J Intell Inf Syst
DOI 10.1007/s10844-017-0443-x

QBEES: query-by-example entity search in semantic knowledge graphs based on maximal aspects, diversity-awareness and relaxation

Steffen Metzger[1] · **Ralf Schenkel**[2] · **Marcin Sydow**[3]

Received: 2 February 2016 / Revised: 10 January 2017 / Accepted: 12 January 2017

Abstract Structured knowledge bases are an increasingly important way for storing and retrieving information. Within such knowledge bases, an important search task is finding similar entities based on one or more example entities. We present QBEES, a novel framework for defining entity similarity based on structural features, so-called aspects and maximal aspects of the entities, that naturally model potential interest profiles of a user submitting an ambiguous query. Our approach based on maximal aspects provides natural diversity awareness and includes query-dependent and query-independent entity ranking components. We present evaluation results with a number of existing entity list completion benchmarks, comparing to several state-of-the-art baselines.

Keywords Semantic knowledge graph · Similar entity search · Aspect model · Diversity · Experimental evaluation

✉ Marcin Sydow
msyd@ipipan.waw.pl

Steffen Metzger
smetzger@mpi-inf.mpg.de

Ralf Schenkel
schenkel@uni-trier.de

[1] Max Planck Institute for Informatics, Saarbrücken, Germany

[2] Universität Trier, Trier, Germany

[3] Institute of Computer Science, Polish Academy of Sciences and Polish-Japanese Academy of Information Technology, Warsaw, Poland

1 Introduction

Nowadays, more and more data becomes available in semantic form, be it, e.g., within the *Linked Open Data* (LOD) cloud (Heath and Bizer 2011), product databases (as used e.g. in shops like Amazon etc.) or huge common knowledge bases (ontologies) like DBpedia (Auer et al. 2007) or YAGO (Hoffart et al. 2013). In consequence, semantic data can be applied for an increasing number of information retrieval (IR) problems, e.g. to support and amend traditional IR methods in document retrieval (Metzger et al. 2011). However, with increasing amounts of information available, automatic IR methods also become more important to navigate the semantic data itself (Tunkelang 2009). One typical information retrieval task is the search for similar data given an example. While explicit search interfaces allow a fine-tuned control, many use cases rather suggest implicit query interfaces. Whenever a retrieval task is too complicated to be expressed explicitly by average users, is vague in nature or unclear to the user herself, an implicit search interface is the natural, user-friendly choice.

Consider, for instance, the task to search for possible replacements of a particular part in your production process, e.g. to optimize the overall costs. Instead of specifying all the properties of the part, the natural choice would be to provide a system with the pointer to an (already existing) semantic description. Other applications include recommendation systems and general purpose entity search engines that provide similar entities given one or several examples. With such an engine, a user might look (interactively) for movie directors that also are actors, thus, providing Quentin Tarantino and Clint Eastwood the user might expect to find Sylvester Stallone and Peter Jackson. Or she might identify powerful female politicians by providing Angela Merkel and Margaret Thatcher as examples. Note that in all cases we assume that the user expects similar entities to have similar 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.

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-)bodybuilders, governors of California, actors that appeared in The Expendables or men who cheated on their wife. Our approach tries to capture these different *branches* of similar entities and to provide the most similar entities along each such possible query interpretation.

An approach that becomes a standard tool in dealing with unknown user intent behind an ambiguous query is *diversity awareness*. The general idea is to avoid redundancy among the returned results. Diversity awareness has been successfully applied in various application domains such as web search (e.g. (Agrawal et al. 2009)), relational database querying (e.g. (Vee et al.)) or recently in entity summarization in semantic knowledge graphs (Sydow et al. 2013).

The *aspect-based* approach described in this paper models different potential user interests of an ambiguous query. Moreover, the concept of maximal aspects, introduced in Section 4.4, naturally incorporates diversity awareness as it automatically avoids redundancy in the corresponding entity sets (see the theorem and explanation in Section 5.2).

In principle, a search by example in a semantic dataset can be considered a faceted search (Tunkelang 2009) 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 no other entity shares a larger property combination that includes these properties with the query. 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. In principle, our approach to this point is very similar to a skyline search (Hose and Vlachou 2012). 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 bodybuilders that won Mr. Olympics", but also in general for Austrian bodybuilders or, for instance, Austrian sportsmen or bodybuilders of any nationality as well.

While our model is general enough to be applied to the family of "Query by Entity ExampleS" (QBEES) use cases, we focus on list completion, since this is a well-defined IR task with standard evaluation test-data available.

Contributions The main *contributions* of this paper are 1) the discussion in detail of our *aspect-based* entity model with its natural potential for modeling user interests and diversity awareness, 2) its extension by ad-hoc aspect relaxation, and 3) an evaluation of the approach.

The core idea of aspect-based similar entity retrieval has been published previously (Metzger et al. 2013, 2014). In this extended journal version, we provide additional material including:

- an extended discussion of the aspect model,
- an extended presentation of the algorithm including additional details regarding entity popularity,
- a detailed discussion how to deal with incomplete knowledge bases and inconsistent example entities,
- a completely novel and detailed model analysis section,
- a presentation of new evaluation data sets QALD3 and QALD4,
- additional evaluation experiments on QALD3 and QALD4 data sets, and
- an extended related work section.

Outline In Section 2 related work is discussed and in Section 3 basic assumptions and notation are provided. Section 4 introduces entity aspects, the basis of our model. Our main model to compute similar entities is discussed in Section 5. Section 6 presents the ad-hoc relaxation, and Section 7 discusses a solution for dealing with incomplete knowledge bases and inconsistent example entities. The evaluation results are discussed in Section 8, and Section 9 concludes the paper.

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 (Balog et al. 2011) and INEX (Demartini et al. 2009) introduced 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 (Hoffart et al. 2013), Freebase (Bollacker et al. 2007), and DBpedia (Auer et al. 2007), and the Linked Open Data (LOD) cloud, the Semantic Search Challenge[1] has extended this to semantic (RDF) data graphs with entities as nodes. The same scenario is considered in this paper.

Core ingredients of many entity search systems are similarity measures. A large body of existing work exploits the graph structure to determine entity similarity. One of the earliest approaches was SimRank (Jeh and Widom 2002) which considers two entities as similar if their context is similar. Tversky (Tversky 1977) provided one of the early set-theoretic approaches to feature based similarity. 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 (Haveliwala 2003), with a focus on relational data graphs (Agarwal et al. 2006; Minkov and Cohen 2010).

Albertoni and De Marino suggest a pair-wise asymmetric entity similarity (Albertoni and Martino 2008) based on the overlap of a candidate resource's features with selected features of the query resource that is combined with typical ontology concept similarity methods based on entity types. Our system is in so far similar, as we consider similarity also in an asymmetric way and our model shares the containment approach to some degree based on the aspects we identify. However, their system expects domain experts to explicitly determine the features that are mostly responsible for entity similarity (optionally in an interactive way). 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 refine the query without particular knowledge about the underlying system or features. (Rodríguez and Egenhofer 2004) proposes an asymmetric similarity measure called Matching-Distance Similarity Measure (MDSM) 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. (Janowicz et al. 2007) 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 et al. (Pirrò and Euzenat 2010) combine the main ideas from an ontology oriented concept similarity with a feature model from the information theory domain resulting in a feature and information theoretic measure (FaITH) based on the ratio of intrinsic information content from two concepts' closest common ancestor to the sum of their unique information content.

Another group of approaches uses features extracted from the context of entities to determine their similarity. Keßler et al. (Keßler et al. 2007) examine the impact of context on semantic similarity measures. More recently, entity similarity measures were proposed that include textual features (such as terms in the entity's URI or labels) and/or structural

[1] https://km.aifb.kit.edu/ws/semsearch11/

features (categories or types of the entity). Balog et al. (Balog et al. 2011) propose to use language models that include terms and categories. Bron et al. (Bron et al. 2013), which is closest to our work, exploit additional textual descriptions of the target group of entities in addition to example entities. They combine a term-based language model with a structural model constructed from types of and facts about the entity. The term-based language model for an entity is constructed from terms appearing in facts about that entity and in descriptions of types and other entities connected to it. In the structural model, entities are represented by their facts, similar to our type and factual aspects, with a uniform weight. Given a set of example entities, types and facts that appear in many examples have higher weight when retrieving results. In contrast, our query model does not include a term 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; our model includes priors for entity importance; and our model allows to give different weights to different features. We experimentally compare our model to their model in Section 8. A very recent work by Zhiltsov et al. (Zhiltsov et al. 2015) represents textual and structural features of an entity as a multi-fielded document and applies fielded sequential dependance models for ranking. However, as that work focuses on keyword-based ad-hoc retrieval of entities, it cannot be directly applied to our problem. Yu et al. (Yu et al. 2012) 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 labeled test data. Wang and Cohen (Wang and Cohen 2007) 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.

Mottin et al. (2014a, b) introduced the concept of exemplar queries, which is similar to the problem considered in this paper in the sense that an example result is used instead of a query. However, the setting in their XQ system is strictly different since it considers examples in the form of a connected subgraph of entities, not single entities, and determines result subgraphs based on their similarity to the query graph. The problem is therefore in some sense easier, as more information can be exploited for identifying query results. All methods require that the query is a connected graph, not a single entity, since similarity is defined based on edge-level isomorphisms; this solution therefore cannot be applied to our setting.

The GQBE system by Jayaram et al. (Jayaram et al. 2014) is similar to XQ, but does not use connected subgraphs, but just entities that form a query result as input; the meaningful connections between those entities are explored by the system. Again, the main difference to our system is that we consider only single entities as inputs and results, not combinations, and hence have fewer information for identifying relevant results. Since the scoring used in GQBE relies on weighted edges in subgraphs isomorphic to the query graph, this solution cannot be applied to our setting.

Recently, the issue of diversity awareness concerning entity-related computations on semantic knowledge graphs was studied in the context of diversity aware entity summarization in (Sydow et al. 2013) and (Kosiński et al. 2013). In entity summarization, an entity q and a numerical limit $k \in N$ is given and the task is to return a summary of the entity q in the form of a small knowledge subgraph containing at most k facts concerning q. In (Sydow et al. 2013) it is demonstrated that diversity awareness is appreciated by users. In (Kosiński et al. 2013) diversity awareness is considered as an optimization problem and a heuristic algorithm is presented. While we focus on entity similarity search, our aspect model might be useful for modeling user interests in other (than similar entity search) application domains, including the entity summarization problem. To the best of our knowledge such an approach has not been considered in this setting, yet.

3 Knowledge graph

A Knowledge Graph KG is a directed multigraph that consists of three basic components, a *Fact Graph* FG, an *Ontology Tree* O, and a set of type assignment arcs TA connecting the two. The set of vertices V of KG is partitioned into the set of *entities* E (e.g. `Warsaw`, `Poland`) and a set of *classes* C (e.g. `person` or `city`) such that $V = E \cup C$.

Existing knowledge bases represented in RDF(S) (Hitzler et al. 2010) can easily be represented as a knowledge graph, by converting factual RDF statements into edges in FG, RDFS subclass relations into edges in O, and `rdf:type` statements that assign entities to classes into edges in TA. More advanced features of RDF(S), including data types, property hierarchies, and domain and ranges of properties are not represented in the KG.

3.1 Fact graph FG

The fact graph $FG = (E, F)$ is a directed multigraph where nodes in E represent *entities* (e.g. `Warsaw`, `Poland`). A pair of nodes connected by a labeled arc in $f \in F$ represents an instance of a binary relation between them and $f's$ label represents its semantic kind (meaning) (e.g. `isCapitalOf`).

An arc, together with its ends thus represents a *fact* about the entities (e.g. "Warsaw is the capital of Poland").

The fact graph is a multigraph, since there are possibly multiple parallel arcs between the same pair of entities (e.g. "Warsaw is capital of Poland" and "Warsaw is the largest city in Poland"). We will use the notation `relation(arg1,arg2)` for any directed arc with label `relation` in FG, O, or TA that points from node `arg1` to `arg2`. In this notation the fact "Warsaw is the capital of Poland" is represented by an arc that can be denoted as `isCapitalOf(Warsaw,Poland)`.

Note that for simplicity we do not distinguish between named entities and literals, i.e. the target of a labeled arc in F can also be a literal value, e.g. a date.

3.2 Ontology tree O and type assignment TA

Each class $c \in C$ represents some *type* of entities (e.g. person). Thus, each arc of the form `hasType(anEntity,aClass)` in KG means that the entity `anEntity` is an instance of the class `aClass`.

The class nodes are connected by directed arcs labeled as `subClassOf`, such that `subClassOf(classA,classB)` indicates that `classB` is a concept that contains the more specific concept `classA`. For instance, `subClassOf(composer,musician)` indicates that every composer is also a musician. These arcs form a tree where the root represents the most general class of entities (e.g., `owl:Thing` or `wordnet_entity`). Note that this type hierarchy is transitive, i.e. given in addition `subClassOf(musician,person)` then it automatically also holds `subClassOf(composer,person)`. In other words, we assume the ontology to contain the transitive closure for the `subClassOf` relation. Due to transitive inheritance, each entity is implicitly also an instance of all classes that are more general than an explicitly mentioned class. As an example, the explicit arc `hasType(Chopin,composer)` implies also an implicit arc `hasType(Chopin,person)`. Notice that an entity may be an instance of several different classes so that none of them is more general than another (e.g. `hasType(Chopin,composer)`, `hasType(Chopin,pianist)`).

Given two classes `A` and `B`, we say that `A` *is more specific than* `B` (or `B` is more general than `A`) if `subclassOf(A,B)` holds. We abbreviate this by $A < B$.

Table 1 summarizes the notations introduced in this section.

4 Aspect-based entity model

In this section, we introduce the notion of a *maximal aspect of an entity*, a natural concept that is key to our approach and, in addition, provides a natural diversity awareness of the results.

We first describe a 3-level aspect-based entity characterization model and define *basic*, *compound* and *maximal* aspects that are later used to precisely model the concept of entity similarity in our approach. The model is flexible and has many potential applications. It is derived purely from the structure of the knowledge graph and thus is orthogonal to other approaches to characterize entities in knowledge graphs (e.g. text-based).

4.1 Aspects of an entity

Let us now introduce the concept of an *aspect* with an example. Given an entity $q \in E$ such as `Chopin` (a famous Polish composer), consider all arcs that are incident with q in KG. These arcs can either represent a type of the entity q (e.g. `hasType(Chopin,composer)`) or facts concerning q (e.g. `bornIn(Chopin,Poland)`). 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 an arc with a variable represented by '.' we obtain a *logical predicate* with one free variable that represents a *basic aspect* of that entity. For example, a factual arc `bornIn(Chopin,Poland)` and a type arc `hasType(Chopin,composer)` naturally induce predicates of the form

Table 1 Denotations

FG	$= (E, F)$ fact graph
E	the set of entities (i.e. nodes in the fact graph FG)
F	factual arcs (i.e. arcs in FG)
O	$= (C, S)$ ontology tree
C	the set of classes (i.e. nodes in the ontology tree O)
S	"subclass" arcs (i.e. arcs in O)
TA	type assignment ("hastype") arcs (connecting FG with O)
R	the set of all relations (arc labels) in KG
V	$= E \cup C$, the set of nodes in KG
W	$= F \cup S \cup TA$, the set of edges in KG
KG	$= FG + O + TA = (V, W)$ the whole knowledge graph

`bornIn(.,Poland)` and `hasType(.,composer)` that represent the "basic properties" of this entity of "being born in Poland" and "being a composer", respectively. We call such logical predicates *factual aspect* and *type aspect* of the entity, respectively. The aspects of an entity includes all type aspects, even if some types of the entity are more general than others. In the example, we would consider a type arc of the form `hasType`Chopinperson even though $person$ is a more general type than $composer$.

In addition we also consider *relational aspects* that capture the information which relations an entity is involved with (e.g. whether a person appeared in a movie). We represent this information by replacing the remaining argument of a factual aspect by a free variable ? such that `actedIn(.,?)` indicates that an entity acted in at least one movie. Precisely, we define `relation(.,?)` by $\exists_{y \in E}$ s.t. `relation(.,y)` $\in F$ holds. Thus, `bornIn(.,?)` represents the aspect of "being born somewhere". Notice that it could be considered both as a relaxation of a factual aspect and restriction of a type aspect (since, for example, only entities of type `person` can be born somewhere).

4.2 3-level aspect-based entity characterization model

For an entity $e \in E$ we consider three types of aspects that form a 3-level entity characterization model:

1. **type aspects** (level 1) of the form `hasType(.,class)` where the set of all type aspects of an entity e reflects the set of all types types $C(e) \subseteq C$ that e is an instance of: $C(e) = \{c \in C | hasType(e, c) \in TA\}$.
2. **relational aspects** (level 2) of the form `actedIn(.,?)` reflecting the multi-set of relations from R that appear in F with e.
3. **factual aspects** (level 3) of the form `r(.,n)` or `r(n,.)`, where $r \in R$ and n is an entity s.t. $r(e, n) \in F$ or $r(n, e) \in F$, representing the set $F(e) \subseteq F$ of facts involving e, i.e. $F(e) = \{r(arg1, arg2) \in F | arg1 = e \vee arg2 = e\}$ (e.g., for `Chopin`, `bornIn(.,Poland)`).

Notice that these three levels of entity characterization form a natural hierarchy, for two reasons. Firstly, each next level of characterization provides information concerning the entity that is in some sense more specific than the previous. Thus, the type aspects on level 1 provide the most general information and restrict the possible types of relations that may be incident to the entity (level 2) and the particular choice of incident relations puts constraints on possible adjacent entities (level 3). E.g the level-2 information `actedIn(.,?)` ("the entity acted in any movie") can be further refined on the level 3 as `actedIn(.,Casablanca)` (acted in the specific movie "Casablanca").

Secondly, each next level concerns information that is topologically "further away" from the entity in the knowledge graph, starting with the information that concerns directly the entity itself (type) to incident relations to adjacent nodes.

Obviously, this model could be further extended for higher number of levels (including arcs and nodes in KG that are increasingly more topologically distant from the entity under study). However, we argue that the most important characteristics concerning the entity are contained in these three levels.

We name the union of the three kinds of aspects for an entity q as the set of its *basic aspects* and denote it as $\mathcal{A}(q)$.

A *compound aspect* $A \subseteq \mathcal{A}(q)$ of entity q is any set of basic aspects of q. For example, for two basic aspects $a_1 =$ `bornIn(.,Poland)`, $a_2 =$ `hasType(.,composer)` $\in$

$\mathcal{A}(q)$ the set $A = \{a_1, a_2\}$ represents the compound aspect of "being a composer born in Poland".

4.3 Entity set of an aspect

For each basic aspect a of some entity q, we naturally define its *entity set* $E(a)$ as the set of all entities $e \in E$ that share this aspect, i.e. contain a in their set of basic aspects $\mathcal{A}(e)$. For example, for the entity $q =$`Chopin` and its basic aspects `hasType(.,composer)`, `bornIn(.,?)` and `bornIn(.,Poland)` their entity sets consist of all entities that are composers, born somewhere and born in Poland, respectively.

We naturally extend the above definition of entity set $E(a)$ (for a basic aspect a) to the concept of entity set $E(A)$ of a compound aspect A as the set of all entities that share *all* basic aspects in A (in the former example: all the entities that are both composers and are born in Poland).

For any two compound aspects A and B where $A \subset B$, we say A *subsumes* B, as it is "more general", i.e. it is potentially shared by more entities (in particular by a superset of entities).

4.4 Maximal aspect of an entity

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

We say that a compound aspect $A \subseteq \mathcal{A}(q)$ of entity q is a *maximal aspect* of q if and only if it satisfies the following two conditions:

1. $E(A)$ 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).

Note that for efficiency reasons, we filter maximal aspects such that they include only most specific types; this does not change semantics (as adding or removing a more general type to an aspect A that already includes a specific type does not change $E(A)$), but improves efficiency of computations.

$M(q)$ denotes the *family of all maximal aspects of* q.

4.5 Maximal aspects for a set of entities

We naturally extend all the concepts related to aspects of a single entity $q \in E$ for the case of a *set of entities* $Q \subseteq E$. 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* $Q \subseteq E$, $M(Q)$ by 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 not in* Q.

More precisely, for a query entity set Q we say that a compound aspect $A_Q \subseteq \mathcal{A}(Q)$ is *a maximal aspect for* Q if and only if it satisfies the following two conditions:

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

Illustrating example Let us illustrate the concept of the *maximal aspect* with the following example. Assume an entity set $Q = \{$`Schwarzenegger`, `Stallone`$\}$ is given. Each of the entities is an American action movie actor, a director, and a bodybuilder, and Schwarzenegger is in addition a politician.

Assume that the set of all basic aspects $A(Q)$ shared by all entities in Q is $A(Q) = \{$`hasType(.,ActionMovieActor)`, `livesIn(.,USA)`, `hasType(.,MovieDirector)`, `hasType(.,Bodybuilder)`$\}$ (Stallone is not a politician).

Assume that there is no other entity in the knowledge base that is at the same time an American action movie actor, a director and a bodybuilder (except Schwarzenegger and Stallone).

Then, consider two compound aspects for Q:
$A_1 = \{$`hasType(.,ActionMovieActor)`, `livesIn(.,USA)`$\}$
$A_2 = \{$`hasType(.,ActionMovieActor)`, `livesIn(.,USA)`, `hasType(.,MovieDirector)`$\}$

Assume that $E(A_1) \setminus Q \neq \emptyset$ i.e. there are entities outside Q that share compound aspect A_1, e.g. Clint Eastwood; similarly with A_2. Thus, each of the two compound aspects A_1 and A_2 is shared by Clint Eastwood. In other words, the first condition of the definition of maximal aspects holds for both A_1 and A_2. But, since $A_1 \subseteq A_2$, A_1 is *not* a maximal aspect (since it can be extended to A_2, for example).

But, if we additionally assume that there are no other (than Schwarzenegger and Stallone) entities in the knowledge base that are American action movie actors, directors that are at the same time bodybuilders then the compound aspect A_2 *is a maximal aspect* for Q, because adding any more basic aspect from $\mathcal{A}(Q)$ to A_2 would make it being satisfied only by entities from Q (the second condition of the definition).

The main idea of our approach is the following. Given the query in the form of the example entity set Q, we compute the family of maximal aspects for Q. Then, for each maximal aspect, we return the entities that satisfy it as the candidate results.

The concept of maximal aspect, that is key to our approach, is designed so that it has the following properties:

– the entities satisfying a maximal aspect are maximally similar to the entities given as the example in terms of its structural properties in the semantic knowledge graph
– it is a minimum necessary possible relaxation of the set of all basic aspects $\mathcal{A}(Q)$
– in addition, the results generated by different maximal aspects are naturally *diversified* as is precisely explained in the Section 5.2

Intuitionally, each separate maximal aspect for Q represents a different potential user interest hidden behind the ambiguous query Q.

4.6 Type filtering

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. We will now explain how to include such type constraints into the aspect model.

First, we determine the set $\mathcal{T}$ of *typical types* allowed as types of returned entities, independent of the query. How exactly this set is determined is driven by application requirements. A natural approach is to restrict this set to types that are not too general; general types would not really help in finding relevant results. In our implementation, a type is general if it has at least 100,000 instances in the knowledge base; for the YAGO knowledge base used in the experiments, this excludes 45 types. The original QBEES paper (Metzger et al. 2013, 2014) used two different approaches, a manually defined cut of the type hierarchy and an approach that forbids both general types and YAGO types derived from Wikipedia categories. We will examine in the experimental evaluation which of the methods performs best; as it will turn out, forbidding general types is most effective for queries with just a single entity, whereas larger queries do not require strict type filtering.

Second, for a query Q, we need to compute the set of *typical types* $\mathcal{T}(Q)$ *for* Q. All result entities must belong to at least one of the types in $\mathcal{T}(Q)$, hence any type aspects not included in $\mathcal{T}(Q)$ are removed from the set $\mathcal{A}(Q)$ of basic aspects shared by the query entities. When computing $\mathcal{T}(Q)$, we first identify the set S of types that are shared by all entities in Q that are also in $\mathcal{T}$, i.e. we compute $S := \mathcal{T} \cap \bigcap_{q\in Q} C(q)$. Then the most specific types in S are identified by filtering out all classes that are a super class of another class in the intersection. This yields $\mathcal{T}(Q)$ as $\{s \in S | \nexists s' \in S subclassof(s', s) \in O\}$. As an example, consider the query $Q = \{$`Schwarzenegger`, `Stallone`$\}$, then the two entities might share the types `actor`, `AmericanActor` and `entity`, but $\mathcal{T}(Q)$ may only include `actor` (or `AmericanActor` in case it is also present in the set $\mathcal{T}$), thus, enforcing the resulting entities to be some kind of actor.

However, $\mathcal{T}(Q)$ as computed above may be empty, e.g. because the entities do not share any type within $\mathcal{T}$, most likely due to faulty or missing information in the knowledge graph or a fuzzy input. In this case, we compute $\mathcal{T}(Q)$ as $\mathcal{T} \cap \cup_{q\in Q} C(Q)$, i.e. we intersect $\mathcal{T}$ with the union of $C(q)$ over $q \in Q$. In such a case the corresponding basic aspects will also be added into $\mathcal{A}(Q)$, but their effect in some of the ranking approaches is reduced (see Section 5.5). As an example, consider the query $Q = \{$`Schwarzenegger`, `Terminator`$\}$ that combines an actor with a movie. As the only common type `entity` of the two query entities would usually be too general, the initial computation of $\mathcal{T}(Q)$ yields an empty set, and hence $\mathcal{T}(Q)$ would be set to include both `actor` and `movie`, provided both are included in $\mathcal{T}$.

Third, an important aspect is when to filter entities. Here, the most natural approach is clearly to filter entities before they should be returned to the user, i.e., at the very end of the processing. Unfortunately, this is not very efficient since many entities may be discarded. We therefore propose to filter out maximal aspects that do not include at least one typical type (or a more specific sub-type) as a basic aspect. This is more efficient since no results are discarded after they were generated, but it could be less effective since some aspects (and their entities) are excluded beforehand. We will examine the effects of late and early type filtering in the experimental evaluation.

5 Finding similar entities

Let us now consider a QBEES retrieval task: Given a set of *query entities* Q, we want to retrieve a ranked list of the k most similar entities $\mathcal{R}$, where similarity is primarily defined by aspects shared with the entities in Q.

5.1 Entity similarity based on maximal aspects

Our approach is based on the fundamental observation that if A is a maximal aspect of q, all entities $e \in E(A) \setminus \{q\}$ are "maximally" similar to q wrt to a specific set of basic aspects represented by A. Thus, for a given entity set Q, the most similar entities, in the aspect sense, can be found in entity sets of maximal aspects of Q.

5.2 Maximal aspects support diversity and intent-awareness

As we observed in Section 1, one of the main problems to be addressed in the context of suggesting or recommending a small set of items is the problem of the *unknown intent* of the user submitting an *ambiguous* query.

This is especially important in the case studied here, where the query does not contain any explicit query intent or user interest, since it is just a set of example entities without any keywords or text that might suggest the actual user intent.

Thus, given a query set Q it is not obvious which properties of the given query entities are important to the user and the system should carefully select the (low) number of returned entities in order to maximize the chances that the implicit user interest behind the query will be satisfied at least partially.

A typical solution in such cases is the *diversity-aware approach*, i.e. constructing the result set so that it is diverse, in our particular case, it should represent multiple possible interpretations of the implicit user interest.

In our approach, the user interests are naturally modeled by compound aspects of entities. For example, the user may be interested in "American actors that are also politicians" or "movie directors that are bodybuilders", etc. Thus, returning an entity from an entity set of any maximal aspect naturally covers some possible implicit user interest while keeping the result maximally similar to the query set.

In addition, having the diversity awareness in mind, the concept of maximal aspect is designed so that it not only expresses the closest similarity in aspect space but, importantly, naturally provides diversity awareness.

We present this in the form of the following theorem:

Theorem 1 *Let Q be a (query) set of entities and $A_Q \neq B_Q$ be two different (non-empty) maximal aspects of Q. Then, $E(A_Q)$ and $E(B_Q)$ do not share any entities, except those in Q (i.e. $(E(A_Q) \cap E(B_Q) \setminus Q) = \emptyset$).*

Proof Assume, $e \in E(A_Q) \cap E(B_Q)$ for some entity $e \notin Q$. This implies that e shares all the basic aspects from both A_Q and B_Q. Let us introduce denotation $C_Q = A_Q \cup B_Q$. Thus, e shares all basic aspects from C_Q what implies that $e \in E(C_Q)$. But, since A_Q and B_Q are different and non-empty, C_Q strictly contains both A_Q and B_Q what would contradict the maximality property of them. □

Due to the above theorem, the set of candidate similar entities to be returned is *partitioned* by the entity sets of maximal aspects.

In other words, each new maximal aspect considered when computing the result set brings at least one new entity to this result set (i.e. the entities are not repeated for different maximal aspects). Avoiding redundancy in the result entity set can be viewed as one of possible manifestations of diversity of the result set.

Equivalently, in our model, each entity in the result set represents exactly one maximal aspect of the query, which makes the interpretation of each resulting entity more clear.

Furthermore, selecting by the user an entity from the result set can serve as a kind of relevance feedback in a possible interactive extension of our model to make it possible to refine the results in an iterative way. Such an extension of our model deserves a separate publication and is preliminarily presented and discussed in (Sydow et al. 2016; Sobczak et al. 2015).

Notice that one of the main goals of diversity is to make it possible to refine the unknown user need represented by an ambiguous query. Thus, entity sets of maximal aspects of Q seem to be the right tool to guide the process of selecting a diverse set of entities that are maximally similar to the query set Q and make it possible to refine the underspecified user information need.

5.3 The algorithm

To solve the task, we select k entities with the following procedure (initially $\mathcal{R} = \emptyset$):

1. Compute the family of maximal aspects $M(Q)$ of Q.
2. Filter the maximal aspects by *type constraints*,
3. Rank the maximal aspects,
4. Pick the entity e not in $Q \cup \mathcal{R}$ with the largest popularity $pop(e)$ from the top aspect $A \in M(Q)$, add e to $\mathcal{R}$, and remove A if $E(A)$ does not include any entities not included in $\mathcal{R}$,
5. Repeat steps 3 and 4 until k entities are picked, i.e. $|\mathcal{R}| = k$, or no aspects are left.

Algorithm 1 The algorithm for finding similar entities

```
Input: $Q$: set of entities
Output: $\mathcal{R}$: result list of entities most similar to $Q$
 1: compute family of maximal aspects $M(Q)$ of $Q$
 2: Filter $M(Q)$ by type constraints
 3: $\mathcal{R} \leftarrow \emptyset$
 4: while $|\mathcal{R}| < k \wedge |M(Q)| > 0$ do
 5:     rank maximal aspects
 6:     $A \leftarrow$ top aspect in $M(Q)$
 7:     $e \leftarrow \arg\max\{pop(e) | e \in E(A) \setminus \{Q \cup \mathcal{R}\}$
 8:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{e\}$
 9:     if $E(A) \setminus (\mathcal{R} \cup Q) == \emptyset$ then $M(Q) \leftarrow M(Q) \setminus \{A\}$
10:     end if
11: end while
12: return $\mathcal{R}$
```

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: $\mathcal{A}(Q) = \cap_{q \in Q} \mathcal{A}(q)$. 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 Sections 4.6 and 7). From $\mathcal{A}(Q)$, we can compute the corresponding family of maximal aspects $M(Q)$. Each such maximal aspect $A \in M(Q)$ induces the set of entities $E(A)$ that share a maximal set of properties with *all* query entities, but there may be multiple such maximal aspects. We discuss our algorithm to identify maximal aspects in Section 5.4.

2. Type constraints 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 discussed how to compute $\mathcal{T}(Q)$ in Section 4.6.

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 `hasType(.,person)` and `hasBirthplace(.,Austria)` while another one might consist of `hasType(.,GovernorOfCalifornia)`. Hence, we rank the maximal aspects (see Section 5.5). Among other factors, this ranking takes into account the entity sets of the maximal aspects aspects, but only the entities that have not yet been retrieved as results. Thus the rank of an aspect can change whenever an entity from its entity set is picked in step 4, hence the aspects are re-ranked in every step.

Depending on the application scenario a ranking method might need to introduce some additional diversification, i.e. ensure entities are picked from different maximal aspects. In that case, the rank of an aspect may depend on all entities already retrieved as results, which is another reason why the aspects are re-ranked after each entity pick.

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. Thus, we model the importance or popularity $pop(e)$ for each entity e, that can be estimated in various ways, including the knowledge graph structure or click information, etc. (see Section 5.6). Once an entity has been picked from an aspect A's entity set $E(A)$, we check whether the aspect can contribute more entities to the result $\mathcal{R}$. If $E(A) \setminus (\mathcal{R} \cup Q) = \emptyset$, we call the aspect *empty with respect to* $\mathcal{R}$ and remove it; we will simply say that an aspect is *empty* when it is clear to which set of entities we refer.

5.4 Finding maximal aspects

In order to identify maximal aspects, we need to consider all aspects that are subsets of $\mathcal{A}(Q)$ and decide whether they satisfy the two conditions defining a maximal aspect. We start by computing the set $I_{\mathcal{A}(Q)}$ of aspects that satisfies the first condition: for each aspect $A \in I_{\mathcal{A}(Q)}$ it holds: $A \subseteq \mathcal{A}(Q) \wedge (E(A) \setminus Q) \neq \emptyset$.

Let us assume $n = |\mathcal{A}(Q)|$ and that there are at most k entities in E. Now, for each basic aspect $a \in \mathcal{A}(Q)$ we compute its entity set $E(a)$, which we can do in $O(n \cdot k)$. Then we consider the union $U(\mathcal{A}(Q))$ of all entity sets of all basic aspects $a \in \mathcal{A}(Q)$, i.e. $U(\mathcal{A}(Q)) = \cup_{a \in A(Q)} E(a)$, 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. This can be done again in $O(n \cdot k)$ by maintaining $\mathcal{A}_Q(e)$ as a list or hash set for each entity and filling it up by iterating over the elements of $E(a)$ for all $a \in \mathcal{A}(Q)$.

Now, this gives us the set of aspects $I_A(Q) = \{A_Q(e) | e \in U(A(Q)) \setminus Q\}$ that are subsets of $\mathcal{A}(Q)$ and whose entity sets contain another entity than Q. To find the subset of maximal aspects, we need to check the second maximality condition, i.e. maximality wrt inclusion. Hence, 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 $B \supset A$. If we assume $r = |I_{\mathcal{A}(Q)}|$, then this requires to check at most r aspects against r aspects ($O(r^2)$) and we remove at most r elements ($O(r)$). Thus the whole operation takes $O(r^2 + r)$, and in total with the first condition this makes $O(n \cdot k + r^2 + r)$. While r is theoretically bound by k as well as 2^n, in practice the knowledge graph is not that dense, such that the real values are much smaller. In fact, the crucial part is the first component ($O(n \cdot k)$), as the number of entities for some basic aspects can be relatively large. In particular, retrieval of the entity set for each basic aspect is crucial as this requires database access, which is why we cache entity sets in a most-recently-used cache.

Claim: This procedure provides us with the maximal aspects: $I_{\mathcal{A}(Q)} = M(Q)$. *Proof:* 1) For each $\mathcal{A}_Q(e) \in I_{\mathcal{A}(Q)}$ there is some $e \in E(A)$ that generated A with $e \notin Q$ by construction. 2) By the filtering of subsuming aspects, each aspect A left in $I_{\mathcal{A}(Q)}$ cannot be extended with other basic aspects without violating requirement 1), as this would mean there is a maximal aspect not in $I_{\mathcal{A}(Q)}$. 3) Completeness: There cannot be a maximal aspect which we do not generate as a candidate this way. 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 $B \subset \mathcal{A}(Q)$ (otherwise not an aspect of Q). Hence, $e \in U(\mathcal{A}(Q))$ and thus, e generates a candidate aspect $C := \mathcal{A}_Q(e)$. Then either

- $C = B$, then $B \in I_{\mathcal{A}(Q)}$,
- or $C \supset B$, then B is not a maximal aspect, as it subsumes C,
- $C \subset B$ cannot be, as $e \in B$, thus, e satisfies all basic aspects of B and hence $B \subseteq C$ 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; we will discuss in Section 7 how to extend this model such that we can also deal with this case.

5.5 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, a (basic or compound) aspect A that represents popular entities might be considered more useful for the average query than an aspect representing only entities that are hardly known. Given an estimator for the popularity of an entity, $pop(e)$ (see Section 5.6) , the popularity $pop(A)$ of aspect A can be estimated as the aggregated popularity of its entities, i.e., $pop(A) = \sum_{e \in E(A)} pop(e)$. It is clear that the larger the entity set, the larger the popularity usually will be. Instead of plain popularity, we further normalize by the number of entities, thus focusing on popularity density and giving higher weights to specific aspects. This yields our *simple popularity based ranker*:

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

Second, a compound aspect might be considered interesting if it represents a small number of entities, whereas most of its component aspects represent many entities. To formalize this, we first estimate the value $val(b)$ of a basic aspect b as a normalized version of the number of entities it represents, i.e., $val(b) = 1 - \frac{1}{|E(b)|}$ (which is close to 1 for aspects with many entities). Based on this we can estimate the value of a compound aspect A by the sum of the values of its components, i.e., $val(A) = \sum_{a \in A} val(a)$. We normalize this as follows

$$nval(A) = \frac{val(A)}{\sum_{B \in M(Q)} val(B)} \tag{2}$$

This alone will prefer aspects with many general properties. To give more weight to aspects that represent a rare combination of basic aspects, we combine the normalized value with the size of the aspect, yielding

$$cost(A) := \frac{1}{|E(A)|} \times nval(A) \tag{3}$$

Third, we consider how much of Q's basic aspects an aspect covers, giving more weight to compound aspects that combine many basic aspects. Here, we weigh a basic aspect a by its selectivity $sel(a) = \frac{1}{|E(a)|}$, i.e., we give high weight to basic aspects with few entities. We then consider the ratio $ratio(A, B)$ between two (compound) aspects A, B where $A \subseteq B$, which is defined as follows:

$$ratio(A, B) = \frac{\sum_{a \in A \cap B} sel(a)}{\sum_{b \in A \cup B} sel(b)} \overset{A \subseteq B}{=} \frac{\sum_{a \in A} sel(a)}{\sum_{b \in B} sel(b)} \tag{4}$$

This yields 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 ($dist$).

$$\begin{aligned} dist(A) &:= ratio(A, \mathcal{A}(Q)) \times pop(A) \\ distp(A) &:= ratio(A, \mathcal{A}(Q)) \end{aligned} \tag{5}$$

As shown in Section 7, in some cases we allow aspects to be included in $\mathcal{A}(Q)$ even if they are not shared by all entities in Q. For each such basic aspect b the selectivity $sel(b)$ is weighed by the ratio of entities in Q that share the aspect b, i.e. $sel(b) = \frac{1}{|E(b)|} \frac{|\{q \in Q | b \in A(q)\}|}{|Q|}$. The same holds for $val(b)$ analogously.

5.6 Entity popularity

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 $\sum_{e \in E} pop(e) = 1$. 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 (Brin and Page 1998) 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, i.e., a graph that has the same vertices as KG and an edge set W' that contains an edge $\{u, v\}$ whenever an arc from u to v or from v to u exists in W. We chose this undirected representation since each arc in the knowledge graph has a corresponding natural inverse arc (e.g., a `hasBirthplace` arc has the natural inverse `isBirthplaceOf`) that may not be explicitly modeled in KG. The recursive definition of our graph-based popularity measure $pop_G(e)$ is then

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

where $deg(f)$ is the degree of vertex f. We iteratively solve this equation through a power iteration implemented in Apache Giraph,[2] setting $\epsilon = 0.15$.

For the YAGO knowledge base used in our experiments, we can exploit that it was created from Wikipedia and a mapping from YAGO entities to Wikipedia articles is straightforward. For Wikipedia, click frequencies for each page are available,[3] 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

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

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 chemist, 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. This is driven by the assumption that a user might rather have aspects in mind that contain on average more well-known entities.

6 Relaxing aspects

Our algorithm from Section 5.1 removes a maximal aspect as soon as all its entities are picked. This is desired when we aim at retrieving only the most similar entities from each 'similarity branch' represented by the different maximal aspects. However, if only the now

[2] http://giraph.apache.org/

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

empty maximal aspect reflects the user's general information need, entities from any other maximal aspect may be non-relevant, and if we want to retrieve all relevant entities we might need to explore this branch further. As an example, consider a now empty maximal aspect $A_1 = \{$`hasType(.,actor)`, `bornIn(.,Austria)`, `actedIn(.,The Terminator)`$\}$ corresponding to "Actors born in Austria that starred in Terminator". If the user is interested in all actors that appeared in Terminator, we will potentially miss all non-Austrian actors. Thus, we want to allow a *relaxation* of the now empty maximal aspect's constraints to cover more entities of the same branch of similar entities.

A *first approach towards relaxation* is removing one or more aspects from an empty maximal aspect, until the resulting set(s) of aspects are maximal again, now in a slightly modified sense. We extend the notion of maximal aspects $M(Q)$ to $M(B, Q')$ allowing to restrict the set of basic aspects that can be included in a maximal aspect as follows. Given a set of basic aspects B and a set of entities Q', we define the set $M(B, Q')$ of maximal aspects constructed from the basic aspects in B relative to the entities in Q' such that for any aspect $A \in M(B, Q')$ it holds: 1) $E(A)$ contains at least one entity not in Q' 2) A is maximal wrt inclusion 3) $A \subseteq B$. With this notation, the set of maximal aspects for a set Q of entities as introduced in Section 4 can now be written as $M(\mathcal{A}(Q), Q)$.

To relax an empty maximal aspect A we take the original query entities Q and all entities in $\mathcal{R}$ (which are already picked as result) into account and compute $M(A, Q \cup \mathcal{R})$, which can be done with the algorithm from Section 5.4. This yields 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 type constraints. We call this approach `recursive plain` or short `rec`. Note that at any time all aspects considered are maximal with respect to $Q \cup \mathcal{R}$.

A *second approach towards relaxation* exploits any type aspects that may be included in the empty maximal aspect. As we discussed in Section 4.4, we include only the most specific common types in a maximal aspect. Relaxation is now an opportunity to replace a type with a more general type, according to the type hierarchy S. As an example, consider the case where A represents "Austrian actors that appeared in Terminator", i.e., $A = \{$`hasType(.,AustrianActor)`, `actedIn(.,Terminator)`$\}$. Using the approach explained before, we could only find $\{$`hasType(.,AustrianActor)`$\}$ or $\{$`actedIn(.,Terminator)`$\}$ as new maximal aspects. We now exploit the fact that `AustrianActor` is a subclass of `actor` and relax `hasType(.,AustrianActor)` to `hasType(.,actor)`. This yields the new maximal aspect $\{$`hasType(.,actor)`, `actedIn(.,Terminator)`$\}$. Formally, to relax an empty aspect A including type relaxation, we first extract all type aspects $T \subset A$, then consider all aspects T' with more general types according to S. The new set of maximal aspects to replace A is then computed as $M(A \cup T', Q \cup \mathcal{R})$. We refer to this relaxation strategy as `recursive full` or short `recf`. While this approach helps to increase the overall recall, it is also quite costly since a large number of potential maximal aspects must be checked.

In an attempt to include type relaxation while keeping the runtime cost of relaxing a maximal 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. This *one-step* approach relaxes an aspect A by individually considering each basic aspect $a \in A$ and verifying if an aspect A' yielded by relaxing a is a maximal aspect with respect to $Q \cup \mathcal{R}$. A basic aspect a is either relaxed by dropping it, or by replacing its type by a direct super type in case a is a type aspect. Note that if there are multiple direct super types for the type of a type aspect, we generate a relaxed aspect A' based on each such direct super type. Algorithm 2 summarizes this approach. We refer to this approach as `1-step`. Figure 1 illustrates the above variants of relaxation.

Algorithm 2 One-step heuristic for aspect relaxation

```
Input: $A$: empty maximal aspect
Output: $\mathcal{M}$: set of maximal aspects after relaxing $A$
 1: $\mathcal{M} \leftarrow \emptyset$
 2: for $a \in A$ do
 3:     if $a$=type(.,t) $\wedge$ $\exists t'$ s.t. subclassOf(t,t')$\in S$ then
 4:         choose most specific $t'$ s.t. subclassOf(t,t')$\in S$
 5:         $tmp \leftarrow A \setminus \{a\} \cup \{$type(.,t')$\}$
 6:     else
 7:         $tmp \leftarrow A \setminus \{a\}$
 8:     end if
 9:     if $tmp$ maximal aspect then
10:         $\mathcal{M} \leftarrow \mathcal{M} \cup \{tmp\}$
11:     end if
12: end for
```

7 Dealing with incomplete knowledge bases and inconsistent examples

In the discussion so far, we always assumed that the underlying knowledge base was complete, i.e., that it represented all important facts for an entity needed for finding similar entities. In practice, this is often too optimistic. Recent facts about entities (such as recent movies of an actor) may be missing, old information (such as the position of a politician) may not be updated, and type assignments may be incomplete (such as animated cartoons being assigned to movies or cartoons, but not both) or even wrong (possibly caused by confusing an entity with another that has the same name). When facing such problems, the intersection of the basic aspects of hint entities will often include only very generic aspects,

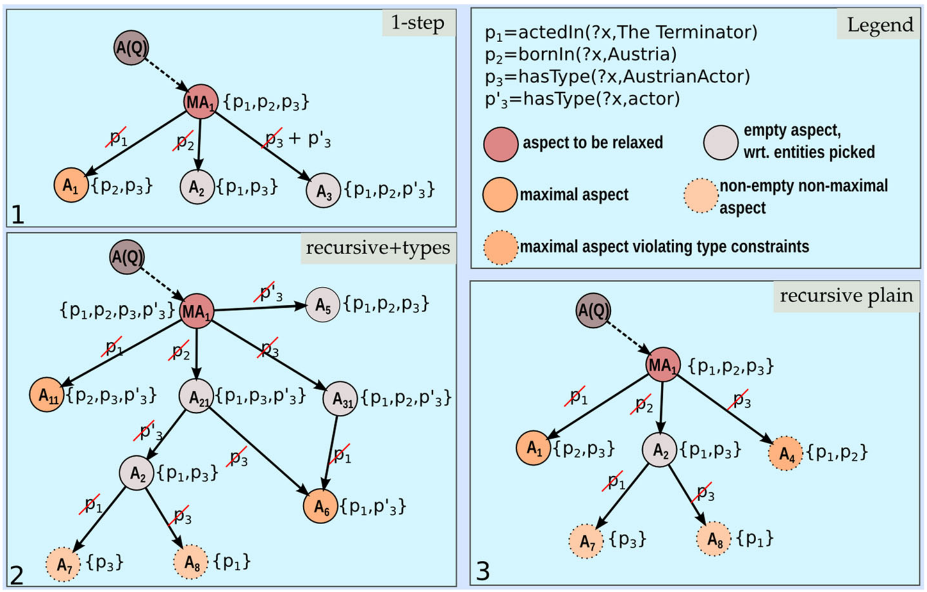

Fig. 1 Relaxing Aspects variations

e.g., `type(.,entity)` or `wasBornOn(.,?)` and thus will generate huge amounts of entities, most of which will be useless for the user.

A second source for such problems are inconsistent examples given by the user. Since the user is unaware how an entity is internally modeled, she may provide examples that are totally different in nature. For example, consider topic 67 from the INEX 2007 benchmark used for the evaluation in Section 8 that asks for "Ferris and observation wheels", including the Sky Dream in Fukuoka (modeled in the YAGO knowledge base as an attraction without any further non-type facts), the London Eye (modeled as a building, with facts about height and creation date), and the Wiener Riesenrad (also modeled as a building). If these three examples are given, they have nothing in common except the type "entity", even though they are very similar for a user, and our method will not retrieve any useful entity as the maximal aspect generates way too many entities to rank them in a useful way. Relaxation as explained in the previous section cannot help here since there is nothing to relax in a maximal aspect consisting only of the most general type. Formally, we say that a maximal aspect is **too large** if it generates too many entities, where the actual limit is a parameter that is set to 10,000 in our system.

We solve this problem by loosening the requirements for shared aspects if all maximal aspects produced by our method are too large. In this case, we also consider basic aspects that are not shared by all $|Q|$ example entities, but by $|Q|-1$, and recompute maximal aspects for this set of basic aspects. We keep lowering this *share threshold* until at least one maximal aspect is produced that is not too large; this is always the case when the share threshold reaches a value of 1 unless none of the example entities has at least one specific fact; in that rare case, it would be impossible to generate useful results anyway.

In the example above, we will ignore the maximal aspect constructed from all three examples, and will create maximal aspects based only on the London Eye and the Wiener Riesenrad, ignoring the Sky Dream. Note that, with examples that are not inconsistent, but for which the knowledge base is incomplete, it is possible that multiple maximal aspects are created based on basic aspects shared by different subsets of the examples' basic aspects. We can therefore deal with both incomplete knowledge bases and inconsistent examples. We will demonstrate the value of this in Section 8.2.6.

8 Evaluation

8.1 Setup

We use the "core" variant of the Wikipedia-based general knowledge base YAGO2 (Hoffart et al. 2013) in version 2.3.0 as the underlying knowledge graph for the following evaluation. We ignore the following meta-relations: `hasGloss`, `hasPreferredMeaning`, `hasPreferredName`, `hasWikipediaUrl`, `isCalled`, `means`, `subclassOf`, `subpropertyOf`.

Bron et al. (Bron et al. 2013) introduced three data sets, 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 data set consists of several topics, to which entities have been assigned with graded relevance assessments. In the same spirit, we created two new data sets based on the QALD (Question Answering over Linked Data) evaluation

Table 2 Number of topics and queries per Dataset

Dataset	#topics	#queries
inex2007	23	862
inex2008	48	1729
semsearch2011	39	1449
qald3	21	771
qald4	23	836

campaign,[4] more precisely the *dbpedia-train-answers* data set from the QALD 3 campaign and the *qald-4_multilingual_train_withanswers* data set from the QALD 4 campaign. Among the queries in the original data sets we used those whose answers contain multiple entities from DBpedia (i.e., those containing the properties `answertype="resource"`, `aggregation="false"`, `onlydbo="false"`). We transformed the filtered queries to the same format as queries from INEX. The questions in English were used as the topics (needed for Bron et al.) and the answers as gold standard.

We mapped the original entities (given as DBpedia URIs) to YAGO where possible and removed them otherwise, removing topics where no or only one relevant entity was left. We use the data set from INEX 2007 (inex2007) to analyse our model variations in Section 8.2 and the data sets from INEX 2008 (inex2008), the Semantic Search Workshop (semsearch2011), QALD3 and QALD4 as test data to compare our approach against a random walk with restart and the approach introduced by Bron et al. (Bron et al. 2013) in Sections 8.3.1 and 8.3.2. We randomly generate example entity queries of different query size, i.e. number of example entities, for each topic, where the examples are taken from the ground truth. 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 (see Table 2).

As evaluation measures, we use mean average-precision (map), mean normalized discounted cumulative gain (mndcg), precision, recall, and mean reciprocal rank (mrr) for rankings of length 10 and 100. Precision is the fraction of relevant entities in a ranking over the length of the ranking, recall the number of relevant entities in a ranking over all relevant entities to be retrieved. The average precision (ap) for a query is the sum over all precision values measured at ranks where a relevant entity has been returned divided by the number of relevant entities. The reciprocal rank (rr) is based on the first rank r that provides a relevant entity, based on this the reciprocal rank is $1/r$. 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 n, however, the contribution is discounted on a logarithmic scale ($\frac{rel_i}{log_2(i)}$). To compute the ndcg a ranking's dcg value is normalized by the dcg of a perfect ranking. The mean version of ap, ndcg and rr is computed by averaging over a set of topics (Manning et al. 2008); in our case, we average over all queries of a data set with the same number of example entities. The assessments for semsearch2011 are graded with values from 0 to 2, but we assume a relevance threshold of 1 for computing map, i.e. any assessment other than 0 was considered relevant. For all computations, the example entities

[4]http://qald.sebastianwalter.org/. The preprocessed datafiles used in our experiments are available on e-mail request.

were not retrieved by the algorithms and not considered for recall-based measures; as a consequence, results for different numbers of examples cannot be directly compared since the queries and the ground truth are different.

8.2 Model analysis

In this section we present the result of an analysis of several components of our model using the inex2007 data set.

8.2.1 Entity importance estimation

We first evaluate the impact of the two entity importance estimators from Section 5.6. As clearly visible in Fig. 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.

8.2.2 Type filtering

We now compare the effect of the the different variants for filtering results by type discussed in Section 4.6. The main alternatives are *when to filter* (`late`, i.e., filter on the level of results; `early`, i.e., filter on the level of maximal aspects; `none`, i.e., do not perform any type filtering) and *what to filter*. For early filtering, we consider the two variants used in the original QBEES paper, i.e., filtering out a manual set of types (`early+manual`) or generic types and types derived from Wikipedia categories (`early+gen+wiki`), filtering out general types with more than 100000 entities (`early+>100000`), and no type filtering, i.e., simply making sure that each maximal aspect includes at least one type aspect (`early+none`). For late filtering, we examine the two thresholds 10,000 and 100,000; a result entity is filtered out if its most specific type includes at least that many entities.

Figure 3 shows MAP and nDCG results for `dist`, results for other measures and other rankers are similar. It is evident that late filtering performs best for small numbers of examples, but it is also very inefficient since many potential results are filtered out after computing them. Early filtering is more efficient, and the different variants perform very

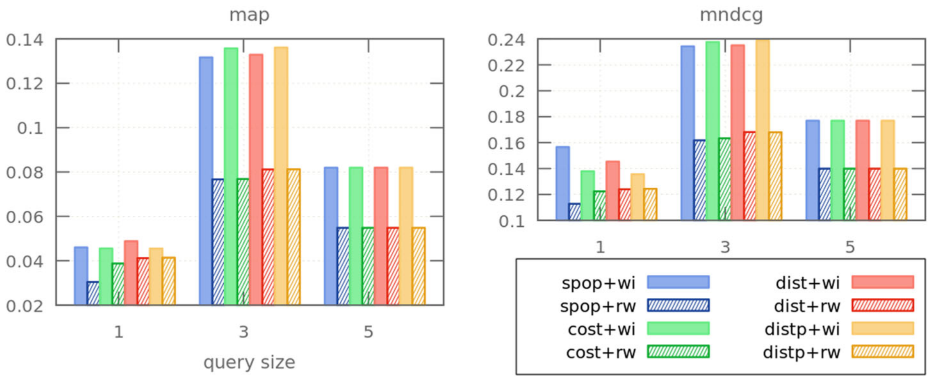

Fig. 2 Importance Estimators Top10 for INEX2007

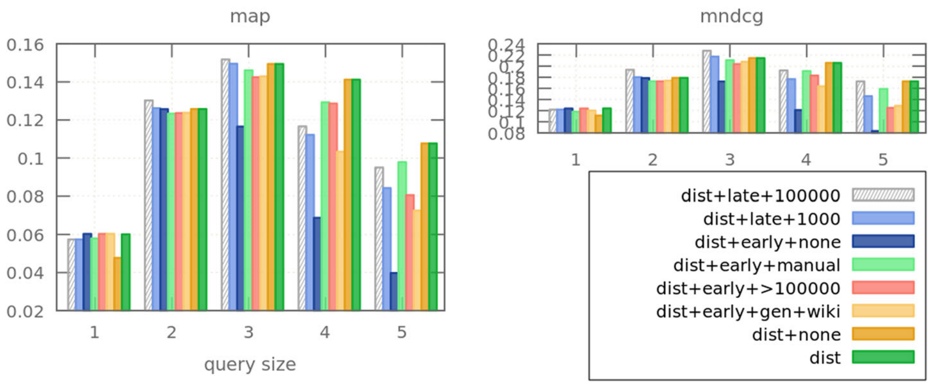

Fig. 3 Type Filtering Variants on Top100 for INEX2007

similarly. Interestingly, no type checking at all performs best when at least two examples are given. For the remainder of the experiments we thus use a hybrid type filtering approach that, for one-example queries, uses early filtering of general types, and does not perform any type filtering otherwise (indicated as `dist`).

8.2.3 *Type constraints*

As the original topics of the INEX data sets come with target categories for the entities based on Wikipedia categories, we automatically mapped these to YAGO Wikicategory classes where possible and used them as a constraint, comparing this to identifying typical types as constraints with our method from Section 4.6. In cases where the category mapping failed, we fall back to our type finding approach for determining constraint types.

Figure 4 compares the results of this topic category based approach(`+cats` versions) with our type finding approach(`+tf` versions) introduced in Section 4.6. The results show that our own method performs better than 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.

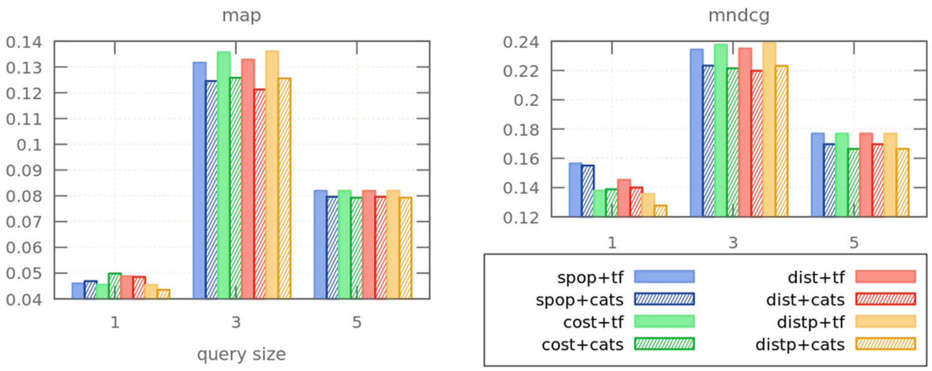

Fig. 4 Topic Constraints Top10 - INEX2007

8.2.4 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 Fig. 5) 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.

8.2.5 Relaxing aspects

In Section 6 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), the recursive approach with specific types (_rec) and the recursive approach using full type aspects (_recf).

Figures 6 and 7 provide the MAP and nDCG values for rankings of length 10 and 100, and Figs. 8 and 9 provide recall and average ranking length for rankings of length 10 and 100.

When looking at the MAP and nDCG values, it is clearly visible that compared to the non-relaxing variants, the relaxing approaches provide a better 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 8.2.7).

Considering the recall (see Figs. 8 and 9), 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 recursive relaxing approaches gain the most by additional

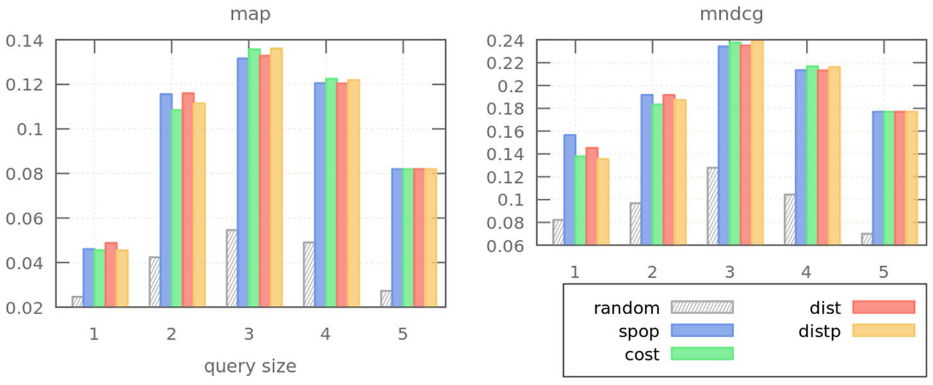

Fig. 5 Ranking Approaches vs Random Ranking Top-10: MAP (*left*), nDCG (*right*) for INEX2007

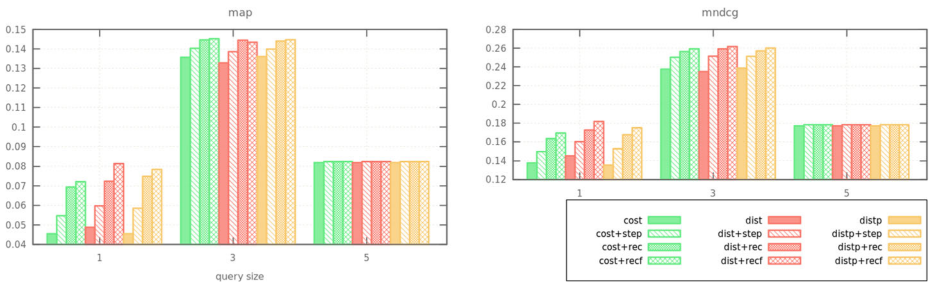

Fig. 6 Aspect relaxation variations Top-10: MAP and nDCG for INEX2007

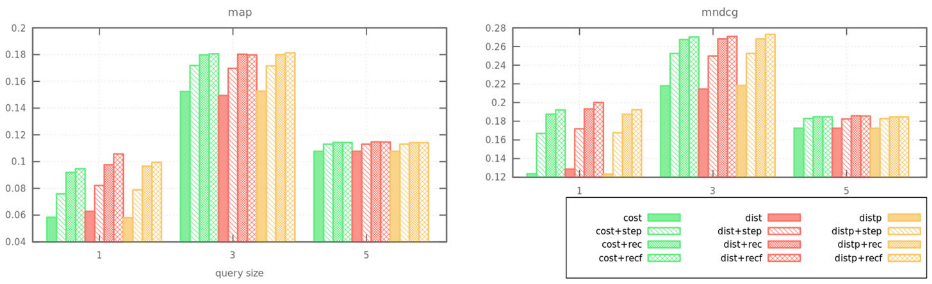

Fig. 7 Aspect relaxation variations Top-100: MAP and nDCG for INEX2007

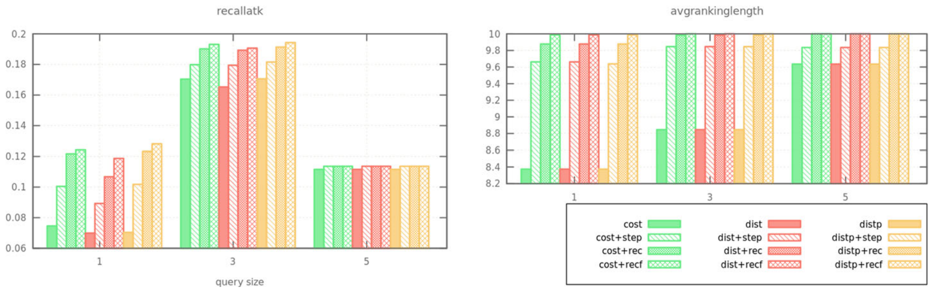

Fig. 8 Aspect relaxation variations Top-10: recall and average ranking length for INEX2007

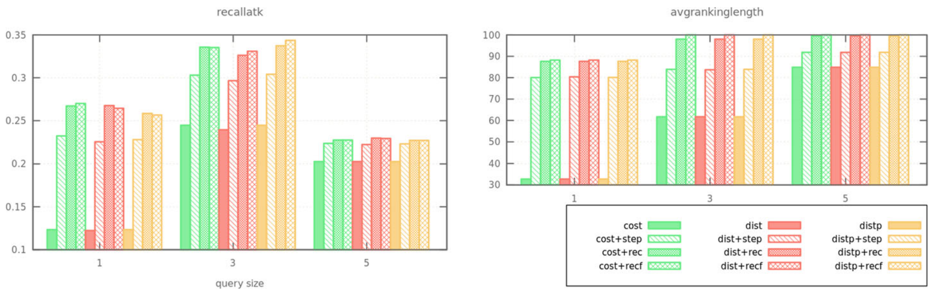

Fig. 9 Aspect relaxation variations Top-100: recall and average ranking length for INEX2007

relevant entities as indicated by Fig. 9. The main reason for this is that they retrieve more entities overall, shown by much larger average ranking lengths than the non-relaxing versions produce.

8.2.6 Impact of share thresholds

Share thresholds were introduced before to cope with inconsistent examples or missing information in the knowledge base, which can result in maximal aspects too large to be useful. Figures 10 and 11 show the impact of using share thresholds on MAP and nDCG on the inex2007 data set, with and without additional one-step relaxation. The runs with share thresholds are marked with `+st`.

It is evident from the figures that MAP and nDCG increase a lot for five example entities. At the same time, additional relaxation can further improve result quality for top-100 results; this additional benefit is enabled only by the smaller maximal aspects now generated. For fewer results, the effect is less pronounced, and for just one example, share thresholds cannot make a difference. Using recursive relaxation instead of one-step relaxation provides only minor additional improvements; we do not present these results for space reasons. Due to this big advantage, the combination of share thresholds and relaxation (usually one-step due to runtime advantages) will be the method of choice.

Table 3 summarizes the impact of share thresholds on the query execution. The table shows, for each number of examples, how many queries were evaluated with a given final share threshold. As an example, for queries with five examples, only for 74 of the 174 queries maximal aspects constructed from aspects shared by all five examples were used; for 24 of these queries, aspects shared by only two entities were finally used to generate results. The majority of these cases is rooted in the incompleteness of YAGO or inconsistent examples in the benchmarks (where some results labeled as relevant in the ground truth clearly do not match the intent of the topic, such as the entity “Giacomo Casanova” for the topic on movies about Venice); only a minority is caused by problems mapping entities from DBPedia or Wikipedia to YAGO. This clearly shows the need for loosening the strict requirement of aspects being shared by all examples.

8.2.7 Performance

Components We implemented our method in Java 8, storing the YAGO ontology in a PostgreSQL 9.5 database. To analyze the performance of our approach, we selected 200 queries from the INEX 2007 data set and processed them in random order to mimic a reasonably realistic workload. We ran every test 3 times and averaged the runtime measures over all 3 runs. All tests were run as a single thread on an Intel 6700K processor (4GHz) with 4 cores

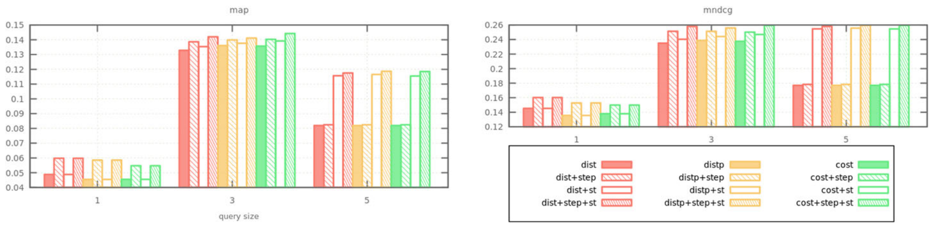

Fig. 10 Share Threshold, Top-10: MAP and nDCG for INEX2007

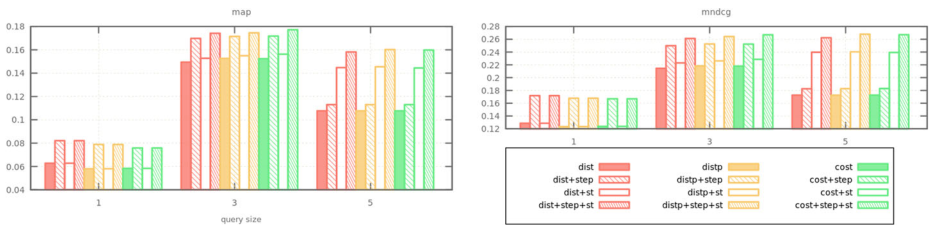

Fig. 11 Share Threshold, Top-100: MAP and nDCG for INEX2007

and 16 GB main memory. The data was served from a Postgres 9.5 database run on the same machine under Windows 10.

We first aim at identifying the critical components for runtime. 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 $I_{\mathcal{A}(Q)}$ 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 4 provides the averaged results. Note that for this experiment, every query is run with a cold cache, i.e. after each query the system is completely restarted. The variant used no relaxation and the `distp` ranking. 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.

It is evident that the most crucial aspect is the retrieval of data from the database dwarfing all computations. Hence, we make use of caching for the entity sets of basic properties in the following experiment, where at most 10G of the machine's main memory are used for caching.

Performance of rankers We now want to evaluate the average performance of the different rankers for the 200 queries. Caching is activated for this analysis, but the cache is empty at the very first query for each ranker 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. Figures 12 and 13 show the average time it takes to compute the top-10 and top-100 results, respectively, for a single query for different settings of our model: `cost` and `distp` are the different ranking approaches with all the default settings discussed before without relaxation; results for `dist` and `spop` are very similar. The `_1step` variants relax using the *one-step* heuristic, the `_rec` variants use recursive relaxation, the `_recf` variants use recursive full relaxation, the `_st` variants have share threshold

Table 3 Effective Share Thresholds used for INEX 2007 queries (top100 results)

# ex.	Thresh=1	Thresh=2	Thresh=3	Thresh=4	Thresh=5	Overall
1	163					163
2	67	134				201
3	20	18	150			188
4	3	45	22	100		170
5	0	24	24	18	74	140

Table 4 Runtime Contribution of components

Component	Average time spent
entity set retrieval	2.22 sec
computing aspect candidates	0.12 sec
maximal aspect computation	0.06 sec
ranking and selection	0.32 sec

enabled, and the `_st_1step` variants use one-step relaxation with share threshold enabled. The chart shows average time overall queries ('avg' on the left) and averages over queries with a certain number of examples ('avg-i').

The results show that, as long as we are not using recursive relaxation, our approach provides results relatively fast within a few seconds. Clearly the best performance show the variants that make use of share thresholds, where advantage is most obvious for larger numbers of examples. For these queries, maximal aspects without share thresholds are often very huge, hence much data must be loaded for ranking the entities in these aspects. With share thresholds, more, but smaller maximal aspects are generated, hence the time to rank the entities is smaller.

Our straight-forward implementation of the Bron et al. approach takes about 30 seconds per query when evaluated against precomputed per-term index lists loaded from disk for the content part of the score and loading entity information from a database for the structural part. 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. 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. 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.

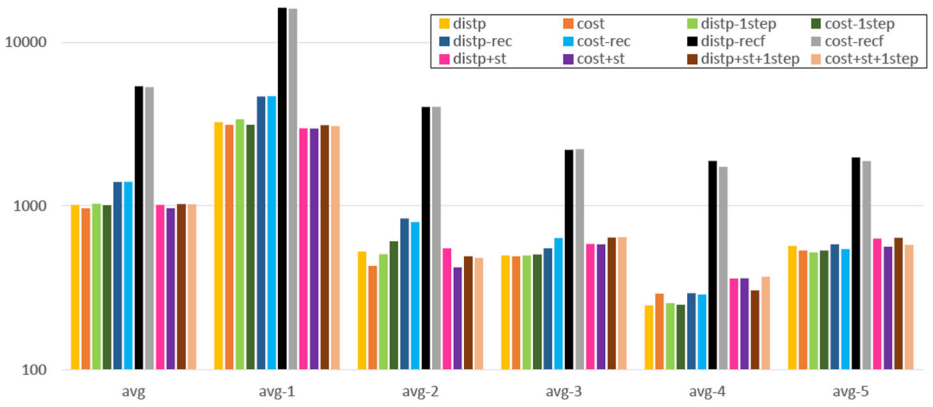

Fig. 12 Average runtimes for top-10 results over 200 queries

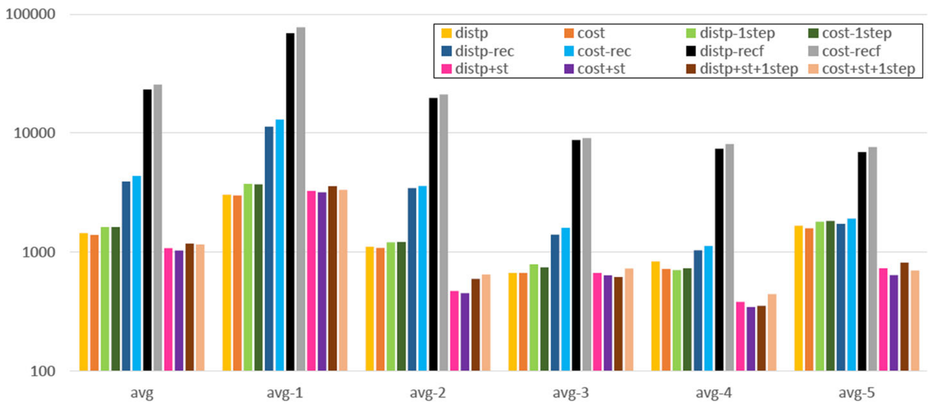

Fig. 13 Average runtimes for top-100 results over 200 queries

8.3 Comparison to competitors

8.3.1 Evaluation on INEX 2008 and SemSearch

We now compare result quality on the inex2008 and semsearch2011 data sets with different variants of our approach to two state-of-the-art competitors: (1) a random walk with restart (RWWR) 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 (Bron et al. 2013) (`bron-hybrid`), since this provides the best results combining a structural score based on the example entities and a textural score based on a short query description. Similar to the random walk used for estimating entity popularity in Section 5.6, in an RWWR a random surfer traverses the knowledge graph in random movements from entity to entity, starting at a randomly selected query entity. With a probability of β, the random surfer jumps directly to a query entity selected uniformly at random instead of following an edge. We computed an approximate solution of the recursive equation of the RWWR in an in-memory implementation with a random jump probability of $\beta = 0.15$ and stopping after 100 steps, ignoring all relations that include literal arguments. Note that for the RWWR we optionally apply a filter on resulting entities, either using the categories provided in the INEX data set for the topic where possible (':c' versions) or using our own typical type identification approach(':tf'). For the structural component of the Bron et al. approach we leave out `means` and `isCalled` facts and the 10 most generic types, all with more than 100,000 instances.

Based on our model analysis findings, we investigate two settings in the following. First, we consider an online, precision-oriented setting with a focus on top-10 results. Second, we consider an offline, recall-oriented setting where we consider the top-100 results. For both settings, we use the 1-step relaxation using only specific types and shared thresholds, since this is clearly the best compromise of result quality and runtime.

Figures 14 and 15 show the map and mndcg values for our approaches as well as the competitors on the inex2008 and semsearch2011 data sets for both ranking-length variants. As the results show, all our approaches behave similarly well, with `spop` lagging somewhat

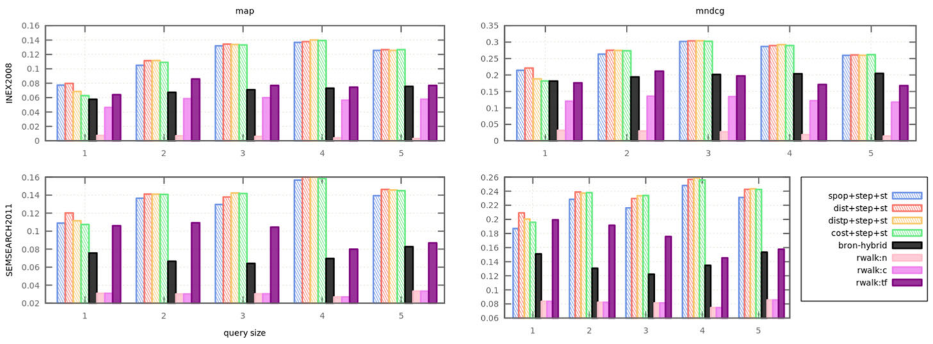

Fig. 14 Approach Comparison one-step relaxing versions Top-10

behind the other variants, in particular in the semsearch2011 data set, probably mislead by maximal aspects with popular entities not relevant for the query. However, as soon as at least two examples are given, our stronger variants outperform all competitors. 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. Note that not all topics have enough entities to produce queries for all query sizes, hence the topic base changes from size to size.

When we consider the offline setting with the longer rankings, we can see a similar picture, our stronger approaches outperform all competitors in most cases with a few exceptions where `bron-hybrid` or the random walk with type filtering reach the same quality level.

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 (Bron et al. 2013), this is not a contradiction. First, our queries differ as we generated them independently. Second, our underlying knowledge graph differs 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.

8.3.2 Evaluation on QALD datasets

We evaluated our approach and our competitors on the newly created datasets qald3 and qald4. The results are presented in Figs. 16 and 17.

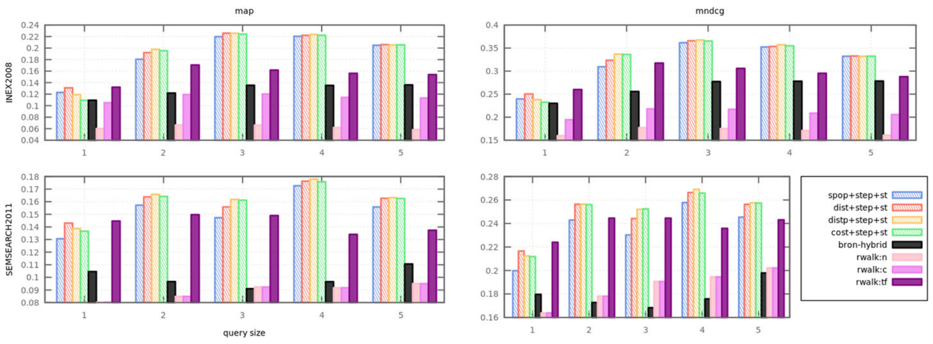

Fig. 15 Approach Comparison one-step relaxing versions Top-100

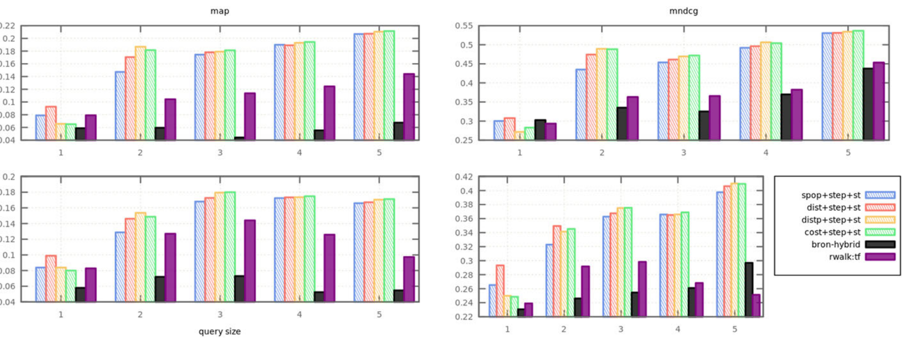

Fig. 16 Approach Comparison: MAP and nDCG for Top-10 on QALD datasets

The experimental results on the QALD data sets are essentially similar to the results achieved on inex2008 and semsearch2011, but absolute values for map and mndcg are much higher. This can be explained by the fact that QALD uses structural (SPARQL) queries to define the ground truth for the topics, and a maximal aspect essentially corresponds to a SPARQL star-shaped query. Our aspect-based method therefore can, for many queries, generate exactly the indented structural query representation. Another clear evidence for this is the fact that only about 25 % of all qald queries with five examples require a share threshold below five, i.e., their maximal aspects constructed from aspects shared by all examples are too large; for inex2007, 50 % of the queries with five examples showed this behavior. Our method does not work similarly well for all queries since some are too complex to represent as an intersection of basic aspects, for example queries including unions and linear queries with diameter of at least one. Extending our framework to also support these more complex structures is an interesting topic for future work.

Comparing to Bron et al., our methods are almost always clearly better, with the exception of top-100 for one-example queries on qald3; this means that our methods can usually better pick up the underlying structure of the results. Comparing to the random walk with restart combined with our type filter, the difference is smaller, but still our methods perform better (and, as an additional advantage shown before, are a lot faster than the random walk).

This advantage in terms of result quality directly translates into an advantage in terms of mean reciprocal rank (mrr), shown in Fig. 18. Independent of the number of example

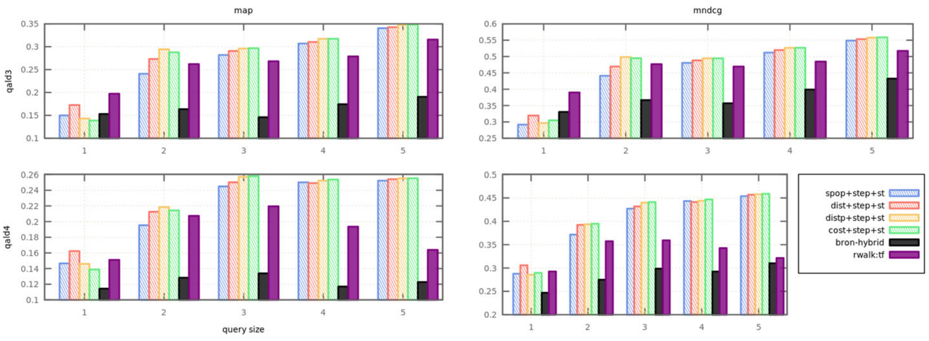

Fig. 17 Approach Comparison: MAP and nDCG for Top-100 on QALD datasets

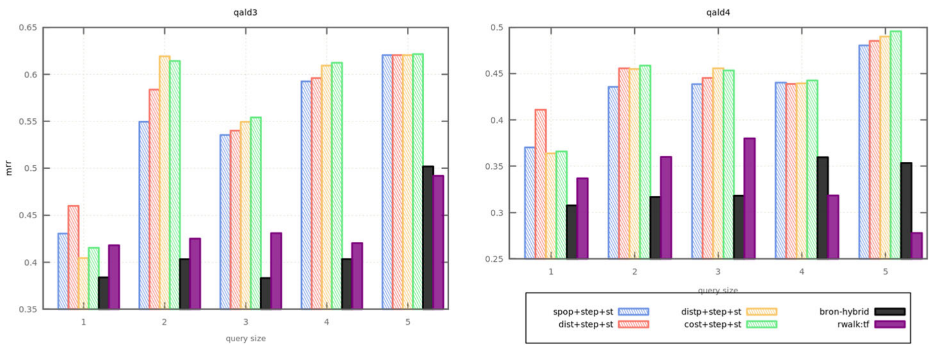

Fig. 18 Approach Comparison: mrr for Top-10 on qald3 and qald4

entities, our methods retrieve the first relevant result always earlier than any of the competitors. Our methods are therefore useful for a user who aims at finding quickly a relevant result.

9 Conclusion

In this paper we discussed our facet aware aspect based entity similarity model in detail and extended it by recursive relaxation of the aspects to expand the search space. We also discussed an approach for dealing with incomplete knowledge bases and inconsistent example entities. We evaluated various system settings in the context of set completion tasks. While our evaluation shows that our system outperforms state of the art models in terms of precision and recall based measures for several benchmark collections, we also showed that finding the right constraint types can be an important part of the problem.

However, there remain several interesting directions for future work. While our entity popularity estimators achieve good results, they are query-independent, and it would interesting to extend them towards 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 associated with the query entities than others.

The experiments on the QALD collections have shown that, while the model is already well performing in many cases, there are a number of interesting paths for making it more powerful, including support for more complex aspects that span multiple facts and disjunctive combinations of aspects.

Another interesting direction is to apply the aspect-based approach to model user interests in other entity-related applications on semantic knowledge graphs, including the entity summarization problem. There are other use cases, like interactive navigation, that might require amendments and offer additional opportunities. It would be also interesting to evaluate how well our approach adapts to other knowledge graphs with richer knowledge, and how schema information could be exploited.

Acknowledgments This work was partially supported by Polish National Science Centre grant "DISQUSS" 2012/07/B/ST6/01239. We would like to thank Grzegorz Sobczak for providing additional experimental evaluation results of QBEES on the QALD3 and QALD4 data sets, his work was supported by the European Union under the European Social Fund Project PO KL "Information technologies: Research and their interdisciplinary applications", Agreement UDA-POKL.04.01.01-00-051/10-00.

References

Agarwal, A., Chakrabarti, S., & Aggarwal, S. (2006). Learning to rank networked entities. In *KDD, pages 14–23*.

Agrawal, R., Gollapudi, S., Halverson, A., & Ieong, S. (2009). Diversifying search results. In *WSDM, pages 5–14*, New York, NY, USA. ACM.

Albertoni, R., & Martino, M.D. (2008). Asymmetric and context-dependent semantic similarity among ontology instances. *J. Data Semantics*, *10*, 1–30.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A nucleus for a web of open data Aberer, K., et al. (Eds.) In *The Semantic Web, volume 4825 of LNCS, pages 722–735*. Springer Berlin Heidelberg.

Balog, K., Bron, M., & de Rijke, M. (2011). Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, *29*(4), 22.

Balog, K., Serdyukov, P., & de Vries, A.P. (2011). Overview of the TREC 2011 entity track. In *TREC*.

Bollacker, K., Tufts, P., Pierce, T., & Cook, R. (2007). A platform for scalable, collaborative, structured information integration. In *In Intl. Workshop on Information Integration on the Web (IIWeb'07)*.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, *30*(1-7), 107–117.

Bron, M., Balog, K., & de Rijke, M. (2013). Example based entity search in the web of data. In *ECIR, pages 392–403*.

Demartini, G., Iofciu, T., & de Vries, A.P. (2009). Overview of the INEX 2009 entity ranking track. In *INEX, pages 392–403*.

Haveliwala, T.H. (2003). Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, *15*(4), 784–796.

Heath, T., & Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web*. Morgan & Claypool Publishers.

Hitzler, P., Kròtzsch, M., & Rudolph, S. (2010). *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press.

Hoffart, J., Suchanek, F.M., Berberich, K., & Weikum, G. (2013). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, *194*, 28–61.

Hose, K., & Vlachou, A. (2012). A survey of skyline processing in highly distributed environments. *The VLDB Journal*, *21*(3), 359–384.

Janowicz, K., Keßler, C., Schwarz, M., Wilkes, M., Panov, I., Espeter, M., & Bäumer, B. (2007). Algorithm, implementation and application of the SIM-DL similarity server. In *GeoS, pages 128–145*.

Jayaram, N., Gupta, M., Khan, A., Li, C., Yan, X., & Elmasri, R. (2014). GQBE: querying knowledge graphs by example entity tuples. In *ICDE, pages 1250–1253*.

Jeh, G., & Widom, J. (2002). SimRank: a measure of structural-context similarity. In *KDD, pages 538–543*.

Keßler, C., Raubal, M., & Janowicz, K. (2007). The effect of context on semantic similarity measurement. In *OTM Workshops (2), pages 1274–1284*.

Kosiński, W., Kuśmierczyk, T., Rembelski, P., & Sydow, M. (2013). Application of ant-colony optimisation to compute diversified entity summarisation on semantic knowledge graphs. In *FedCSIS, volume 1, pages 69–76*.

Manning, C.D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*, New York. Cambridge University Press.

Metzger, S., Elbassuoni, S., Hose, K., & Schenkel, R. (2011). S3K: Seeking Statement-Supporting top-K Witnesses. In *CIKM, pages 37–46*.

Metzger, S., Schenkel, R., & Sydow, M. (2013). QBEES: Query by entity examples. In *CIKM, pages 829–1832*.

Metzger, S., Schenkel, R., & Sydow, M. (2014). Aspect-based similar entity search in semantic knowledge graphs with diversity-awareness and relaxation. In. *Int. J. Conf. on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 60–69.

Minkov, E., & Cohen, W.W. (2010). Improving graph-walk-based similarity with reranking: Case studies for personal information management. *ACM Trans. Inf. Syst.*, *29*(1), 4.

Mottin, D., Lissandrini, M., Velegrakis, Y., & Palpanas, T. (2014). Exemplar queries: Give me an example of what you need. *PVLDB*, *7*(5), 365–376.

Mottin, D., Lissandrini, M., Velegrakis, Y., & Palpanas, T. (2014). Searching with XQ: the exemplar query search engine. In *SIGMOD, pages 901–904*.

Pirrò, G., & Euzenat, J. (2010). A feature and information theoretic framework for semantic similarity and relatedness. In *ISWC, (1) pages 615–630*.

Rodríguez, M.A., & Egenhofer, M.J. (2004). Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure. *Int. J. of Geographical Information Science*, *18*(3), 229–256.

Sobczak, G., Chochół, M., Schenkel, R., & Sydow, M. (2015). iqbees: Towards interactive semantic entity search based on maximal aspects. In *Foundations of Intelligent Systems - 22nd International Symposium, ISMIS 2015, Lyon, France, October 21-23, 2015, Proceedings, pages 259–264*.

Sydow, M., Pikuła, M., & Schenkel, R. (2013). The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *J. Intell. Inf. Syst.*, *41*(2), 109–149.

Sydow, M., Sobczak, G., Schenkel, R., & Mioduszewski, K. (2016). iQbees: Interactive query-by-example entity search in semantic knowledge graphs. In *Proc. of the 2016 Federated Conference on Computer Science and Information Systems Annals of Computer Science and Information Systems*, Vol. 8.

Tunkelang, D. (2009). *Faceted Search*. Synthesis Lectures on Information Concepts: Retrieval, and Services. Morgan & Claypool Publishers.

Tversky, A. (1977). Features of similarity. *Psychological Review*, *84*(2), 327–352.

Vee, E., Srivastava, U., Shanmugasundaram, J., Bhat, P., & Amer-Yahia, S. Efficient computation of diverse query results. In *ICDE, ICDE '08, pages 228–236, Washington, DC, USA, 2008. IEEE Computer Society*.

Wang, R.C., & Cohen, W.W. (2007). Language-independent set expansion of named entities using the web. In *ICDM, pages 342–350*.

Yu, X., Sun, Y., Norick, B., Mao, T., & Han, J. (2012). User guided entity similarity search using meta-path selection in heterogeneous information networks. In *CIKM, pages 2025–2029*.

Zhiltsov, N., Kotov, A., & Nikolaev, F. (2015). Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In *SIGIR, pages 253–262*.