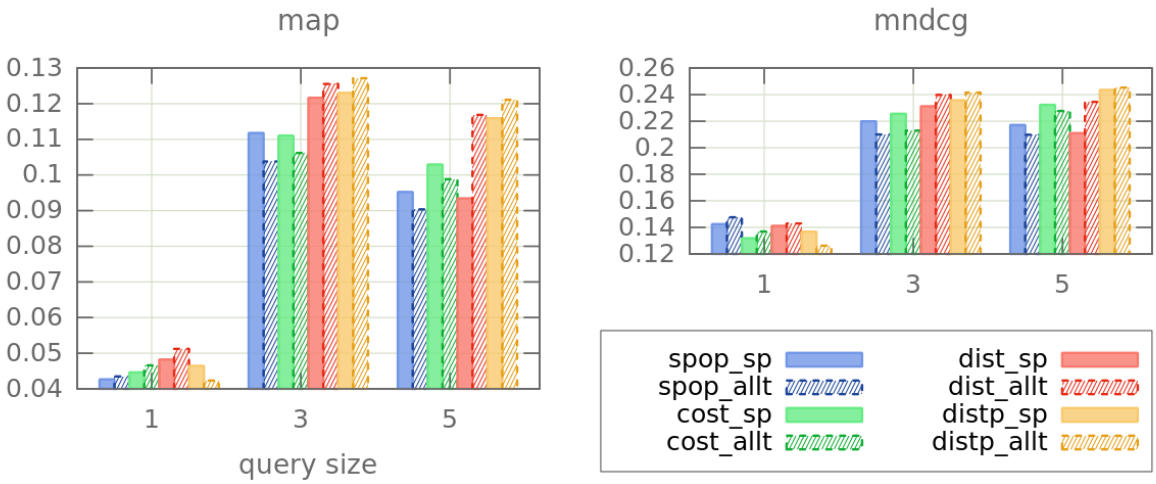


Figure 5.3.: Specific vs. all types Top10 - INEX2007

explicitly stated otherwise. Also remember that the topic constellation also changes for different query sizes, as not all topics are represented in the same percentage for each query size. Besides potential overfitting, this explains why with more information that size 5 queries provide the results are often a little worse than for size 3 queries.

5.6.2.3. Type Constraints

As the original topics of the INEX datasets come with Wikicategory based target categories for the entities, we automatically mapped these to YAGO wikicategory classes where possible and used them as a constraint instead of identifying typical types as constraints with our method from Section 5.4.4. In cases where the category mapping failed, we fall however back to our typical type finding approach for determining constraint types.

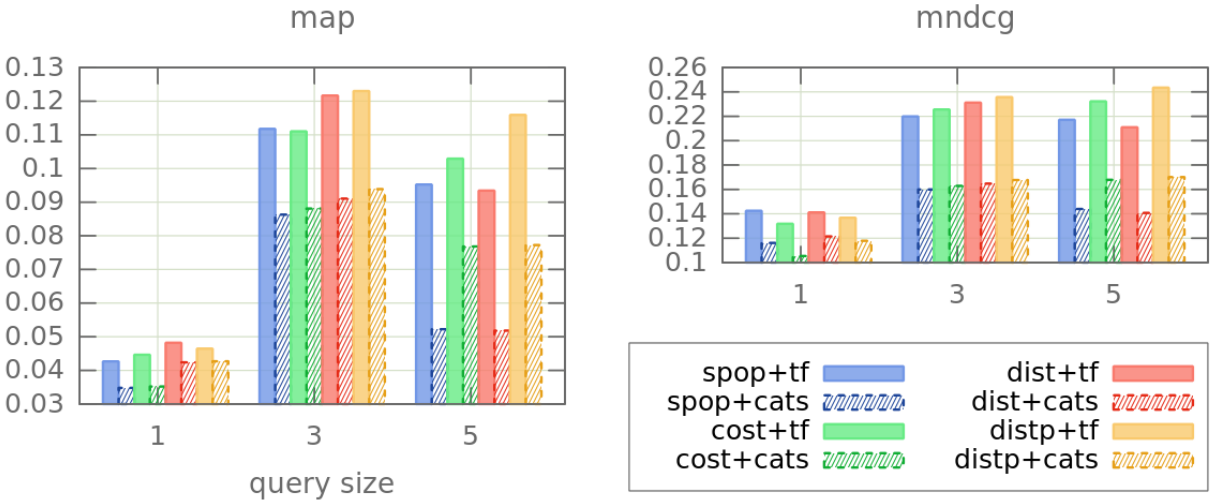


Figure 5.4.: Topic Constraints Top10 - INEX2007

Figure 5.4 compares the results of this topic category based approach(`+cats` versions) with our pure typical type approach(`+tf` versions), see Section 5.4.4 for details. The results show that our own method clearly outperforms the manually selected query categories. The reason for this is that the categories are too specific in general, and YAGO is unfortunately also quite incomplete in this perspective, such that the category constraints often filter out too many entities that just lack the specific type assignment within YAGO.

5.6.2.4. Relation Aspects

We now evaluate the impact of the different ways to include generalized relation aspects introduced in Section 5.4.3.3 on result quality. We compare 1) no relation aspects (none), 2) exactly one per relation if there is at least one instance (any), 3) one for each instance

of a relation (all) and 4) one representing the presence of any relation instance and one representing the exact amount of relation instances (ext).

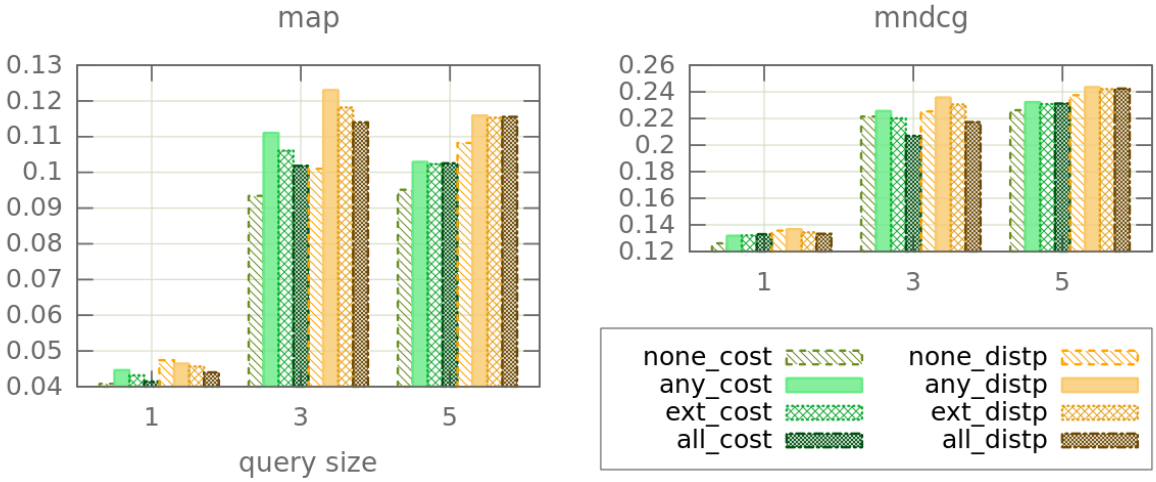


Figure 5.5.: Relation Aspects Top10: MAP, nDCG @ INEX2007

Figure 5.6 indicates that all relation aspect variants help to retrieve correct entities, but can also disturb the ranking in a negative way (for instance, nDCG of `cost` for query size 3). Our compromise of including generalized relation aspects only to represent the fact that the entity has any relation instance is often the most effective. Hence, we typically use the *any* setting in the following, as it seems to provide the best cost/use balance (the more aspects are introduced the more costly the computations).

5.6.2.5. Ranking Benefits

As a baseline providing a lower bound for the ranking approaches, an additional `random` ranking variant has been implemented that simply selects entities randomly from all maximal aspects. If we compare (see Figure 5.6) the result quality of the `random` ranking with that of the other ranking approaches, we can see that all our approaches indeed perform better than a random ranking and thus, the ranking component is shown to be a substantial component in our approach besides the maximal aspect pre-filtering.

5.6.2.6. Relaxing Aspects

In Section 5.5 we discussed three aspect relaxation strategies that relax an empty maximal aspect when more entities are needed. Here we compare the result quality among the different relaxation strategies and put these into perspective by comparing against the non-relaxing variants. As the relaxing approaches aim at providing more entities in the long run, we also look at how they compare at longer rankings of size 100. For simplicity, only the relaxing variations of three ranking approaches `cost`, `dist` and `distp` are compared, leaving `spop` out. As for the relaxing strategies, we compare the one-step approach (_step),

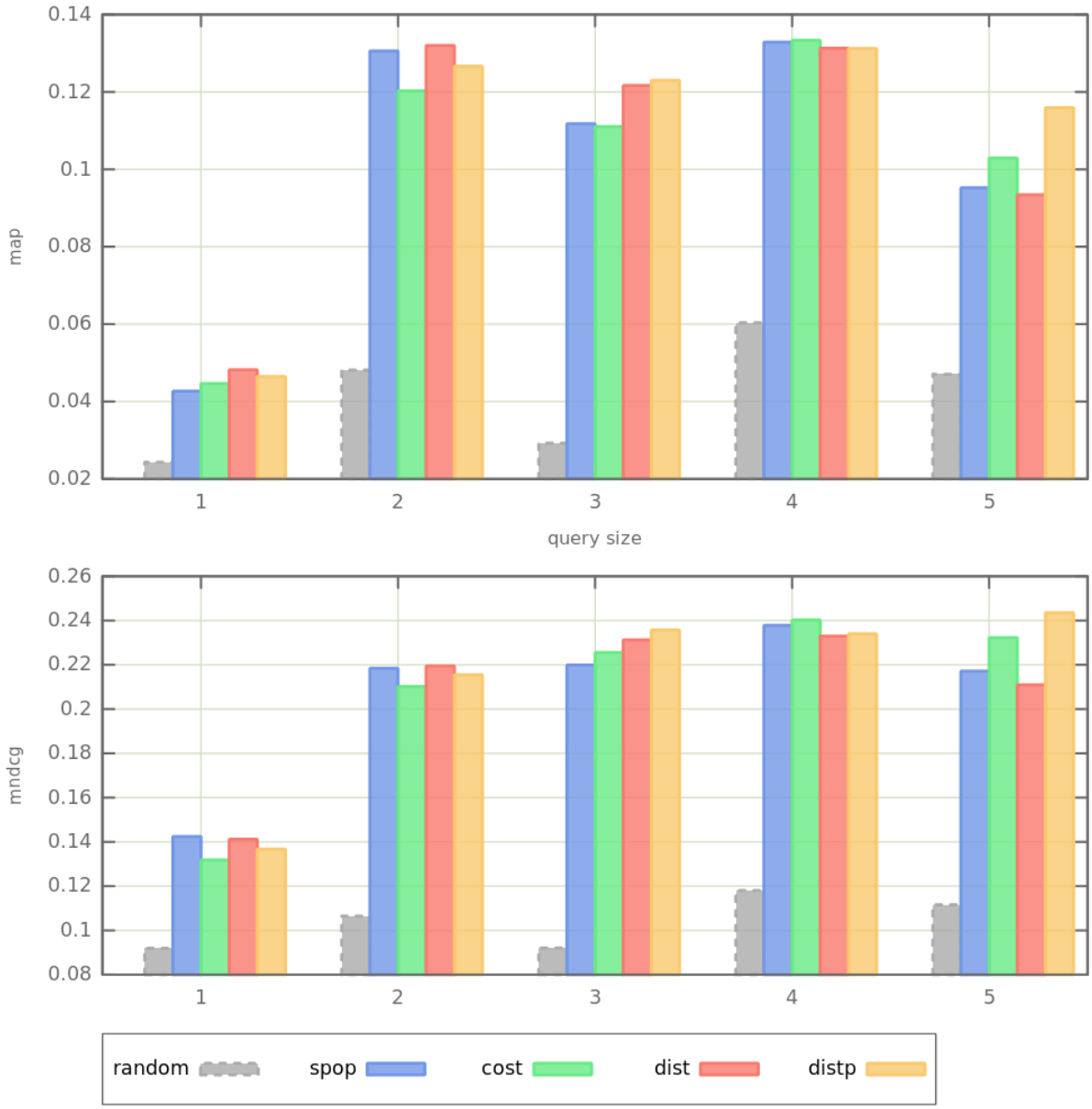


Figure 5.6.: Ranking Approaches vs Random Ranking Top-10: MAP, nDCG @ INEX2007

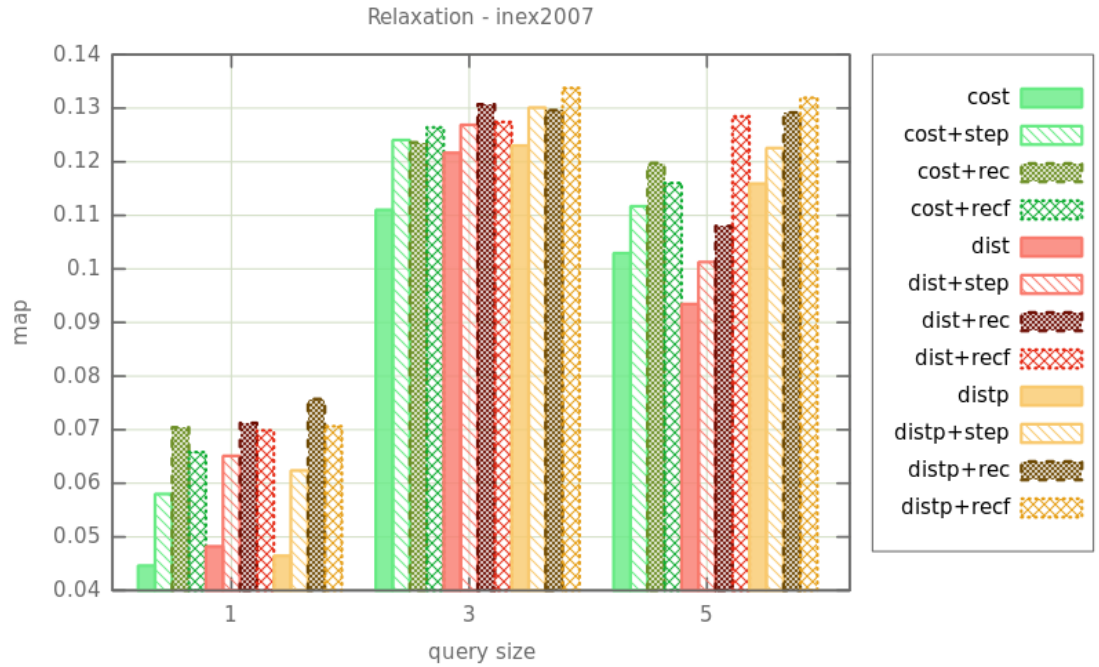
the recursive approach with specific types (_rec) and the recursive approach using full type aspects (_recf).

Figures 5.7(a), 5.8(a), 5.9(a), and 5.10(a) provide the MAP, nDCG, recall and precision values for rankings of length 10, while Figures 5.7(b), 5.8(b), 5.10(b) and 5.9(b) provide the respective values for rankings of size 100.

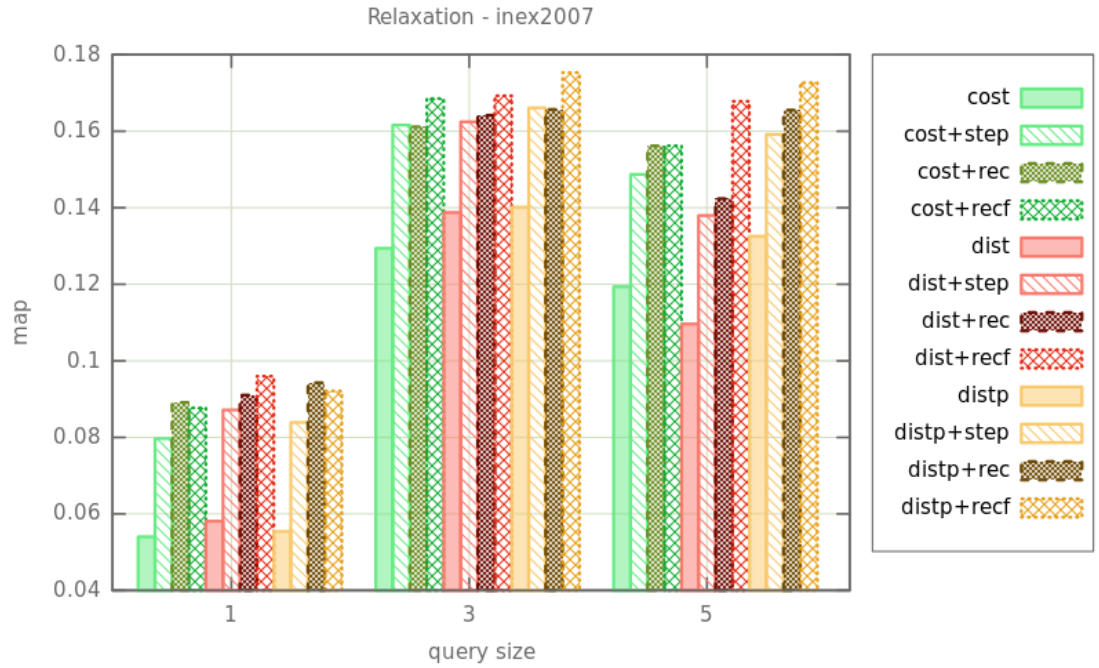
When looking at the MAP and nDCG values, it is clearly visible that compared to the non-relaxing variants, the relaxing approaches provide supreme result quality, especially when considering longer rankings. Amongst the relaxing variants the fully recursive (_recf) version of each ranking approach often leads the field by most measures, while the one-step heuristic and `rec` achieve nearly the same quality and lie close together. Note, however, that the run-time for the recursive approaches, especially with all types, can be significantly larger (see Section 5.6.3.1).

Considering the recall (see Figure 5.9(a) and Figure 5.9(b)), it is evident that a (and probably the) main reason for the supreme result quality of the relaxing approaches is a larger overall recall. Especially in the long-tail the fully recursive relaxing approach gains the most by additional relevant entities as indicated by Figure 5.9(b).

In contrast, the non-relaxing variants often provide a better precision, as can be seen in Figure 5.10(a) and Figure 5.10(b). Note however, that one reason for the superior precision, especially when we look at rankings of size 100 is that the non-relaxing variants often only result in very short rankings, i.e. typically shorter than 100 entities (see Figure 5.11 for the average ranking length). In such cases, when computing the precision the divisor (ranking length) is limited here to the actual ranking length returned.

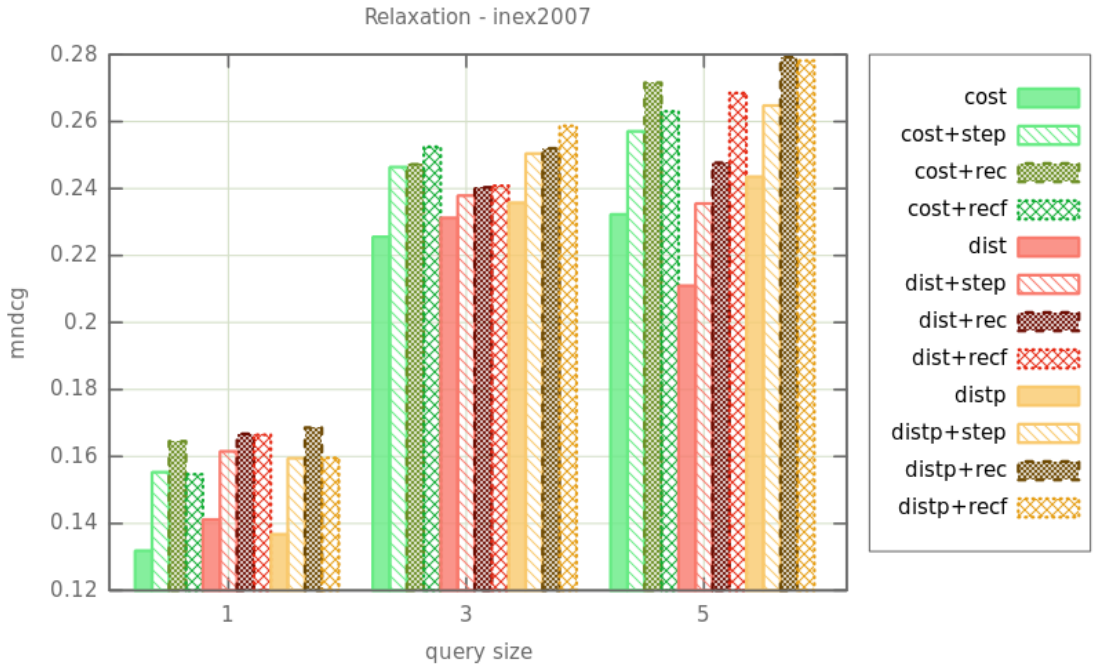


(a) Aspect relaxation variations Top-10: MAP @ INEX2007

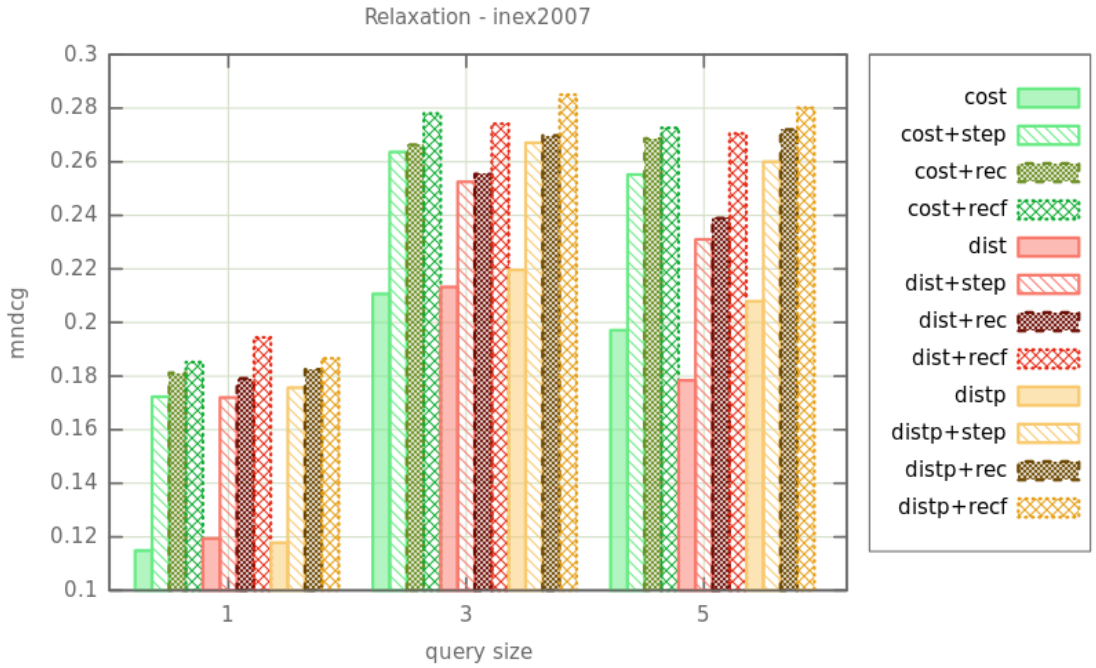


(b) Aspect relaxation variations Top-100: MAP @ INEX2007

Figure 5.7.: Aspect relaxation variations: MAP @ INEX2007

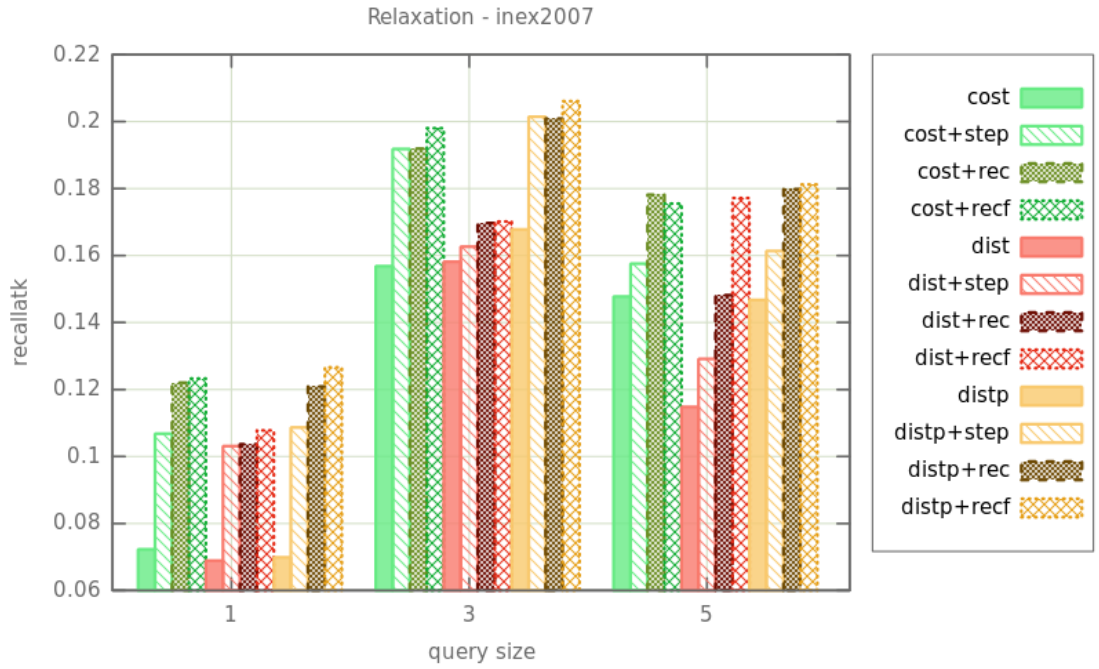


(a) Aspect relaxation variations Top-10: nDCG @ INEX2007

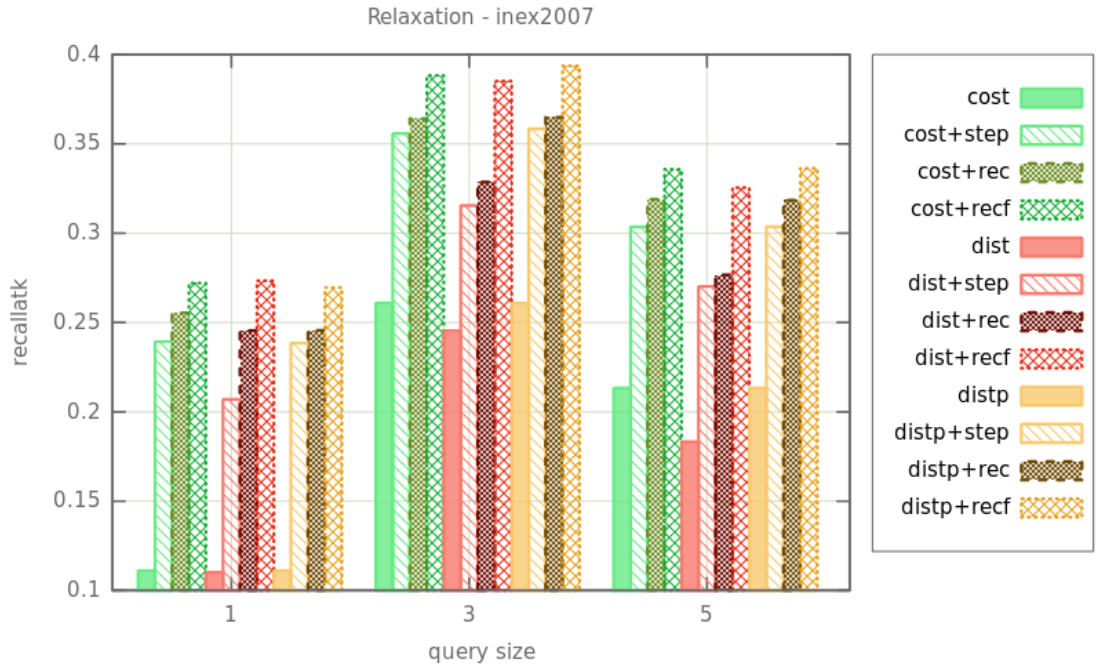


(b) Aspect relaxation variations Top-100: nDCG @ INEX2007

Figure 5.8.: Aspect relaxation variations: nDCG @ INEX2007

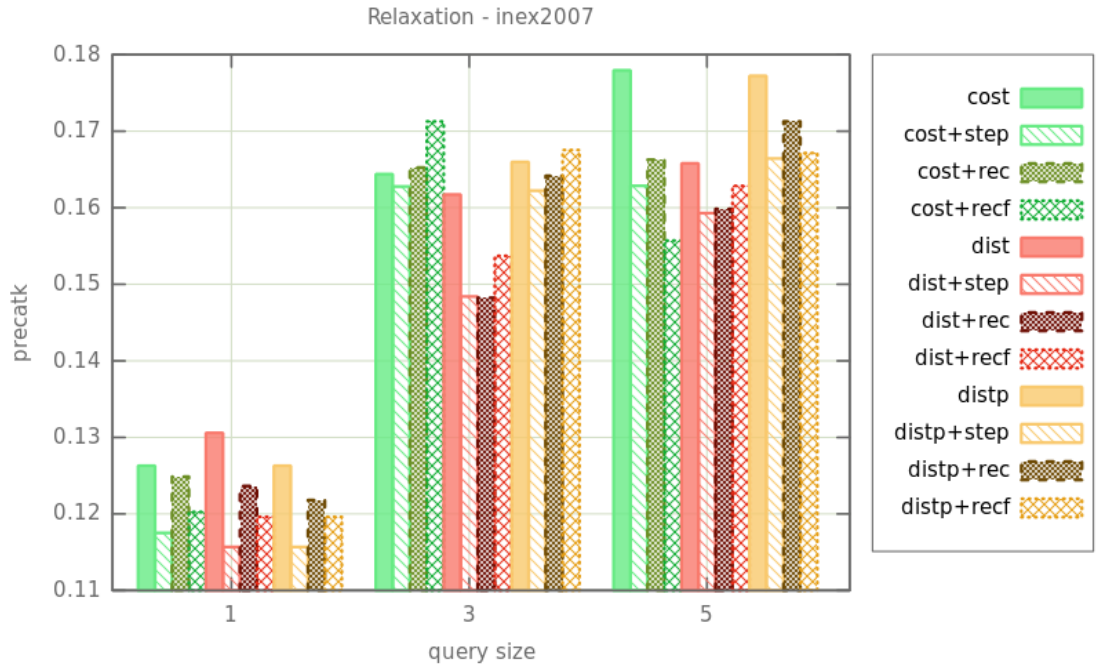


(a) Aspect relaxation variations Top-10: recall @ INEX2007

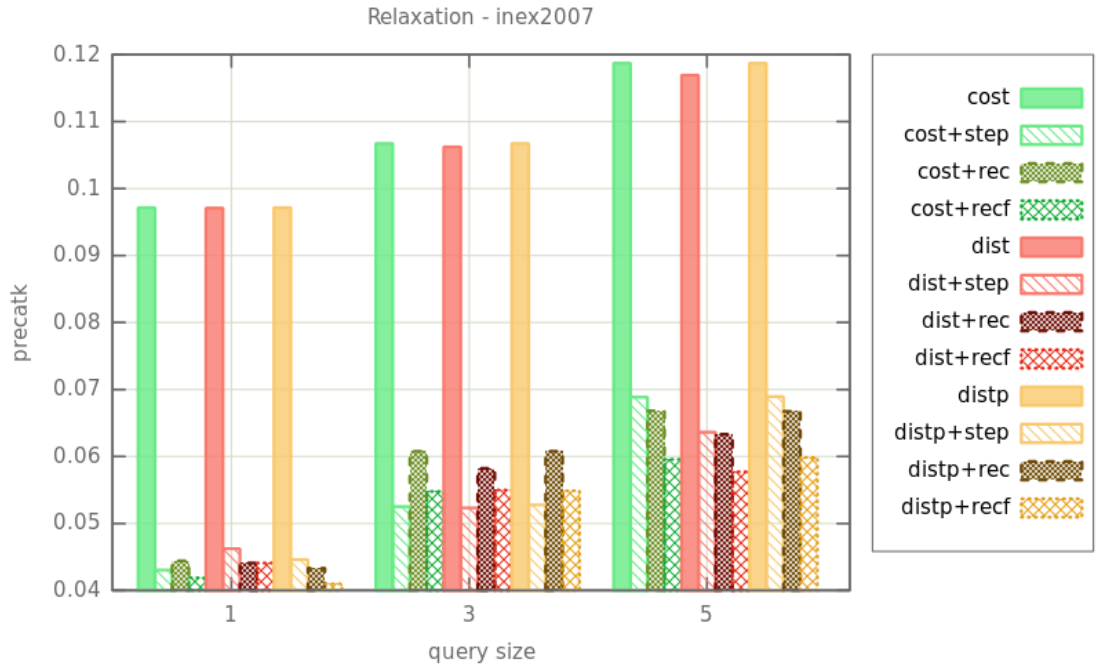


(b) Aspect relaxation variations Top-100: recall @ INEX2007

Figure 5.9.: Aspect relaxation variations: recall @ INEX2007



(a) Aspect relaxation variations Top-10: precision @ INEX2007



(b) Aspect relaxation variations Top-100: precision @ INEX2007

Figure 5.10.: Aspect relaxation variations: precision @ INEX2007

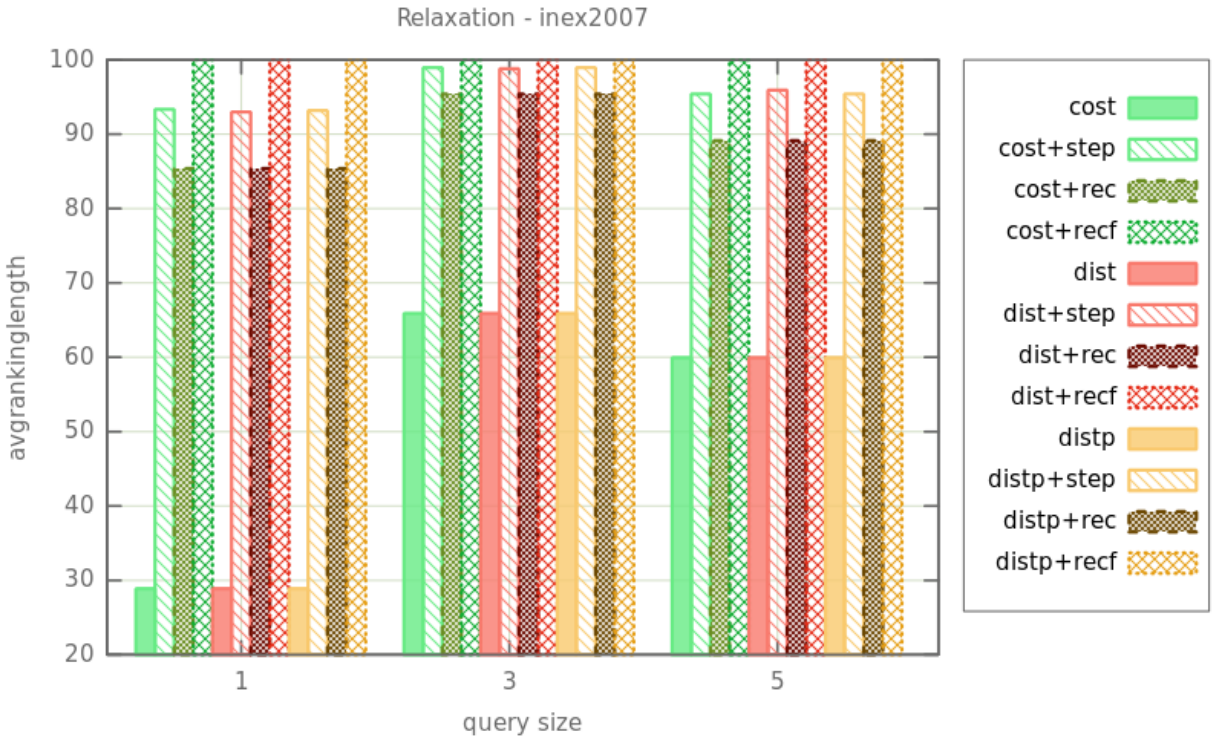


Figure 5.11.: Aspect relaxation variations Top-100: Average Ranking Length @ INEX2007

5.6.3. Comparison to Competitors

We now compare the different variants of our approach to two state-of-the-art competitors: (1) a random walk with restart at the query nodes (`rwalk:c`, `rwalk:tf`, `rwalk:n`) as a graph-based baseline, and (2) the hybrid approach suggested by Bron et al. in [BBdR13] (`bron-hybrid`), since this provides the best results combining a structural part based on the example entities and a textural part based on a short query description. The random walk with restart was executed in Apache Giraph with a random jump probability of 0.15. Note that in (1) we optionally apply a filter on resulting entities, either using the categories provided in the INEX dataset for the topic where possible (':c' versions) or using our own typical type identification approach(':tf'). For the Bron et al. approach we leave out `means` and `isCalled` facts and the 10 most generic types, i.e. all with more than 100000 instances, since their weight is so small that they don't influence the result, but increase runtime significantly. For the random walk we ignore all relations that include literal arguments.

5.6.3.1. Performance

Components In Section 5.4.3.1 we discussed our algorithm to compute maximal aspects. In order to identify the critical components for runtime let us consider the following setup. We randomly choose 40 single-entity queries and for each generate 20 entity suggestions. We analyze the time needed for A) retrieval of the entity sets of all basic aspects of $\mathcal{A}(Q)$, B) the computation of the set $I_{\mathcal{A}(Q)}$ of generated by the entities of $\mathcal{A}(Q)$, C) the computation of maximal aspects by applying the subsumption test on $I_{\mathcal{A}(Q)}$, and D) the ranking and selecting of entities. Table 5.3 provides the averaged results. Note that all experiments are done with a cold cache, i.e. after each query the system is completely restarted. The variant used no relaxation and the `distp` ranking. This test ran as a single thread on a machine with a six-core AMD Phenom X6 1090T processor (3.2 Ghz) and 8 GB main memory (of which at most 2GB could be allocated) with (local) network access to the database. Note that when measuring the run-times we left out the initialization phase setting up the system once for all queries, in which, for instance, caches are initialized, the connection to the database established and tested and the type hierarchy is loaded into main memory; this holds for this and the next test.

component	average time spent
entity set retrieval	22.8 sec
computing aspect candidates	1.5 sec
maximal aspect computation	0.002 sec
ranking and selection	0.25 sec

Table 5.3.: Runtime Contribution of components

As we can see, the most crucial aspect is the retrieval of data from the database dwarfing

all computations. Hence, we made use of caching for the entity sets of basic properties.

Real World To estimate the average performance of our approach in an application setting, we have selected 200 queries from the INEX 2007 dataset and processed them in random order to mimic a reasonably realistic workload. Caching is activated for this analysis, but cold at the very first query in each variant, to provide a realistic setting. As the computation took place on a shared infrastructure, we ran the test 3 times and averaged the runtime measures over all 3 runs. In Table 5.4 you can find the average time it takes to process a single query for different settings of our model: `spop`, `cost`, `dist` and `distp` are the different ranking approaches with all the default settings discussed before in Section 5.6.2 without relaxation. The `_1step` variants relax using the *one-step* heuristic, the `_recf` variants use all types and full recursive relaxation, while the `_allt` versions use all types, but no relaxation. This test was run as a single thread on a 8x Dual-Core AMD Opteron(tm) Processor 8222 that has 256 GB main memory of which at most 7 GB was allocated.

ranker setting	time per query		ranker setting	time per query
`spop`	2.77 sec		`spop_1step`	3.56 sec
`cost`	1.79 sec		`cost_1step`	2.34 sec
`distp`	1.77 sec		`distp_1step`	2.34 sec
`spop_allt`	3.11 sec		`spop_recf`	35.6 sec
`cost_allt`	2.94 sec		`cos_recf`	36.7 sec
`distp_allt`	3.1 sec		`distp_recf`	41.7 sec

Table 5.4.: Runtimes

The results show that as long as we are not using all the types together with a fully recursive relaxation, our approach provides results relatively fast within a few seconds, while the Bron et al. approach takes about 31.6 seconds per query. While we did not measure the runtime of our random walk implementation within the same setup, we can safely say that on average the time to process a query is above 1 minute (in fact we observed times averaging around 15 minutes per query). However, note that both our implementations for `bron-hybrid` and for the random walk are simple and not performance-optimized, such that better performance results may certainly be possible. Still, there is also room for improvement with our own approach, s.t. it seems likely that with some improvement and the right underlying architecture, that, for instance, loads the ontology in its entirety into main memory, real-time processing for an interactive setting is possible, at least with our faster approaches. There is also some room for performance improvement at the implementation level, as we, for instance, do several naive data type conversions (e.g. from List to Set etc.) that could be avoided. On the other hand, the full relaxation is mainly aimed at use-cases that need larger result sets, like an offline set-completion, where its slower runtime should not be a problem.

5.6.3.2. Result Quality

For the result quality evaluation we use the inex2008 and semsearch2011 datasets, since we "trained" our model in some sense on the inex2007 dataset, i.e. picking the configuration variants that proved most promising there as discussed in the previous section 5.6.2. However, for completeness, we also provide some evaluation data comparing our approaches against the `bronetal-hybrid` approach on the inex2007 dataset.

Online Scenario First let us investigate an interactive set-completion scenario.

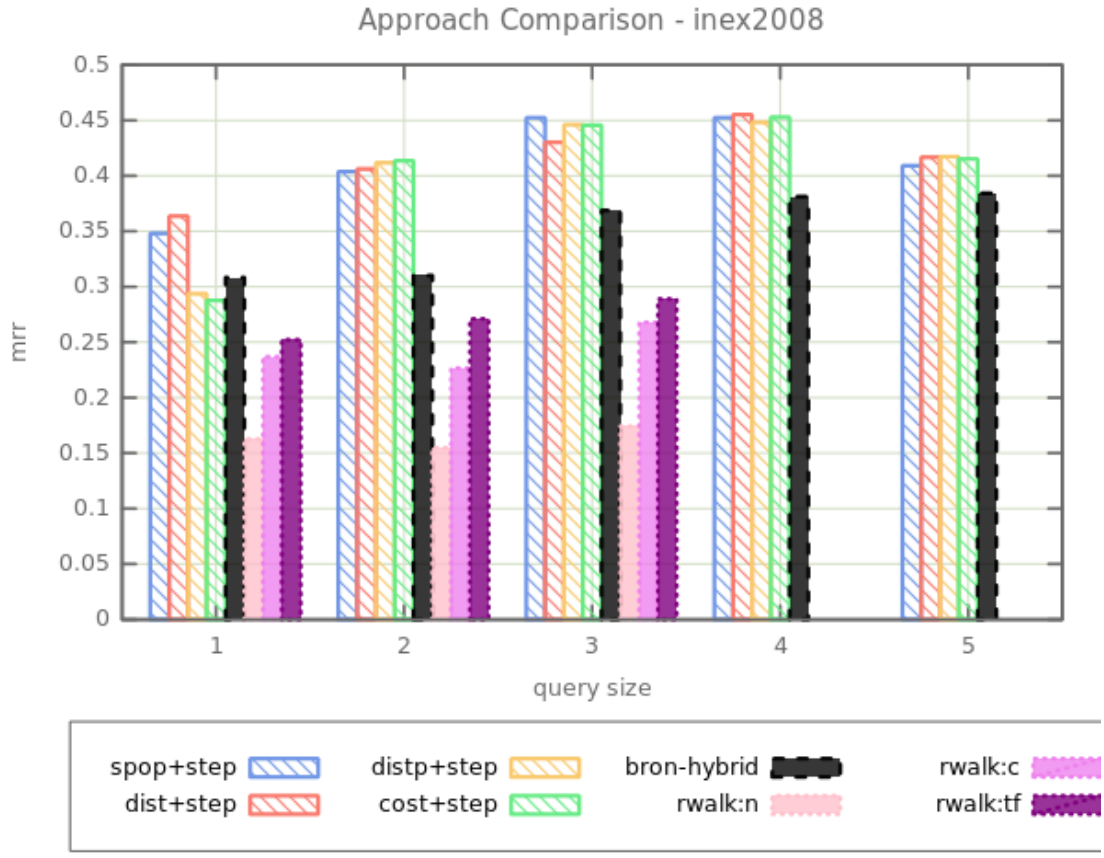


Figure 5.12.: Approach Comparison Top-10: mrr @ INEX2008

The focus in such a setting lies on a relative quick retrieval of a small set of entity candidates, because 1) users typically are not willing to look through more than 10-20 results [JSS00][6] and 2) in an online set-completion scenario, the first entity selected by a user can be used to refine the query. Therefore we consider our `1step` relax approaches (with specific types) and rankings of size 10.

[6] Actually the average user looks according to [JSS00] on average at 2.35 result pages per query.

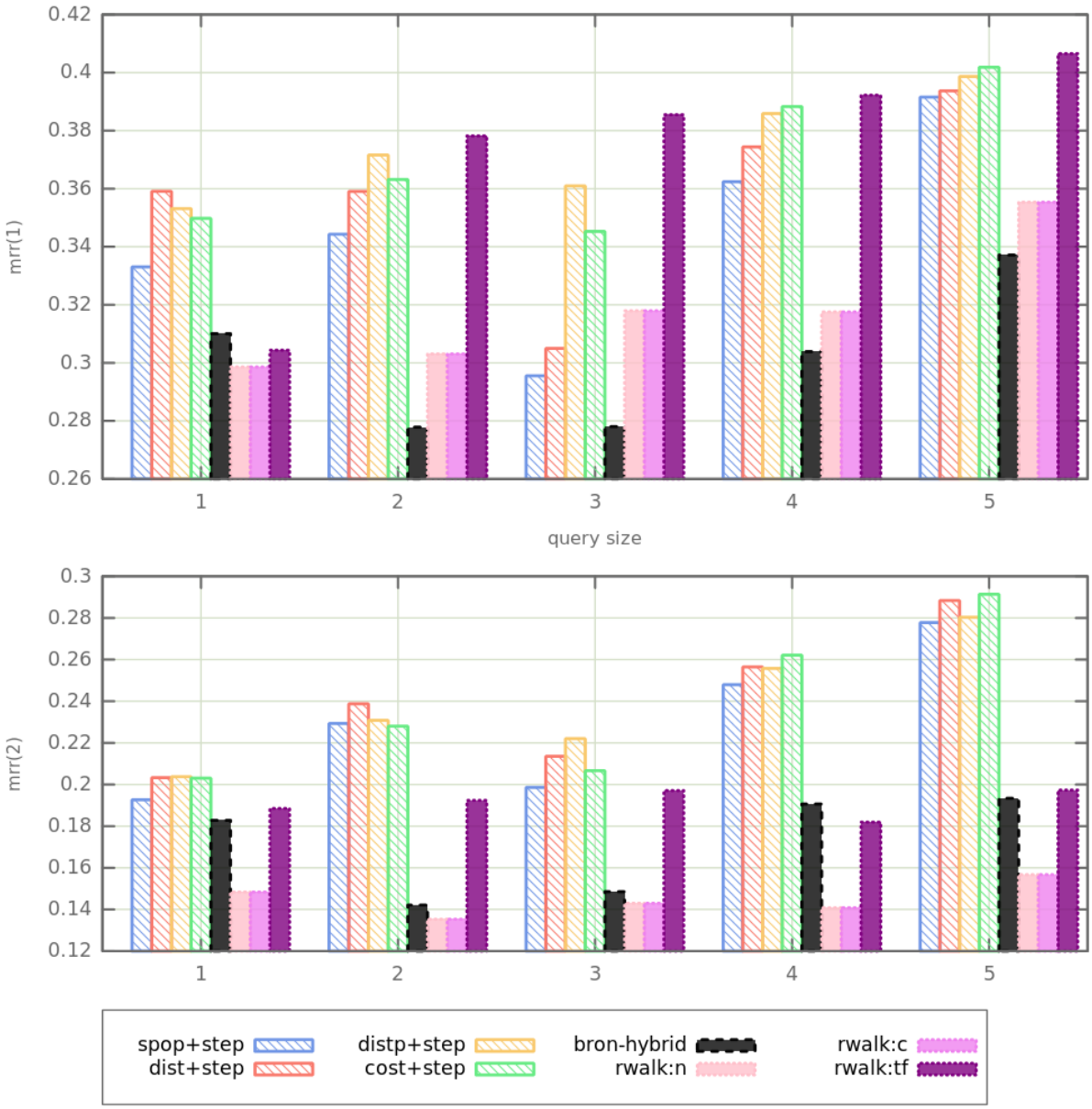


Figure 5.13.: Approach Comparison Relaxing Versions Top-10: mrr @ SEMSEARCH2011

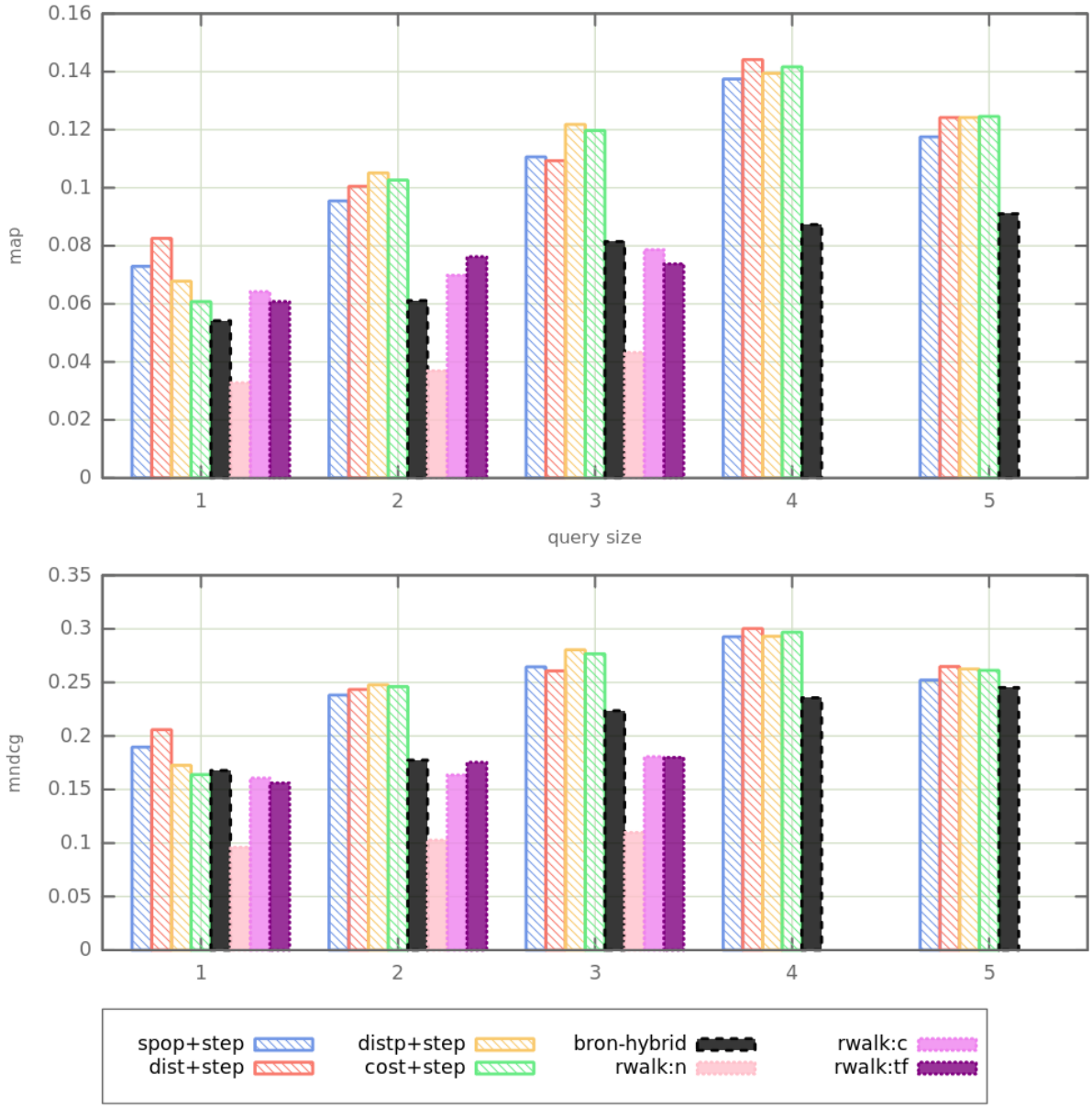


Figure 5.14.: Approach Comparison Top-10: MAP,nDCG @ INEX2008

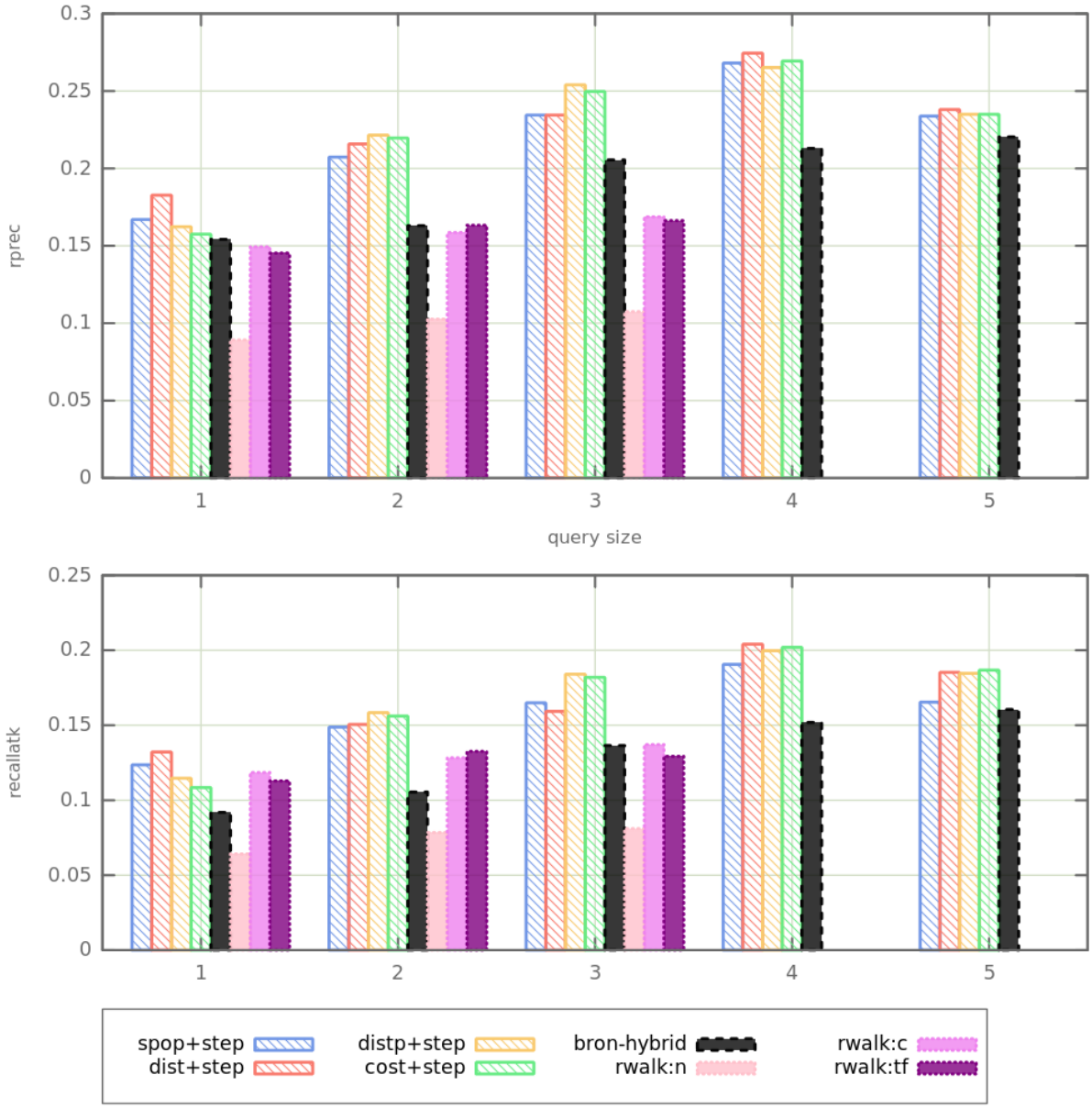


Figure 5.15.: Approach Comparison Top-10: r-prec, recall @ INEX2008

In an iterative setting with query refinement, it is important to quickly find the first relevant entity, as then the query can be refined and a new set of entities retrieved that is hopefully more relevant overall than the earlier results. Figure 5.12 shows the mrr values as an indicator for how quickly the first relevant entity can be found for the inex2008 dataset. On this dataset all our approaches provide the first valuable result as least as early or earlier than the other approaches with the exception of single entity queries where two of our approaches fall behind.

Figure 5.13 provides the mrr values on the semsearch2011 dataset. However, we distinguish between following our default setting to use relevance grade 1 as a binary threshold (mrr (1)), and using relevance grade 2 as a stricter binary threshold (mrr (2)). When considering the standard threshold results, the random walk with filtering support is better in providing the first relevant results. On closer inspection however, we find that several of the results judged with a relevance score of 1 are actually incorrect entities that are very close to the correct ones. For instance, we have the entities `Languages of Angola` and `Languages of Brazil` as relevant entities for the set of all nations where Portugese is an official language (topic 29); for topic 38 (states that border Oklahoma) we have several interstates of Oklahoma, crossing into neighbouring states. In a rough similarity setting these entities are surely related to the query, but they are not actually correct answers in a strict sense. Due to our strict type adherence it is clear that this can be a drawback for our approach. The entities judged with a relevance score of 2 seem on average to be more correctly fitting to the topic. Thus, when we only consider the entities relevant that are judged with a score of 2, our approaces also perform better with regard to the mrr values compared to the random walk, see mrr(2) in Figure 5.13. It is natural that the random walk is more robust when it comes to such mixed-in entities that are related to the query, but not fully matching it: the random walk is mainly based on relatedness of entities, while our approach is more strict in requiring shared properties and shared types in particular.

Figure 5.14 and Figure 5.16 show the MAP and nDCG values for our approaches as well as the competitors on the inex2008 and semsearch2011 datasets, respectively, while Figure 5.15 and Figure 5.17 show the r-prec and recall values for both datasets. Note that the random walk computation was so slow that we left it out for the higher query sizes in the inex2008 dataset. For completeness, we also show the MAP and nDCG (Figure 5.18) as well as the r-prec and recall (Figure 5.19) results on the inex2007 dataset, but only in comparison to `bron-hybrid`.

As these results show, all our approaches behave similarly well, with `spop` often lagging slightly behind the other variants. In most cases our variants are on par or outperform all competitors, only in one case `spop` performs slightly worse than the random walk with type filter (query size 3 on the semsearch2011 dataset). The random walk benefits strongly from entity filtering (':c',':tf' versions vs ':n' version), which, based on the wide spread of entities the random walk covers as candidates, seems logical. The results indicate a general quality dampening for our approaches at query size 5 in the inex datasets, this may be due to over-fitting to very specific shared aspects.

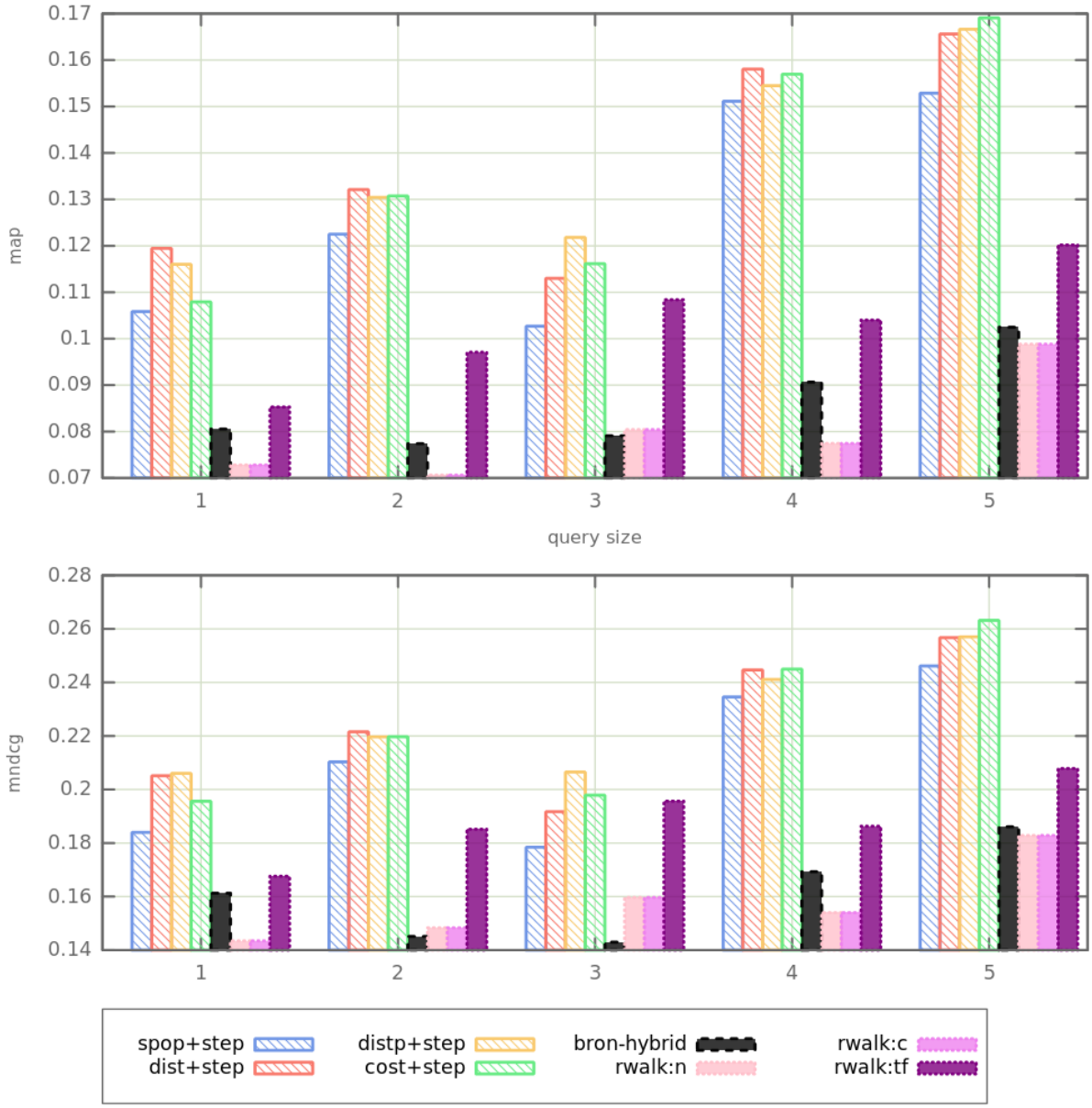


Figure 5.16.: Approach Comparison Top-10: MAP, nDCG @ SEMSEARCH2011

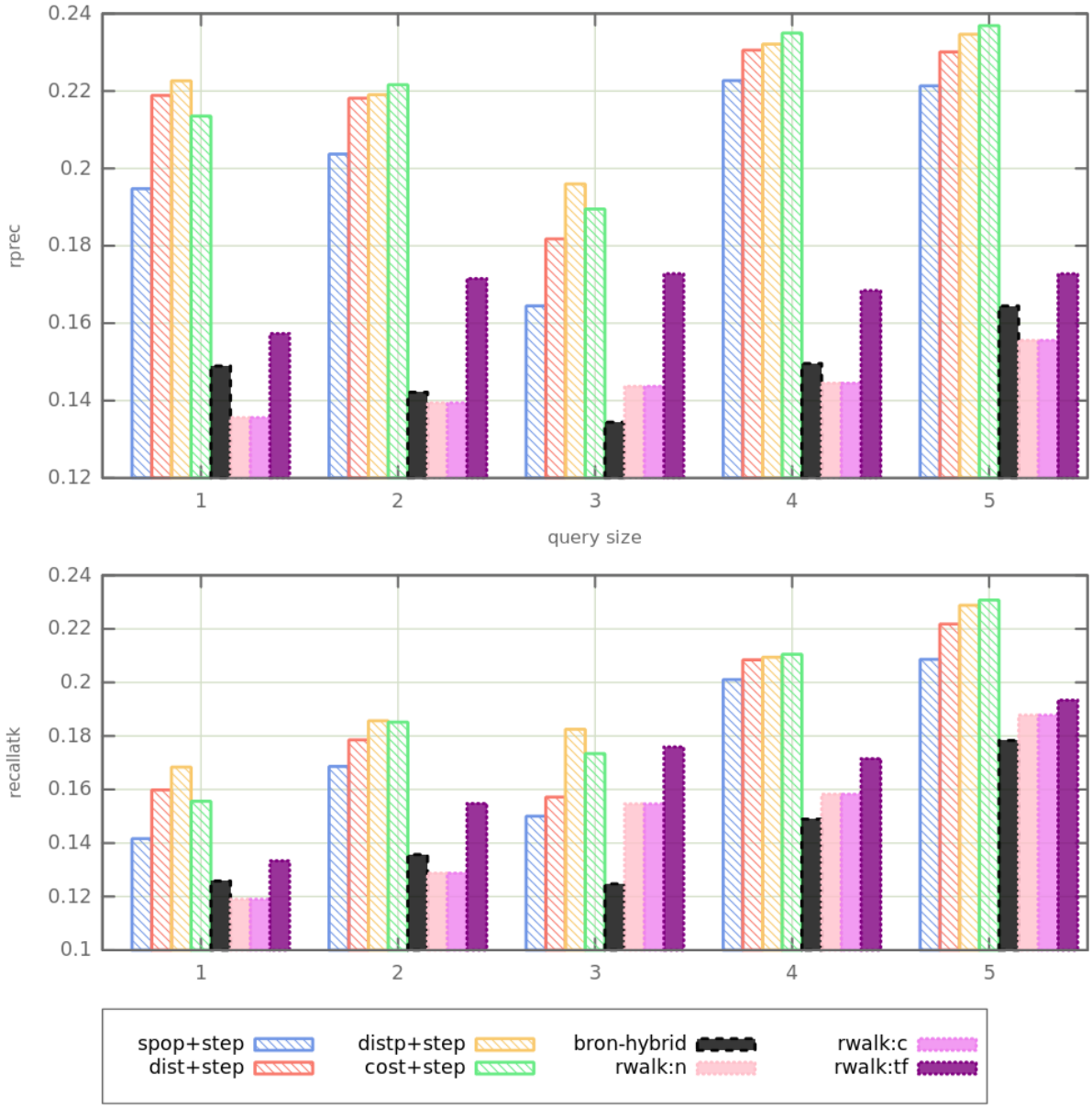


Figure 5.17.: Approach Comparison Top-10: r-prec, recall @ SEMSEARCH2011

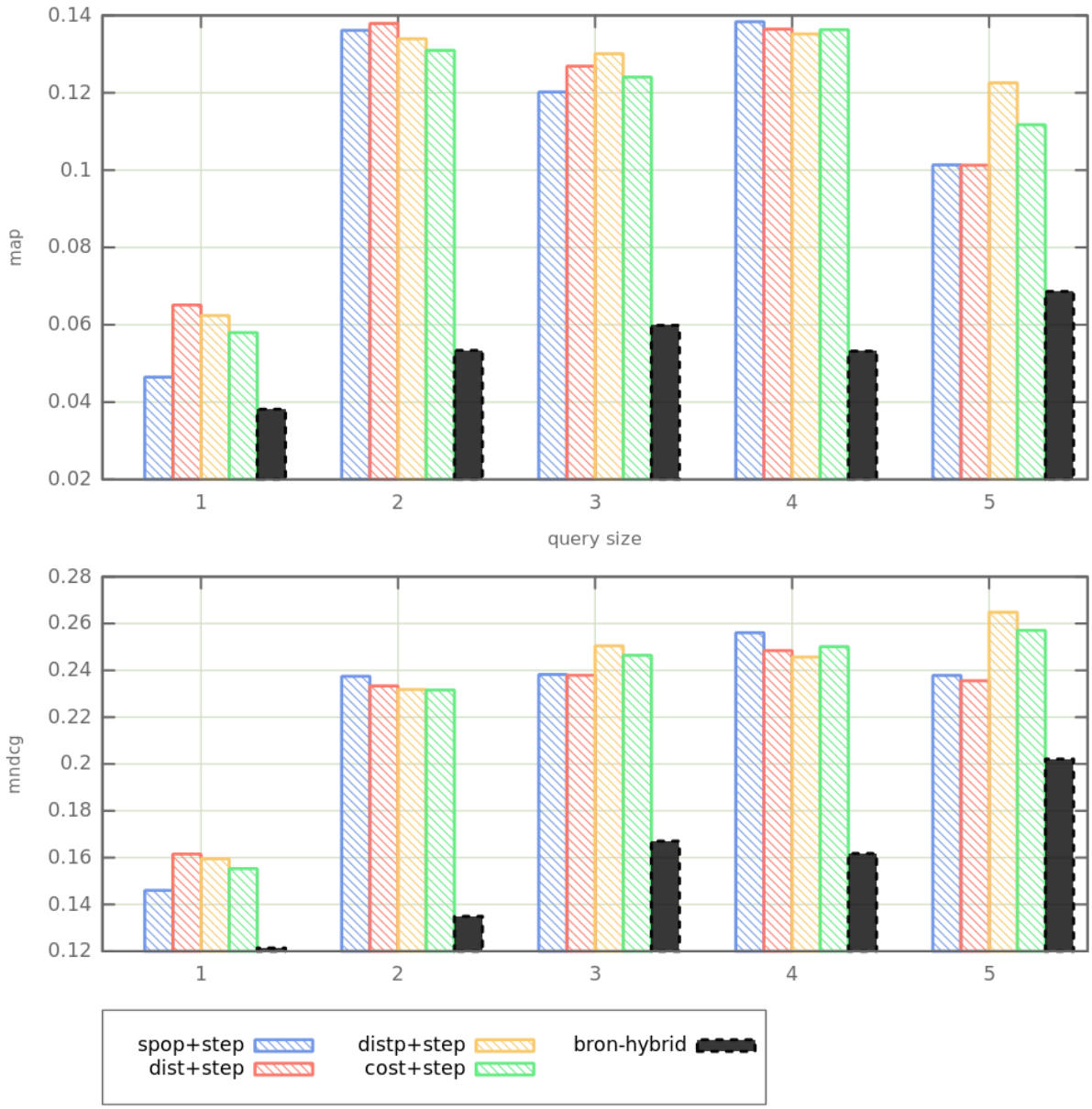


Figure 5.18.: Approach Comparison Top-10: MAP, nDCG @ INEX2007

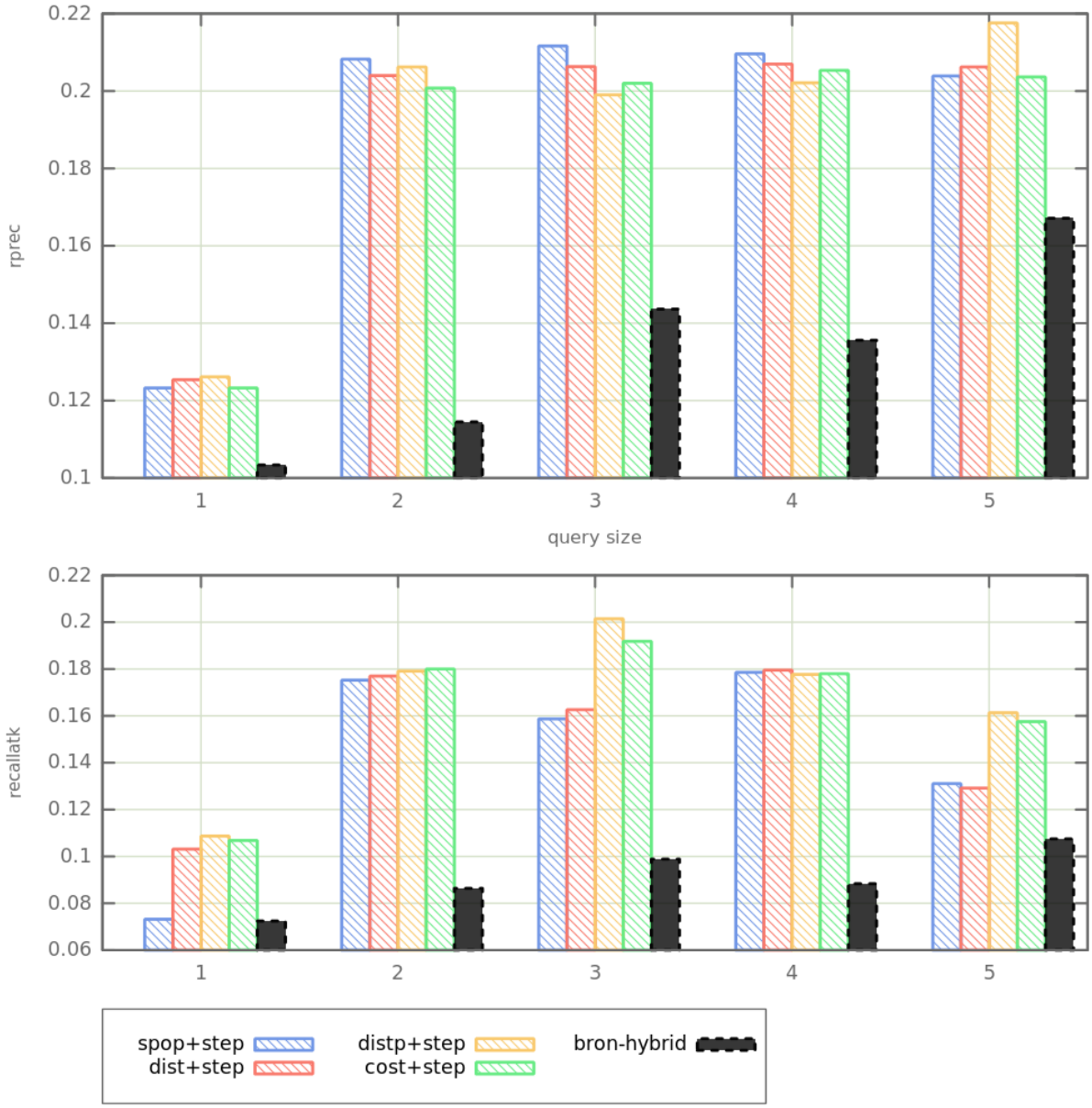


Figure 5.19.: Approach Comparison Top-10: r-prec, recall @ INEX2007

Offline Scenario Now, let us consider an offline set-completion setting by investigating rankings of size 100. For this setting we use our fully recursive variants (`recf` versions). Figure 5.20 and Figure 5.23 show the MAP and nDCG results, while Figure 5.21 and Figure 5.24 provide recall and r-prec on both datasets.

Here we can see a similar picture, our stronger approaches (`cost`, `dist`, `distp`) outperform all competitors accross the board with one exception (nDCG on queries of size 3 on the semsearch2011 dataset). The weaker `spop` variant falls behind more often, e.g. for queries of size 1 on the inex2008 dataset. Note that the relaxing versions reduce the quality dampening for size 5 queries of the inex2008 dataset that we observed for the `1step` versions to an insignificant level. This supports our theory that the dampening is due to over-fitting aspects, which the relaxation would break up. In Figure 5.26 the MAP and nDCG values are also provided for inex2007. While we can only provide a comparison against the `bron_hybrid` ranking, the findings are very similar to the ones on the other two datasets.

While mrr values would be less important in such a scenario, Figure 5.22 provides the mrr values for the inex2008 dataset. For the semsearch2011 dataset Figure 5.25 provides the mrr values for using both binary thresholds, i.e. grading value of 1 as threshold (mrr (1)) and grading value of 2 as threshold (mrr (2)). As the relaxing approaches have the potential to provide less diverse results early on compared to the non-relaxing variants (by first exploring a maximal aspect by relaxation), they are weaker with respect to mrr values. However, mrr is also less important in an offline setting, where it is more important to find the entities of interest in a single effort rather than one early. Figure 5.22 shows that with the exception of size 1 queries the stronger relaxing aspect-based approaches outperform the other methods on the inex datasets, but in several cases the gap becomes much smaller than with the non-relaxing versions. For the semsearch2011 dataset the same effect as with the shorter rankings can be seen; when looking at less strict setup (mrr (1)) the random walk with filtering support provides some relevant entities often the fastest, but when considering the strict setup, the aspect based approaches again provide the first good result slightly earlier most of the time.

While our results for the approach by Bron et al indicate a slightly lower quality (i.e. lower map values) than what was reported in [BBdR13], this is not a contradiction. First, our queries differ as we generated them independently. Second, our underlying knowledge graph differs, as they use DBpedia [ABK+07], and most importantly by mapping to YAGO and ignoring entities that could not be mapped the set of relevant entities is smaller, thus, if the distribution of relevant results in rankings is similar, this typically leads to lower measurements for map.

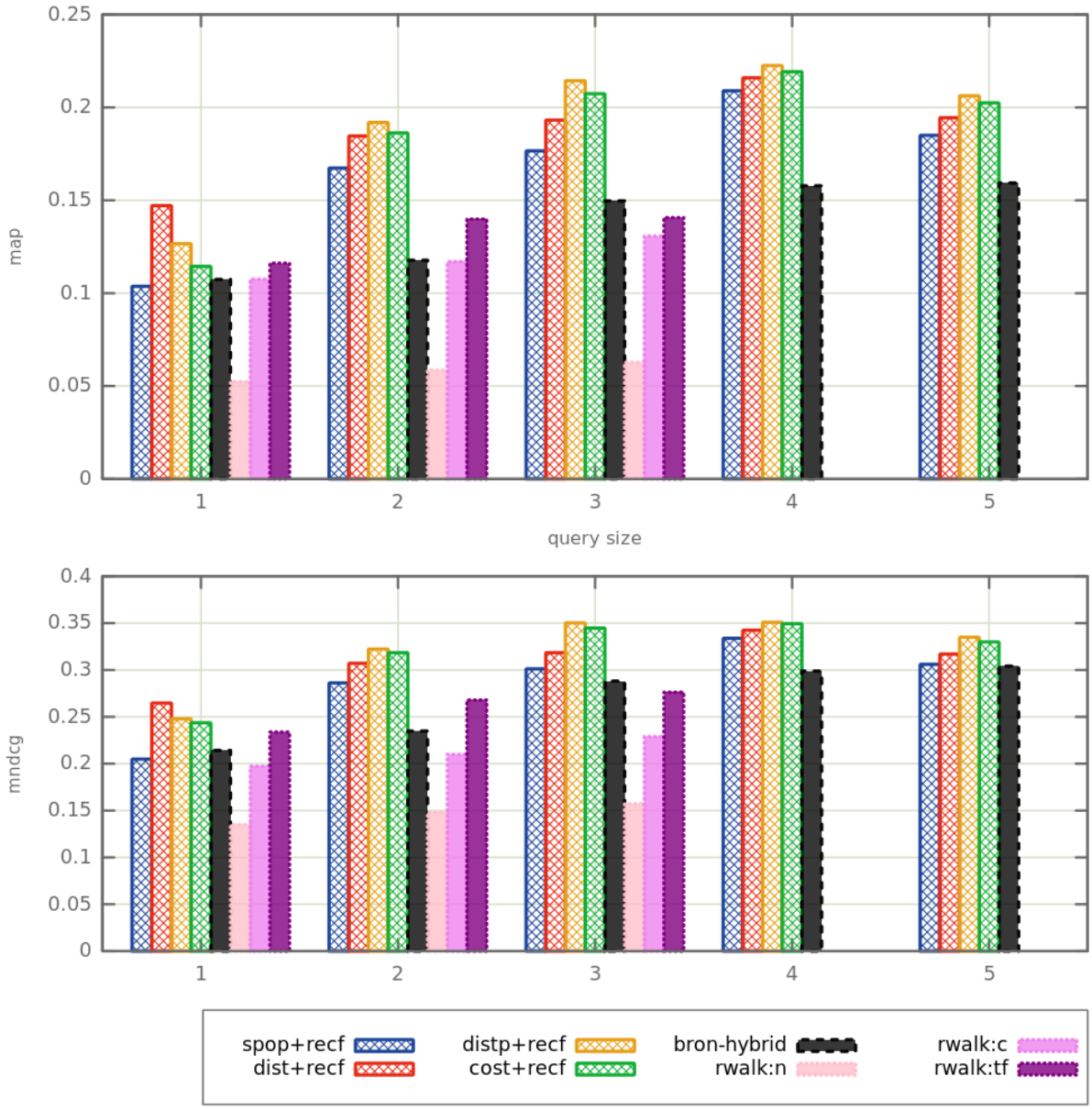


Figure 5.20.: Approach Comparison Relaxing Versions Top-100: MAP, nDCG @ INEX2008

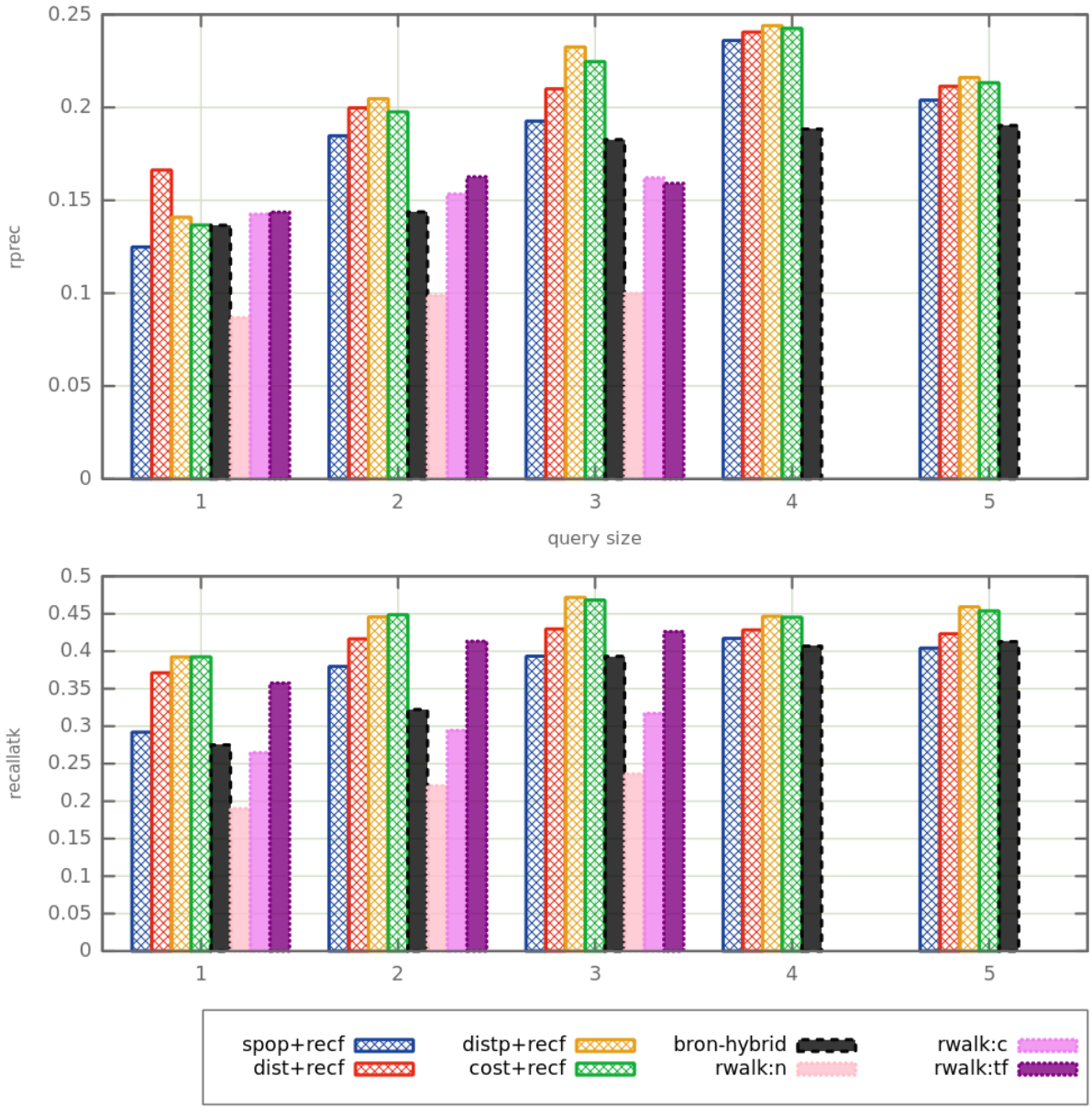


Figure 5.21.: Approach Comparison Relaxing Versions Top-100: r-prec, recall @ INEX2008

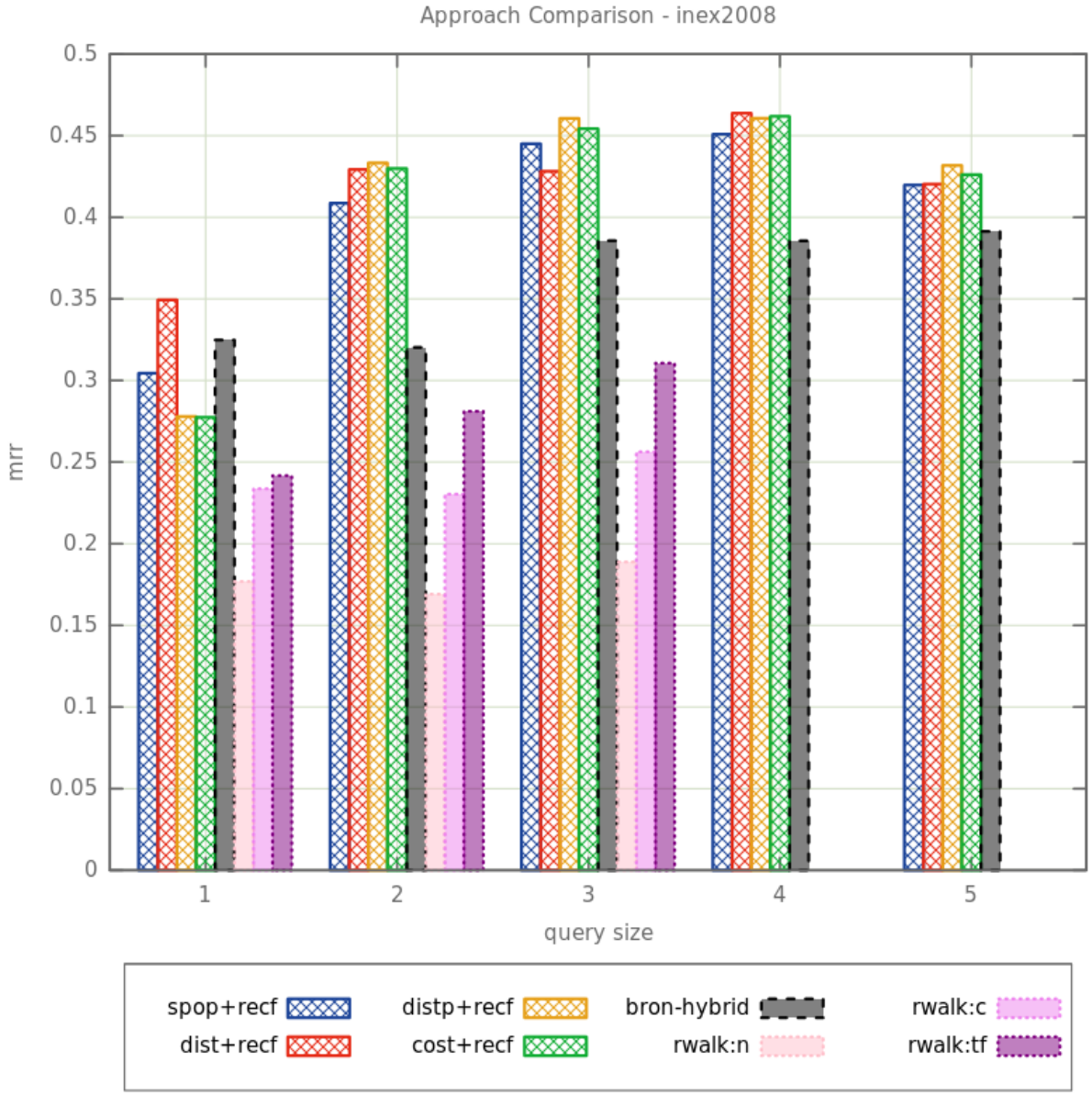


Figure 5.22.: Approach Comparison Relaxing Versions Top-100: mrr @ INEX2008

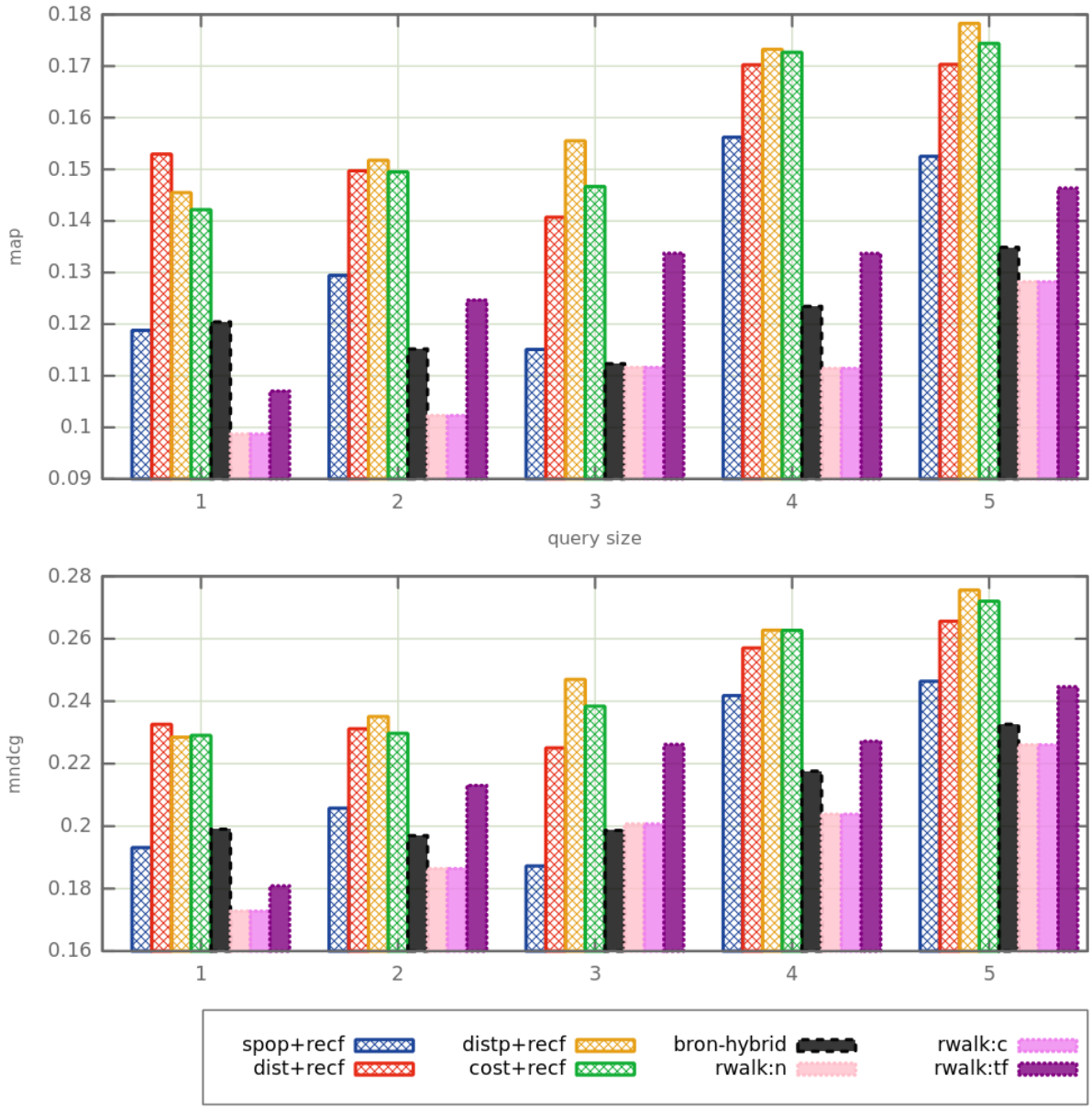


Figure 5.23.: Approach Comparison Relaxing Versions Top-100: MAP, nDCG @ SEMSEARCH2011

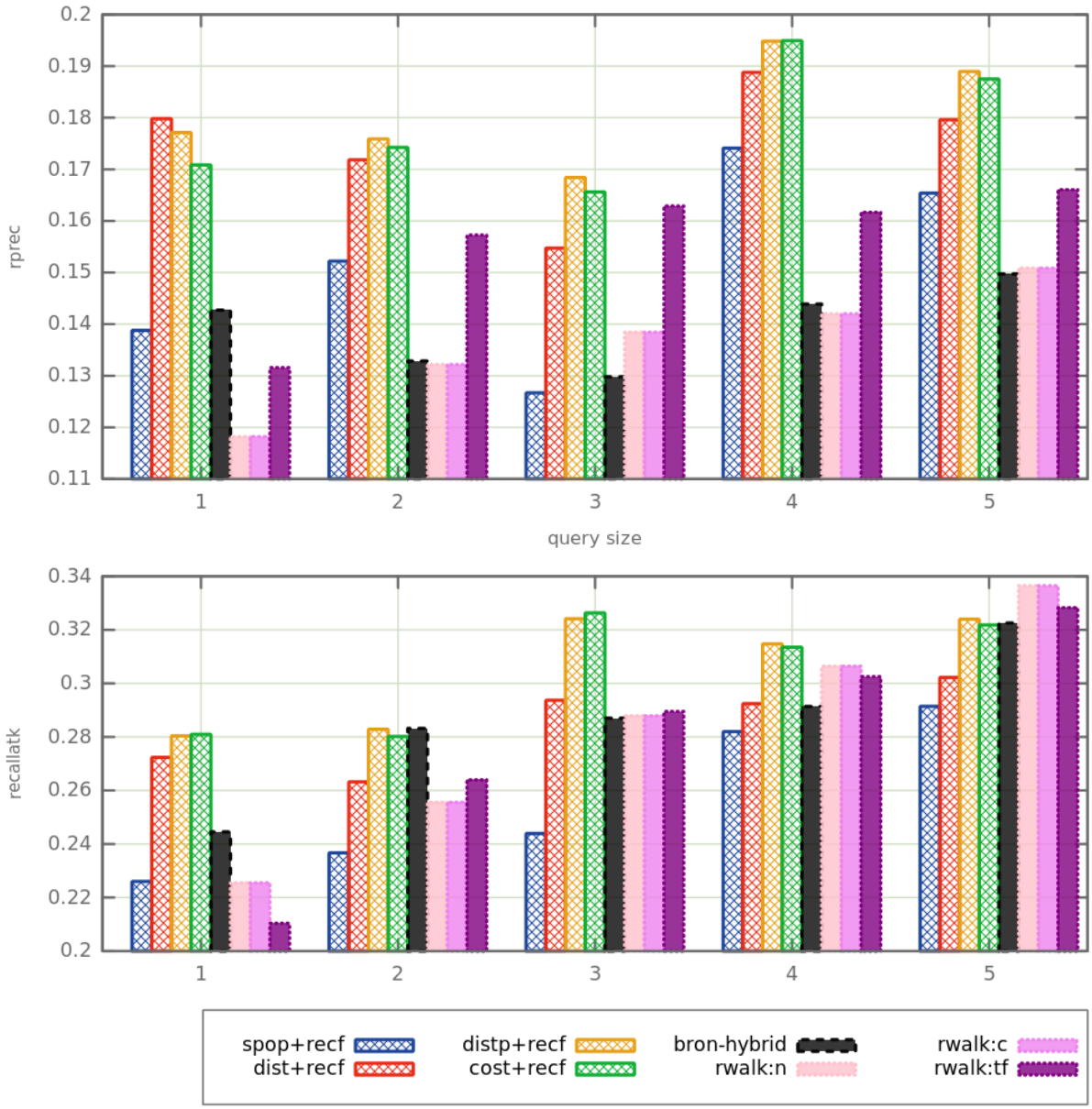


Figure 5.24.: Approach Comparison Relaxing Versions Top-100: r-prec, recall @ SEMSEARCH2011

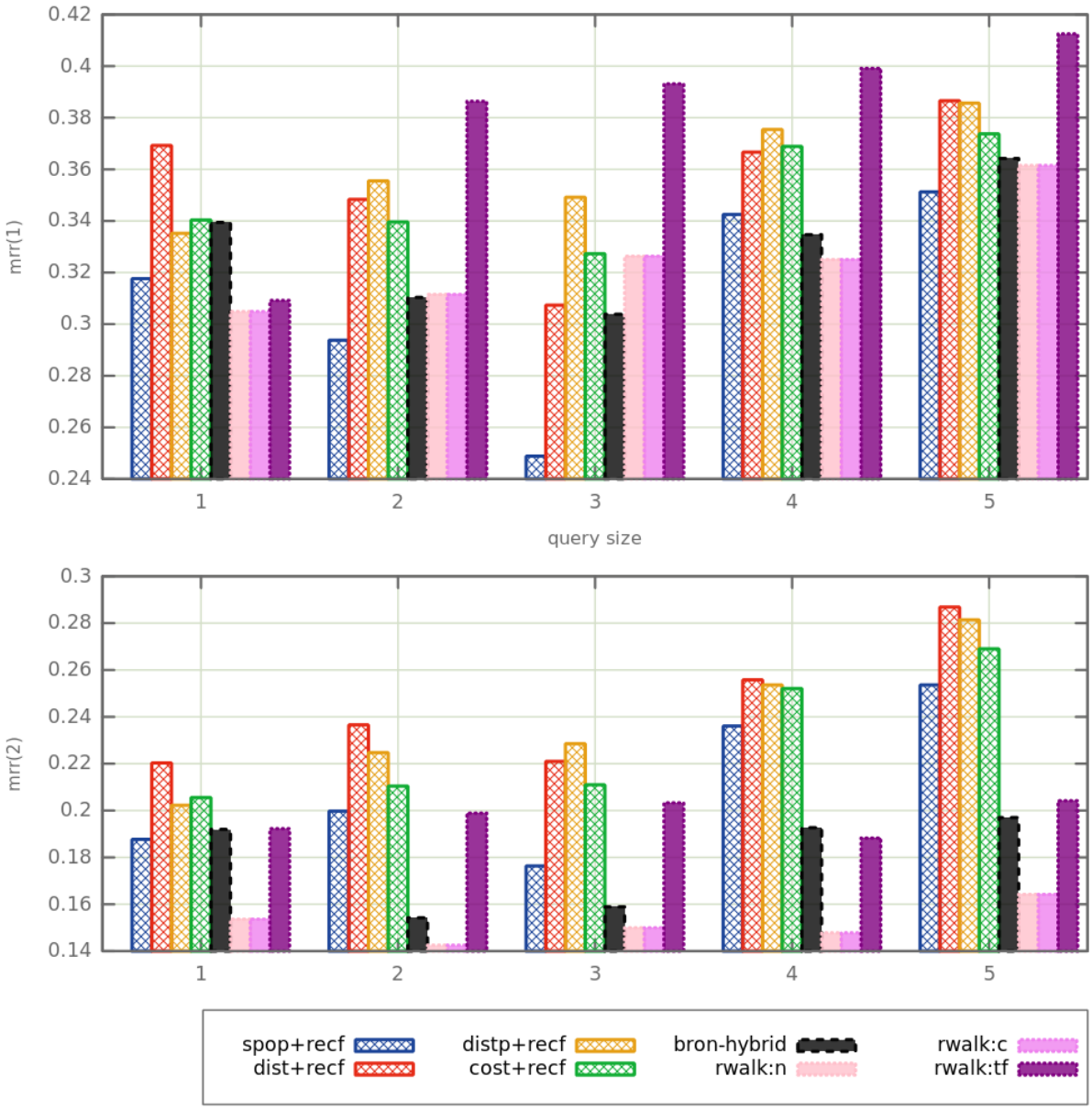


Figure 5.25.: Approach Comparison Relaxing Versions Top-100: mrr @ SEMSEARCH2011

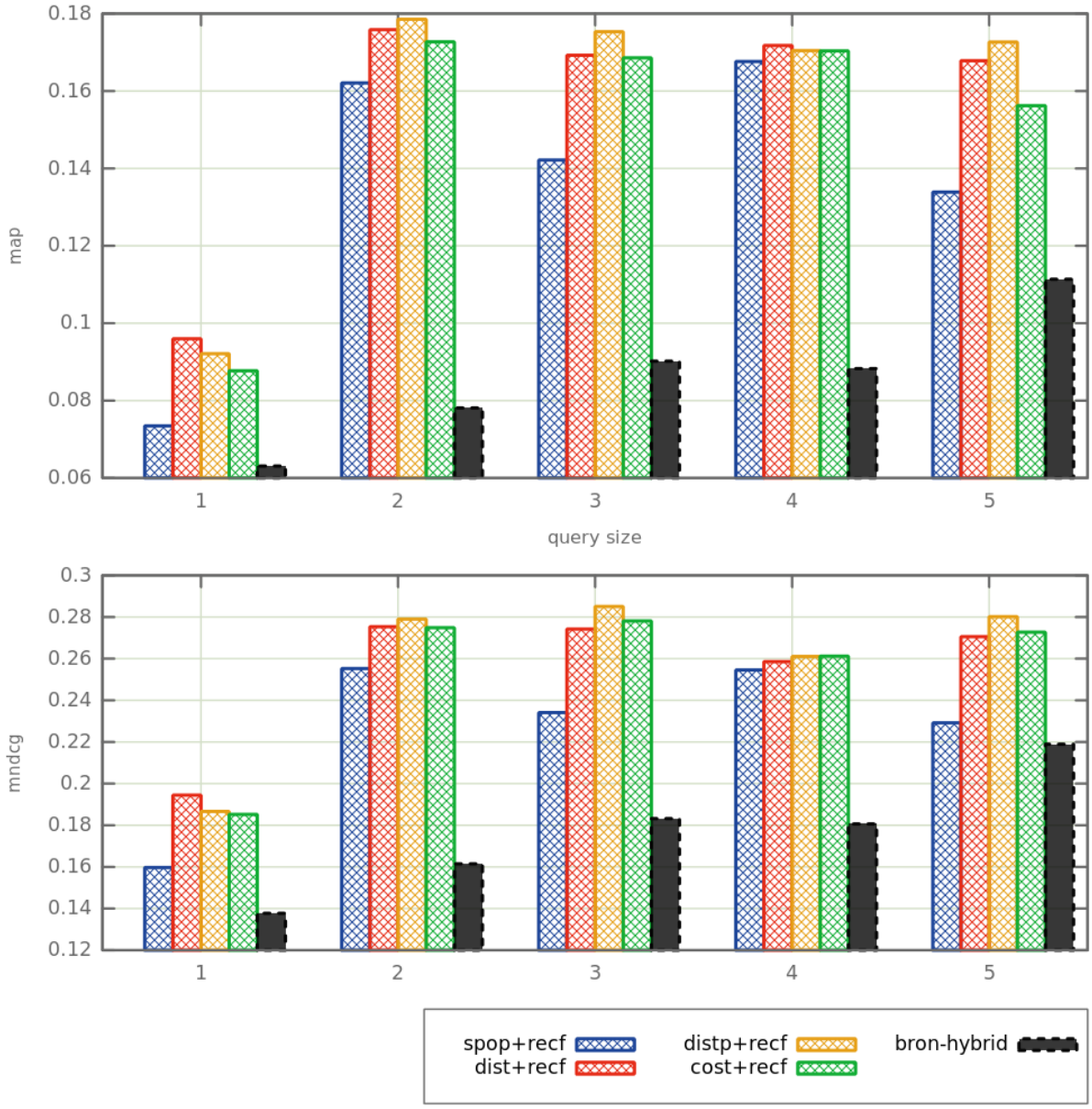


Figure 5.26.: Approach Comparison Relaxing Versions Top-100: MAP, nDCG @ INEX2007

5.7. Summary and Future Work

In this chapter we discussed our aspect aware aspect based entity similarity model in detail and extended it by relaxation of the aspects to expand the search space. We evaluated various system settings in the context of set completion tasks. While our evaluation shows that it can outperform state of the art models in several cases, we also showed that the relaxation improves result quality in such scenarios and that finding the right constraint types can be an important part of the problem. Our approach is by design less robust to malformed input, e.g. mistakenly chosen entities in the query set of entities, than traditional approaches like random walks. This problem could be reduced to some degree to the cost of increased complexity by introducing a probabilistic model when considering more than one entity, such that all basic aspects of all entities are considered but based on the likelihood that the user had them in mind, e.g. by analyzing how many of the entities share each basic aspect.

There remain several other interesting directions for future work. While our ranking methods achieve good results there is room for improvement, e.g. by identifying not only globally popular entities but predicting representative entities for a chosen aspect. Similarly, aspect selection could be guided by identifying aspects that are more commonly than others associated with the query entities. There are also additional use-cases, like interactive navigation, that might require amendments and offer additional opportunities. Finally, it would be interesting to evaluate how well our approach adapts to other knowledge graphs of higher or lower density.

6. Collaborative Knowledge Networks - The Semantic Web of the Future?

6.1. Motivation

With this work we provide methods to integrate user feedback into an extraction process, verify extracted information by tracing it to its document sources and to navigate an ontology. These methods aim at allowing collaboration in the extraction process amongst several human users and with an extraction system. However, if we think this one step further, there are not only different humans that might want to collaborate on separate knowledge projects, but collaboration among different such projects to combine their results could provide a much larger and more complete knowledge base. At the moment, there exist several academic common knowledge ontology projects based on information extraction [SKW07, HSBW13, ABK+07, CBK+10, EFC+11] not to mention commercial ontology projects, which partially are also accessible publicly [gkg, BEP+08, ocy]. While there are some efforts to integrate other projects, e.g. YAGO includes Wordnet [Mil95] and both YAGO [HSBW13] and DBpedia [ABK+07] provide `sameAs` links to match entities with those of the other ontology, this is a tedious task and users have to find out themselves in how far ontologies are compatible. In particular, to make use of `sameAs` links they need to use tools that support such logical statements and thus, for instance, include entities or statements linked in such a way in query answers. Speaking of tools to query and reason on ontological data, each such tool could also be considered as an *knowledge agent* working on ontological data, potentially generating new information, e.g. by deduction. Similarly extraction systems can be seen as knowledge agents providing (or rather translating) information. The collaboration of such different agents, however, is a complicated process from a user's point of view. It is hard to overlook which knowledge agents work well together, which ontologies, extraction systems, reasoning tools or the combination of which of those can actually provide the information a user needs. In addition, any effort of a normal user in correcting or adapting an ontology she uses by downloading it first and then accessing it with any fitting tool is typically only useful for herself and rarely finds the way back to the official ontology release version. To allow an easier co-operation of various knowledge management and generation *agents* that also propagate feedback from users or other automatic agents (e.g., deducted feedback) we suggest *Colledge* (short for *Collaborative Knowledge*), a peer-to-peer based network that implements a simple communication protocol to form a

collaborative knowledge network, as a basic architecture for a more interactive Semantic Web.

By modeling knowledge seekers and providers as cooperative agents acting within a social trust network our vision resembles a college, where students gain knowledge by accessing information from their teachers and information media such as books, exchange what they have learned enriched with their own modifications amongst themselves, which increases the trust they have in their knowledge, and finally may also feed back corrections and extensions to their teachers and the broader public.

In principle, our network consists of nodes that implement one or several components of today's semantic web, e.g. information caching, extraction or reasoning. Each node implements an extended query interface that allows for personalization, constraints etc. and returns a result along with meta-information such as provenance. Additionally, each node accepts incoming knowledge and may use this to optimize its own data or its processes to obtain knowledge, e.g. tuning an extraction engine.

From a user's perspective knowledge shall be distributed over this network similar to files over a file-sharing network, thus knowledge discovery is integrated into the network architecture, i.e. node discovery, and the query processing over the network. In order to automatize the selection of trusted quality information sources, a trust network is suggested and leveraged to enable a global confidence measure integrating trust into information sources and confidence of sources in their own data.

The remainder of this chapter is structured as follows. While Section 6.3 gives more details on the current state of the art based on an example application, Section 6.4 outlines the envisioned knowledge network, and Section 6.5 highlights different aspects of the proposed architecture and problems to overcome. First however, we discuss some related work in Section 6.2.

6.2. Related Work

With more and more information becoming available on the Web, the question arises how to efficiently use this global source of knowledge in an automation friendly way.

The solution offered by the Semantic Web is to convert all information in machine-readable formats, such as RDF, and offer access over standardized interfaces, e.g. via a file or a SPARQL interface.

Although this may sound like the perfect solution, it only addresses one part of the problem, namely how to standardise information access. Other problems are addressed independently. Focusing on RDF and SPARQL, for instance, today's research already proposes strategies for efficient distributed query processing [SHH+11, Har11, LT11, QL08, LWB08], search [CQ09, DFJ+04, ODC+08, HHU+11], source selection [UHK+11, ACHZ09, HS12], reasoning [MSH09, MDT11], ontology alignment [SAS11], and semantic heterogeneity and query rewriting [GKLM11]. There are first approaches combining the steps of information extraction and query processing instead of considering them as strictly consecutive steps [JIDG09]. Furthermore, provenance provided by extraction systems can be used to retrieve further information [MEHS11]. Similarly, there is ongoing work into answering knowledge queries using crowd sourcing [FKK+11, SLB12] and the SQoUT project [JDG08, JIDG09, JIG08] as well as [EHFI12] explore the integration of information extraction into an SQL query execution database engine, therefore bringing information stored in a database together with an extraction agent. And finally, distributed search based on peer-to-peer networks has been studied for a while as well [MTW05, LC07].

In an effort to enable collaborative knowledge management [SMMC09] suggests a network of semantic wikis that basically work as a version control system merging different versions maintained independently. While this allows human collaboration and feedback integration it does not include other automatic agents such as reasoners and information extraction approaches. It also lacks any notion of trust handling. Information trust on the other hand is a well known problem in (social) networks [GKRT04, ZL05] and general web retrieval [WM11].

6.3. The State-of-the-Art

As an example, let us assume a startup company wants to create a website about local cinemas and the movies they show. Instead of modelling everything themselves, the company decides to use an RDF-based representation and tries to find appropriate existing ontologies.

Although the Semantic Web provides several search engines [CQ09, DFJ+04, ODC+08, HHU+11] that might *help locate relevant existing ontologies*, the process of finding *interesting sources*, *judging their quality*, choosing the most relevant and useful ontologies and *adapting* (e.g. alignment, consolidation, reasoning, correcting, etc.) them involves a high degree of human interaction. Additionally, a human needs to check the accuracy of data used or trust the publisher of the data. For large publishers of general-domain information, like mainstream movies, this might be relatively simple, but it needs an expert for less well known domains, like independent French movies, or highly specialised domains, like quantum physics.

Let us assume the company has found a set of ontologies that cover most of the required information. The first obstacle is that RDF data is provided in different ways on the Web, e.g. SPARQL endpoints, RDF dumps, dereferenceable URIs, annotated web pages, web services, etc. Some approaches offer a way to query the data on the Web without the need for a local component, e.g. SPARQL endpoints or SWSE [HHU+11], which crawls all the data and stores it into a repository that can be queried. When using such services, however, it is impossible to make corrections to the data or change it in any way, nor are guarantees for data freshness or correctness provided.

Assume, for instance, while the publishers of the original data might find it correct to place the "Lord of the Rings" movies into the horror genre, the company might disagree and change the genre accordingly. Even if the publishers accept bug fixes, they would not adapt their data according to the company's needs. Thus, for the company *personalizing* the ontology is a hard task if the data is not stored locally, e.g. when using remote SPARQL endpoints. If the data comes from different ontologies the company may need to merge the contained information exploiting `rdf:sameAs` links and applying techniques originating from *ontology alignment, entity consolidation, and reasoning*. Most of these aspects require significant effort. For instance, there are several reasoning frameworks available, which provide different benefits and have their own drawbacks. A user first needs to get a deep understanding in reasoning aspects in order to choose the system that fits her needs and then configure it accordingly.

Another issue is *data freshness*, e.g. the company may need to make sure that it has information on current movies as well as the schedule and number of free seats at local cinemas. If information is only updated in large data dumps comprising of a complete domain ontology, accessing up-to-date information is quite inefficient. Using a SPARQL endpoint view onto a regularly updated dataset may alleviate some of these practical problems, as only specific information can be retrieved, yet the update rate of the data behind the SPARQL interface is totally up to the data publisher. In particular, a user of the interface has no idea how fresh received information is. While movie information will

not change that often the company might want to be sure free seat information is really up-to-date. In a SPARQL live-view there is no built-in possibility to *negotiate the update strategy* with the publishing node. Even more critical, there is no *standardized way* to inform the user *how recent information* retrieved *is*.

However, the company might decide to employ their own *information extraction* system instead to read information directly from the cinemas' online booking systems. Additionally, the company might provide suggestions of similar older movies on their web page along with links to buy these old movies. In order to establish movie similarity links, the company might decide to use *crowd sourcing* tasks. In both cases, the company would need to implement the corresponding components and integrate them. It would be hard to share the implementation as it would be designed to fit into the company's framework. Similarly sharing (partial) results would mean additional effort to publish the data and make it known to the outside world.

Even if the company decides to return some of the additional value it created in fixing and extending the base ontologies to the community that offered these so generously for free, the only way is to communicate with the publishers and explicitly report modifications.

The whole process outlined here is a tedious task done by many users over and over again because there is no straightforward way to share the result with other users.

In summary, the main problems we identify is a) the lack of any standardized way to provide and integrate feedback, b) the lack of a standardized way to negioate data freshness with data sources, c) the lack of trust management and d) the lack of a standardized protocol for (online) collaboration of different types of knowledge agents.

6.4. Colledge Network Architecture

So far, in the Semantic Web there is basically no interaction or feedback between publishers and consumers. This is different in social networks where we have all become used to facebook-style "like" flags and the possibility to comment on any content. Furthermore, the recent success of cloud applications in the private sector has shown that complex processes can be made available to a broader user community if their complexity is hidden behind a common interface.

We envision a system that allows users (incl. developers) to ignore technical details and that incorporates the interactive nature of social networks for knowledge harvesting and exchange. Similar to file sharing networks, the system is based upon a self-organizing peer-to-peer network, where each node serves as information provider and seeker while nodes are expected to cooperate to answer particular information needs. Components of the Semantic Web as e.g. extraction systems, reasoners, query engines etc., are assumed to participate in the network by implementing standardized query and communication interfaces, thus, typically complex tasks can be provided over a unified interface and be partially automatised. Additionally, steps typically seen as consecutive are interleaved and may benefit from one another as well as different users accessing the system can benefit from results retrieved for others or from modifications suggested by other users. Basically interaction with the network consists of three steps:

1. asking a query,
2. retrieving the results,
3. and possibly providing feedback.

Each of these steps is broadcast throughout the network and each node on the way may learn from information passed to or through it. This basic layout and interaction model is outlined in Figure 6.1.

When a user issues a query, a single node might not be able to answer it with locally available information. Therefore, during query processing, the node might decide to apply distributed query processing techniques (cost model, indexing, statistics, query optimization, etc.) to optimize queries over multiple sources.

In contrast to existing systems, there are no restrictions on the way how (sub)queries are answered at each node, e.g., over a local triple store, using a wrapper for a relational database, involving further nodes, web services, applying a reasoning system, extracting information on-the-fly from the Web or a local corpus, as a crowd sourcing task, etc. Thus, ontological knowledge published by running a knowledge network node, similar to a SPARQL endpoint, is accessible in the same way as information acquired at query time, e.g. by a node employing information extraction or crowd sourcing methods.

Similar to social networks, the system considers and exchanges feedback on its content and its own components. A user, for instance, can give explicit or implicit feedback on the

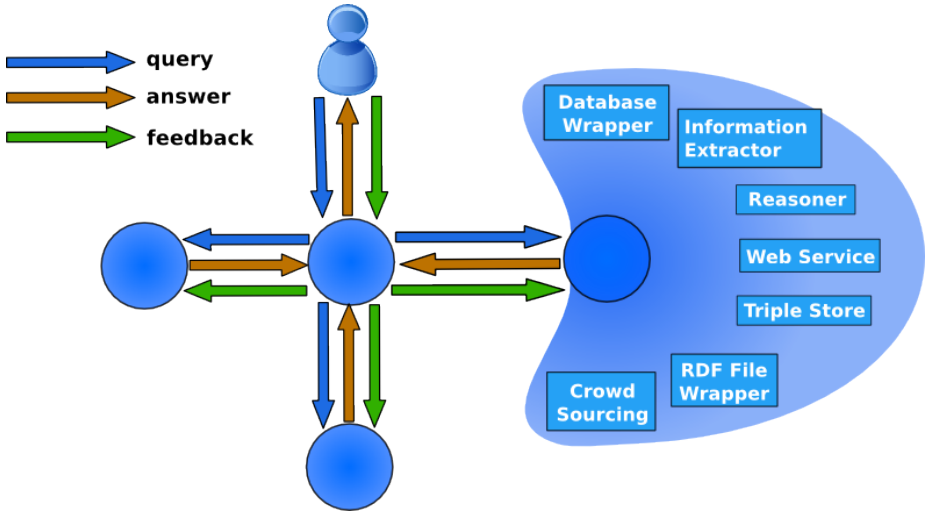


Figure 6.1.: Network Architecture Outline

Sources: User clipart symbol from OpenCliparts.org (public domain)

quality of a query result. This feedback is not only useful to improve the node's locally stored data, but can be shared with other nodes to improve their knowledge bases. Additionally nodes, can employ user feedback to improve their knowledge acquisition methods, e.g. to tune an information extraction system or to adjust the trust in a particular worker when a crowd sourcing approach is used.

Also similar to social networks, the system needs to consider the concept of "friends" or trust, respectively. Not all the sources can be trusted, neither in the data they provide nor in their feedback. Hence, nodes need to provide a trust evaluation and a trust network needs to be established. Thus, when answering queries, provenance is an essential aspect. Provenance information can be used, first, to establish whether information received is trustworthy, and second to find original source nodes that provide interesting information to the network. This way, high quality nodes can be found and be queried directly in the future. However, for basic network exploration, nodes will also need to provide meta-information about themselves, similar to the VoID ontology descriptions used for linked open data[1] allowing other nodes to decide whether they might be interested in a node's information services. Additionally, nodes can offer update subscriptions to deal with dynamic data.

User Perspective From a user's perspective, the system is similar to a social file sharing network, where all users can participate in as soon as they know any node that already participates. Ideally, the user formulates a query and the system does everything automatically – it identifies relevant sources, rewrites and extends the query, considers mappings between ontologies, applies reasoning to derive further information etc.

The system also allows a user to restrict and personalize a query based on her own

[1] http://www.w3.org/TR/void/

beliefs, e.g., if aliens exist or not. On the other hand, a user might want to test her beliefs by issuing a consistency query, that explicitly aims to find indications that contradict her basic beliefs. Therefore, the system also incorporates reasoning as well as reasoning nodes so that a user does not need to find, install, and configure a reasoning framework herself.

In order to help the user formulate structured queries, the system can make use of approaches that map natural language queries into structured queries. In addition, the system is interactive, i.e., in case of ambiguities with respect to query interpretation, the system will ask the user and learn from her responses.

Let us consider the application scenario from Section 6.3, where a company builds a website about local cinemas, providing information such as their location, which movies are shown and how many places are left for a particular showing.

With Colledge, the company first sets up a node within the network or accesses an existing one. This will require some effort the first time it is done, but that node can be reused for multiple applications.

Once the node has run an initialization phase to discover its neighbourhood, the company needs to manually find out the best way to canonically formulate their queries and then train the node for typical queries. However, instead of reading ontology descriptions and searching for fitting ontologies, a human simply issues natural language queries and decides whether the results fit her expectations. From the results she can also derive typical entity and relation names to use in the automatic query generation later employed by the web application. For instance, a developer who wants to program a method to retrieve all horror movies currently shown in "Steve's Cinema" might simply issue a natural language query for "Steve's Cinema shows what horror movies?" which might be translated into a query with the SPARQL triple patterns (`cs:StevesCinema`, `cs:shows`, `?m`).(`?m`, `rdf:type`, `imdb:horror`). The system might then come up with triples of the form (`cs:StevesCinema`, `cs:shows`, `imdb:LotR`).(`imdb:LotR`, `rdf:type`, `imdb:horror`) at which point the developer can directly use canonic expressions to formulate the query in the future. Note that the possible complexity of the natural language interface depends mainly on the capabilities of the translation module, we are using a simple example here.

Figure 6.2 outlines a possible distribution of the example query through the network. Note that several types of nodes are involved (see Section 6.5.2). The query enters the network at Q, a simple relay node that separates the query q into its two components (triple patterns) $q1$ and $q2$. An aggregator relay node then retrieves answers for $q1$ from two nodes offering information on movies, while $q2$ is processed by a reasoning node, which reduces $q2$ to $q3$ and $q4$, retrieved from an aggregator memory node AG. This aggregator uses amongst others an extraction node (SC) which regularly parses the website of Steve's Cinema and informs AG about updates. Alternatively SC could not offer an update subscription, but extract only at query time. However, in this example the data should change at predictable times, thus a fixed extraction schedule may be more plausible.

Once the answer is received, the company might flag the answer or a particular triple as invalid, e.g. since the company does not consider "Lord of the Rings" a horror movie.

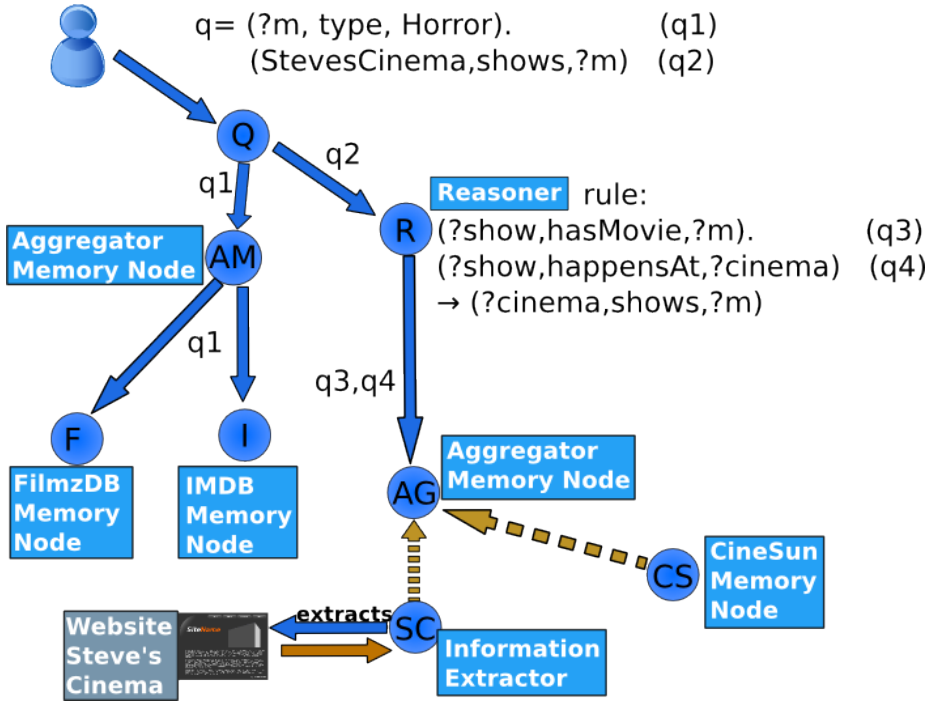


Figure 6.2.: Query Example

Sources: User and website symbol from OpenCliparts.org (public domain)

This feedback will be sent back to all nodes involved in retrieving the query answer and they may decide to accept or ignore this opinion. If the source node accepts the feedback as valid, the company's query node will never see the answer again. Even if the feedback is not initially accepted, the query node will always filter this particular answer out, either by placing it as a query constraint along with any such query or by filtering the answers by comparing them with the local user provided knowledge.

In the next section we shall discuss some aspects in more detail.

6.5. Colledge Core Aspects

In the following, we outline some of the core aspects considered by the envisioned framework and identify some of the main challenges that remain to be solved.

6.5.1. Data and Query Model

While the proposed framework could be based on any data model or query language, we focus on RDF and SPARQL as these are the de facto standards used in the Semantic Web. Thus, the basic component of interactions within the knowledge network are SPARQL queries, which we restrict for now to the pattern matchching queries introduced in Definition 2.1.16, and query answers (Definition 2.1.17). In extension to the definitions in Chapter 2, we assume in the following that each statement comes with a *confidence value* $conf(s) \in [0, 1]$ indicating how likely the statement expresses a true piece of information – facts are statements with $conf(s) = 1$. The confidence can be expressed by a separate statement of the form (`s`, `hasConfidence`, `c`) where $c \in [0, 1]$.

In distributed systems, the terms *query coordinator* or *query origin* denote the node in the network that issues a query, i.e., the node at which a user (or any other software running on the node) initiates the query processing. In the following, we will use these terms to denote which node a query originates from and therefore where the final result to a query has to be available.

6.5.2. Knowledge Network and Node Classification

Formally, a knowledge network can be defined as a set of knowledge nodes N connected by a set of directed edges E, where each edge from node n_1 to node n_2 indicates that n_1 is aware of n_2's existence and knows how to access n_2's data. Each node $n \in N$ needs to implement a query and a feedback interface so that it can receive and answer queries and learn from feedback.

The query interface can be implemented on top of an extended SPARQL endpoint or in principle by any other kind of service generating RDF data. Thus, from a technical point of view, we distinguish three main categories of nodes:

1. *Relay nodes* maintain indexes and operate as query re-routers.

2. *Memory nodes* provide access to stored semantic data, e.g. by masking an RDF triple store, reading an RDF file or wrapping another structured format (e.g. a database).

3. *Generation nodes* generate information on-the-fly. For the time being, we envision nodes applying information extraction, crowd sourcing, reasoning nodes, or sensor nodes. While information extraction nodes employ an information extraction system to generate information from source documents (at least partially) at query time, crowd sourcing nodes use crowd sourcing tasks to generate RDF triples [FKK+11,

SLB12]. Reasoning nodes use logical rules to (recursively) derive knowledge from existing data during runtime.

In practice, a node may belong to multiple categories.

6.5.3. Provenance and Trust

To determine the quality in terms of soundness and trustworthiness of query results in a distributed setup, an important piece of meta information is how the results were derived from existing knowledge and where this knowledge originates from. Thus, during query processing additional meta information needs to be exchanged.

To formally define the origin of data as part of the meta data, we use the notion of a *knowledge source*.

Definition 6.5.1 (Knowledge Source, Primary Source)**.** A *knowledge source* of a statement s provides a representation of s. A knowledge source of a statement not referencing another knowledge source as its origin is called a *primary source* of the statement.

To identify the origin of information, nodes answering a (sub)query provide provenance information on their local results. In the simplest case, provenance information for a result triple resembles a chain:

Definition 6.5.2 (Simple Provenance Chain)**.** Given a query q and a node n that produces statement s as triple result to q, a *simple provenance chain* is a sequence of knowledge sources $n, n_1, \ldots, n_m$ reflecting the nodes via which statement s has been retrieved.

Ignoring reasoning for the moment, for each piece of information, there is at least one simple provenance chain from a *primary source* to its final recipient. The trustworthiness of a piece of information is usually based on such provenance with special regard to primary sources. Typically, confidence in a statement s

- increases the more *provenance chains* are based on different primary sources,
- increases the more different provenance chains are based on the same primary source ps, but the trust held in ps can never be exceeded,
- increases the shorter the path to a primary source is.

Consider the example in Figure 6.2 and sub-query $q1$. Assume that node I returns a response triple $s =$(`LotR`, `type`, `Horror`). Then Q, AM, I is a simple provenance chain and I is the primary source for s. Q might now either trust the information chain or verify the data directly at the (primary) source, in this case at I.

Either way, at some point, the *confidence* in s needs to be estimated. This will depend on the *trust* in the knowledge sources of the remaining provenance chain. Basically, in order to believe a statement s to be true, one needs to trust each node along the way to

be honest. Thus given a simple provenance chain $q, n_1, ..., n_w$ and a trust function t on knowledge sources, we can estimate the confidence $conf(s)$ in s at node q by

$$conf(s) = \prod_{i \in \{1,..,w\}} t(n_i) \tag{6.1}$$

Knowledge sources might also be aware that some information they provide is less certain. An information extraction system might for instance know that its extraction methods are less reliable for some relations or particular statements. Similarly a node n that forwards knowledge it gained from other nodes, e.g., acting as a proxy or knowledge mirror, might provide its own estimation of the statement's reliability, e.g. based on its own trust-based confidence evaluation based on its own sources. It may do this in addition or instead of provenance information, such that the query node can choose to consider such a node n as the source of the information, at least with regard to its confidence estimation. Thus, a node i might provide its own confidence estimation $conf_i(s)$, indicating how confident it is to have understood or measured the information correctly.

Note that this is only a simplified outline for basic cases. In practice, provenance information for a query answer resembles a directed acyclic graph where nodes represent logical operators. Consider, for instance, s being retrieved not only from I but also from F via AM. Then, at AM we can believe in s if we trust I or F. Similarly, reasoning nodes can reduce a statement s by applying logical rules to a set of pre-condition statements $c_1, ..., c_n$, resulting in s being true iff we believe in $c_1, ..., c_n$, thus introducing additional logical nodes in the provenance graph.

There are methods in probabilistic reasoning to derive a probabilistic confidence from such provenance graphs [SORK11]. While the worst case runtime for such methods can be exponential, they can still be applied in real-time [MDT11] as provenance graphs usually have a limited size.

These trust computations can either be done at the query origin or a node might delegate trust estimations by trusting the confidence estimations received from neighbours along with query answers. Recursively applied, confidence is computed stepwise based on the local trust function at each node the query passes. This approach also has the advantage that it allows for more efficient optimization within the network. For instance, partial results can be ordered by local confidence and only the top results be returned. The final setting may apply a mix of these approaches which may be depending on negotiations between a querying and the answering node. After all, some nodes might not be willing or able to provide provenance information but might be able to give a confidence estimation or the other way around. However, from the query node's point of view, local confidence computions have the drawback that the original source becomes obfuscated, especially if provenance information is omitted. Thus, information received from several nodes can originate in the same primary source without the query node being aware of it, which may lead to an overestimation of the confidence in the information.

6.5.4. Generation Nodes & Data Freshness

Our knowledge network includes nodes that can acquire information at query time to deal with changing information. Important examples include information extraction and crowd sourcing, but also sensor and reasoning nodes.

We assume that only a minority of all statements change over time and their change frequency varies heavily, e.g. consider a change in temperature versus the change of a movie theatre's schedule. Thus, it is not reasonable to update information more often than it actually changes. On the other hand, a generation node might have limited update capabilities. If we imagine a sensor node measuring the temperature at the North Pole or analysing the state of the coffee machine in the kitchen through a web-cam, taking a particular measurement is a matter of milliseconds, still there might be a measurable cost associated with sending a signal from the North Pole. An extraction node re-parsing several web pages or a crowd sourcing node, for instance, retrieving crowd opinions on the quality of a current movie either by graded rating or a 500 word review – in Gaelic, may take seconds, minutes or even longer. Thus, a node may be limited physically to a maximal update frequency. Additionally, limitations like available computation power, parallel queries, and monetary cost can add a dynamic limit on the actual update frequency. Depending on the actual information need, information freshness requirements may also be variable. A querying node might be satisfied if the information retrieved contains all updates to a certain time point or contains on average only a certain percentage of outdated triples. For instance, if the user needs to estimate whether to bring a bikini or a snow suit for her holidays, it is tolerable if the temperature at her holiday destination was acquired a few days ago. Similarly, if calculating the average number of movie ticket sales over a weekend to calculate whether a movie achieved a box office record, a margin of error might be acceptable in a first estimation, thus a certain percentage of visitor numbers being not up-to-date would be acceptable.

Thus, in our model a querying node can attach a hard or soft data freshness constraint to its query. In the soft case, the answering node can decide to answer although it cannot satisfy the constraint, but a hard constraint indicates the query coordinator will throw away any result not within the constraints anyway, thus it can spare such answers.

However, while a node may not be able to always provide the most up-to-date information, the least it needs to provide is meta-information on how fresh its information is.

Finally, aggregator nodes can divert traffic from primary source nodes by caching their knowledge on particular knowledge domains, e.g. on movies. In the example in Figure 6.2 the nodes AM and AG are examples of such aggregator nodes. An aggregator node may decide to subscribe to an aggregated node's updates of certain information types, if the aggregated node offers such an update stream. A typical example where this makes sense is information that changes infrequently, such as a movie schedule. In the example in Figure 6.2 AG has subscribed to update streams from CS and SC and thus, never needs to ask for information covered by these subscriptions.

6.5.5. Personalization

While there is often great emphasis on constructing consistent knowledge spaces, we explicitly suggest the creation of an inconsistent knowledge space within the network. The knowledge network should represent human understandings of the world, which are often enough inconsistent, especially when beliefs held by different people are mixed. Thus, to retrieve meaningful answers, the system may need to decide at query time which particular statements to accept as base facts. This decision should depend on the beliefs held by the user. These can be explicitly stated at query time as query constraints or gathered over time from user feedback (see Section 6.5.7). For instance, a user query may be extended by the statement (`LotR`, `type`, `Horror`) to indicate that the user thinks the movie "Lord of the Rings" is a horror movie, and thus the query result for movie suggestions to see with her young nephew should not contain it. Again, such query constraints could be applied as a filter once all query answers reach the query node along with full provenance information, but in most cases it would be more efficient to provide such constraints as part of the query, such that results in conflict with the query constraints are filtered out locally within the network.

Note that the full power of query constraints is only used when reasoning is employed. Basically either the query node first derives as many consequences from any user specific query constraints and adds these as well or individual nodes need to employ reasoning, possibly via a reasoning node in order to include implied consequences derived from the constraints specified.

6.5.6. Query Processing

The first step of processing a query that cannot be answered locally is to locate nodes with matching knowledge. In today's Linked Open Data (LOD) setup, the user needs to manually identify candidate nodes and add them to their federation [SHH+11]. While such a manual setup should still be possible to add known trustworthy nodes, we envision that knowledge nodes automatically discover other nodes using techniques similar to peer-to-peer networks, e.g., flooding, routing indices etc. (for an overview over peer-to-peer network approaches see [ATS04]).

Once potential sources are identified, the query processor can simply split up the query and ask every known node, similar to the current approach for LOD. In Figure 6.2 the query node first splits the original query q into its two sub-queries $q1$ and $q2$, solves the sub-queries and then joins the results. Each query executed yields information about which nodes can provide which kind of information, including information about transitive neighbours and especially their primary sources. Thus, a node can iteratively learn more and more about nodes in the network. Unlike current LOD settings, however, these nodes can be of very different kind, so the quality of information provided by a node can vary and needs to be estimated. This allows for ranking nodes and asking nodes according to the ranking in batches until the result is satisfying. Typically, memory nodes will provide results faster than generation nodes, but there may also be differences amongst generation

nodes. This node efficiency can be learnt iteratively just as the trust distribution. To speed up the learning process for both properties, i.e., information availability and query processing efficiency, we require that every node provides general meta-information, such as its type and the knowledge domain it covers, similar to the voID descriptions used in the LOD world [ACHZ09].

Given both user-defined goals and per-node estimations for quality and efficiency, a cost model is needed to determine optimal query executions. Existing work on integrating information extraction on-the-fly into SQL [EHFI12, JIDG09] often includes such cost models, but they need to be extended with per statement confidence. Additionally, query execution in our network is distributed and thus it could leverage existing distributed SPARQL approaches [SHH+11]. As query run-time at different nodes may vary largely, we envision an incremental processing strategy, reporting initial results early while continuing evaluation in the background, such that they are available when a user asks for more or more convincing results. To alleviate this, a querying node can attach time or quality constraints to a query, such that a queried node can either tune its execution plan or directly refuse execution.

An important aspect not considered by existing systems are the different update rates of different types of information. Depending on how often the information changes and how up-to-date the information needs to be, it may not always be necessary to re-extract. This balance between data dynamic and user constraints on data freshness needs to be integrated into the cost model as well. For the problem of out-dated SPARQL endoints that aggregate and cache information from other Linked Data sources Umbrich et al [UKHP12] suggest to split queries based on 'coherence estimates'. While this is suggesting a similar behaviour as we suggest, the approach considers only how to retrieve the most up-to-date information, yet it ingnores that an application might not require certain information to be the most up-to-date.

6.5.7. Feedback

User feedback in our setting can be given either explicitly by pointing out that particular statements are not true according to the user's belief system, or implicitly, e.g. by learning from query constraints used by the user, by results she usually accepts and vice versa by learning from results that seem not to satisfy the user, such that she asks for more. Thus, by constantly harvesting a user's feedback and query constraints, a knowledge node may generate user belief statistics, i.e. a set of statements a user finds valid or invalid associated with confidence values indicating how certain the node is that the user holds this belief. The system can then re-use this information for personalization and to improve in several ways:

1. It may include statements a user beliefs in as default query constraints.

2. It may use statements a user beliefs in as answers for incoming queries, citing the user as source.

3. It may adjust the (user specific) trust function for a source whenever the user accepts or rejects statements that stem from this source.

4. If the node has an information extraction component it may adapt confidence estimations for the method used to extract the statement.

While feedback is typically collected after a query has been answered, the system might also directly ask the user during query execution to clarify the query or her assumptions about the knowledge domain. For instance the system could

- try to clarify disambiguation problems to understand the query, e.g. asking which "Einstein" the user meant, the famous scientist or the dog from the "Back to the Future" movies,

- ask whether it can extend or restrict the query in order to improve the result quality or the performance, e.g. by replacing an entity type like `GermanActor` by a more general type like `Actor` or vice versa, or

- try to clarify some basic beliefs that are decisive for the query execution, e.g. whether the user accepts that mankind has visited the moon or not.

While such interaction may make a huge difference in terms of execution cost, when the right questions are asked, the number of user questions needs to be limited such that the user does not get annoyed by answering her own query. Hence this could also be integrated into the cost model with an additional user discomfort cost and a maximum discomfort threshold depending on the user.

6.5.8. Collaboration

Arguably the main difference we would like to encourage is that of (semi-)automatic collaboration. While today, knowledge publishers can only gain from one another if they manually talk to each other or copy parts of another ontology (in case they do not agree with parts of its modelling), which might be considered plagiarism by the original authors, we want to explicitly encourage and support information exchange. In particular this means nodes in our network architecture should collaborate and exchange:

- ontological knowledge relevant for query answering,

- provenance information and thus, knowledge about their information sources,

- index information recommending expert nodes,

- user feedback by forwarding feedback to relevant nodes, and

- meta information on good extraction, crowd sourcing or sensor settings

We have previously discussed the typical answering of queries as well as how users can generate feedback and how it can be used locally to improve a node's performance. However, once a query is answered and feedback from a user gathered, it is the responsibility of the query node to try pay back some value to the nodes that helped to answer the query. Thus, user feedback should be forwarded to all nodes involved in the query answering, more precisely to all nodes potentially affected by the feedback. For instance, if user feedback claims a certain statement to be false, feedback shall be sent to nodes that were responsible in producing that part of the answer. However, the query node may decide to contact additional nodes, if it thinks these may also benefit due to its semantic node index. Additionally a query node may have contacted different nodes and thus it might have received different, potentially disagreeing results. It should also inform nodes that produced disagreeing results about the opinion of its counterpart. This also holds for any node that issued a sub-query.

Consider the example in Figure 6.2 and assume the query results amongst others in the IMDB node I returning the triple $s = (\texttt{LotR}, \texttt{type}, \texttt{Horror})$ towards the aggregator node AM, while F provides a triple $\overline{s} = (\texttt{LotR}, \texttt{not\_type}, \texttt{Horror})$ indicating the opposite. Since I enjoys more trust, at the query node, s will be retrieved and be part of a query answer. If now a user refutes this part of the answer, Q will forward the feedback to AM, which will forward it to I and M, which may update their knowledge. Q and AM may also adapt their trust function for I and M.

Similarly nodes can exchange index information, which can be especially helpful for initial node discovery. While we already suggested that each node provides an interface for self-description, this could be extended to also offer descriptions on other nodes, based on the index information available. This way, a node new to the network could ask initially known neighbours explicitly for the best expert nodes to be asked about certain kinds of information, thus short-cutting the discovery process further.

Additionally, nodes may gather and exchange user belief statistics (if users agree) in order to generate user clusters to improve indexing and query processing as well as try to establish belief worlds, i.e. sets of statements agreed on by a substantial amount of users. This can help again in improving result quality by employing automatic query rewriting techniques, e.g. adding more constraints from the belief world a user belongs to. If user statistics can be exchanged, this could even lead to social interaction amongst users that share similar beliefs, which might also simply mean they are interested in similar knowledge domains.

Finally, the network may collaboratively solve larger tasks like ontology matching [SAS11] and entity consolidation in a distributed way. For instance, during query processing reasoning nodes might logically derive new `sameAs` links. This information can be distributed through the network as part of a query answer or in a dedicated exchange step. The more nodes agree on a `sameAs` statement, the more likely it will establish itself as a fact in large portions of the network and thus become accepted truth. This can help in ontology matching as well as entity consolidation. There could also be dedicated nodes coordinating such global endeavours, e.g. by collecting `sameAs` candidates and spreading candidates accepted by enough reasoning nodes deliberately within the network.

6.6. Summary and Future Work

In this section we envisioned a distributed self-organizing social knowledge network to enable collaborative knowledge generation and management. We gave an overview over several important research areas related to semantic knowledge generation and management and outline a way to combine these issues in a shared architecture. Our envisioned network aims to lower the participation threshold and encourage a higher level of knowledge exchange and interactivity by integrating knowledge acquisition with knowledge management, retrieval and user feedback. While trying to ease access and still give users as many personalization freedoms as possible, participating nodes are also free to decide about their own policy.

While we sketch a rough outline of a knowledge network of the future, an implementation of this architecture, however, lies out of the scope of this work. In each aspect we discussed there remain open problems to overcome before such a network can be realized. However, in her master's thesis [Gon13] Anastasiya Goncharova has implemented a first prototype system that locally simulates all three protocol steps (query, answer, feedback) of node communication in a basic implementation. Her system splits and forwards queries and provides provenance information on a per-statement level that allows the query node to observe the information flow back to the source node. She also analyed some information trust estimation and node selection methods. Please find further details in [Gon13].

Among the major obstacles that need to be tackled towards such a knowledge network, we leave one aspect relatively untouched, namely that of security. While a trust network might protect against some forms of knowledge oriented attacks, other forms of network attacks, like denial of service attacks and data redundancy are not addressed at all.

7. Conclusion

Summary In this work we have provided three components aimed at supporting users in extracting information into ontologies, and in the maintenance and navigation of ontologies. First, a framework integrating user feedback into the extraction process and the ontology (Chapter 3). Second, methods to support fact verification and exploration by identifying the most relevant source documents for a set of statements using extraction information (Chapter 4) and briefly discussed the application of semantic annotations to navigate documents by semantic similarity. And third, entity similarity measures allowing to explore an ontology along links based on similarity rather than relatedness (Chapter 5).

In addition we discussed a particular extraction scenario for which a workflow environment has been designed, we suggest an extraction pattern based approach for ad-hoc fact extraction (or fact confirmation). We also propose a peer-to-peer based framework as a next-generation knowledge network.

Outlook On each topic area touched, there remains room for improvement and there are multiple possibilities for future work. For instance, we only touched the topic of personalisation and trust in user feedback and it remains an open question how to learn the type hierarchy in a similarly interactive fashion as the entities, their names and other relation instances. We also did not explore any user involvement in extending the set of rules used for reasoning, e.g. maybe from massive opposing or supporting feedback general rules that make the feedback unnecessary could be deduced. In our witness retrieval approach we only used pagerank as a document trust approximation. This proved to have no impact in our experiments, and in general is only a rough easy to obtain estimation for actual trust. It may be too fine grained on the one hand, e.g. source trust might be determined by the URL domain, e.g. trusting Wikipedia or the Washington Post more than a random blog. Then, on the other hand, this might require a more fine-grained approach that applies to paragraphs rather than documents. A related task would be to identify the actual author of a textual phrase, e.g. deciding whether the sentence is meant to represent the article's author's opinion or whether it is a quote. Another possibility would be to aim at grasping the general tone and topic domain of the article, e.g. trusting a scientifically written article more than one filled with esoteric vocabulary on questions of physics. There also remain open problems for entity similarity based ontology navigation, e.g. in identifying characteristic basic aspects for entities or the most likely entity to represent a compound aspect, e.g. while Angela Merkel is quite famous, she is probably not the the first person that comes to mind when it comes to the set of all (female) scientists (she holds a Phd

degree).

After all, we provide some methods that could help human users in several ways to employ extraction techniques for their tasks at hand, but until information extraction systems truely engage users in a userfriendly way and become tools of everyday use in professions such as journalism and social science that can be employed as easily to a set of documents as today's data visualisations can be applied to spreadsheet information there are still a few more steps to be taken.

8. Bibliography

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives, *DBpedia: A nucleus for a web of open data*, The Semantic Web (Karl Aberer et al., eds.), LNCS, vol. 4825, Springer Berlin Heidelberg, 2007, pp. 722–735.

[ACA06] Alekh Agarwal, Soumen Chakrabarti, and Sunny Aggarwal, *Learning to rank networked entities*, KDD, 2006, pp. 14–23.

[ACHZ09] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao, *Describing Linked Datasets – On the Design and Usage of voiD, the Vocabulary of Interlinked Datasets*, LDOW, 2009.

[Ade98] Brad Adelberg, *Nodose - a tool for semi-automatically extracting structured and semistructured data from text documents.*, SIGMOD Record, 1998, pp. 283–294.

[AG00] Eugene Agichtein and Luis Gravano, *Snowball: Extracting relations from large plain-text collections*, Proceedings of the Fifth ACM Conference on Digital Libraries (New York, NY, USA), DL '00, ACM, 2000, pp. 85–94.

[AM08] Riccardo Albertoni and Monica De Martino, *Asymmetric and context-dependent semantic similarity among ontology instances*, J. Data Semantics **10** (2008), 1–30.

[ama] *Amazon.com*, `http://www.amazon.com`.

[Ari76] Roger Ariew, *Ockham's razor: A historical and philosophical analysis of ockham's principle of parsimony*, University of Illinois press, Champaign-Urbana, 1976.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis, *A survey of peer-to-peer content distribution technologies*, ACM Comput. Surv. **36** (2004), no. 4, 335–371.

[BAdR09] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke, *A language modeling framework for expert finding*, Inf. Process. Manage. **45** (2009), no. 1, 1–19.

[Bar77] J. Barwise, *An introduction to first-order logic*, Handbook of Mathematical Logic (J. Barwise, ed.), North-Holland, Amsterdam, 1977, pp. 5–46.

[Bat95] M Bates, *Models of natural language understanding*, Proceedings of the National Academy of Sciences **92** (1995), no. 22, 9977–9982.

[BB98] Amit Bagga and Breck Baldwin, *Entity-based cross-document coreferencing using the vector space model*, Proceedings of the 17th International Conference on Computational Linguistics - Volume 1 (Stroudsburg, PA, USA), COLING '98, Association for Computational Linguistics, 1998, pp. 79–85.

[BBdR11] Krisztian Balog, Marc Bron, and Maarten de Rijke, *Query modeling for entity search based on terms, categories, and examples*, ACM Trans. Inf. Syst. **29** (2011), no. 4, 22.

[BBdR13] Marc Bron, Krisztian Balog, and Maarten de Rijke, *Example based entity search in the web of data*, ECIR, 2013, pp. 392–403.

[BBH+09] Falk Brauer, Wojciech M. Barczynski, Gregor Hackenbroich, Marcus Schramm, Adrian Mocan, and Felix Förster, *RankIE: Document Retrieval on Ranked Entity Graphs*, PVLDB **2** (2009), no. 2, 1578–1581.

[BBM09] Wojciech M. Barczynski, Falk Brauer, and Adrian Mocan, *ExplainIE - Explaining Information Extraction Systems*, ICIQ, 2009, pp. 268–269.

[BCD+07] Stephan Bloehdorn, Philipp Cimiano, Alistair Duke, Peter Haase, Jörg Heizmann, Ian Thurlow, and Johanna Voelker, *Ontology-based Question Answering for Digital Libraries*, Proceedings of the European Conference on Research and Advanced Technologies for Digital Libraries (ECDL) (Berlin, Germany) (László Kovács, Norbert Fuhr, and Carlo Meghini, eds.), Lecture Notes in Computer Science, no. 4675, Springer, 2007, pp. 14–25.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (eds.), *The description logic handbook: Theory, implementation, and applications*, Cambridge University Press, New York, NY, USA, 2003.

[BCSW07] Holger Bast, Alexandru Chitea, Fabian M. Suchanek, and Ingmar Weber, *ESTER: efficient search on Text, Entities, and Relations*, 30th International Conference on Research and Development in Information Retrieval (SIGIR'07) (Amsterdam, Netherlands) (Charlie Clarke, Norbert Fuhr, and Noriko Kando, eds.), Association for Computing Machinery (ACM), Association for Computing Machinery (ACM), 2007, pp. 671–678.

[BDT08] Xi Bai, Renaud Delbru, and Giovanni Tummarello, *RDF snippets for Semantic Web search engines*, OTM Conferences (2), 2008, pp. 1304–1318.

[BE08] Michele Banko and Oren Etzioni, *The tradeoffs between open and traditional relation extraction*, ACL, The Association for Computer Linguistics, 2008, pp. 28–36.

[BEP+08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, *Freebase: a collaboratively created graph database for structuring human knowledge*, SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data (New York, NY, USA), ACM, 2008, pp. 1247–1250.

[BGE+08] Michel Buffa, Fabien Gandon, Guillaume Ereteo, Peter Sander, and Catherine Faron, *Sweetwiki: A semantic wiki*, Web Semantics: Science, Services and Agents on the World Wide Web **6** (2008), no. 1, 84 – 97, Semantic Web and Web 2.0.

[BIPM97] John Bear, David J. Israel, Jeff Petit, and David L. Martin, *Using information extraction to improve document retrieval*, TREC, 1997, pp. 367–377.

[BL98] T. Berners-Lee, *RFC 2396: Uniform Resource Identifiers (URI)*, Tech. report, MIT, 1998.

[BLM06] Maria Luisa Bonet, Jordi Levy, and Felip Manyà, *A complete calculus for max-sat*, SAT (Armin Biere and Carla P. Gomes, eds.), Lecture Notes in Computer Science, vol. 4121, Springer, 2006, pp. 240–251.

[Blo10] Sebastian Blohm, *Large-scale pattern-based information extraction from the world wide web.*, Ph.D. thesis, Karlsruhe Institute of Technology, 2010, http://d-nb.info/1000088529, pp. 1–236.

[Boo86] Time-Life Books, *Artificial intelligence*, Understanding computers, Time-Life Books, 1986.

[BP98] Sergey Brin and Lawrence Page, *The anatomy of a large-scale hypertextual web search engine*, Computer Networks **30** (1998), no. 1-7, 107–117.

[BP06] Razvan Bunescu and Marius Pasca, *Using Encyclopedic Knowledge for Named Entity Disambiguation*, Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06) (Trento, Italy), 2006, pp. 9–16.

[Bri99] Sergey Brin, *Extracting Patterns and Relations from the World Wide Web*, WebDB, 1999, pp. 172–183.

[BSdV11] Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries, *Overview of the TREC 2011 entity track*, TREC, 2011.

[BTPC07] Kurt Bollacker, Patrick Tufts, Tomi Pierce, and Robert Cook, *A platform for scalable, collaborative, structured information integration*, Intl. Workshop on Information Integration on the Web (IIWeb'07), 2007.

[BTV08] Alessio Bechini, Andrea Tomasi, and Jacopo Viotto, *Enabling ontology-based document classification and management in ebxml registries*, Proceedings of the ACM symposium on Applied computing (New York, NY, USA), ACM, 2008, pp. 1145–1150.

[BZ10] Roi Blanco and Hugo Zaragoza, *Finding support sentences for entities*, SIGIR, 2010, pp. 339–346.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell, *Toward an architecture for never-ending language learning*, Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010), 2010.

[CC06] M. Fernández I. Cantador and P. Castells, *Core: A tool for collaborative ontology reuse and evaluation*, 4th International Workshop on Evaluation of Ontologies for the Web (EON 2006) at the 15th International World Wide Web Conference (WWW 2006) (Edinburgh, UK), May 2006.

[CCB06] Jon Curtis, John Cabral, and David Baxter, *On the application of the Cyc ontology to word sense disambiguation*, In Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference, 2006, pp. 652–657.

[CGN05] N. Chatterjee, S. Goyal, and A. Naithani, *Resolving pattern ambiguity for english to hindi machine translation using WordNet*, Workshop on Modern Approaches in Translation Technologies, 2005.

[Chi98] Nancy A. Chinchor, *Proceedings of the Seventh Message Understanding Conference (MUC-7) named entity task definition*, Proceedings of the Seventh Message Understanding Conference (MUC-7) (Fairfax, VA), April 1998, version 3.5, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/, p. 21 pages.

[CHM11] Michael J. Cafarella, Alon Y. Halevy, and Jayant Madhavan, *Structured data on the web*, Commun. ACM **54** (2011), no. 2, 72–79.

[CKGS06] Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, and Khaled F. Shaalan, *A survey of web information extraction systems*, IEEE Trans. Knowl. Data Eng. **18** (2006), no. 10, 1411–1428.

[CL01] Chia-Hui Chang and Shao-Chen Lui, *Iepad: Information extraction based on pattern discovery*, Proceedings of the 10th International Conference on World Wide Web (New York, NY, USA), WWW '01, ACM, 2001, pp. 681–688.

[CM08] Ed H. Chi and Todd Mytkowicz, *Understanding the efficiency of social tagging systems using information theory*, HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia (New York, NY, USA), ACM, 2008, pp. 81–88.

[CMBT02a] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan, *GATE: A framework and graphical development environment for robust NLP tools and applications.*, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 2002.

[CMBT02b] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan, *GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications*, Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), 2002.

[CQ09] Gong Cheng and Yuzhong Qu, *Searching Linked Objects with Falcons: Approach, Implementation and Evaluation*, Int. J. Semantic Web Inf. Syst. **5** (2009), no. 3, 49–70.

[CR93] Alain Colmerauer and Philippe Roussel, *The birth of prolog*, HOPL Preprints (John A. N. Lee and Jean E. Sammet, eds.), ACM, 1993, pp. 37–52.

[CRSE07] Michael J. Cafarella, Christopher Re, Dan Suciu, and Oren Etzioni, *Structured Querying of Web Text Data: A Technical Challenge.*, the Conference on Innovative Data Systems Research (CIDR), Online Proceedings, 2007, pp. 225–234.

[DFJ+04] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs, *Swoogle: a search and metadata engine for the semantic web*, CIKM, 2004, pp. 652–659.

[DHJ+13] Milad Daivandy, Denis Hünich, Rene Jäkel, Steffen Metzger, Ralph Müller-Pfefferkorn, and Bernd Schuller, *Heterogeneous resource federation with a centralized security model for information extraction*, Journal of Internet Services and Applications **4** (2013), no. 1, 10.

[DHM+04] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang, *Similarity search for web services*, Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30, VLDB '04, VLDB Endowment, 2004, pp. 372–383.

[DIdV09] Gianluca Demartini, Tereza Iofciu, and Arjen P. de Vries, *Overview of the INEX 2009 entity ranking track*, INEX, 2009, pp. 254–264.

[Dom98] J. Domingue, *Tadzebao and webonto: Discussing, browsing, editing ontologies on the web*, 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, 1998.

[DSJS10] Anish Das Sarma, Alpa Jain, and Divesh Srivastava, *I4e: Interactive investigation of iterative information extraction*, Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '10, ACM, 2010, pp. 795–806.

[Dub10] Julien Dubois, *Introduction to Jersey—a Standard, Open Source REST Implementation*, `http://www.oracle.com/technetwork/articles/java/jersey-part1-159891.html`, June 2010.

[EBC06] Oren Etzioni, Michele Banko, and Michael J. Cafarella, *Machine reading*, Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2, AAAI'06, AAAI Press, 2006, pp. 1517–1519.

[EBSW08] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld, *Open information extraction from the web*, Commun. ACM **51** (2008), no. 12, 68–74.

[ECD+04] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates, *Web-scale Information Extraction in Knowitall: (Preliminary Results)*, Proceedings of the 13th International Conference on World Wide Web (New York, NY, USA), WWW '04, ACM, 2004, pp. 100–110.

[EFC+11] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam, *Open information extraction: The second generation.*, IJCAI (Toby Walsh, ed.), IJCAI/AAAI, 2011, pp. 3–10.

[EHFI12] Amr El-Helw, Mina H. Farid, and Ihab F. Ilyas, *Just-in-time information extraction using extraction views*, Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '12, ACM, 2012, pp. 613–616.

[EHL+08] Kathrin Eichler, Holmer Hemsen, Markus Löckelt, Günter Neumann, and Norbert Reithinger, *Interactive dynamic information extraction*, KI (Andreas Dengel, Karsten Berns, Thomas M. Breuel, Frank Bomarius, and Thomas Roth-Berghofer, eds.), Lecture Notes in Computer Science, vol. 5243, Springer, 2008, pp. 54–61.

[EHMS10] Shady Elbassuoni, Katja Hose, Steffen Metzger, and Ralf Schenkel, *ROXXI: Reviving witness dOcuments to eXplore eXtracted Information*, Proceedings of the VLDB Endowment **3** (2010), no. 1-2, 1589–1592.

[EHSH04] Marc Ehrig, Peter Haase, Nenad Stojanovic, and Mark Hefke, *Similarity for Ontologies – a Comprehensive Framework*, PAKM'04, 2004.

[ERS+09] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum, *Language-model-based ranking for queries on RDF-graphs*, CIKM, 2009, pp. 977–986.

[FBCC+10] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty, *Building Watson: An Overview of the DeepQA Project*, AI Magazine **31** (2010), no. 3, 59–79.

[FFR97] Adam Farquhar, Richard Fikes, and James Rice, *The ontolingua server: A tool for collaborative ontology construction*, Int. J. Hum.-Comput. Stud. **46** (1997), no. 6, 707–727.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning, *Incorporating non-local information into information extraction systems by gibbs sampling*, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (Stroudsburg, PA, USA), ACL '05, Association for Computational Linguistics, 2005, pp. 363–370.

[Fie00] Roy Thomas Fielding, *Architectural styles and the design of network-based software architectures*, Phd thesis, University of California, 2000.

[FKK+11] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin, *CrowdDB: answering queries with crowdsourcing*, SIGMOD, 2011, pp. 61–72.

[FKL+10] Ronald Fagin, Benny Kimelfeld, Yunyao Li, Sriram Raghavan, and Shivakumar Vaithyanathan, *Understanding queries in a search database system*, PODS, 2010, pp. 273–284.

[FL04] David Ferrucci and Adam Lally, *Uima: an architectural approach to unstructured information processing in the corporate research environment*, Nat. Lang. Eng. **10** (2004), no. 3-4, 327–348.

[FN71] Richard E. Fikes and Nils J. Nilsson, *Strips: A new approach to the application of theorem proving to problem solving*, Proceedings of the 2Nd International Joint Conference on Artificial Intelligence (San Francisco, CA, USA), IJCAI'71, Morgan Kaufmann Publishers Inc., 1971, pp. 608–620.

[FP10] Manaal Faruqui and Sebastian Padó, *Training and evaluating a german named entity recognizer with semantic generalization*, Proceedings of KONVENS 2010 (Saarbrücken, Germany), 2010.

[Fra96] Alexander Franz, *Automatic ambiguity resolution in natural language processing*, Lecture Notes in Artificial Intelligence, vol. 1171, Springer-Verlag, Berlin, 1996.

[FT02] Roy T. Fielding and Richard N. Taylor, *Principled design of the modern web architecture*, ACM Trans. Internet Technol. **2** (2002), no. 2, 115–150.

[gei] *Geizhals Preisvergleich*, `http://www.geizhals.de`.

[Gen91] Michael R. Genesereth, *Knowledge interchange format*, KR (James F. Allen, Richard Fikes, and Erik Sandewall, eds.), Morgan Kaufmann, 1991, pp. 599–600.

[gkg] *Google Knowledge Graph*, `http://www.google.com/insidesearch/features/search/knowledge.html?hl=en`.

[GKLM11] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu, *View selection in semantic web databases*, PVLDB **5** (2011), no. 2, 97–108.

[GKRT04] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins, *Propagation of trust and distrust*, Proceedings of the 13th International Conference on World Wide Web (New York, NY, USA), WWW '04, ACM, 2004, pp. 403–412.

[GL90] Ramanathan V. Guha and Douglas B. Lenat, *CYC: A Midterm Report*, AI Magazine **11** (1990), no. 3, 32–59.

[GMB11] Bela Gipp, Norman Meuschke, and Joeran Beel, *Comparative evaluation of text- and citation-based plagiarism detection approaches using guttenplag*, JCDL '11, 2011, pp. 255–258.

[Gon13] Anastasiya Goncharova, *Provenance and Feedback Based Trusted Source Selection in Collaborative Knowledge Networks*, Master's thesis, Universität des Saarlandes, Saarbrücken, 2013.

[goo] *Google Shopping*, `http://www.google.com/shopping`.

[GS96] Ralph Grishman and Beth Sundheim, *Message understanding conference-6: A brief history*, Proceedings of the 16th Conference on Computational Linguistics - Volume 1 (Stroudsburg, PA, USA), COLING '96, Association for Computational Linguistics, 1996, pp. 466–471.

[GSW05] Jens Graupmann, Ralf Schenkel, and Gerhard Weikum, *The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents*, Proceedings of the 31st International Conference on Very Large Data Bases, VLDB '05, VLDB Endowment, 2005, pp. 529–540.

[Har11] Olaf Hartig, *Zero-knowledge query planning for an iterator implementation of link traversal based query execution*, ESWC (1), 2011, pp. 154–169.

[Hav03] Taher H. Haveliwala, *Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search*, IEEE Trans. Knowl. Data Eng. **15** (2003), no. 4, 784–796.

[HBR14] Xuedong Huang, James Baker, and Raj Reddy, *A historical perspective of speech recognition*, Communications of the ACM **57** (2014), no. 1, 94–103.

[Hea92] Marti Hearst, *Direction-based text interpretation as an information access refinement*, Text-Based Intelligent Systems, 1992, pp. 257–274.

[HHD+07] Andreas Harth, Aidan Hogan, Renaud Delbru, Jürgen Umbrich, Seán O'Riain, and Stefan Decker, *SWSE: Answers Before Links!*, Semantic Web Challenge, 2007.

[HHU+11] Aidan Hogan, Andreas Harth, Jürgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker, *Searching and browsing linked data with SWSE: The semantic web search engine*, J. Web Sem. **9** (2011), no. 4, 365–401.

[Hie00] Djoerd Hiemstra, *A probabilistic justification for using tf x idf term weighting in information retrieval*, Int. J. on Digital Libraries **3** (2000), no. 2, 131–139.

[Hie03] ———, *Statistical language models for intelligent xml retrieval*, Intelligent Search on XML Data, 2003, pp. 107–118.

[HLN04] W. Hunt, L.V. Lita, and E. Nyberg, *Gazetteers, WordNet, Encyclopedias, and The Web: Analyzing Question Answering Resources*, Tech. Report CMU-LTI-04-188, Language Technologies Institute, Carnegie Mellon, 2004.

[HMS11] Katja Hose, Steffen Metzger, and Ralf Schenkel, *Sharing Knowledge between Independent Grid Communities*, 6th International Workshop on Applications of Semantic Technologies (AST 2011) (Berlin, Germany) (Catherina Burghart, Stephan Grimm, Andreas Harth, and Jens Wissmann, eds.), 2011.

[HS12] Katja Hose and Ralf Schenkel, *Towards Benefit-Based RDF Source Selection for SPARQL Queries*, Semantic Web Information Management (SWIM) (Scottsdale, AZ), 2012.

[HSBW13] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum, *Yago2: A spatially and temporally enhanced knowledge base from wikipedia*, Artificial Intelligence **194** (2013), no. 0, 28–61.

[HYB+11] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum, *Robust Disambiguation of Named Entities in Text*, Proceedings of the Conference on Empirical Methods in Natural Language Processing (Stroudsburg, PA, USA), EMNLP '11, Association for Computational Linguistics, 2011, pp. 782–792.

[IG12] Sergej Isak-Geidel, *Informationsextraktion in deutschen Texten*, Master's Thesis, Saarland University, Saarbrücken, Germany, 2012.

[IHMS11] Hassan Issa, Katja Hose, Steffen Metzger, and Ralf Schenkel, *Advances Towards Semantic Plagiarism Detection*, Workshop Information Retrieval at LWA 2011 (Magdeburg, Germany) (Ingo Frommholz and Claus-Peter Klas, eds.), 2011.

[imd] *IMDB.org*, `http://www.imdb.org`.

[Iss12] Hassan Issa, *Semantic Aware Document Similarity Search*, Master's Thesis, Saarland University, Saarbrücken, Germany, January 2012.

[IW06] Georgiana Ifrim and Gerhard Weikum, *Transductive learning for text classification using explicit knowledge models*, the European Conference on Principles and Practice of Knowledge Discovery (PKDD) (Berlin, Germany) (Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, eds.), Lecture Notes in Artificial Intelligence, vol. 4213, Springer, 2006, pp. 223–234.

[JDG08] Alpa Jain, AnHai Doan, and Luis Gravano, *Optimizing SQL queries over text databases*, ICDE, 2008, pp. 636–645.

[jer] *Jersey - RESTful Web Services in Java*, `https://jersey.java.net`.

[JIDG09] Alpa Jain, Panagiotis G. Ipeirotis, AnHai Doan, and Luis Gravano, *Join optimization of information extraction output: Quality matters!*, ICDE, 2009, pp. 186–197.

[JIG08] Alpa Jain, Panagiotis G. Ipeirotis, and Luis Gravano, *Building query optimizers for information extraction: the SQoUT project*, SIGMOD Record **37** (2008), no. 4, 28–34.

[JK02] Kalervo Järvelin and Jaana Kekäläinen, *Cumulated gain-based evaluation of ir techniques*, ACM Trans. Inf. Syst. **20** (2002), no. 4, 422–446.

[JKS+07] Krzysztof Janowicz, Carsten Keßler, Mirco Schwarz, Marc Wilkes, Ilija Panov, Martin Espeter, and Boris Bäumer, *Algorithm, implementation and application of the SIM-DL similarity server*, GeoS, 2007, pp. 128–145.

[JM80] Frederick Jelinek and Robert L. Mercer, *Interpolated estimation of markov source parameters from sparse data*, In Proceedings of the Workshop on Pattern Recognition in Practice (Amsterdam, The Netherlands: North-Holland), May 1980, pp. 381–397.

[JM00] Daniel Jurafsky and James H. Martin, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, 1st ed., Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[JMD+12] René Jäkel, Steffen Metzger, Jason Milad Daivandy, Katja Hose, Dennis Hünich, Ralf Schenkel, and Bernd Schuller, *Interactive Information Extraction based on Distributed Data Management for German Grid Projects*, EGI Community Forum 2012 / EMI Second Technical Conference (EGICF12-EMITC2) (Munich, Germany), Proceedings of Science, Proceedings of Science, 2012, pp. 1–10.

[JSS00] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic, *Real life, real users, and real needs: a study and analysis of user queries on the web*, Information Processing & Management **36** (2000), no. 2, 207 – 227.

[JW02] Glen Jeh and Jennifer Widom, *SimRank: a measure of structural-context similarity*, KDD, 2002, pp. 538–543.

[Kat97] Boris Katz, *Annotating the world wide web using natural language.*, RIAO (Luc Devroye and Claude Chrisment, eds.), 1997, pp. 136–159.

[KB07] Paul Kalmar and Matthias Blume, *Fico: Web person disambiguation via weighted similarity of entity contexts*, Proceedings of the 4th International Workshop on Semantic Evaluations (Stroudsburg, PA, USA), SemEval '07, Association for Computational Linguistics, 2007, pp. 149–152.

[KCKW11] Matthias Klusch, Xiaoqi Cao, Patrick Kapahnke, and Stefan Warwas, *Semantics and Agents for Intelligent Simulation and Collaboration in the 3D Internet*, SKG, IEEE, 2011, pp. 9–16.

[KK09] Matthias Klusch and Patrick Kapahnke, *OWLS-MX3: An Adaptive Hybrid Semantic Service Matchmaker for OWL-S*, Proceedings of 3rd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, vol. 525, CEUR-WS.org, 2009.

[KKK+11] Manolis Koubarakis, Kostis Kyzirakos, Manos Karpathiotakis, Charalampos Nikolaou, Michael Sioutis, Stavros Vassos, Dimitrios Michail, Themistoklis Herekakis, Charalampos Kontoes, and Ioannis Papoutsis, *Challenges for Qualitative Spatial Reasoning in Linked Geospatial Data*, IJCAI 2011 Workshop on Benchmarks and Applications of Spatial Reasoning (BASR-11), 2011, pp. 33–38.

[KL89] Michael Kifer and Georg Lausen, *F-logic: A higher-order language for reasoning about objects, inheritance, and scheme*, Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '89, ACM, 1989, pp. 134–146.

[KL94] Kevin Knight and Steve K. Luk, *Building a large-scale knowledge base for machine translation.*, National Conference on Aritificial Intelligence, AAAI, 1994, pp. 773–778.

[kno] *knoodl, web-based modular set of ontology oriented tools*, `http://www.knoodl.com`.

[Kow88] Robert A. Kowalski, *The early years of logic programming*, Commun. ACM **31** (1988), no. 1, 38–43.

[KPS+05] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James Hendler, *Swoop: A web ontology editing browser*, Journal of Web Semantics **4** (2005), 2005.

[KSRC09] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti, *Collective annotation of Wikipedia entities in web text*, KDD, 2009, pp. 457–466.

[KW12] Erdal Kuzey and Gerhard Weikum, *Extraction of temporal facts and events from wikipedia*, Proceedings of the 2Nd Temporal Web Analytics Workshop (New York, NY, USA), TempWeb '12, ACM, 2012, pp. 25–32.

[LC01] Bo Leuf and Ward Cunningham, *The wiki way: quick collaboration on the web*, Addison-Wesley, London, March 2001.

[LC07] Jie Lu and Jamie Callan, *Content-Based Peer-to-Peer Network Overlay for Full-Text Federated Search*, RIAO (David Evans, Sadaoki Furui, and Chantal Soulé-Dupuy, eds.), CID, 2007.

[LCC+11] Brian Lowe, Brian Caruso, Nick Cappadona, Miles Worthington, Stella Mitchell, and Jon Corson-Rikert, *The vitro integrated ontology editor and semantic web application.*, ICBO (Olivier Bodenreider, Maryann E. Martone, and Alan Ruttenberg, eds.), CEUR Workshop Proceedings, vol. 833, CEUR-WS.org, 2011.

[LD08] Pieter Leenheer and Christophe Debruyne, *Dogma-mess: A tool for fact-oriented collaborative ontology evolution*, On the Move to Meaningful Internet Systems: OTM 2008 Workshops (Robert Meersman, Zahir Tari, and Pilar Herrero, eds.), Lecture Notes in Computer Science, vol. 5333, Springer Berlin Heidelberg, 2008, pp. 797–806.

[LDH06] Jing Liu, Xin Dong, and Alon Y. Halevy, *Answering structured queries on unstructured data*, WebDB, 2006.

[LKR+08] Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and H. V. Jagadish, *Regular expression learning for information extraction*, Proceedings of the Conference on Empirical Methods in Natural Language Processing (Stroudsburg, PA, USA), EMNLP '08, Association for Computational Linguistics, 2008, pp. 21–30.

[LLYM04] Shuang Liu, Fang Liu, Clement Yu, and Weiyi Meng, *An effective approach to document retrieval via utilizing WordNet and recognizing phrases*, the Annual International ACM SIGIR conference on Research and development in information retrieval (SIGIR) (New York, NY, USA), ACM, 2004, pp. 266–272.

[LPW05] Michael D. Lee, Brandon Pincombe, and Matthew Welsh, *An Empirical Evaluation of Models of Text Document Similarity*, pp. 1254–1259, Erlbaum, Mahwah, NJ, 2005.

[LRNdS02] Alberto H.F. Laender, Berthier Ribeiro-Neto, and Altigran S. da Silva, *DEByE – Data Extraction By Example*, Data & Knowledge Engineering **40** (2002), no. 2, 121 – 154.

[LT11] Günter Ladwig and Thanh Tran, *SIHJoin: Querying remote and local linked data*, ESWC (1), 2011, pp. 139–153.

[LW10] Xiao Ling and Daniel S. Weld, *Temporal information extraction.*, AAAI (Maria Fox and David Poole, eds.), AAAI Press, 2010.

[LWB08] Andreas Langegger, Wolfram Wöß, and Martin Blöchl, *A semantic web middleware for virtual data integration on the web*, ESWC, 2008, pp. 493–507.

[Man07] Christoph Mangold, *A survey and classification of semantic search approaches*, IJMSO **2** (2007), no. 1, 23–34.

[MC08] Luke K. McDowell and Michael Cafarella, *Ontology-driven, unsupervised instance population*, Web Semant. **6** (2008), no. 3, 218–236.

[MC10] Einat Minkov and William W. Cohen, *Improving graph-walk-based similarity with reranking: Case studies for personal information management*, ACM Trans. Inf. Syst. **29** (2010), no. 1, 4.

[MCWD06] Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira, *An introduction to the syntax and content of Cyc*, Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, 2006, pp. 44–49.

[MDT11] Timm Meiser, Maximilian Dylla, and Martin Theobald, *Interactive reasoning in uncertain RDF knowledge bases*, CIKM, 2011, pp. 2557–2560.

[MEHS11] Steffen Metzger, Shady Elbassuoni, Katja Hose, and Ralf Schenkel, *S3K: Seeking Statement-Supporting top-K Witnesses*, 20th ACM Conference on Information and Knowledge Management (CIKM 2011) (Glasgow, UK), 2011.

[MFRW00a] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder, *An Environment for Merging and Testing Large Ontologies*, Proc. of the Seventh International Conference on Principles of Knowledge (Breckenridgeand Coloradoand United States), April 2000.

[MFRW00b] Deborah L. Mcguinness, Richard Fikes, James Rice, and Steve Wilder, *The chimaera ontology environment*, Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000), 2000.

[MHS12] Steffen Metzger, Katja Hose, and Ralf Schenkel, *Colledge - a vision of collaborative knowledge networks*, 2nd International Workshop on Semantic Search over the Web (SSW 2012) (Istanbul, Turkey), ACM, 2012, p. .

[Mil95] George A. Miller, *Wordnet: A lexical database for english*, Commun. ACM **38** (1995), no. 11, 39–41.

[MP98] E. Marsh and D. Perzanowski, *Muc-7 evaluation of ie technology: Overview of results*, Proceedings of the Seventh Message Understanding Conference (MUC-7), `http://www.itl.nist.gov/iaui/894.02/-related_projects/muc/index.html`, 1998.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to information retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[MSH09] Boris Motik, Rob Shearer, and Ian Horrocks, *Hypertableau Reasoning for Description Logics*, Journal of Artificial Intelligence Research **36** (2009), 165–228.

[MSHS12] Steffen Metzger, Michael Stoll, Katja Hose, and Ralf Schenkel, *LUKe and MIKE: Learning from User Knowledge and Managing Interactive Knowledge Extraction*, 21st ACM International Conference on Information and Knowledge Management (CIKM 2012) (Maui, USA), 2012.

[MSS13] Steffen Metzger, Ralf Schenkel, and Marcin Sydow, *QBEES: Query by Entity Examples*, 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013) (San Francisco, USA), 2013.

[MTS+06] Hendry Muljadi, Hideaki Takeda, Aman Shakya, Shoko Kawamoto, Satoshi Kobayashi, Asao Fujiyama, and Koichi Ando, *Semantic wiki as a lightweight knowledge management system*, Proceedings of the First Asian Conference on The Semantic Web (Berlin, Heidelberg), ASWC'06, Springer-Verlag, 2006, pp. 65–71.

[MTW05] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum, *MINERVA∞: A Scalable Efficient Peer-to-Peer Search Engine*, USENIX 2005 (Grenoble,

France), Lecture Notes in Computer Science, vol. 3790, Springer, 2005, pp. 60–81.

[muc98] *Proceedings of the seventh message understanding conference (muc-7)*, April 1998, version 3.5, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/.

[mys] *MySQL*, `http://www.mysql.com`.

[NMS+07] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, and Wei-Ying Ma, *Web object retrieval*, WWW, 2007, pp. 81–90.

[NP01] Ian Niles and Adam Pease, *Towards a standard upper ontology*, Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001 (New York, NY, USA), FOIS '01, ACM, 2001, pp. 2–9.

[NS07] David Nadeau and Satoshi Sekine, *A survey of named entity recognition and classification*, Linguisticae Investigatiaones **30** (2007), no. 1, 3–26, Publisher: John Benjamins Publishing Company.

[NTW11] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum, *Scalable knowledge harvesting with high precision and high recall*, WSDM, 2011, pp. 227–236.

[NWS12] Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek, *Patty: A taxonomy of relational patterns with semantic types.*, EMNLP-CoNLL, ACL, 2012, pp. 1135–1145.

[ocy] *OpenCyc,* the publicly available version of Cycorp's commercial Cyc ontology and reasoning platform, `http://www.cyc.com/platform/opencyc`.

[ODC+08] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello, *Sindice.com: a document-oriented lookup index for open linked data*, Int. J. Metadata Semant. Ontologies **3** (2008), 37–52.

[OH94] Akitoshi Okumura and Eduard Hovy, *Building Japanese-English dictionary based on ontology for machine translation*, HLT '94: Proceedings of the workshop on Human Language Technology (Morristown, NJ, USA), Association for Computational Linguistics, 1994, pp. 141–146.

[ope08] *Open NLP*, Website: http://opennlp.sourceforge.net, 2008.

[OVBD06] Eyal Oren, Max Völkel, John G. Breslin, and Stefan Decker, *Semantic wikis for personal knowledge management*, Database and Expert Systems Applications, vol. 4080/2006, Springer Berlin / Heidelberg, September 2006, pp. 509–518.

[owl] *OWLGrEd, a graphical ontology editor*, `http://owlgred.lumii.lv`.

[PBCE+10] Martin Potthast, Alberto Barrón-Cedeño, Andreas Eiselt, Benno Stein, and Paolo Rosso, *Overview of the 2nd International Competition on Plagiarism Detection*, CLEF'10, 2010.

[PC98] Jay M. Ponte and W. Bruce Croft, *A language modeling approach to information retrieval*, SIGIR, ACM, 1998, pp. 275–281.

[PE10] Giuseppe Pirrò and Jérôme Euzenat, *A feature and information theoretic framework for semantic similarity and relatedness*, International Semantic Web Conference (1), 2010, pp. 615–630.

[PIW10] Jeffrey Pound, Ihab F. Ilyas, and Grant Weddell, *Expressive and flexible access to web-extracted data: a keyword-based structured query language*, SIGMOD, 2010, pp. 423–434.

[PK00] Thierry Poibeau and Leila Kosseim, *Proper name extraction from non-journalistic texts.*, CLIN (Walter Daelemans, Khalil Sima'an, Jorn Veenstra, and Jakub Zavrel, eds.), Language and Computers - Studies in Practical Linguistics, vol. 37, Rodopi, 2000, pp. 144–157.

[pos] *PostgreSQL*, `http://www.postgresql.org`.

[pri] *Pricegrabber.com*, `http://www.pricegrabber.com`.

[PWTY08] Thomas Penin, Haofen Wang, Thanh Tran, and Yong Yu, *Snippet Generation for Semantic Web Search Engines*, ASWC, 2008, pp. 493–507.

[QF93] Yonggang Qiu and Hans-Peter Frei, *Concept based query expansion*, SIGIR, 1993, pp. 160–169.

[QL08] Bastian Quilitz and Ulf Leser, *Querying distributed RDF data sources with SPARQL*, ESWC, 2008, pp. 524–538.

[RE04] M. Andrea Rodríguez and Max J. Egenhofer, *Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure*, International Journal of Geographical Information Science **18** (2004), no. 3, 229–256.

[RH02] Deepak Ravichandran and Eduard Hovy, *Learning surface text patterns for a question answering system*, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (Stroudsburg, PA, USA), ACL '02, Association for Computational Linguistics, 2002, pp. 41–47.

[RN03] Stuart J. Russell and Peter Norvig, *Artificial intelligence: A modern approach*, 2 ed., Pearson Education, 2003.

[RRDA11] L. Ratinov, D. Roth, D. Downey, and M. Anderson, *Local and global algorithms for disambiguation to wikipedia*, ACL, 2011.

[RZ09] Stephen E. Robertson and Hugo Zaragoza, *The probabilistic relevance framework: Bm25 and beyond*, Foundations and Trends in Information Retrieval **3** (2009), no. 4, 333–389.

[Sar08] Sunita Sarawagi, *Information extraction*, Foundations and Trends in Databases **1** (2008), no. 3, 261–377.

[SAS11] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart, *PARIS: Probabilistic Alignment of Relations, Instances, and Schema*, PVLDB **5** (2011), no. 3, 157–168.

[Sch06] S. Schaffert, *Ikewiki: A semantic wiki for collaborative knowledge management*, Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE '06. 15th IEEE International Workshops on, 2006, pp. 388–396.

[Sga82] Petr Sgall, *Natural language understanding and the perspectives of question answering.*, COLING, 1982, pp. 357–364.

[Sha92] Stuart C. Shapiro, *Encyclopedia of artificial intelligence*, 2nd ed., John Wiley & Sons, Inc., New York, NY, USA, 1992.

[SHH+11] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt, *FedX: Optimization techniques for federated query processing on linked data*, ISWC, 2011, pp. 601–616.

[SHMS11] Michael Stoll, Katja Hose, Steffen Metzger, and Ralf Schenkel, *Interaktive Wissensextraktion und Wissenssuche*, LWA 2011 : Technical Report ; Report of the symposium "Lernen, Wissen, Adaptivität 2011" of the GI special interest groups KDML, IR and WM (Magdeburg) (Myra Spiliopoulou, Andreas Nürnberger, and Rene Schult, eds.), Otto von Guericke Universität, 2011, pp. 205–206.

[sima] *Similar Pages, a similar page search engine*, `http://www.similarpages.com`.

[simb] *Similar Sites, a similar page search engine*, `http://www.similarsites.com`.

[SIW06] Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum, *Leila: Learning to extract information by linguistic analysis*, Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge (Sydney, Australia), Association for Computational Linguistics, July 2006, pp. 18–25.

[SKW07] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum, *YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia*, 16th International World Wide Web Conference (WWW 2007) (Banff, Canada) (Carey L. Williamson, Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, eds.), ACM, 2007, pp. 697–706.

[SLB12] Joachim Selke, Christoph Lofi, and Wolf-Tilo Balke, *Pushing the boundaries of crowd-enabled databases with query-driven schema expansion*, PVLDB **5** (2012), no. 6, 538–549.

[SLKL05] Mu-Hee Song, Soo-Yeon Lim, Dong-Jin Kang, and Sang-Jo Lee, *Automatic classification of web pages based on the concept of domain ontology*, Asia-Pacific Software Engineering Conference (2005), 645–651.

[SMMC09] Hala Skaf-Molli, Pascal Molli, and Gérôme Canals, *Swooki: Supporting disconnection in a peer-to-peer semantic wiki*, Proceedings of the 5th French-Speaking Conference on Mobility and Ubiquity Computing (New York, NY, USA), UbiMob '09, ACM, 2009, pp. 91–94.

[SMMS02] Ljiljana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic, *User-driven ontology evolution management*, Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web (London, UK, UK), EKAW '02, Springer-Verlag, 2002, pp. 285–300.

[SORK11] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch, *Probabilistic databases*, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2011.

[SPA] *W3C: SPARQL Query Language for RDF*, www.w3.org/TR/rdf-sparql-query/.

[SS09a] Pavel Smrz and Marek Schmidt, *Information extraction in semantic wikis*, SemWiki (Christoph Lange, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel, eds.), CEUR Workshop Proceedings, vol. 464, CEUR-WS.org, 2009.

[SS09b] Steffen Staab and Rudi Studer, *Handbook on ontologies*, 2nd ed., Springer Publishing Company, Incorporated, 2009.

[SSM05] Bertrand Sereno, Simon Buckingham Shum, and Enrico Motta, *Claimspotter: an environment to support sensemaking with knowledge triples*, IUI, 2005, pp. 199–206.

[SSS08] Kinga Schumacher, Michael Sintek, and Leo Sauermann, *Combining Fact and Document Retrieval with Spreading Activation for Semantic Desktop Search*, ESWC, 2008, pp. 569–583.

[SSVW09] Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin, *A new hybrid dependency parser for German*, Von der Form zur Bedeutung: Texte automatisch verarbeiten / From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009 (Tübingen) (C. Chiarcos, R. E. de Castilho, and M. Stede, eds.), 2009, pp. 115–124.

[SSW09] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum, *SOFIE: A Self-Organizing Framework for Information Extraction*, International World Wide Web conference (WWW 2009) (New York, NY, USA), ACM Press, 2009.

[Sto12] Michael Stoll, *MIKE - managing interactive knowlegde extraction*, Master's thesis, Universität des Saarlandes, Saarbrücken, April 2012.

[stu] *StumbleUpon, a social tagging search engine*, `https://www.stumbleupon.com`.

[Suc08] Fabian Suchanek, *Automated construction and growth of a large ontology*, Doctoral dissertation, Universität des Saarlandes, December 2008.

[SVT+12] Thomas Steiner, Ruben Verborgh, Raphaël Troncy, Joaquim Gabarró, and Rik Van de Walle, *Adding realtime coverage to the google knowledge graph.*, International Semantic Web Conference (Posters & Demos) (Birte Glimm and David Huynh, eds.), CEUR Workshop Proceedings, vol. 914, CEUR-WS.org, 2012.

[THK+04] Heike Telljohann, Erhard Hinrichs, Sandra Kübler, Ra Kübler, and Universität Tübingen, *The tüba-d/z treebank: Annotating german with a context-free backbone*, In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004, 2004, pp. 2229–2235.

[TKS02] Erik F. Tjong Kim Sang, *Introduction to the conll-2002 shared task: Language-independent named entity recognition*, Proceedings of CoNLL-2002, Taipei, Taiwan, 2002, pp. 155–158.

[TNFM11] Tania Tudorache, Natalya Fridman Noy, Sean M. Falconer, and Mark A. Musen, *A knowledge base driven user interface for collaborative ontology development*, IUI (Pearl Pu, Michael J. Pazzani, Elisabeth André, and Doug Riecken, eds.), ACM, 2011, pp. 411–414.

[Tun09] Daniel Tunkelang, *Faceted search*, Synthesis Lectures on Information Concepts, Retrieval, and Services **1** (2009), no. 1, 1–80.

[UBL+12] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano, *Template-based question answering over rdf data*, Proceedings of the 21st international conference on World Wide Web, 2012, pp. 639–648.

[UHK+11] Jürgen Umbrich, Katja Hose, Marcel Karnstedt, Andreas Harth, and Axel Polleres, *Comparing data summaries for processing live queries over linked data*, World Wide Web **14** (2011), no. 5-6, 495–544.

[UKHP12] Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira, *Freshening up while staying fast: Towards hybrid sparql queries*, EKAW (Annette ten Teije, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles, and Nathalie Hernandez, eds.), Lecture Notes in Computer Science, vol. 7603, Springer, 2012, pp. 164–174.

[USBL06] Victoria S. Uren, Simon Buckingham Shum, Michelle Bachler, and Gangmin Li, *Sensemaking tools for understanding research literatures: Design, implementation and user evaluation*, International Journal of Man-Machine Studies **64** (2006), no. 5, 420–445.

[vit] *Vitro, integrated ontology editor and semantic web application*, `http://vitro.mannlib.cornell.edu`.

[VK06] Denny Vrandecic and Markus Krötzsch, *Reusing ontological background knowledge in semantic wikis*, Proceedings of the First Workshop on Semantic Wikis – From Wikis to Semantics (Budva, Montenegro) (Max Völkel, Sebastian Schaffert, and Stefan Decker, eds.), Juni 2006.

[VKV+06] Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer, *Semantic wikipedia*, Proceedings of the 15th International Conference on World Wide Web (New York, NY, USA), WWW '06, ACM, 2006, pp. 585–594.

[Vra12] Denny Vrandečić, *Wikidata: A new platform for collaborative data collection*, Proceedings of the 21st International Conference Companion on World Wide Web (New York, NY, USA), WWW '12 Companion, ACM, 2012, pp. 1063–1064.

[VVMD03] Maria Vargas-Vera, Enrico Motta, and John Domingue, *AQUA: An ontology-driven question answering system*, New Directions in Question Answering (Menlo Park, CA, USA) (Mark T. Maybury, ed.), AAAI Press, 2003, pp. 53–57.

[WC07] Richard C. Wang and William W. Cohen, *Language-independent set expansion of named entities using the web*, ICDM, 2007, pp. 342–350.

[wika] *Wikidata - the free knowledge base*, `http://www.wikidata.org`.

[wikb] *Wikipedia, the free encyclopedia*, `http://www.wikipedia.org`.

[WM11] Minji Wu and Amélie Marian, *A framework for corroborating answers from multiple web sources*, Information Systems **36** (2011), no. 2, 431 – 449, Special Issue: Semantic Integration of Data, Multimedia, and Services.

[wng] *WisNetGrid project homepage*, `http://www.wisnetgrid.org`.

[Wor04a] World Wide Web Consortium, *RDF Primer*, `http://www.w3.org/TR/rdf-primer`, 2004.

[Wor04b] ———, *RDF Schema (RDFS)*, `http://www.w3.org/TR/rdf-schema`, 2004.

[Wor06] ———, *Defining N-ary Relations on the Semantic Web*, `http://www.w3.org/TR/swbp-n-aryRelations`, 2006.

[Wor12] ———, *OWL 2 Web Ontology Language*, `http://www.w3.org/TR/owl2-overview`, 2012.

[WR07] Andreas Wagner and Marc Rössler, *Walu - eine annotations- und lernumgebung für semantisches texten*, LWA (Alexander Hinneburg, ed.), Martin-Luther-University Halle-Wittenberg, 2007, pp. 154–158.

[WTA10] Xinglong Wang, Jun'ichi Tsujii, and Sophia Ananiadou, *Disambiguating the species of biomedical named entities using natural language parsers*, Bioinformatics **26** (2010), no. 5, 661–667.

[WW10] Debora Weber-Wulff, *Test cases for plagiarism detection software*, 2010, `http://www.plagiarismadvice.org/conference`.

[xai] *xaitKnow,* a commercial virtual world knowledge management tool for computer game development, `http://www.xaitment.com/english/products/xaitknow/xaitknow.html`.

[XWN01] Jinxi Xu, Ralph M. Weischedel, and Chanh Nguyen, *Evaluating a probabilistic model for cross-lingual information retrieval*, SIGIR, 2001, pp. 105–110.

[YSGK13] Taha Yasseri, Anselm Spoerri, Mark Graham, and János Kertész, *The most controversial topics in wikipedia: A multilingual and geographical analysis*, CoRR **abs/1305.5566** (2013).

[YSN+12] Xiao Yu, Yizhou Sun, Brandon Norick, Tiancheng Mao, and Jiawei Han, *User guided entity similarity search using meta-path selection in heterogeneous information networks*, CIKM, 2012, pp. 2025–2029.

[Zha08] ChengXiang Zhai, *Statistical language models for information retrieval a critical review*, Found. Trends Inf. Retr. **2** (2008).

[ZL01] Chengxiang Zhai and John Lafferty, *A study of smoothing methods for language models applied to ad hoc information retrieval*, Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (New York, NY, USA), SIGIR '01, ACM, 2001, pp. 334–342.

[ZL05] Cai-Nicolas Ziegler and Georg Lausen, *Propagation models for trust and distrust in social networks*, Information Systems Frontiers **7** (2005), no. 4-5, 337–358.

[ZS09] Andrea Zielinski and Christian Simon, *Morphisto –an open source morphological analyzer for german*, Proceedings of the 2009 Conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008 (Amsterdam, The Netherlands, The Netherlands), IOS Press, 2009, pp. 224–231.

A. Appendix

A.1. Provenance based Document Retrieval

A.1.1. Queries

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
1	(#2009834: Abraham_Lincoln, diedOnDate, 1865-04-15) (#2011306: Abraham_Lincoln, wasBornOnDate, 1809-02-12)	In this query we are interested in the birth and death of Abraham Lincoln. The main topic is the life(span) of Lincoln, so anything that talks about his life,e.g. any biography etc., is on topic.
2	(#491394119: Abdul_Qadeer_Khan, isCitizenOf, Pakistan)	In this query we are interested in Khan's role as a Pakistani citizen. Any biographical piece about his life or part of it in/for Pakistan is on topic.
3	(#490134099: Adolf_Hitler, isMarriedTo, Eva_Braun)	In this query we are interested in the relationship between Hitler and his wife Eva Braun. Any section talking about their relationship/marriage in a bit more detail than that they where married makes a document on topic. Biographies of Hitler therefore are usually on topic.
4	(#766998042: Adolf_Hitler, isLeaderOf, Germany)	In this query we are interested in Hitler's role as head of the German state, i.e. anything talking about how he took power, he's decisions as chancellor etc. is on topic. Documents providing some information about him in general are at least somewhat on topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
5	(#488194439: Adolf_Hitler, created, Mein_Kampf)	In this query we are interested in Hitler as author of Mein Kampf. Anything that talks about the book and Hitler is on topic, further information about the life of Hitler, e.g. a biography just mentioning he wrote the book is at least somewhat on topic.
6	(#2107542: Albert_Einstein, wasBornOnDate, 1879-03-14)	In this query we are interested in Einsteins birthdate. However in general any biographic information about his birth or early years is on topic. Other information only focussing e.g.about his science work is rather somewhat on topic.
7	(#489848255: Albert_Einstein, wasBornIn, Ulm) (#766991119: Albert_Einstein, wasBornOnDate, 1879-##-##)	In this query we are interested in Einsteins birth, the date and the location. Any biographic information about his birth or early years is on topic. Documents focussing on topics only related to him, e.g. his science theories are somewhat on topic or off topic depending on how much it is about Einstein himself.
8	(#2106754: Albert_Einstein, isKnownFor, General_relativity)	In this query we are interested in Einsteins invention of the theory of greater relativity. Anything talking about him coming up with this theory or greater relativity in general is on topic. Documents with only some information on Einstein in general is rather somewhat on topic.
9	(#489636323: Albert_Einstein, graduatedFrom, ETH_Zurich)	In this query we are interested in Einsteins relation to the ETH, documents talking about his time at or his work for ETH is on topic. Documents giving further information about him or ETH are at least somewhat on topic or on topic, depending on the depth.
10	(#2106742: Albert_Einstein, isKnownFor, Special_relativity) (#2106754: Albert_Einstein, isKnownFor, General_relativity)	In this query we are interested in Einsteins invention of the theory of (greater and special) relativity and how they relate to each other. Documents about him coming up with the relativity theory in general or giving further information about relativity are on topic. Documents providing some general information about Einstein are at least somewhat on topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
11	(#2106742: `Albert_Einstein`, `isKnownFor`, `Special_relativity`)	In this query we are interested in Einsteins invention of the theory of special relativity. Documents about him coming up with the theory or special relativity in general are on topic. Documents providing some other information about Einstein are at least somewhat on topic.
12	(#490086591: `Albert_Einstein`, `isMarriedTo`, `Mileva_Marić`)	In this query we are interested in Einsteins relation to Marie Maric. Any document including further information about their relation is on topic, other information about Einstein or Maric is at least somewhat on topic.
13	(#490646779: `Allianz_Arena`, `isLocatedIn`, `Munich`)	In this query we are interested in the Allianz Arena's relation to Munich, i.e. any document giving further information on its location, why it's been built etc. is on topic. Any document giving other information about the stadium or Munich is somewhat on topic.
14	(#490265927: `Ali_Khamenei`, `isLeaderOf`, `Iran`)	In this quer we are interested in Khameinis work as leader of Iran. Any document providing information about his actions as Iran's leader, how he came to power etc. is on topic. Other information about his life or Iran is somewhat on topic.
15	(#488200323: `Aleksandr_Solzhenitsyn`, `created`, `The_Gulag_Archipelago`)	In this query we are interested in Solzhenitsyn being the author of Gulag Archipelago. Any document providing further information about his time of writing, the content of the book etc. is on topic. Any nonrelated information about his life is somewhat on topic.
16	(#766997813: `Aleksandr_Solzhenitsyn`, `hasWonPrize`, `Nobel_Prize`)	In this query we are interested in Solhenitsyn being awarded the Nobel Prize. Any document providing information about the work for which he has been given the award or the circumstances under which he was given the Nobel Price is on topic. Any other information about the author is at least somewhat on topic.
17	(#490265663: `Angela_Merkel`, `isLeaderOf`, `Germany`)	In this query we are interested in Merkel's work as leader of Germany. Any document providing information about her actions as Germany's leader or the circumstances she came to power etc. are on topic. General information about Merkel or Germany is somewhat on topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
18	(#490149519: Ann_Dunham, isMarriedTo, Lolo_Soetoro) (#490406415: Ann_Dunham, hasChild, Barack_Obama)	In this query we are interested in the young years of Obama, specifically the time he lived with his stepfather Soetoro (and his mother). Any document talking about the relation of Soetoro to his stepson including the relation to his mother and of his mother to Obama is on topic as are biographic informations from this time. General information about Obama, Ann Dunham or Soetoro is at least somewhat on topic.
19	(#490264747: Arnold_Schwarzenegger, isLeaderOf, California)	In this query we are interested in Schwarzenegger's work as leader of California. Any document providing information about his actions as 'Governator' and how he came to power are of interest. General information about Schwarzenegger before his political career or about California are somewhat on topic.
20	(#490086763: Arnold_Schwarzenegger, isMarriedTo, Maria_Shriver)	In this query we are interested in the relation of Schwarzenegger with his wife Maria Shriver. Any document providing further information about their relation is on topic. Any general information about Schwarzenegger or Maria Shriver is at least somewhat on topic.
21	(#491282547: Arnold_Schwarzenegger, isPoliticianOf, California)	In this query we are interested in the role of Schwarzenegger as a Californian politician. So anything talking about his rise in politics becoming the 'Governator' or his political actions are of interest. General information about Schwarzenegger is at least somewhat on topic.
22	(#488195075: Arthur_Conan_Doyle, created, Sherlock_Holmes)	In this query we are interested in Doyle's creation of the famous Sherlock Holmes. Any document providing information about the authors thougths and writings on Holmes are on topic as is further information about the fictous character of Holmes. Any other information about his life unrelated to his writing career are somewhat on topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
23	(#766992663: Aung_San_Suu_Kyi, hasWonPrize, Nobel_Peace_Prize)	In this query we are interested in Aung San Suu Kyi winning the Nobel Peace Price. So any document providing information about her polticial activities etc. are on topic. General biographic data as well as general information about the Nobel price - including other winners of it - is somewhat on topic.
24	(#2699342: Aung_San_Suu_Kyi, wasBornOnDate, 1945-06-19)	In this query we are interested in the birth of Auung San Suu Kyi. Any further biographical data is on topic.
25	(#491330415: Australia, participatedIn, Vietnam_War)	In this query we are interested in Australia's participation in the Vientam war. Any document providing further information about Australia's role in the Vietnam war is on topic. Any document giving general information about Australia is somewhat on topic. Documents about the Vietnam war in general are usually on topic, except if they exclusively concentrate on a specific area, like American veterans, that has nothing to do with Australia.
26	(#766993608: Barack_Obama, graduatedFrom, Columbia_University)	In this query we are interested in whether Barack Obama graduated from Columbia University and if so the circumstances. Thus any document providing further information about Obama and at least partially talking about his college time is on topic. Other documents mentioning Obama sufficiently can be somewhat on topic.
27	(#490110447: Barack_Obama, isMarriedTo, Michelle_Obama)	In this query we are interested in the marriage of Barack Obama with Michelle Obama. Thus any document providing further information about Obama, Michelle and them benig married is on topic. Documents only partially talking about one of them are somewhat on topic..
28	(#490283527: Barack_Obama, isLeaderOf, United_States)	In this query we are interested in Barack Obama being the leader of the United States of America. Thus any document focussing on him and his presidency (and/or actions done as President of the states) is on topic, documents only covering him partially (under different aspects) are somewhat on topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
29	(#489636387: Bill_Clinton, graduatedFrom, Georgetown_University)	In this query we are interested in Clintons graduation at Georgetown. Thus any document providing further information about Clinton that at least partially covers his youth is on topic. Other documents mentioning him sufficiently can be somewhat on topic.
30	(#766992749: Boris_Yeltsin, isLeaderOf, Russia)	In this query we are interested in Yeltsin being the leader of Russia. Thus any document focussing on him and his presidency is on topic, documents only covering him partially (under different aspects) are somewhat on topic.
31	(#766990639: Clint_Eastwood, actedIn, Gran_Torino_(film))	In this query we are interested in Clint Eastwoods role in Gran Torino. Any Document focussing on the movie, his involvement or his general acting career is on topic. Documents partially covering Eastwood are somewhat on topic.
32	(#489090955: Clint_Eastwood, actedIn, Space_Cowboys)	In this query we are interested in Clint Eastwoods role as actor in Space Cowboys. Any Document focussing on the movie, his involvement or his general movie career is on topic. Documents partially covering Eastwood are somewhat on topic.
33	(#766997681: Clint_Eastwood, actedIn, The_Bridges_of_ Madison_County)	In this query we are interested in Clint Eastwoods role in Bridges of Madison County. Any Document focussing on the movie, his involvement or his general acting career is on topic. Documents partially covering Eastwood are somewhat on topic.
34	(#490086867: Bill_Clinton, isMarriedTo, Hillary_Rodham_Clinton)	In this query we are interested in Bill Clintons marriage to Hillary Clinton. Any document focussing on one of them (and at least partially their relationship) is on topic. Any document giving only some information about one of them is somewhat on topic.
35	(#489848447: Bill_Clinton, wasBornIn, Hope,_Arkansas)	In this query we are interested in Clinton's birthplace. Any document focussing on him providing biographical information or his birthplace is ontopic. Documents providing some other pieces of information about him are somewhat on topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
36	(#766998415: Bill_Gates, created, Microsoft)	In this query we are interested in Bill Gates relationship to Microsoft. Any document concentrating on Gates being the head of MS, on MS or Gates in general is on topic. If it is only partially about one of them it's somewhat on topic.
37	(#490298227: Cuba, hasCapital, Havana)	In this query we are interested in Cubas capital. Any document concentrating on Havanna and/or Cuba in general is on topic. If it is only partially about one of them it's somewhat on topic.
38	(#490290919: Czech_Republic, hasCapital, Prague)	In this query we are interested in the capital of the Czech Republic. Any document concentrating on Prague and/or the Republic in general is on topic. If it is only partially about one of them it's somewhat on topic.
39	(#488192715: George_Orwell, created, Animal_Farm)	In this query we are interested in the Orwell's novell Animal Farm. Any document concentrating on Orwell and/or Animal Farm in general is on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing, it is off topic.
40	(#489636547: George_H._W._Bush, graduatedFrom, Yale_University)	In this query we are interested in whether and how President George Bush Sr. graduated at Yale. Any document concentrating on Bush and/or Yale in general is on topic. However, if a document is talking about Bush in a way very unrelated to his education, you may decide that document is only somewhat on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing, it is off topic.
41	(#766992743: Jacques_Chirac, influences, Nicolas_Sarkozy)	In this query we are interested in whether and how Jacques Chirac influences/influenced his successor Sarkozy. Any document concentrating on one of them in general is on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing, it is off topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
42	(#490095839: Joe_Biden, isMarriedTo, Jill_Biden)	In this query we are interested in the vice-president of the US Joe Biden and his wife Jill. Any document concentrating on one of them in general is on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing and does not give further information, it is off topic.and
43	(#489854675: Joe_Biden, wasBornIn, Scranton,_Pennsylvania)	In this query we are interested in the vice-president of the US Joe Biden's birthplace Scranton. Any document concentrating on one of them in general is on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing without offering more information, it is off topic.
44	(#490372387: Joseph_Stalin, hasChild, Svetlana_Alliluyeva) (#490372391: Joseph_Stalin, hasChild, Yakov_Dzhugashvili)	In this query we are interested in two of Stalins children. Any document concentrating on one of them is generally on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing without offering more information, it is off topic.
45	(#490291183: Lebanon, hasCapital, Beirut)	In this query we are interested in the capital of Beirut. Any document concentrating on one of them in general is on topic. If it is only partially about one of them it's somewhat on topic. If it only mentions the fact of interest in passing without offering more information, it is off topic.
46	(#205413258: John_F._Kennedy, diedOnDate, 1963-11-22) (#490209903: John_F._Kennedy, diedIn, Dallas)	In this query we are interested in the assassination of JFK. Any document concentrating on his death (or at least dealing intensively with it while concentrating on JFK in general) is on topic. If it is only about JFKs actions as politician in some lawmaking discussion etc. or focussing on something else having some smaller paragraph about JFK's death it's somewhat on topic. If it only mentions the fact of interest in passing without offering more information, it is off topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
47	(#766990670: Marie_Curie, wasBornOnDate, 1867-##-##) (#766991506: Marie_Curie, diedIn, Paris) (#766993550: Marie_Curie, diedOnDate, 1934-##-##)	In this query we are interested in biographical information about Marie Curie. Any document concentrating on her life is on topic. Documents only discussing the content of her work are generally rather somewhat on topic as well as documents containing only smaller paragraphs about her. Docs mentionings the fact of interest in passing without offering more information are off topic.
48	(#489849719: Marie_Curie, wasBornIn, Warsaw) (#7279954: Marie_Curie, wasBornOnDate, 1867-11-07)	In this query we are interested in biographical information about Marie Curie. Any document concentrating on her life is on topic. Documents only discussing the content of her work are generally rather somewhat on topic as well as documents containing only smaller paragraphs about her. Docs mentionings the fact of interest in passing without offering more information are off topic.
49	(#491333387: Kuwait, participatedIn, Gulf_War) (#491333419: United_States, participatedIn, Gulf_War) (#491333431: Iraq, participatedIn, Gulf_War)	In this query we are interested in some participants in the Gulf war. Any document concentrating on the participation of one of those nations in the war or about the war itself or about Iraq with the war playing some role are on topic. Documents focussing about one participant country in general, e.g. other activities nothing to do with the war are somewhat on topic as well as documents containing only smaller paragraphs about the topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
50	(#491376123: France, participatedIn, War_in_Afghanistan_ (2001-present)) (#491376171: Germany, participatedIn, War_in_Afghanistan_ (2001-present)) (#491376175: United_States, participatedIn, War_in_Afghanistan_ (2001-present)) (#491376195: Taliban, participatedIn, War_in_Afghanistan_ (2001-present))	In this query we are interested in some participants in the Afghan war. Any document concentrating on the participation of one of those nations in the war or about the war itself or about Afghanistan with the war playing some role are on topic. Documents focussing about one participant country in general, e.g. other activities nothing to do with the war, are somewhat on topic as well as documents containing only smaller paragraphs about the topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.
51	(#766992696: Henry_Kissinger, hasWonPrize, Nobel_Peace_Prize) (#766992709: Le_Duc_Tho, hasWonPrize, Nobel_Peace_Prize) (#766997850: Dalai_Lama, hasWonPrize, Nobel_Peace_Prize) (#766998665: Kofi_Annan, hasWonPrize, Nobel_Peace_Prize)	In this query we are interested in some nobel peace price winners. Any document concentrating on one of them or the nobel peace price are on topic. Documents containing only smaller paragraphs about the topic are somewhat on topic. Documents mentionings the fact(s) of interest in passing without offering more information are off topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
52	(#766992689: Gertrude_B._Elion, hasWonPrize, Nobel_Prize) (#766997903: Joseph_Stiglitz, hasWonPrize, Nobel_Prize)	In this query we are interested in some nobel price winners. Any document concentrating on one of them or the nobel peace price are on topic. Documents containing only smaller paragraphs about the topic are somewhat on topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.
53	(#766997875: Harold_Pinter, hasWonPrize, Nobel_Prize) (#766998640: Elfriede_Jelinek, hasWonPrize, Nobel_Prize)	In this query we are interested in some nobel price winners. Any document concentrating on one of them winning the nobel price (or that focus on one of them and mention the winning of the price in some part) are on topic. Documents containing only smaller paragraphs about the topic are somewhat on topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.
54	(#766992695: Henri_Becquerel, hasWonPrize, Nobel_Prize) (#766992711: Marie_Curie, hasWonPrize, Nobel_Prize) (#766998653: Irène_Joliot-Curie, hasWonPrize, Nobel_Prize)	In this query we are interested in some nobel price winners. Any document concentrating on one of them or the nobel peace price are on topic. Documents containing only smaller paragraphs about the topic are somewhat on topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.
55	(#489490819: Mel_Brooks, directed, Blazing_Saddles) (#489497883: Mel_Brooks, directed, Young_Frankenstein)	In this query we are interested in two Mel Brooks (as director) movies. Any document concentrating on one of them or Mel Brooks and his film making are on topic. Documents containing only smaller paragraphs about the topic are somewhat on topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.

Table A.1.: The queries used in the S3K evaluation

ID	Fact Set	Description
56	(#490088359: `Leon_Trotsky`, `isMarriedTo`, `Aleksandra_Sokolovskaya`) (#490088363: `Leon_Trotsky`, `isMarriedTo`, `Natalia_Sedova`)	In this query we are interested in the wives of Trotsky. Any document concentrating on Trotsky or his relation to one of them are on topic. Documents containing only smaller paragraphs about the topic are somewhat on topic. Docs mentionings the fact(s) of interest in passing without offering more information are off topic.

A.2. Query by Entity Examples Search

A.2.1. Class concepts used for typical types identification

Table A.2.: The classes τ used to determine the typical types of a query (omitting 'wordnet_' prefix)

Typical Types	
geoclass_populated_place	ability_105616246
act_100030358	action_100037396
activity_100407535	actor_109765278
address_108491027	administrative_district_108491826
administrator_109770949	album_106591815
alumnus_109786338	animal_100015388
area_108497294	aristocrat_109807754
artist_109812338	associate_109816771
association_108049401	asteroid_109208702
athlete_109820263	attribute_100024264
auditory_communication_107109019	beginning_100235435
biography_106515827	body_107965085
body_of_water_109225146	broadcast_106254007
broadcasting_station_102903405	building_102913152
building_complex_102914991	business_108061042
celestial_body_109239740	change_100191142
change_of_state_100199130	chemical_114806838
chordate_101466257	city_108524735
civilization_108111783	clergyman_109927451
clothing_103051540	club_108227214
cognition_100023271	commune_108541609
communicator_109610660	community_108223802
company_108058098	composer_109947232
computer_game_100458890	concept_105835747
container_103094503	content_105809192
contest_107456188	continent_109254614
conveyance_103100490	country_108544813

Continued on next page

Table A.2 – continued from previous page

craft_103125870	creation_103129123
creativity_105624700	creator_109614315
definite_quantity_113576101	device_103183080
direction_106786629	director_110014939
discography_106488224	district_108552138
diversion_100426928	dwelling_103259505
educational_institution_108276342	educator_110045713
enterprise_108056231	entertainer_109616922
ethnic_group_107967382	event_100029378
facility_103315023	fiction_106367107
fictional_character_109587565	film_maker_110088390
football_player_110101634	fundamental_quantity_113575869
game_100430606	game_100456199
gathering_107975026	geographic_point_108578706
geological_formation_109287968	group_action_101080366
head_110162991	head_of_state_110164747
history_106514093	home_108559508
house_103544360	housing_103546340
idea_105833840	imaginary_being_109483738
institution_108053576	intellectual_109621545
journalist_110224578	language_106282651
language_unit_106284225	lawyer_110249950
leader_109623038	literary_composition_106364329
living_thing_100004258	location_100027167
material_114580897	matter_100020827
medium_106254669	message_106598915
military_officer_110317007	military_unit_108198398
minor_planet_109355623	mountain_109359803
movie_106613686	municipality_108626283
music_107020895	musical_composition_107037465
musical_organization_108246613	musician_110339966
musician_110340312	natural_elevation_109366317
natural_object_100019128	negotiator_110351874

Continued on next page

Table A.2 – continued from previous page

network_108434259	novel_106367879
official_110372373	organization_108008335
painter_110391653	part_113809207
performer_110415638	person_100007846
plant_103956922	player_110439851
poet_110444194	point_108620061
political_unit_108359949	politician_110451263
press_106263369	print_media_106263609
producer_110480018	product_104007894
professional_110480253	protocol_106665108
psychological_feature_100023100	publication_106589574
radio_station_104044119	railway_station_104049098
region_108630039	region_108630985
representative_110522035	residence_108558963
river_109411430	road_104096066
rule_106652242	ruler_110541229
scholar_110557854	school_108276720
scientist_110560637	season_115239579
serviceman_110582746	settlement_108672562
ship_104194289	show_106619065
signal_106791372	singer_110599806
site_108651247	skilled_worker_110605985
social_event_107288639	social_group_107950920
song_107048000	spiritual_leader_109505153
state_100024720	state_108168978
state_108654360	station_104306080
stream_109448361	structure_104341686
substance_100019613	symbol_106806469
team_108208560	terminal_104412901
thing_100002452	time_period_115113229
town_108665504	traveler_109629752
uniform_104509592	unit_of_measurement_113583724
urban_area_108675967	vehicle_104524313

Continued on next page

Table A.2 – continued from previous page

vertebrate_101471682	vessel_104530566
village_108672738	way_104564698
work_104599396	worker_109632518
writer_110794014	writing_106362953