QBEES: Query by Entity Examples

Steffen Metzger
Max Planck Institute for Informatics
Saarbrücken, Germany
smetzger@mpi-inf.mpg.de

Ralf Schenkel
Universität Passau
Passau, Germany
schenkel@ifis.uni-passau.de

Marcin Sydow
Polish-Japanese Institute of Information Technology and Institute of Computer Science, Polish Academy of Sciences
Warsaw, Poland
msyd@poljap.edu.pl

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 only on structural features, so-called aspects, of the entities, that 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.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

Keywords

list completion; entity search

1. INTRODUCTION

More and more data is available in semantic form, e.g., within the Linking Open Data cloud [8], product databases or common knowledge ontologies like DBpedia [2] or YAGO [9]. In consequence, information retrieval methods become more important to navigate the semantic data [12]. One typical IR task is the search for similar information pieces 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 explicitly expressed by average users, is vague in nature or unclear to the user herself, an implicit search interface is a natural user-friendly choice. Consider, for instance, the task to replace a particular worker of a company's workforce or searching for all possible replacements of a particular part in a production process. Instead of specifying all the relevant abilities of the worker or all the properties of the part, the natural choice would be to just provide the pointer, e.g. a URI, to its (already existing) semantic description. Other applications include general purpose entity search engines that provide similar entities given one or several examples. With such an engine, a user might look for movie directors that were also active actors, thus providing Quentin Tarantino and Clint Eastwood the user might expect to find Sylvester Stallone and Peter Jackson.

A central problem in this setting is the inherent ambiguity of examples. For instance, assume that a user provides "Arnold Schwarzenegger" as an example. The user's interest could be in other Austrian (ex-)body-builders, governors of California or actors that appeared in "The Expendables". In principle, a search by example in a semantic data-set can be considered a faceted search [12] 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 to the different possible facets of an example. In this paper we propose a model that captures all possible facets in so called aspects of the query. Thus our model can make the facet responsible for the inclusion of any returned entity transparent to the user.

The main *contributions* of this paper are 1) the introduction of an *aspect-based* entity model, 2) the presentation of various aspect-oriented ranking variations, and 3) a preliminary evaluation of the approach.

2. RELATED WORK

Entity search has been considered extensively in the literature, often with a focus on unstructured or semistructured data. The entity tracks at TREC [3] and INEX [6] introduced mainly two different retrieval tasks: finding related entities (with a textual description of the relationship and a target category), and entity list completion (with a list of example entities and a textual description). While the majority of test collections has been built based on unstructured text and semistructured XML documents, recent developments such as the Semantic Search Challenge[1] have extended this to semantic (RDF) data that forms a data graph with entities as nodes, the same scenario considered in this paper.

Core ingredients of many entity search systems are similarity measures for entities. A large body of work exists that exploits the graph structure for determining how sim-

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2507873.

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

ilar two entities are. One of the earliest approaches was SimRank [10] which considers two entities as similar if their context is similar. 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 [7], with a focus on relational data graphs [1, 11].

Another group of approaches uses features extracted from the context of entities to determine their similarity, including textual features (terms in the entity's uri or appearing in documents close to the entity) and structural features (categories or types of the entity). Balog et al. [4] propose to use language models that include terms and categories. Bron et al. [5], which is closest to our work, combines a term-based language model with a simple structural model including uniformly weighted facts about the entity. In contrast, our query model does not include a keyword component, our set of structural features in the aspects is more general, and our model allows to give different weights to different features. We experimentally compare our model to their structural model in Section 6.

Yu et al. [14] solve a slightly different problem where entities similar to a single query entity are computed, exploiting a small number of example results. Focusing on heterogeneous similarity aspects, they propose to use features based on so-called meta paths between entities and several path-based similarity measures, and apply learning-to-rank methods for which they require labelled test data. Wang and Cohen [13] 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.

3. KNOWLEDGE GRAPH

A Knowledge Graph (KG) consists of two basic components: A *Fact graph* FG and an ontology O.

The Fact graph FG is a directed multigraph where each node represents some *entity* (e.g. `Warsaw`, `Poland`). Each pair of nodes connected by a labelled arc represents an instance of a binary relation between two entities where the arc label represents the kind of relation (e.g. `isCapitalOf`), thus representing a *fact* about the entities (e.g. "Warsaw is the capital of Poland"). In the following we will use the notation `relation(arg1,arg2)` for any arc with label `relation` in KG that connects nodes `arg1` and `arg2`. In this notation the fact "Warsaw is the capital of Poland" is represented as `isCapitalOf(Warsaw,Poland)`.

Each node in the ontology tree O represents a class (type) of entities (e.g. `person` or `city`). The root of this tree represents the most general class of entities (e.g., `owl:Thing` or `wordnet_entity`) The nodes in the ontology tree are connected by directed arcs labelled as `subClassOf`.

The fact graph FG connects the entities to the ontology O: an arc of the form `hasType(anEntity,aClass)` represents the information that the entity `anEntity` is an instance of class `aClass`. Due to inheritance, each such entity is implicitly also an instance of all classes that are more general than the 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)`).

4. ASPECT MODEL OF ENTITIES

Given an entity q (e.g. `Chopin`), consider all arcs that are incident with q in KG. These arcs can either represent facts concerning q (e.g. `bornIn(Chopin,Poland)`) or a type of the entity q (e.g. `hasType(Chopin,composer)`). For any entity q (i.e. a node in FG), each such arc represents some "atomic property" of this entity (e.g. birthplace, type, occupation); the entity is characterised by the set of all "atomic properties".

By replacing the particular entity q in such an arc with a variable x we obtain a *logical predicate* with one free variable, e.g. a factual arc `bornIn(Chopin,Poland)` naturally *induces* a predicate of the form `bornIn(x,Poland)` that represents the "basic property" of this entity of "being born in Poland".

We call such a predicate a *basic aspect* of the entity. As a further relaxation, we also include basic aspects where only the label remains (e.g., `bornIn(x,y)`). Now, we define the *entity set of an aspect*. Each basic aspect a of entity q defines the set of all entities that share this aspect with entity q. For example, for the basic aspect `bornIn(x,Poland)` its entity set consists of all entities that are born in Poland. We call this the *entity set* of aspect a and denote it as $E(a)$.

Let the set of all basic aspects of an entity q be denoted as $\mathcal{A}(q)$. A *compound aspect* of entity q is any subset A of $\mathcal{A}(q)$. E.g. for two basic aspects $a_1 = \texttt{bornIn}(\texttt{x}, \texttt{Poland})$, $a_2 = \texttt{hasType}(\texttt{x}, \texttt{composer}) \in \mathcal{A}(q)$ the set $A = \{a_1, a_2\}$ represents a compound aspect of "being a composer born in Poland". We naturally extend the definition of entity set to compound aspects $E(A)$ as the set of all entities that share *all* basic aspects in A (in the former example: all the entities in KG that are both composers *and* are born in Poland).

4.1 Similarity by Maximal Aspects

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

We call a compound aspect A of entity q a *maximal aspect* of q iff it satisfies the two conditions:

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

Notice that if A is a maximal aspect of q, all enitities $e \in E(A) \setminus \{q\}$ are "maximally" similar with respect to a specific set of basic aspects to q. We denote the family of all maximal aspects of entity q as $M\mathcal{A}(q)$.

5. ASPECT BASED ENTITY RETRIEVAL

We now discuss how our aspect model is used to retrieve entities given example entities. Formally the task is defined as follows: Given a set of *query* entities Q as initial *hints*, we want to retrieve a set of entities that are similar, i.e. share some properties with the entities in Q.

Basically our general approach to select k entities consists of the following steps. 1. identify the family of maximal aspects $M\mathcal{A}(Q)$ of Q. 2. filter the maximal aspects by types typical for the entities in Q, 3. rank the maximal aspects, 4. pick the entity with largest popularity $pop(e)$ from the top aspect and update the aspect's rank, 5. redo step 4 until k entities are picked.

1. Maximal Aspects. Given a set of entities Q, first for each entity $q \in Q$ the set of its basic aspects $\mathcal{A}(q)$ is computed. Then the shared properties are identified by intersecting the aspect sets $\mathcal{A}(Q) = \cap_{q \in Q} \mathcal{A}(q)$. This provides the corresponding family of maximal aspects $M\mathcal{A}(Q)$. (For a set of entities Q, we extend the definition of a maximal aspect set such that the entity set of a maximal aspect must contain at least one entity not in Q.)

2. Typical Types. One of our basic assumptions is that the goal is to find other entities of relatively equal type. For instance, given a city, the output should be other cities and not, e.g., a country, because 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 descendant thereof) as basic aspect. Some details aside, $\mathcal{T}(Q)$ consists of all types shared by all entities in Q excluding some very general classes. If this yields an empty set, we also allow types that are shared by a majority of $q \in Q$.

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(x,person)` and `hasBirthplace(x,Austria)` while another one might consist of `hasType(x,GovernorOfCalifornia)`. Hence, in order to decide which aspect set is more likely to be of interest, we rank the maximal aspects (see Section 5.1).

4. Picking an entity. Similarly to aspects, the entities in the entity set of an aspect may have different likelihoods of importance to a user, especially for relatively broad aspects. We use two different entity importance measures that provide an estimated popularity $pop(e)$ for an entity e. First, we use the stationary probabilities of a random walk on KG. Alternatively, as a YAGO-specific method we estimate popularity based on the click count of the Wikipedia page corresponding to the entity.

5.1 Aspect Ranking

Given a set of query entities Q, we rank aspects in a language-model-style approach, i.e. each aspect A is ranked according to $P(A|Q)$, which we model as:

$$P(A|Q) \propto P(Q|A) \times P(A) \qquad (1)$$

where $P(Q|A)$ is the likelihood to generate the original query entities given the aspect A and $P(A)$ is the likelihood to pick A (from all maximal aspects). In order to estimate $P(Q|A)$ and $P(A)$ we employ different approximations that are combined in 4 rankers. These ranking approaches are based on the following components. Given an estimator for the popularity of an entity, $pop(e)$, the popularity of an aspect can be estimated as the aggregated popularity of its entities, i.e. $pop(A) = \sum_{e \in E(A)} pop(e)$ normalized by the sum over the popularity of all entities. A basic aspect b might be considered for its worth or likelihood to be generated $v(b)$. We estimate the value of a basic aspect by its selectivity, i.e. $v(b) = \frac{1}{|E(b)|}$ or its inverse $v^{-1}(b) = 1 - \frac{1}{|E(b)|}$. The value ratio $ratio(A, B)$ between two (compound) aspects A, B can then be computed as

$$ratio(A, B) = \frac{\sum_{a \in A} v(a)}{\sum_{b \in B} v(b)} \qquad (2)$$

Similarly the ‘cost ’of a compound aspect A can be computed by $cost(A) = \sum_{a \in A} v^{-1}(a)$ and normalized by the overall cost as follows

$$ncost(A) = \frac{cost(A)}{\sum_{B \in M\mathcal{A}(Q)} cost(B)} \qquad (3)$$

We now consider the following ranking functions:

spop $P(A|Q) \propto P(Q|A) \times P(A) = \frac{1}{|E(A)|} \times pop(A)$

cost $P(A|Q) \propto P(Q|A) \times P(A) = \frac{1}{|E(A)|} \times ncost(A)$

dist $P(A|Q) \propto P(Q|A) \times P(A) = ratio(A, \mathcal{A}(Q)) \times pop(A)$

distp $P(A|Q) = ratio(A, \mathcal{A}(Q))$

Note that we can easily introduce a diversity aspect into the ranking by remembering from which aspects entities have already been picked.

6. EVALUATION

Setup. We evaluate our model using the Wikipedia-based knowledge base YAGO [9]. For our preliminary evaluation, we adopt two datasets from [5] based on the INEX 2007 and INEX 2008 Entity Tracks, mapping entities to YAGO where possible and removing the others. For each topic in the datasets, we consider 1-5 example entities and randomly generate 10 distinct queries for each size as long as there remains at least one other relevant entity. For queries of size 1 to 3 in the inex2008 dataset we created up to 5 queries due to the long runtimes of the random walk algorithm used as a baseline. This results in 862 queries on 23 topics for the inex2007 dataset and 1244 queries on 48 topics in the inex2008 dataset.

We compute mean average-precision(map) and mean normalized discounted cumulative gain (mndcg) for rankings of length 100. While the assessments are graded values from 0 to 2, we considered any assessment other than 0 relevant.

Entity Importance Estimation. We first evaluate the effect of the two different entity importance estimators, random walk and Wikipedia click counts. As clearly visible in Figure 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 estimations in the following.

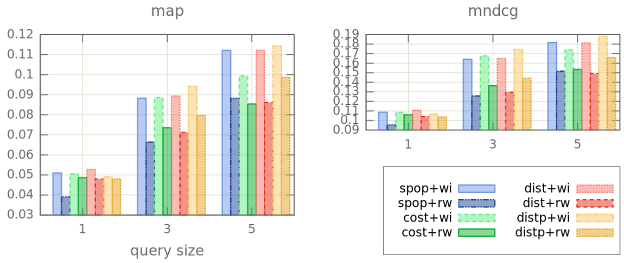

Figure 2: Importance Estimators - INEX2007

Type constraints. As the original topics of the INEX datasets come with Wikicategory based target categories

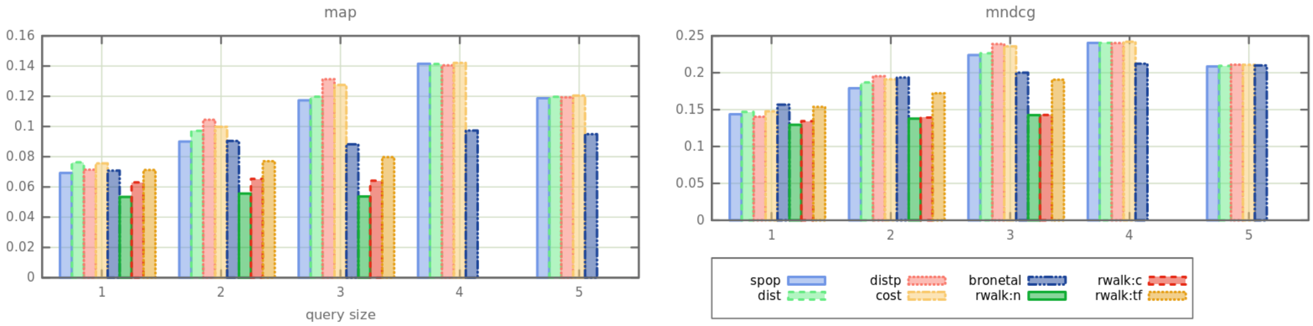

Figure 1: Approach Comparison - INEX2008

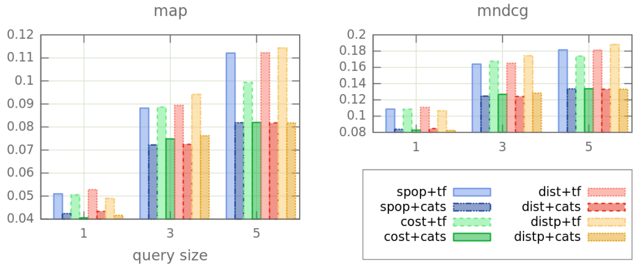

Figure 3: Topic Category Constraints - INEX2007

for the entities, we automatically mapped these to YAGO wikicategory-classes and used them as a constraint for suggested entities. Figure 3 shows that this does not work well, since the resulting classes are too over-fitting, i.e. unfortunately the YAGO data is quite incomplete in this perspective, such that the class constraints often filter out too many entities. Note that in cases where the automatic category mapping failed, we fell back to the default method (which is what we compare against).

Competitors. We now evaluate our approach with its different ranking approaches (`spop`, `cost`, `dist`, `distp`) against (1) a random walk with restart at the query nodes, (`rwalk:c`, `rwalk:tf`, `rwalk:n`) as a graph-based baseline, and (2) the structure-only approach suggested by Bron et al. in [5] (`bronetal`). Note that in (1) we optionally applied a filter on resulting entities, either using the categories provided in the INEX dataset for the topic where possible (':c' versions) or using our own typical type identification approach(':tf').

Results. Figure 1 shows the map and mndcg values for all approaches on the inex2008 dataset. Note that the random walk computation was so slow that we left it out for the higher query sizes. As the results show, the random walk benefits strongly from entity filtering (':c',':tf' versions vs ':n' version). Note that all our approaches behave similarly well. While for queries of size 1, our approach provides roughly the same quality as the approach suggested by Bron et al and the best random walk, for larger queries our aspect based approach outperforms both. The mndcg values indicate a quality dampening at query size 5, this is probably due to over-fitting and lower agreement for the typical types.

7. CONCLUSION

In this paper we presented a facet aware entity similarity model and evaluated its use for set completion tasks. While our preliminary evaluation shows that it can outperform state of the art structure-only models in several cases, the narrow focus on very specific similar entities can also be a drawback. In particular, future work will need to look into relaxing maximal aspects when they are too narrow and thus exclude other similar results that are not contained in a maximal aspect from the result ranking.

Acknowledgements: The work is supported by Polish National Science Centre grant DEC-2012/07/B/ST6/01239

8. REFERENCES

[1] Alekh Agarwal et al. Learning to rank networked entities. In *KDD*, pages 14–23, 2006.

[2] Sören Auer et al. DBpedia: A nucleus for a web of open data. In Karl Aberer et al., editors, *The Semantic Web*, volume 4825 of *LNCS*, pages 722–735. Springer Berlin Heidelberg, 2007.

[3] Krisztian Balog et al. Overview of the TREC 2011 entity track. In *TREC*, 2011.

[4] Krisztian Balog et al. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29(4):22, 2011.

[5] Marc Bron et al. Example based entity search in the web of data. In *ECIR*, pages 392–403, 2013.

[6] Gianluca Demartini et al. Overview of the INEX 2009 entity ranking track. In *INEX*, pages 254–264, 2009.

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

[8] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1(1):1–136, 2011.

[9] Johannes Hoffart et al. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194(0):28–61, 2013.

[10] Glen Jeh and Jennifer Widom. SimRank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[11] Einat Minkov and William W. Cohen. Improving graph-walk-based similarity with reranking: Case studies for personal information management. *ACM Trans. Inf. Syst.*, 29(1):4, 2010.

[12] Daniel Tunkelang. Faceted search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–80, 2009.

[13] Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *ICDM*, pages 342–350, 2007.

[14] Xiao Yu et al. User guided entity similarity search using meta-path selection in heterogeneous information networks. In *CIKM*, pages 2025–2029, 2012.