Information Retrieval Journal (2019) 22:285–323
https://doi.org/10.1007/s10791-018-9346-x

KNOWLEDGE GRAPHS AND SEMANTICS IN TEXT ANALYSIS AND RETRIEVAL

Identifying and exploiting target entity type information for ad hoc entity retrieval

Darío Garigliotti[1] · **Faegheh Hasibi**[2] · **Krisztian Balog**[1]

Received: 8 May 2018 / Accepted: 25 November 2018 / Published online: 5 December 2018

Abstract
Today, the practice of returning entities from a knowledge base in response to search queries has become widespread. One of the distinctive characteristics of entities is that they are typed, i.e., assigned to some hierarchically organized type system (type taxonomy). The primary objective of this paper is to gain a better understanding of how entity type information can be utilized in entity retrieval. We perform this investigation in two settings: firstly, in an idealized "oracle" setting, assuming that we know the distribution of target types of the relevant entities for a given query; and secondly, in a realistic scenario, where target entity types are identified automatically based on the keyword query. We perform a thorough analysis of three main aspects: (i) the choice of type taxonomy, (ii) the representation of hierarchical type information, and (iii) the combination of type-based and term-based similarity in the retrieval model. Using a standard entity search test collection based on DBpedia, we show that type information can significantly and substantially improve retrieval performance, yielding up to 67% relative improvement in terms of NDCG@10 over a strong text-only baseline in an oracle setting. We further show that using automatic target type detection, we can outperform the text-only baseline by 44% in terms of NDCG@10. This is as good as, and sometimes even better than, what is attainable by using explicit target type information provided by humans. These results indicate that identifying target entity types of queries is challenging even for humans and attests to the effectiveness of our proposed automatic approach.

Keywords Entity retrieval · Entity types · Semantic search · Query understanding

✉ Darío Garigliotti
dario.garigliotti@uis.no

Faegheh Hasibi
faegheh.hasibi@ntnu.no

Krisztian Balog
krisztian.balog@uis.no

[1] University of Stavanger, Stavanger, Norway

[2] Norwegian University of Science and Technology, Trondheim, Norway

1 Introduction

Entities, such as people, organizations, or locations are natural units for organizing information; they can provide not only more focused responses, but often immediate answers, to many search queries (Pound et al. 2010). Entities can improve the user experience throughout the entire search process, by enabling techniques of query assistance, content understanding, result presentation, and contextual recommendations (Balog 2018). Indeed, entities play a key role in transforming search engines into "answer engines" (Mika 2013). The pivotal component that sparked this evolution is the increased availability of structured data published in knowledge bases, such as DBpedia, Freebase, or the Google Knowledge Graph. Knowledge bases are now primary sources of information for entity-oriented search (Balog 2018). Major web search engines also shaped users' expectations about search applications; the single-search-box paradigm has become widespread, and ordinary users have little incentive (or knowledge) to formulate structured queries. The task we consider in this paper, referred to as *ad hoc entity retrieval* (Pound et al. 2010), corresponds to this setting: returning a ranked list of entities from a knowledge base in response to a keyword user query.

One of the unique characteristics of entity retrieval that distinguishes it from document retrieval is that entities are typed. *Entity types* (or *types* for short) are semantic categories that group together entities with similar properties. "An analogy can be made to object-oriented programming, whereby an entity of a type is like an instance of a class" (Balog 2018). Types are typically organized in a hierarchy, which we will refer to as *type taxonomy* hereinafter. Each entity in the knowledge base can be associated with (i.e., is an *instance of*) one or more types. For example, using the DBpedia Ontology, the type of the entity `Albert Einstein` is `Scientist`; according to Wikipedia's category system, that entity belongs to the types `Theoretical physicists` and `People with acquired Swiss citizenship`, among others. It is assumed that by identifying the types of entities sought by the query (*target types*, from now on), one can use this information to improve entity retrieval performance (Zhu et al. 2008; Demartini et al. 2010a; Pehcevski et al. 2010; Bron et al. 2010; Balog et al. 2011; Raviv et al. 2012; Kaptein and Kamps 2013); see Fig. 1 for an illustration. The main high-level

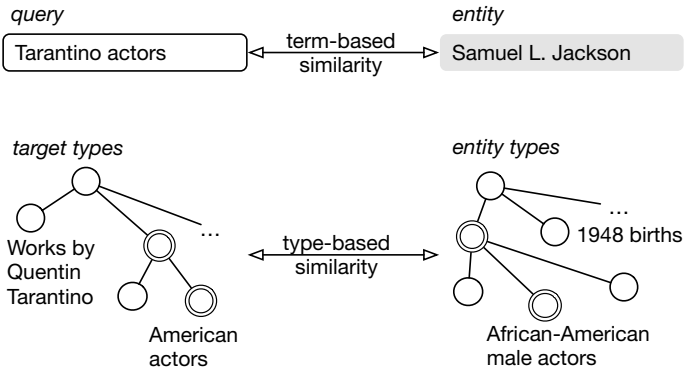

Fig. 1 Illustration of type-aware entity retrieval, where the target types sought by the query (left) are matched against the types that are assigned to the given entity in the knowledge base (right)

research question we are concerned with in this study is the following: *How can one exploit entity type information to improve ad hoc entity retrieval?*

The concept of entity types, while seemingly straightforward, turns out to be a multifaceted research problem that has not yet been thoroughly investigated in the literature. Most of the research related with the usage of type information has been conducted in the context of the INEX Entity Ranking track (Demartini et al. 2010b). There, it is assumed that the user complements the keyword query with one or more target types, using Wikipedia's category system as the type taxonomy. The focus has been on expanding the set of target types based on hierarchical relationships and dealing with the imperfections of the type system (Demartini et al. 2010a; Balog et al. 2011; Pehcevski et al. 2010; Kaptein and Kamps 2013). Importantly, these developments have been motivated and driven by the peculiarities of Wikipedia's category system. It is not known whether the same methods prove effective, and even if these issues persist at all, in case of other type taxonomies. One important contribution of this paper is that we consider and systematically compare multiple type taxonomies (DBpedia, Freebase, Wikipedia, and YAGO). Additionally, there is the issue of representing entity type information, more specifically, to what extent the hierarchy of the taxonomy should be preserved. Yet another question is how to combine type-based and text-based matching in the retrieval model. Therefore, the research questions we address are as follows:

- *RQ1* What is the impact of the particular choice of type taxonomy on entity retrieval performance?
- *RQ2* How can one represent hierarchical entity type information for entity retrieval?
- *RQ3* How can one combine term-based and type-based matching for entity retrieval?

To answer the above questions, we conduct a series of experiments for all possible combinations of three dimensions:

i. The way term-based and type-based information is combined in the retrieval model (Sect. 3).
ii. The hierarchical representation of entity type information (Sect. 4.1).
iii. The choice of the type taxonomy (Sect. 4.2).

Using a standard entity retrieval test collection (Hasibi et al. 2017), in Sect. 7 we perform a thorough experimental comparison and analysis of all possible configurations across the above identified three dimensions. Throughout this set of experiments, we make use of a so-called *target type oracle*. We assume that there is an "oracle" process in place that provides us with the correct target types for a given query. We employ this idealized setting to ensure that our results reflect the full potential of using type information, without being hindered by the imperfections of an automated type detector. We find that type information can yield up to 67% relative improvement in terms of NDCG@10 over a strong text-only baseline (cf. Sect. 7.3).

In a realistic setting, target entity types are not provided, but need to be automatically identified based on the keyword query. This gives rise to the following research objective:

- *RQ4* How can one automatically determine the target entity types of a query from a type taxonomy?

We cast the problem of *hierarchical target entity type identification* as a ranking task and present both unsupervised and supervised approaches in Sect. 5. Using a custom-built test collection, based on DBpedia's type system, we find that our supervised approach achieves an NDCG@5 score of 0.6, which represents a relative improvement of 58% over the best performing baseline (cf. Sect. 8.1).

Finally, we wish to test whether the findings we made using the oracle type detector also hold when automatic type detection is used instead. We ask the following question:

– *RQ5* How does type-aware entity retrieval perform using automatic target entity type identification, compared to an "oracle" setting?

We show that using automatic type identification, we can outperform the text-only baseline by 44% in terms of NDCG@10 (cf. Sect. 8.2.2). Interestingly, these results are as good as, and sometimes even higher than, what could be achieved by using explicit target type annotations by humans. This shows that identifying target entity types of queries is challenging even for humans and attests to the effectiveness of our proposed automatic approach.

In summary, our work is the first comprehensive study on the usage of entity type information for entity retrieval. This paper makes the following main contributions:

– Methods for (i) representing types in a hierarchy, (ii) establishing type-based similarity, and (iii) combining term-based and type-based similarities for ad hoc entity retrieval.
– A systematic comparison of four type taxonomies (DBpedia, Freebase, Wikipedia, and YAGO) across the above three dimensions of interest.
– Methods and a purpose-built test collection for automatic entity type identification.
– An experimental evaluation of automatic target entity type identification both intrinsically (in isolation) and extrinsically (on the ad hoc entity retrieval task).

All resources developed within this study (including relevance assessments, pre-computed features, and all the generated rankings) are made publicly available at https://github.com/iai-group/irj-types.

The remainder of this paper is organized as follows. In Sect. 2, we review related research. Section 3 introduces type-aware entity retrieval models. Next, in Sect. 4 we discuss alternative ways of representing hierarchical entity type information and present different type taxonomies. Section 6 describes our experimental setup. Experimental results are discussed in two parts; we first report results using an oracle setting in Sect. 7, then we employ automatic target entity type identification in Sect. 8. Finally, we conclude in Sect. 9.

2 Related work

The task of entity ranking has been studied in different flavors. *Ad hoc entity ranking* takes a keyword query as input and seeks relevant entities to be returned (Pound et al. 2010; Neumayer et al. 2012). *List search* further assumes that sought results are semantically related (e.g., "US presidents since 1960" or "Axis powers of World War II"); these semantic relationships may be specified with a set of target types, or a (small) set of example entities (Balog et al. 2012; Demartini et al. 2010b). *Related entity finding*, a special case of list search, requests result entities to be of a specific type and stand in a particular relation

with a given input entity (e.g., "airlines that currently use Boeing 747 planes") (Balog et al. 2010). Finally, answers to many questions in *question answering* are specific entities (e.g., "Who is the mayor of Berlin?") (Lopez et al. 2013). Our interest in this work lies in the usage of type information for general-purpose entity retrieval against a knowledge base, where queries may belong to any of the above categories.

2.1 Using type information in entity ranking

Early work represented type information as a separate field in a fielded entity model (Zhu et al. 2008). Demartini et al. (2010a) additionally expand type information using the underlying hierarchy. In later works, types are typically incorporated into the retrieval method by combining term-based similarity with a separate type-based similarity component. This combination may be done (i) using a linear interpolation (Balog et al. 2011; Kaptein and Kamps 2013; Pehcevski et al. 2010) or (ii) in a multiplicative manner, where the type-based component essentially serves as a filter (Bron et al. 2010). Raviv et al. (2012) introduce a particular version of interpolation using Markov Random Fields, linearly aggregating each of the scores for the joint distribution of the query with entity document, type, and name. All the mentioned works have consistently reported significant performance improvements when a type-based component is incorporated into the (term-based) retrieval model. However, type-aware approaches have not been systematically compared to date. We formalize these two general combination strategies, interpolation and filtering, in Sect. 3, and then compare them experimentally in Sect. 7.

Different approaches have measured the type-based similarity by using lexical type label similarity (Vercoustre et al. 2008), descriptions of entities (Kaptein and Kamps 2009), overlap ratio of type sets (Weerkamp et al. 2009), and even types added as a separated field in multi-field retrieval (Zhu et al. 2008; Demartini et al. 2008). In this work we use a state-of-the-art solution proposed by Balog et al. (2011) (cf. Sect. 3.3).

2.2 Type taxonomies

The choice of a particular type taxonomy is mainly motivated by the problem setting, depending on whether a wide-coverage type system (like Wikipedia categories) or a curated, well-designed ontology (e.g., the DBpedia Ontology) is desired. The most common type system used in prior work is Wikipedia categories (Demartini et al. 2010a; Balog et al. 2011; Kaptein and Kamps 2013; Raviv et al. 2012; Bron et al. 2010). This is in part for historical reasons, as this was the underlying type system used at the INEX Entity Ranking track, where type information was first exploited. Further choices include the DBpedia Ontology (Balog and Neumayer 2012; Tonon et al. 2013), YAGO types (Demartini et al. 2010a; Sawant and Chakrabarti 2013; Tonon et al. 2013; Nakashole et al. 2013), Freebase (Lin et al. 2012), and schema.org (Tonon et al. 2013). To the best of our knowledge, ours is the first study to compare different type taxonomies for entity retrieval.

2.3 Representations of type information

Target types are commonly considered either as a set (Pehcevski et al. 2010; Demartini et al. 2010a; Raviv et al. 2012; Kaptein and Kamps 2013) or as a bag (weighted set) (Vallet and Zaragoza 2008; Balog et al. 2011; Sawant and Chakrabarti 2013). Various ways of

measuring type-based similarity have been proposed (Vercoustre et al. 2008; Kaptein and Kamps 2009; Weerkamp et al. 2009; Zhu et al. 2008; Demartini et al. 2008). In this work we employ a probabilistic approach that represents entity type information as multinomial probability distributions (Balog et al. 2011) (cf. Sect. 3.3). Within a taxonomy, types are arranged in a hierarchy. (Wikipedia represents a special case here, as its categories do not form a well-defined "is-a" hierarchy.) Several approaches have attempted to expand the set of target types based on the hierarchical structure of the type system (Pehcevski et al. 2010; Balog et al. 2011; Bron et al. 2010; Demartini et al. 2010a; Balog and Neumayer 2012; Tonon et al. 2013). Crucially, the investigation of type hierarchies has been limited to Wikipedia, and, even there, mixed results are reported (Vercoustre et al. 2008; Zhu et al. 2008; Demartini et al. 2008; Jämsen et al. 2008). It remains an open question whether considering the hierarchical nature of types benefits retrieval performance. We aim to fill that gap.

2.4 Target entity type identification

The INEX Entity Ranking track (Demartini et al. 2010b) and the TREC Entity track (Balog et al. 2012) both featured scenarios where target types are provided by the user. When explicit target type information is lacking, one might attempt to infer types from the keyword query. This subtask was introduced by Vallet and Zaragoza (2008) as the *entity type ranking* problem. They extract entity mentions from the set of top relevant passages, then consider the types associated with the top-ranked entities using various weighting functions. Kaptein et al. (2010) similarly use a simple entity-centric model. Manually assigned target types tend to be more general than automatically identified ones (Kaptein and Kamps 2013). Having a hierarchical structure, therefore, makes it convenient to assign more general types. In (Balog and Neumayer 2012), a hierarchical version of the *target entity type identification* task is addressed using the DBpedia Ontology and language modeling techniques. One approach uses an entity-centric strategy. Another one builds a textual type representation by concatenating the descriptions of all its assigned entities. We present a detailed description of these models in Sects. 5.1 and 5.2, and further expand on them in Sect. 5.3. Sawant and Chakrabarti (2013) focus on telegraphic queries and assume that each query term is either a type hint or a "word matcher." They consider multiple interpretations of the query and tightly integrate type detection within the ranking of entities. Their approach further relies on the presence of a large-scale web corpus. We consider target entity types identification using an oracle process, based on the set of known relevant entities (cf. Sect. 6.2), as well as using automatic methods (cf. Sect. 5).

2.5 Entity type assignments

A further complicating issue is that type information associated with entities in the knowledge base is incomplete, imperfect, or missing altogether for some entities. Gangemi et al. (2012) distinguish between *extensional* coverage, i.e., the number of typed resources, and *intensional* coverage, i.e., conceptual completeness. Automatic typing of entities is a possible solution for alleviating some of these problems. For example, approaches to extend entity type assignments in DBpedia include mining associated Wikipedia articles for wikilink relations (Nuzzolese et al. 2012), patterns over logical interpretations of the deeply parsed natural language definitions (Gangemi et al. 2012), or linguistic hypotheses about category classes (Fossati et al. 2015). Several works have addressed entity typing

Table 1 Glossary of the notation used in this paper

Symbol	Description
e	Entity ($e \in \mathcal{E}$)
$\mathcal{E}$	Set of all entities in the knowledge base
$\mathcal{E}_t$	Set of all entities typed with t
q	Query
t	Type ($t \in \mathcal{T}$)
$\mathcal{T}$	Set of all types in the taxonomy
$\mathcal{T}_e$	Set of all types assigned to e ($\mathcal{T}_e \subset \mathcal{T}$)
w	Term (word)
$a(e, t)$	Entity-type association weight for e and t
$f(w, e)$	Frequency of w in (the description of) e
$n(t, e)$	Entity-type association importance for e and t
$\pi(t)$	Parent type of t in the taxonomy
Rel_q	Set of relevant entities for q according to the ground truth
$rel(q, e)$	Relevance level of e for q according to the ground truth
$R_k(q)$	Set of top-k ranked entities for q
$score_M(e, q)$	Retrieval score of entity e for query q, given by model M
$score_{M_{(\phi_i)}}(t, q)$	Target type score of t for query q, given by model M, with array (ϕ_i) of underlying retrieval model parameters (omitted if empty)
$w2v(w)$	Pre-trained *word2vec* word embedding vector for w
$\mathbf{v}^{w2v}_{content}$	Centroid of *word2vec* vectors for all content words in v
$\mathbb{1}(p)$	Binary indicator function which returns 1 iff p is true
θ_e	Entity types distribution for e
θ_q	Target types distribution for q
λ_t	Weight of type-based component in interpolation model
k	Target types ranking cutoff in strict filtering model

over progressively larger taxonomies with finer-grained types (Fleischman and Hovy 2002; Giuliano 2009; Rahman and Ng 2010; Ling and Weld 2012; Yosef et al. 2012). Regarding the task of detecting and typing *emerging entities*, having fine-grained types for new entities is of particular importance for informative knowledge (Lin et al. 2012; Nakashole et al. 2013).

3 Type-aware entity retrieval

In this section, we formally describe the type-aware entity retrieval models we will be using for investigating the research questions stated in Sect. 1. Our contributions do not lie in this part; the techniques we present were shown to be effective in prior research. We refer to Table 1 for the notation used in this paper.

We formulate our retrieval task in a generative probabilistic framework. Given an input query q, we rank entities e according to

$$P(e|q) \propto P(q|e)\, P(e). \tag{1}$$

When uniform entity priors are assumed, the final ranking of entities boils down to the estimation of $P(q|e)$. We consider the query in the term space as well as in the type space. Hence, we write $q = (q_w, q_t)$, where q_w holds the query terms (words) and q_t holds the *target types*. Two ways of factoring the probability $P(q|e)$ are presented in Sect. 3.1. All models share two components: term-based similarity, $P(q_w|e)$, and type-based similarity, $P(q_t|e)$. These are discussed in Sects. 3.2 and 3.3, respectively.

3.1 Retrieval models

We present two alternative approaches for combining term-based and type-based similarity.

3.1.1 Filtering

Assuming conditional independence between the term-based and type-based components, the final score becomes a multiplication of the components:

$$P(q|e) = P(q_w|e)\,P(q_t|e). \tag{2}$$

This approach is a generalization, among others, of the one used in Bron et al. (2010) (where the term-based information itself is unfolded into multiple components, considering not only language models from textual context but also estimations of entity co-occurrences). We consider two specific instantiations of this model:

- *Strict filtering*, where $P(q_t|e)$ is 1 if the sets of target types and entity types have a non-empty intersection, and is 0 otherwise.
- *Soft filtering*, where $P(q_t|e) \in [0..1]$ and is estimated using the approach detailed in Sect. 3.3.

3.1.2 Interpolation

Alternatively, a mixture model may be used, which allows for controlling the importance of each component. Nevertheless, the conditional independence between q_w and q_t is still imposed by this model:

$$P(q|e) = (1 - \lambda_t)P(q_w|e) + \lambda_t P(q_t|e), \tag{3}$$

where $P(q_t|e)$ is estimated using the approach detailed in Sect. 3.3. Examples of using the interpolation model include (Pehcevski et al. 2010; Balog et al. 2011; Raviv et al. 2012; Kaptein and Kamps 2013).

3.2 Term-based similarity

We base the estimation of the term-based component, $P(q_w|e)$, on statistical language modeling techniques since they have shown to be an effective approach in prior work, see, e.g., (Balog et al. 2011; Kaptein and Kamps 2013; Bron et al. 2010; Balog and Neumayer 2013; Hasibi et al. 2016). Specifically, we employ the Sequential Dependence Model (SDM) (Metzler and Croft 2005). Following Hasibi et al. (2017), we set the default parameters 0.8, 0.1, and 0.1 for terms, ordered, and unordered bigram, respectively. We note that the term-based component is not the focus of this work; any other approach could also be plugged in (provided that the retrieval scores are mapped to probabilities).

3.3 Type-based similarity

Rather than considering types simply as a set, we assume a distributional representation of types, also referred to as *bag-of-types*. Namely, a type in the bag may occur with repetitions, naturally rendering it more important. Following Balog et al. (2011), we represent type information as a multinomial probability distribution over types, both for queries and for entities. Specifically, let θ_q denote the target type distribution for the query q (such that $\sum_t P(t|\theta_q) = 1$). We assume that there is some mechanism in place that estimates this distribution; in our experiments, we will rely on an "oracle" that provides us exactly with this information (cf. Sect. 6.2). Further, let θ_e denote the target type distribution for entity e. We assume that a function $n(t, e)$ is provided, which returns 1 if e is assigned to type t, otherwise 0. We present various ways of setting $n(t, e)$ based on the hierarchy of the type taxonomy in Sect. 4. We note that $n(t, e)$ is not limited to having a binary value; this quantity could, for example, be used to reflect how important type t is for the given entity e. We use a multinomial distribution to allow for such future extensions. Based on these raw counts, the type-based representation of an entity e is estimated using Dirichlet smoothing:

$$P(t|\theta_e) = \frac{n(t,e) + \mu P(t)}{\sum_{t'} n(t',e) + \mu}, \tag{4}$$

where the background type model is obtained by a maximum-likelihood estimate:

$$P(t) = \frac{\sum_{e'} n(t,e')}{\sum_{t'} \sum_{e'} n(t',e')}. \tag{5}$$

The smoothing parameter μ in Eq. (4) is set to the average number of types assigned to an entity. In Eqs. (4) and (5), t' is any type in the taxonomy ($t' \in \mathcal{T}$) and e' is any entity in the knowledge base ($e' \in \mathcal{E}$).

With both θ_q and θ_e in place, we estimate type-based similarity using the Kullback–Leibler (KL) divergence of the two distributions:

$$P(q_t|e) = z\left(\max_{e'} KL(\theta_q \parallel \theta_{e'}) - KL(\theta_q \parallel \theta_e)\right), \tag{6}$$

where z is a normalization factor:

$$z = 1 \Big/ \sum_e \max_{e'}(KL(\theta_q \parallel \theta_{e'}) - KL(\theta_q \parallel \theta_e)).$$

Note that the smaller the divergence the more similar the distributions are, therefore in Eq. (6) we subtract it from the maximum KL-divergence, in order to obtain a probability distribution. For further details we refer to Balog et al. (2011).

4 Entity type representation

This section introduces alternative ways of representing hierarchical entity type information (Sect. 4.1) and the different type taxonomies that are considered in our experimental evaluation (Sect. 4.2).

4.1 Hierarchical entity type representation

We consider different ways of representing hierarchical entity type information. Specifically, we investigate how to set the quantity $n(t, e)$, which is needed for estimating type-based similarity between target types of the query and types assigned to the entity in the knowledge base. Before proceeding further, let us introduce some terminology and notation.

- $\mathcal{T}$ is a type taxonomy that consists of a set of hierarchically organized entity types, and $t \in \mathcal{T}$ is a specific entity type.
- $\mathcal{E}$ is the set of all entities in the knowledge base, and $e \in \mathcal{E}$ is a specific entity.
- $\mathcal{T}_e$ is the set of types that are assigned to the entity e in the knowledge base. We refer to this as a set of *assigned types*. Note that $\mathcal{T}_e$ might be an empty set.

We impose the following constraints on the type taxonomy.

i. There is a single root node t_0 that is the ancestor of all types (e.g., < `owl : Thing` >). Since all entities belong to this type, it is excluded from the set of assigned types by definition.
ii. We restrict the type taxonomy to subtype–supertype relations; each type t has a single parent type denoted as $\pi(t)$.
iii. Type assignments are transitive, i.e., an entity that belongs to a given type also belongs to all ancestors of that type: $t \in \mathcal{T}_e \wedge \pi(t) \neq t_0 \implies \pi(t) \in \mathcal{T}_e$.

We further note that an entity might belong to multiple types under different branches of the taxonomy. Assume that t_i and t_j are both types of e. It might be then that their nearest common ancestor in the type hierarchy is t_0.

While $\mathcal{T}_e$ holds the types assigned to entity e, there are multiple ways of turning it into a numerical value, $n(t, e)$, which reflects the type's importance with respect to the given entity. This importance is taken into account when building the type-based entity representation in Eq. (4). In this work, we treat all types equally important for an entity, i.e., use binary values for $n(t, e)$.

We consider the following three options for representing hierarchical type information; see Fig. 2 for an illustration. In our definitions, we use $\mathbb{1}(x)$ as an indicator function, which returns the value 1 if condition x is true and returns 0 otherwise.

- *Types along path-to-top* It counts all types that are assigned to the entity in the knowledge base, excluding the root (from constraint (iii) it follows that $\mathcal{T}_e$ contains all the types in the path to the top-level nodes):

$$n(t, e) = \mathbb{1}\left(t \in \mathcal{T}_e\right).$$

- *Top-level type(s)* Only top-level types are considered for an entity, that is, types that have the root node as their parent:

$$n(t, e) = \mathbb{1}\left(t \in \mathcal{T}_e \wedge \pi(t) = t_0\right).$$

- *Most specific type(s)* From each path, only the most specific type is considered for the entity:

$$n(t, e) = \mathbb{1}\left(t \in \mathcal{T}_e \wedge \nexists\, t' \in \mathcal{T}_e : \pi(t') = t\right).$$

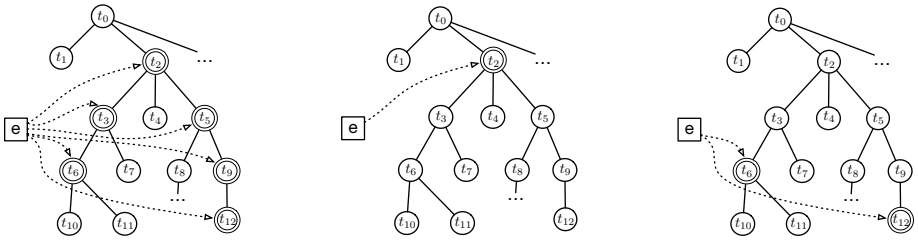

Fig. 2 Alternative ways of representing entity-type assignments with respect to the type taxonomy. The dashed arrows point to the types that are assigned to entity e. The root node of the taxonomy is labeled with t_0

Table 2 Overview of normalized type taxonomies and their statistics. The top block is about the taxonomy itself; the bottom block is about type assignments of entities

Type system	DBpedia	Freebase	Wikipedia categories	YAGO
Number of types	713	1719	423,636	568,672
Number of top-level types	22	92	27	61
Number of leaf-level types	561	1626	303,956	549,754
Height	7	2	35	19
Number of types used	408	1626	359,159	314,632
Number of entities with type	4.87M	3.27M	3.52M	2.88M
Avg. number of types per entity	2.8	4.4	20.8	13.4
Mode depth	2	2	11	4

Even though there may be alternative representations, these three are natural ways of encoding hierarchical information.

4.2 Entity type taxonomies

In this paper we study multiple type taxonomies from various knowledge bases: DBpedia, Freebase, Wikipedia, and YAGO. These vary a lot in terms of hierarchical structure and in how entity-type assignments are recorded. We normalize these type taxonomies to a uniform structure, adhering to the constraints specified in Sect. 4.1. Table 2 presents an overview of the type systems (after normalization). The number of type assignments are counted according to the representation along path-to-top. Properties of the four type systems and details of the normalization process are discussed below.

4.2.1 Type taxonomies

Wikipedia categories The Wikipedia category system, developed and extended by Wikipedia editors, consists of textual labels known as categories. This categorization is not a well-defined "is-a" hierarchy, but a graph; a category may have multiple parent categories and there might be cycles along the path to ancestors (Kaptein et al. 2010). Also, categories

often represent only loose relatedness between articles; category assignments are neither consistent nor complete (Demartini et al. 2010b).

We transformed the Wikipedia category graph, consisting of over 1.16M categories, into a type taxonomy as follows. First, we selected a set of 27 top-level categories covering most of the knowledge domains.[1] These became the top-level nodes of the taxonomy, all with a single common root type $<$ `owl : Thing` $>$. All super-categories that these selected top-level categories might have in the graph were discarded. Second, we removed multiple inheritances by selecting a single parent per category. For this, we considered the population of a category to be the set of its assigned articles. Each category was linked in the taxonomy with a single parent in the graph whose intersection between their populations is the maximal among all possible parents; in case of a tie, the most populated parent was chosen. Under this criterion, and for the purpose of understanding hierarchical relations, any category without a parent was discarded. Lastly, from this partial hierarchy (which is still a graph, not a tree), we obtained the final taxonomy by performing a depth-first exploration from each top-level category, and avoiding to add those arcs that would introduce cycles. This depth-first approach was previously used by Fossati et al. (2015) for enforcing taxonomic constraints on Wikipedia categories. The resulting taxonomy contains over 423k categories and reaches a maximum depth of 35 levels.[2]

DBpedia ontology The DBpedia Ontology is a well-designed hierarchy since its inception; it was created manually by considering the most frequently used infoboxes in Wikipedia. It continues to be properly curated to address some weaknesses of the Wikipedia infobox space. While the DBpedia Ontology is clean and consistent, its coverage is limited to entities that have an associated infobox. It consists of 713 classes, including the root, organized in a hierarchy of 7 levels.

YAGO taxonomy YAGO is a huge semantic knowledge base, derived from Wikipedia, WordNet, and GeoNames (Suchanek et al. 2007). Its type classification schema is constructed by taking leaf categories from the category system of Wikipedia and then using WordNet synsets to establish the hierarchy of classes. The result is a deep subsumption hierarchy, consisting of over 568k classes. We work with the YAGO taxonomy from the current version of the ontology (3.0.2). We normalized it by adding a root node, $<$ `owl : Thing` $>$, as a parent to every top-level type.

Freebase types Freebase has a two-layer categorization system, where types on the bottom level are grouped under high-level domains. We used the latest public Freebase dump (2015-03-31), discarding domains meant for administering the Freebase service itself (e.g.; `base`, `common`). Additionally, we made $<$ `owl : Thing` $>$ the common root of all the domains, and finally obtained a taxonomy of 1719 types.

[1] The selected top-level categories are the main categories for each section of the portal https://en.wikipedia.org/wiki/Portal:Contents/Categories. (As an alternative, we also considered the categories from https://en.wikipedia.org/wiki/Category:Main_topic_classifications, and found that it comprises a similar category selection).

[2] We have confirmed experimentally that enforcing the Wikipedia category graph to satisfy the taxonomical constraints does not hurt retrieval performance. In fact, it is the opposite: it results in small, but statistically significant improvements (Garigliotti and Balog 2017).

4.2.2 Entity-type assignments

Now that we have presented the four type taxonomies, we also need to discuss how type assignments of entities are obtained. We use DBpedia 2015-10 as our knowledge base, which makes DBpedia types, Wikipedia categories, and YAGO type assignments readily available. For the fourth type taxonomy, Freebase, we followed same-as links from DBpedia to Freebase (which exist for 95% of the entities in DBpedia) and extracted type assignments from Freebase. It should be noted that entity-type assignments are provided differently for each of these taxonomies; DBpedia and Freebase supply a single (most specific) instance type for an entity, Wikipedia assignments include multiple categories for a given entity (without any restriction), while YAGO adheres to the representation along path. We treat all entity-type assignments transitively, adhering to constraint (iii) in Sect. 4.1.

5 Target entity type identification

Target entity types may be provided by the user explicitly as part of the search request, for example, via faceted user interfaces. Often, however, users would prefer to use simple keyword queries as input. In that case, target entity types need to be identified automatically based on the keyword query. In this section, we discuss how to assign target entity types to queries from a type taxonomy.

As our starting point, we take the definition of the *hierarchical target type identification* (HTTI) task, as introduced in (Balog and Neumayer 2012): "finding the single most specific type within the ontology that is general enough to cover all relevant entities." We point out two major limitations with this definition and suggest ways to overcome them.

First, it is implicitly assumed that every query must have a *single* target type, which is not particularly useful in practice. Take, for example, the query "Finland car industry manufacturer saab sisu," where both *Company* and *Automobile* are valid types. We shall allow for possibly multiple main types, if they are sufficiently different, i.e., lie on different branches in the taxonomy. Second, it can happen—and in fact it does happen for 33% of the queries considered in Balog and Neumayer (2012)—that a query cannot be mapped to any type in the given taxonomy (e.g., "Vietnam war facts"). However, those queries were simply ignored in Balog and Neumayer (2012). Instead, we shall allow a query not to have any type (or, equivalently, to be tagged with a special `NIL`-type). This relaxation means that we can now take any query as input. Our revised task definition is thus as follows.

Definition 1 Hierarchical target entity type identification (*HTTIv2*) is the task of finding the main target types of a query, from a type taxonomy, such that (i) these correspond to the most specific category of entities that are relevant to the query, and (ii) main types cannot be on the same branch in the taxonomy. If no matching type can be found in the taxonomy then the query is assigned a special `NIL`-type.

Let us note that detecting `NIL`-types is a separate task on its own account, which we are not addressing in this paper. For now, the importance of the `NIL`-type distinction is restricted to how the query annotations are performed.

5.1 Entity-centric model

The entity-centric model can be regarded as the most common approach for determining the target types for a query, see, e.g., Kaptein et al. (2010), Balog and Neumayer (2012), Vallet and Zaragoza (2008). This model also fits the late fusion design pattern for object retrieval (Zhang and Balog 2017). The idea is simple: first, rank entities based on their relevance to the query, then look at what types the top-K ranked entities have. The final score for a given type t is the aggregation of the relevance scores of entities with that type. Formally:

$$score_{EC_{(M,K)}}(t, q_w) = \sum_{e \in R_K(q_w)} score_M(e, q_w)\, a(e, t),$$

where $R_K(q_w)$ is the set of top-K ranked entities for the keyword query q_w. The retrieval score of entity e is denoted by $score_M(e, q_w)$. In our experiments, we consider both Language Models and BM25 as the underlying entity retrieval model M. The rank cut-off threshold K is set empirically. The entity-type association weight, $a(e, t)$, is set uniformly across entities that are typed with t, and is 0 otherwise:

$$a(e,t) = \begin{cases} \frac{1}{|\mathcal{E}_t|} \sum_{e'} \mathbb{1}(e' \in \mathcal{E}_t) & e \in \mathcal{E}_t \\ 0 & \text{otherwise.} \end{cases}$$

We denote this entity-centric target type score by $EC_{M,K}(t, q_w)$.

5.2 Type-centric model

Alternatively, one can also build for each type a direct term-based representation (pseudo type description document), by aggregating descriptions of entities that belong to the given type. Then, those type representations can be ranked much like documents. This model has been presented in Balog and Neumayer (2012) using Language Models, and has been generalized to arbitrary retrieval models (and referred to as the early fusion design pattern for object retrieval) in Zhang and Balog (2017). The pseudo-frequency of word w given type t is defined as:

$$\tilde{f}(w, t) = \sum_e f(w, e)\, a(e, t), \tag{7}$$

where $f(w, e)$ is the frequency of the term w in (the description of) entity e and $a(e, t)$, as before, denotes the entity-type association weight. The relevance score of a type for a given query $q_w = \langle q_1, \ldots, q_n \rangle$ is then calculated as the sum of the individual query term scores:

$$score_{TC_{(M)}}(t, q_w) = \sum_{i=1}^{n} score_M(q_i, \tilde{f}, \varphi)$$

where $score_M(q_i, \tilde{f}, \varphi)$ is the underlying term-based retrieval model M (e.g., LM or BM25), parameterized by φ. This model assigns a score to each query term q_i, based on the word pseudo-frequencies $\tilde{f}$. We denote this type-centric target type score by $TC_M(t, q_w)$.

5.3 Learning to rank

The entity-centric and type-centric models capture different aspects of the task, and it is therefore sensible to combine the two. While this idea has been suggested in Balog and Neumayer (2012), to the best of our knowledge, our work is the first to realize it, using a learning-to-rank (LTR) approach (Garigliotti et al. 2017). In addition, there are other signals that one could leverage, including taxonomy-driven features and type label similarities. Table 3 summarizes the features we use for target entity type identification.

5.3.1 Knowledge base features

We assume that a knowledge base provides a type system of reference along with entity-type mappings. In this setting, features related to the hierarchy of the type taxonomy emerge naturally. In particular, instead of using absolute depth metrics of a type like in Tonon et al. (2016), we use a normalized depth with respect to the height of the taxonomy (feature #13). We also take into account the number of children and siblings of a type (features #14 and #15). Intuitively, the more specific a type, the deeper it is located in the type taxonomy, and the less its number of children, while the more its number of siblings. Hence all three of these features capture how specific a type is according to its context in a type taxonomy. The type coverage (feature #16) is also directly related to the intuition of type specificity; the more general the type, the larger number of entities it tends to cover.

5.3.2 Type label features

We consider several signals for measuring the similarity between the surface form of the type label and the query. The type label length (feature #17) and the IDF-related statistics (features #18–19) are closely related to type specificity. The Jaccard similarities (features #20–21) capture shallow linguistic similarities by n-gram matches between the set of n consecutive terms in the query and the type labels, where $n \leq 2$, since the textual phrases in any of these labels are expected to be short. In particular, the bigram match ($n = 2$) makes sense for capturing some typical type label patterns, e.g., $\langle adjective \rangle \langle noun \rangle$ in `German physicists`. A more constrained version, defined in feature #22, measures the query-type Jaccard similarity over single terms ($n = 1$), which are nouns.

We use pre-trained word embeddings provided by the *word2vec* toolkit (Mikolov et al. 2013). However, we only consider *content words* (linguistically speaking, i.e., nouns, adjectives, verbs, or adverbs). Feature #23 captures the compositional nature of words in type labels:

$$SIMAGGR(t, q) = \cos(\boldsymbol{q}^{w2v}_{content}, \boldsymbol{t}^{w2v}_{content}),$$

where the query and type vectors are taken to be the $w2v$ centroids of their content words. Feature #24 measures the pairwise similarity between content words in the query and the type label:

$$SIMMAX(t, q) = \max_{w_q \in q, w_t \in t} \cos(w2v(w_q), w2v(w_t)),$$

where $w2v(w)$ denotes the *word2vec* vector of term w. Feature #25 $SIMAVG(t)$ is defined analogously, but using *avg* instead of *max*.

Table 3 Features for learning to rank target types

#	Feature	Description	Kind	Value
Baseline features				
1–5	$EC_{BM25,K}(t,q)$	Entity-centric score (cf. Sect. 5.1) of type t for query q, with $K \in \{5, 10, 20, 50, 100\}$ using BM25	Entity-centric	$[0..\infty)$
6–10	$EC_{LM,K}(t,q)$	Entity-centric score (cf. Sect. 5.1) of type t for query q, with $K \in \{5, 10, 20, 50, 100\}$ using LM	Entity-centric	$[0..1]$
11	$TC_{BM25}(t,q)$	Type-centric score (cf. Sect. 5.2) of type t for query q, using BM25	Type-centric	$[0..\infty)$
12	$TC_{LM}(t,q)$	Type-centric score (cf. Sect. 5.2) of type t for query q, using LM	Type-centric	$[0..1]$
Knowledge base features				
13	$DEPTH(t)$	The hierarchical level of type t, normalized by the taxonomy depth	Taxonomy	$[0..1]$
14	$CHILDREN(t)$	Number of children of type t in the taxonomy	Taxonomy	$\{0, \ldots, \infty\}$
15	$SIBLINGS(t)$	Number of siblings of type t in the taxonomy	Taxonomy	$\{0, \ldots, \infty\}$
16	$ENTITIES(t)$	Number of entities assigned to type t	Coverage	$\{0, \ldots, \infty\}$
Type label features				
17	$LENGTH(t)$	Length of (the label of) type t in words	Statistical	$\{1, \ldots, \infty\}$
18	$IDFSUM(t)$	Sum of IDF for terms in (the label of) type t	Statistical	$[0..\infty)$
19	$IDFAVG(t)$	Avg of IDF for terms in (the label of) type t	Statistical	$[0..\infty)$
20–21	$JTERMS_n(t,q)$	Query-type Jaccard similarity for sets of n-grams, for $n \in \{1, 2\}$	Linguistic	$[0..1]$
22	$JNOUNS(t,q)$	Query-type Jaccard similarity using only nouns	Linguistic	$[0..1]$
23	$SIMAGGR(t,q)$	Cosine sim. between the q and t *word2vec* vectors aggregated over all terms of their resp. labels	Distributional	$[0..1]$
24	$SIMMAX(t,q)$	Max. cosine similarity of *word2vec* vectors between each pair of query (q) and type (t) terms	Distributional	$[0..1]$
25	$SIMAVG(t,q)$	Avg. of cosine similarity of *word2vec* vectors between each pair of query (q) and type (t) terms	Distributional	$[0..1]$

Table 4 Query categories in the DBpedia-entity collection

Category	Description	Example
INEX-LD	General keyword queries	"Guitar origin blues"
ListSearch	Entity list queries	"Products of Medimmune, Inc."
QALD-2	Natural language queries	"Who was called Scarface?"
SemSearch ES	Named entity queries	"Brooklyn bridge" or "Ashley Wagner"

6 Experimental setup

We base our experiments on the DBpedia knowledge base (version 2015-10). DBpedia (Lehmann et al. 2015), as a central hub in the Linked Open Data cloud, provides a large repository of entities, which are mapped—directly or indirectly; cf. Sect.4.2.2—to each of the type taxonomies of interest.

6.1 Test collection

Our experimental platform is based on the DBpedia-Entity v2 test collection[3] developed in Hasibi et al. (2017). The dataset contains 467 queries, synthesized from various entity-related benchmarking evaluation campaigns. These range from short keyword queries to natural language questions; see Table 4.

6.2 Target entity types oracle

Throughout our first set of experiments (in Sect. 7), we make use of a so-called *target entity types oracle*. We assume that there is an "oracle" process in place that provides us with the (distribution of) correct target types for a given query. This corresponds to the setting that was employed at previous benchmarking campaigns (such as the INEX Entity Ranking track (Demartini et al. 2010b) and the TREC Entity track (Balog et al. 2012)), where target types are provided explicitly as part of the topic definition. We employ this idealized setting to ensure that our results reflect the full potential of using type information, without being hindered by the imperfections of an automated type detector.

For a given query q, we take $\mathcal{T}_q = \bigcup_{e \in Rel_q} \mathcal{T}_e$, the union of all types of all entities that are judged relevant for that query. Each of these types $t \in \mathcal{T}_q$ becomes a target type, and its probability $P(t|\theta_q)$ is set proportional to the number of relevant entities that have that type. Formally, the oracle O scores target types as follows:

$$score_O(t, q) = \sum_{e \in Rel_q \cap \mathcal{E}_t} rel(e, q), \tag{8}$$

where $\mathcal{E}_t$ denotes the set of entities that are assigned to type t and $rel(e, q)$ is the relevance score of entity e for query q, according to the ground truth. Then, the oracle target distribution is given by:

[3] http://tiny.cc/dbpedia-entity.

Table 5 Statistics of the target entity types oracle

Type system	DBpedia	Freebase	Wikipedia categories	YAGO
Number of types used	213	716	10,852	10,080
Number of queries with type	451	478	469	470
Avg. number of types per query	3.05	19.10	31.32	54.84
Mode depth	4	2	10	6

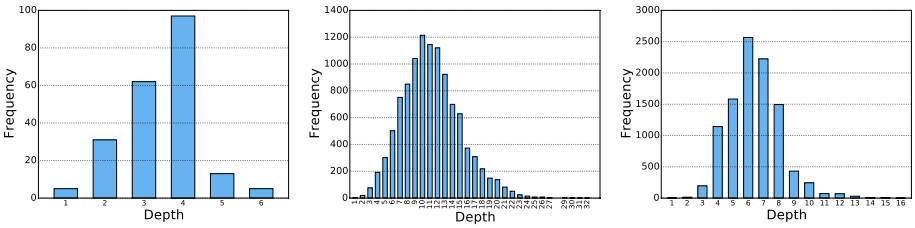

Fig. 3 Distribution of oracle target types within hierarchical levels of the DBpedia, Wikipedia, and YAGO taxonomies. Freebase is omitted since all target types are on the leaf level, i.e., have depth = 2. **a** DBpedia, **b** Wikipedia and **c** YAGO

$$P(t|\theta_q) = \frac{score_O(t,q)}{\sum_{t'} score_O(t',q)}. \tag{9}$$

Table 5 summarizes the statistics of target entity types obtained by the oracle for each type taxonomy. As it can be seen, deeper taxonomies (Wikipedia categories and YAGO) have larger average number of types per query, which is similar to what we observed for entity type assignments in Table 2. For Freebase, the number of target types appears disproportionately large compared to the entity type assignments in Table 2. Figure 3 shows how the target types are distributed hierarchically within each taxonomy.

6.3 Entity retrieval models

As our baseline, we use a term-based approach, specifically the Sequential Dependence Model (SDM) (Metzler and Croft 2005), which we described in Sect. 3.2. We compare three type-aware retrieval models (cf. Sect. 3.1): strict filtering, soft filtering, and interpolation. For the latter, we perform a sweep over the possible type weights $\lambda_t \in [0, 1]$ in steps of 0.05, and use the best performing setting when comparing against other approaches. (Automatically estimating the λ_t parameter is outside the scope of this work.)

6.4 Target entity type identification models

For the entity-centric (Sect. 5.1) and type-centric (Sect. 5.2) models, the Language Modeling (LM) approach uses Dirichlet prior smoothing with the smoothing parameter set to 2000; for BM25, we use $k1 = 1.2$ and $b = 0.75$. For the LTR approach (Sect. 5.3), we

employ the Random Forest algorithm for regression as our supervised ranking method. We set number of trees (iterations) to 1000, and the maximum number of features in each tree, m, to (the ceil of the) 10% of the size of the feature set.

6.5 Type assignments

In the default setting, we include all entities from the knowledge base and use the original set of relevance assessments. By doing so, some entities and queries do not have types assigned from one or more taxonomies. Therefore, we introduce an additional experimental setting, referred to as *1TT*, to ensure that the differences we observe are not a result of missing type assignments.

In the 1TT setting, for each type taxonomy, we restrict our set of entities to those that have at least one type assigned in the taxonomy. We also restrict the set of queries to those that have target types in that type system; queries without any relevant results (as a consequence of these restrictions) are filtered out. This leaves us with a total of 446 queries for DBpedia, 454 for Freebase, 463 for Wikipedia, and 450 for YAGO.

6.6 Test collection of target entity types

We build a test collection for the revised hierarchical target type identification task (cf. Sect. 5). Having the DBpedia Ontology (version 2015-10) as our type taxonomy, we collect relevance labels via crowdsourcing for all the 485 queries in the DBpedia-Entity v1 collection (Balog and Neumayer 2013) (which is a superset of the DBpedia-Entity v2 queries that we use for evaluating entity ranking).

A pool of target entity types is constructed from four baseline methods, taking the top 10 types from each: entity-centric (cf. Sect. 5.1) using K=100, and type-centric (cf. Sect. 5.2), with both BM25 and LM as underlying retrieval methods. Additionally, we included all types returned by the target entity types oracle (cf. Sect. 6.2), to ensure that all reasonable types are considered when collecting human annotations.

We obtained target type annotations via the CrowdFlower crowdsourcing platform. Specifically, crowd workers were presented with a search query (along with the narrative from the original topic definition, where available), and a list of candidate types, organized hierarchically according to the taxonomy. We asked them to "select the single most specific type, that can cover all results the query asks for" (in line with Balog and Neumayer 2012). If none of the presented types are correct, they were instructed to select the "None of these types" (i.e., `NIL`-type) option.

The annotation exercise was carried out in two phases. In the first phase, we sought to narrow down our pool to the most promising types for each query. Since the number of candidate types for certain queries was fairly large, they were broken down to multiple micro-tasks, such that for every top-level type, all its descendants were put in the same micro-task. Each query-type batch was annotated by 6 workers. In the second phase, all candidate types for a query were presented in a single micro-task; candidates include all types that were selected by at least one assessor in phase one, along with their ancestors up to the top level of the hierarchy. Each query was annotated by 7 workers. The Fleiss' Kappa inter-annotator agreement for this phase was 0.71, which is considered substantial.

Note that according to our *HTTIv2* task definition, main target types of a query cannot lie on the same path in the taxonomy. To satisfy this condition, if two types were on the same path, we merged the more specific type into the more generic one (i.e., the more

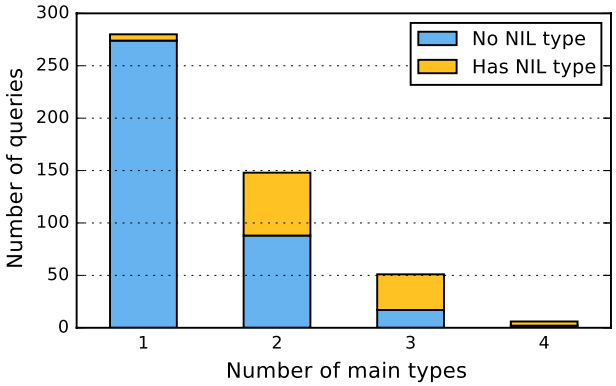

Fig. 4 Distribution of the number of main target types in our test collection (Color figure online)

generic type received all the "votes" of the more specific one). This affected 120 queries. Figure 4 shows the distribution of queries according to the number of main types. In the resulting collection, 280 of all queries (57.73%) have a single target type, while the remainder of them have multiple target types. It is noticeable that as the number of main types increases, so does the proportion of `NIL`-type annotations.

7 Results using Oracle target entity type identification

In this section, we present evaluation results for all combinations of the three proposed dimensions: type taxonomies, type representation modes, and retrieval models. When discussing the results, we use the term *configuration* to refer to a particular combination of type taxonomy, type representation, and retrieval model.

Recall that we distinguish between two settings (cf. Sect. 6.5): ranking all entities in the knowledge base (ALL) and considering only entities that have types assigned to them in a given type taxonomy (1TT). Tables 6 and 7 show results corresponding to these two settings, respectively. Our main evaluation metric is normalized discounted cumulative gain with a cutoff of 10 (NDCG@10); we also report on NDCG@100. We test statistical significance to measure our confidence in rejecting the null hypothesis that states that our improvements occur by chance. Specifically, we use a two-tailed paired t test at $p < 0.05$ and $p < 0.001$, denoted by † and ‡, respectively. For an easier visual inspection, the NDCG@10 scores are also plotted in Fig. 5, where the red line corresponds to the term-based baseline.

7.1 Type taxonomy

Let us begin with our first research question (RQ1), which concerns the impact of the particular choice of type taxonomy.

– *RQ1* What is the impact of the particular choice of type taxonomy on entity retrieval performance?

Table 6 Entity retrieval performance using oracle target types, returning all entities from the knowledge base (ALL). For the interpolation model, λ_t is value of the best empirically found interpolation parameter. Performance is measured in terms of NDCG@10 and NDCG@100

Model	Strict filtering		Soft filtering		Interpolation		
	@10	@100	@10	@100	@10	@100	λ_t
Baseline (Metzler and Croft 2005)	0.4185	0.5143	0.4185	0.5143	0.4185	0.5143	–
DBpedia							
Along path	0.3998	0.4947	**0.4440**†	**0.5279**†	0.4549‡	0.5337‡	0.25
Top-level	0.3998	0.4947	0.4307	0.5187	0.4414‡	0.5253‡	0.25
Most specific	**0.4389**	**0.5186**	0.4404†	0.5259†	**0.4579**‡	**0.5366**‡	0.15
Freebase							
Along path	0.4113	0.5003	**0.4766**‡	**0.5486**‡	**0.4702**‡	**0.5453**‡	0.35
Top-level	0.4113	0.5003	0.4758‡	0.5461‡	0.4690‡	0.5428‡	0.40
Most specific	**0.4306**	**0.5127**	0.4734‡	0.5467‡	0.4664‡	0.5432‡	0.35
Wikipedia							
Along path	0.4310‡	0.5170	0.4256	0.5215	0.4283‡	0.5211‡	0.05
Top-level	0.1102	0.3243	0.2707	0.4271	0.4185	0.5143	0.00
Most specific	**0.5362**‡	**0.5775**‡	**0.4742**‡	**0.5506**‡	**0.4603**‡	**0.5432**‡	0.25
YAGO							
Along path	0.3814	0.4770	**0.4718**‡	0.5483‡	**0.4647**‡	0.5421‡	0.35
Top-level	0.3814	0.4770	0.4186	0.5129	0.4314‡	0.5223‡	0.25
Most specific	**0.4235**	**0.5038**	0.4685‡	**0.5492**‡	0.4561‡	**0.5429**‡	0.20

For a given type taxonomy and a given retrieval model, the best performance across the type representation modes according to each metric is remarked in bold

Statistical significance, tested using a two-tailed paired t test at $p < 0.05$ and $p < 0.001$, is denoted by † and ‡, respectively

It is clear that Wikipedia, in combination with the *Most specific* type representation, performs best for both settings (ALL and ITT, Fig. 5c, g), and yields substantial and highly significant improvements for all three retrieval models. As for the other two type representations for Wikipedia, performance slightly improves for *Along path* (significant for 1TT). *Top-level* Wikipedia types do not contribute when using the interpolation model ($\lambda_t = 0$), and are rather harmful when using either strict or soft filtering.

DBpedia and Freebase also show improvements for the ALL setting in all configurations, except the strict filtering model (Fig. 5a, b). The improvements for these smaller, shallower taxonomies are highly significant for all configurations in the 1TT setting (Fig. 5e, f). The case of YAGO is similar: all but the strict filtering configurations improve in the ALL setting (Fig. 5d), and all 1TT configurations yield highly significant improvements (Fig. 5h).

Comparing the results for the *Top-level* representation between YAGO and Wikipedia, it is clear that the *Top-level* Wikipedia categories that were chosen for enforcing taxonomic constraints are not appropriate for conveying entity type information.

Table 7 Entity retrieval performance using oracle target types, considering only entities that have types assigned to them in the respective type taxonomy (1TT). For the interpolation model, λ_t is value of the best empirically found interpolation parameter. Performance is measured in terms of NDCG@10 and NDCG@100

Model	Strict filtering		Soft filtering		Interpolation		
	@10	@100	@10	@100	@10	@100	λ_t
DBpedia							
Baseline (Metzler and Croft 2005)	0.3036	0.4119	0.3036	0.4119	0.3036	0.4119	–
Along path	0.4600‡	0.5079‡	**0.4353**‡	**0.4894**‡	**0.4172**‡	**0.4746**‡	0.65
Top-level	0.4600‡	0.5079‡	0.4196‡	0.4779‡	0.4141‡	0.4725‡	0.70
Most specific	**0.5092**‡	**0.5393**‡	0.4309‡	0.4864‡	0.4075‡	0.4696‡	0.55
Freebase							
Baseline (Metzler and Croft 2005)	0.3322	0.4403	0.3322	0.4403	0.3322	0.4403	–
Along path	0.4513‡	0.5106‡	0.4471‡	**0.5085**‡	0.4408‡	0.5021‡	0.65
Top-level	0.4513‡	0.5106‡	**0.4492**‡	0.5076‡	**0.4443**‡	**0.5031**‡	0.70
Most specific	**0.4736**‡	**0.5251**‡	0.4393‡	0.5034‡	0.4300‡	0.4966‡	0.60
Wikipedia							
Baseline (Metzler and Croft 2005)	0.3666	0.4727	0.3666	0.4727	0.3666	0.4727	–
Along path	0.4121‡	0.5000‡	0.4145‡	0.5014‡	0.3944‡	0.4877	0.40
Top-level	0.0963	0.2950	0.2193	0.3777	0.3666	0.4727	0.00
Most specific	**0.5874**‡	**0.6071**‡	**0.4741**‡	**0.5393**‡	**0.4474**‡	**0.5180**‡	0.65
YAGO							
Baseline (Metzler and Croft 2005)	0.3076	0.4180	0.3076	0.4180	0.3076	0.4180	–
Along path	0.4325‡	0.4904‡	**0.4453**‡	**0.5041**‡	**0.4313**‡	**0.4879**‡	0.75
Top-level	0.4325‡	0.4904‡	0.3630‡	0.4476‡	0.3807‡	0.4513‡	0.85
Most specific	**0.4843**‡	**0.5231**‡	0.4347‡	0.4998‡	0.4211‡	0.4850‡	0.70

For a given type taxonomy and a given retrieval model, the best performance among the type representation modes (and including the baseline) according to each metric is remarked in bold

Statistical significance, tested using a two-tailed paired t test at $p < 0.05$ and $p < 0.001$, is denoted by † and ‡, respectively

7.2 Type representation

The second research question (RQ2) is about type representation.

– *RQ2* How can one represent hierarchical entity type information for entity retrieval?

The question has a clear answer: keeping only the *Most specific* types in the hierarchy provides the best performance across the board, for all configurations. This fact is also in line with findings in past work (cf. Sect. 2). As for the other two representations, *Along path* is the better performing representation. The difference between *Along path* and *Top-level*, however, are generally small, in particular for the smaller taxonomies, DBpedia and Freebase.

Overall, we have verified that hierarchical relationships from ancestor types result in improved retrieval effectiveness, but simply resorting to the *Most specific* type assignments in the knowledge base is the most effective way of representing entity type information.

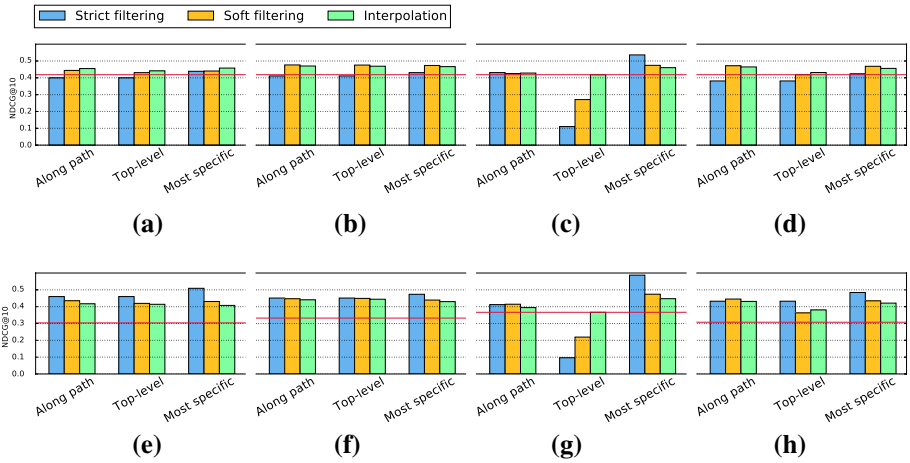

Fig. 5 Entity retrieval performance for all combinations of type taxonomies, type representation modes, and retrieval models. Top: all entities in the knowledge base (ALL); bottom: only entities with types from the given type taxonomy (1TT). The red line corresponds to the term-based baseline(s). Performance is measured by NDCG@10. **a** ALL, DBpedia, **b** ALL, Freebase, **c** ALL, Wikipedia, **d** ALL, YAGO, **e** 1TT, DBpedia, **f** 1TT, Freebase, **g** 1TT, Wikipedia and **h** 1TT, YAGO (Color figure online)

7.3 Type-aware entity retrieval

Our third research question (RQ3) concerns the type-aware retrieval model.

– *RQ3* How can one combine term-based and type-based matching for entity retrieval?

According to the 1TT setting (Table 7), strict filtering with the *Most specific* type representation is the best retrieval model for all configurations (achieving, in particular, a relative improvement of 67% in terms of NDCG@10 on DBpedia types), significantly outperforming the baseline in all cases. This no longer holds in the ALL setting (Table 6). The soft filtering and interpolation models perform best for all taxonomies, with small differences between the two, depending on the type representation. In the ALL setting, strict filtering always performs the worst and is often even below the baseline when *Along path* or *Top-level* type representation is used.

Comparing the respective λ_t type weights in Tables 6 and 7, it is noticeable that the interpolation model relies more on the type component in the 1TT than in the ALL setting. This is makes perfect sense since in the ALL setting many entities lack type information in the knowledge base.

Note that the interpolation model has a parameter λ_t that controls the weight of the type-based component. Figure 6 shows the performance of the interpolation model when varying the value of λ_t. We observe that with the exception of Wikipedia using the *Top-level* type representation, type information always improves over the baseline. In the ALL setting, performances generally peak in the 0.2–0.4 range, while for 1TT it is higher, around 0.5–0.7.

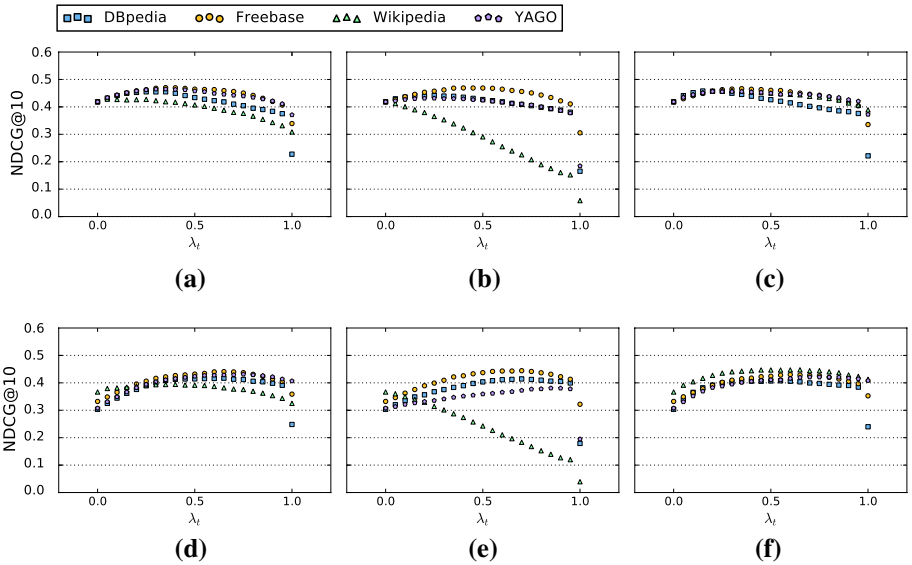

Fig. 6 Retrieval performance using the interpolation model with different type weights, λ_t. Top: all entities in the knowledge base (ALL); bottom: only entities with types from the given type taxonomy (1TT). The leftmost data points ($\lambda_t = 0$) correspond to the term-based baseline. **a** ALL, Types along path, **b** ALL, Top-level type(s), **c** ALL, Most specific type(s), **d** 1TT, Types along path, **e** 1TT, Top-level type(s) and **f** 1TT, Most specific type(s)

7.4 Analysis

We perform a more detailed analysis of particular configurations in order to gain a deeper understanding of each of the dimensions of entity type information. We focus on the bottom row of bar plots in Fig. 5e–h, that is, the 1TT experimental setting. There, as we previously explained, it is ensured that the differences we observe are not a result of missing type assignments. Figure 7 shows, for each of the selected configurations, the differences in NDCG@10 scores (ΔNDCG@10 hereinafter) on the level of individual queries between a given configuration and the corresponding (term-based) baseline. For the ease of visual comprehension, queries are ordered by ΔNDCG@10. Table 8 lists specific queries with the largest ΔNDCG@10 differences, along with their "best" oracle target types (according to the scoring function defined by Eq. 8).

The best performing configuration uses *Most specific* Wikipedia types with the strict filtering model (Fig. 5g). As it can be observed in Fig. 7a, most of the queries are improved while none of them are hurt. The queries with the highest ΔNDCG@10 are mostly named entity queries, with a very suitable best oracle target type to be used as strict filter (first block of Table 8).

By changing type representation from *Most specific* to *Top-level*, the performance for Wikipedia types with strict filtering goes from best to worst (Fig. 5g). As we can see in Fig. 7b, the vast majority of queries is negatively affected. Queries that are most harmed are natural language queries like "classis does the Millepede belong to" and "is the residence of the prime minister of Spain" (second block of Table 8). For queries that are improved, we observe that the best oracle target types are high-level and with a large

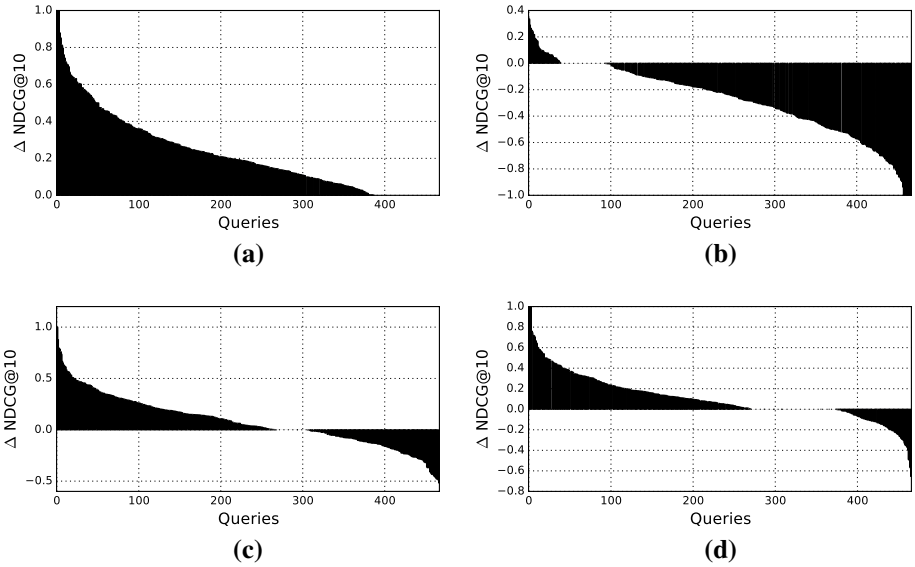

Fig. 7 Differences in NDCG@10 per query between a given type-aware entity retrieval configuration and its corresponding (term-based) baseline, using only entities with types from that type taxonomy (1TT). **a** Strict filtering, most specific Wikipedia types, **b** Strict filtering, top-level Wikipedia types, **c** Interpolation, most specific Wikipedia types and **d** Interpolation, most specific DBpedia types

descendants subtree like "Social sciences," corresponding more to broad knowledge domains that result in more permeable filters.

Let us now observe the results after changing another dimension of the best performing setting: *Most specific* Wikipedia types now with the *interpolation* model (Fig. 5g). Figure 7c shows the distribution of ΔNDCG@10 values. Four out of the 5 most improved queries are named entity queries (third block of Table 8), whereas the most hurt queries belong to diverse categories. The λ_t parameter is set here to 0.65, which means that the type component is given slightly more weight than the term component. Yet, mean performance is much lower for the interpolation model than for strict filtering (0.4474 vs. 0.5874).

Finally, from this last configuration (interpolation with *Most specific* Wikipedia types), we compare against the configuration where the third dimension, type taxonomy, is changed to a smaller, shallower taxonomy like DBpedia (Fig. 5e). The best performing λ_t is set here to 0.55, which represents a balanced interpolation. The ΔNDCG@10 values in Fig. 7d are positive for a large proportion of queries. Around 100 queries are unaffected, while a few are moderately negatively impacted by the type-aware model. It is difficult to identify any patterns here (last block of Table 8), apart from noting that most queries with the largest impact (either positive or negative) are natural language queries. Moreover, given that the DBpedia taxonomy has a depth of 6 levels, its shallowness has a consequence that even *Most specific* types are not specific enough.

Additionally, to answer the question whether the same queries are helped/hurt using different taxonomies, we show in Table 9 the top ten queries according to average Δ NDCG@10 across all four taxonomies. All the queries in this configuration result to have non-negative ΔNDCG@10 in every type taxonomy. We can observe that most of the queries that are helped are named entity queries.

Table 8 Queries with the largest ΔNDCG@10 differences with respect to the term-based baseline, using oracle target types and the 1TT setting

Δ	Query	Best oracle target type(s)
Strict filtering, most specific, Wikipedia		
+ 1.0	SemSearch_ES-129 (“pizza populous Detroit MI”)	American pizza, pizza varieties
+ 1.0	SemSearch_ES-75 (“Sagemont Church Houston TX”)	Lists of churches in the US, Cathedrals in the US
+ 1.0	QALD2_te-53 (“is the ruling party in Lisbon”)	Liberal-conservative parties
+ 1.0	SemSearch_ES-104 (“bourbonnais il”)	Amtrak accidents, railway accidents in Illinois
+ 0.88	SemSearch_ES-96 (“New England Coffee”)	Whitbread divisions and subsidiaries
Strict filtering, top-level, Wikipedia		
+ 0.34	SemSearch_ES-99 (“University of York”)	Geography, social sciences
+ 0.33	INEX_XER-63 (“Hugo awarded best novels”)	Social sciences, arts
+ 0.29	SemSearch_ES-137 (“steak express”)	Social sciences, philosophy
+ 0.26	QALD2_tr-91 (“organizations were founded in 1950”)	Social sciences, society
+ 0.25	INEX_LD-20120131 (“Vietnam travel national park”)	Social sciences, geography
− 1.0	QALD2_te-76 (“List the children of Margaret Thatcher”)	Social sciences, Philosophy
− 1.0	SemSearch_ES-111 (“eagle rock ca”)	Society, history
− 1.0	QALD2_tr-13 (“classis does the Millepede belong to”)	Social sciences, nature
− 1.0	QALD2_te-90 (“is the residence of the prime minister of Spain”)	Arts, philosophy
− 1.0	SemSearch_ES-107 (“concord steel”)	Social sciences, philosophy
Interpolation, most specific, Wikipedia		
+ 1.0	SemSearch_ES-129 (“pizza populous detroit mi”)	American pizza
+ 1.0	QALD2_te-53 (“is the ruling party in Lisbon”)	Liberal-conservative parties
+ 0.88	SemSearch_ES-124 (“Motorola bluetooth hs850”)	Bluetooth, electronics companies of the US
+ 0.8	SemSearch_ES-50 (“laura steele bob and tom”)	American comedy radio programs
+ 0.79	SemSearch_ES-96 (“New England Coffee”)	Whitbread divisions and subsidiaries
− 0.47	SemSearch_ES-22 (“city of Charlotte”)	City councils in the US, years in North Carolina
− 0.48	SemSearch_ES-98 (“University of Texas at Austin”)	Association of American Universities

Table 8 (continued)

Δ	Query	Best oracle target type(s)
− 0.48	QALD2_tr-54 (“was the wife of US president Lincoln”)	First ladies of the US, Lincoln family
− 0.49	TREC_Entity-10 (“Campuses of Indiana University”)	Joint venture schools
− 0.52	QALD2_tr-16 (“the capitals of all countries in Africa”)	Capitals in Africa
Interpolation, most specific, DBpedia		
+ 1.0	SemSearch_ES-129 (“pizza populous detroit mi”)	Food
+ 1.0	QALD2_te-43 (“all breeds of the German Shepherd dog”)	Book
+ 1.0	INEX_LD-2012309 (“residents small island city state Malay Peninsula Chinese”)	Settlement
+ 1.0	QALD2_te-55 (“Greek goddesses dwelt on Mount Olympus”)	Mythological figure
+ 0.76	QALD2_te-42 (“is the husband of Amanda Palmer”)	Writer
− 0.4	QALD2_te-27 (“Sean Parnell is the governor of US state”)	Office holder
− 0.49	QALD2_tr-54 (“was the wife of US president Lincoln”)	Office holder
− 0.5	QALD2_tr-4 (“river does the Brooklyn Bridge cross”)	Bridge
− 0.57	QALD2_te-90 (“is the residence of the prime minister of Spain”)	Building
− 0.65	SemSearch_ES-89 (“University of North Dakota”)	University

Table 9 Queries with the largest average ΔNDCG@10 improvements across all type taxonomies, using most specific types with strict filtering and the 1TT setting

Avg. Δ across taxos.	Query	ΔNDCG@10			
		DBpedia	Free-base	Wiki-pedia	YAGO
+ 1.0	SemSearch_ES-129 ("pizza populous Detroit MI")	*+ 1.0*	*+ 1.0*	*+ 1.0*	*+ 1.0*
+ 0.69	SemSearch_ES-4 ("NAACP Image Awards")	*+ 0.79*	*+ 0.56*	*+ 0.85*	*+ 0.56*
+ 0.68	SemSearch_ES-115 ("goodwill of Michigan")	*+ 0.64*	*+ 0.64*	*+ 0.63*	*+ 0.78*
+ 0.6	INEX_XER-140 ("Airports in Germany")	*+ 0.59*	*+ 0.59*	*+ 0.57*	*+ 0.63*
+ 0.59	QALD2_tr-42 ("are the official languages of the Philippines")	*+ 1.0*	*+ 0.49*	*+ 0.5*	*+ 0.36*
+ 0.59	SemSearch_ES-78 ("sharp pc")	*+ 0.65*	*+ 0.33*	*+ 0.71*	*+ 0.65*
+ 0.58	SemSearch_ES-1 ("44 magnum hunting")	*+ 0.32*	*+ 1.0*	*+ 0.47*	*+ 0.52*
+ 0.58	SemSearch_ES-124 ("motorola bluetooth hs850")	+ 0.03	*+ 0.7*	*+ 0.65*	*+ 0.92*
+ 0.57	SemSearch_ES-50 ("laura steele bob and tom")	*+ 0.49*	*+ 0.67*	*+ 0.8*	*+ 0.32*
+ 0.56	QALD2_te-60 ("a list of all lakes in Denmark")	*+ 0.54*	*+ 0.51*	*+ 0.61*	*+ 0.57*

Differences greater than 0.05 are shown in italic

8 Results using automatic target entity type identification

In the previous section, we have presented results using an idealized setting, where target types were provided by an oracle. We now instead use the methods we introduced Sect. 5 to automatically identify target entity types (Sect. 8.1), and subsequently use these for type-aware entity retrieval (Sect. 8.2). In this part, we focus only on DBpedia, for the following main reason. Both for evaluation and for supervised learning, one needs relevance assessments for target entity types of queries. For the two large taxonomies, human assessments are problematic; Wikipedia is not a proper type taxonomy and is huge, while YAGO does not provide human-readable labels for types. As we have seen in the previous section, Freebase behaves similarly to DBpedia, but it is not particularly interesting in the taxonomical sense, given that it has only two levels. This leaves us with DBpedia. The DBpedia Ontology is small enough to be manageable by humans, and is a proper taxonomy. For the details on the construction of the test collection for target type identification, we refer back to Sect. 6.6.

We note that none of the elements of our our supervised learning approach are specific to this taxonomy, and our methods for target entity type identification can be applied on top of any type taxonomy.

8.1 Target entity type identification

The research question we seek to answer concerns the automatic identification of entity types:

– *RQ4* How can one automatically determine the target entity types of a query from a type taxonomy?

Table 10 Target entity type identification performance, measured in terms of NDCG@1 and NDCG@5

Method	NDCG@1	NDCG@5
EC, BM25 ($k = 10$)	0.1335	0.2657
EC, LM ($k = 10$)	0.1039	0.2625
TC, BM25	0.2305	0.3216
TC, LM	0.2508	0.3757
LTR	**0.4420**	**0.5968**

The best performing method according to each metric is remarked in bold

First, we evaluate target entity type identification intrinsically. We follow (Balog and Neumayer 2012) and approach the task as a ranking problem and report on NDCG at rank positions 1 and 5. Following Garigliotti et al. (2017), the `NIL`-type labels are ignored in our experimental evaluation. For the LTR method, we used 5-fold cross-validation.

Table 10 presents the evaluation results. For each of the underlying retrieval models, BM25 and LM, we select only a single entity-centric (EC) method according to the best performing cut-off K (here, $K = 10$ for both models). We find that our supervised learning (LTR) approach significantly and substantially outperforms all baseline methods (with $p < 0.001$ using a two-tailed paired t test), in particular, achieving an NDCG@5 score of 0.6.

8.1.1 Analysis of LTR features

We analyze the discriminative power of our features, by sorting them according to their information gain, measured in terms of Gini importance. Table 11 presents the resulting features order, with their respective information gains; this is also shown as the vertical bars in Fig. 8. The top 3 features are: $SIMMAX(t, q)$, $SIMAVG(t, q)$, and $SIMAGGR(t, q)$. This underlines the effectiveness of textual similarity, enriched with distributional semantic representations, measured between the query and the type label. Then, we incrementally add features, one by one, according to their importance and report on performance in NDCG@5 metric in Table 11 (and also shown as the line plot in Fig. 8). In each iteration, we set the m parameter of the Random Forests algorithm to 10% of the size of the feature set.

8.2 Type-aware entity retrieval

Next, we turn to extrinsic evaluation of the automatically identified target types, by using them for ad hoc entity retrieval (that is, our end-to-end task):

– *RQ5* How does type-aware entity retrieval perform using automatic target entity type identification, compared to an "oracle" setting?

In this section, we present evaluation results for all combinations of retrieval models, type representation modes, and target entity type identification models. We refer to the latter models simply as *identification models* when discussing the results. We then use the term *configuration* now to refer to a particular combination of retrieval model, type representation, and identification model. Throughout this section, we will focus on the 1TT setting.

Table 11 Performance of our LTR approach, measured by NDCG@5, after incrementally adding features proportional to their information gain, measured by Gini score

Added feature	NDCG@5	Gain
$SIMMAX(t,q)$	0.3672	0.1138
+ $SIMAVG(t,q)$	0.3863	0.1097
+ $SIMAGGR(t,q)$	0.4186	0.1045
+ $ENTITIES(t)$	0.4888	0.0479
+ $EC_{LM,5}$	0.5551	0.0461
+ $EC_{LM,10}$	0.5519	0.0444
+ $EC_{BM25,100}$	0.5555	0.0443
+ $EC_{BM25,50}$	0.5700	0.0417
+ $TC_{BM25}(t,q)$	0.5644	0.0416
+ $EC_{BM25,20}$	0.5757	0.0385
+ $EC_{LM,20}$	0.5661	0.0385
+ $EC_{LM,50}$	0.5782	0.0371
+ $CHILDREN(t)$	0.5905	0.0362
+ $EC_{BM25,10}$	0.5856	0.0340
+ $SIBLINGS(t)$	0.5868	0.0326
+ $EC_{LM,100}$	0.5882	0.0307
+ $IDFSUM(t)$	0.5992	0.0301
+ $EC_{BM25,5}$	0.6003	0.0276
+ $IDFAVG(t)$	0.5924	0.0240
+ $DEPTH(t)$	0.5983	0.0190
+ $JNOUNS(t,q)$	0.5898	0.0185
+ $JTERMS_1(t,q)$	0.5944	0.0178
+ $LENGTH(t)$	0.5967	0.0126
+ $TC_{LM}(t,q)$	0.6001	0.0078
+ $JTERMS_2(t,q)$	0.5980	0.0008

Table 12 shows the results for all configurations. We also present results using the target type labels provided by the human assessors as target entity types, referred as *Oracle 2*. Additionally, we report on our original oracle as well (which uses the type assignments of relevant results), referred to as *Oracle 1*. Again, our main evaluation metric is NDCG@10, and we also report on NDCG@100. The NDCG@10 scores are also plotted in Fig. 9 for an easier visual inspection, with the red line corresponding to the term-based baseline.

For the strict filtering model, we introduce a cut-off parameter k. This parameter controls how many of the highest ranked types we consider as target types. Clearly, that the larger the cut-off value k, the more lenient the filtering gets. Hence, we perform a sweep over the possible cut-offs $k \in \{5, 10, 15, \ldots, 100\}$ to calculate $P(q_t|e)$, and use the best performing setting when comparing against other approaches.

8.2.1 Type representation

We now revisit our second research question (RQ2) about type representation, and answer it for automatically identified target entity types. We observe that for all methods, *Along path* and *Top-level* representations provide better performance compared to the Most specific representation. The difference between these representations is small, which is similar

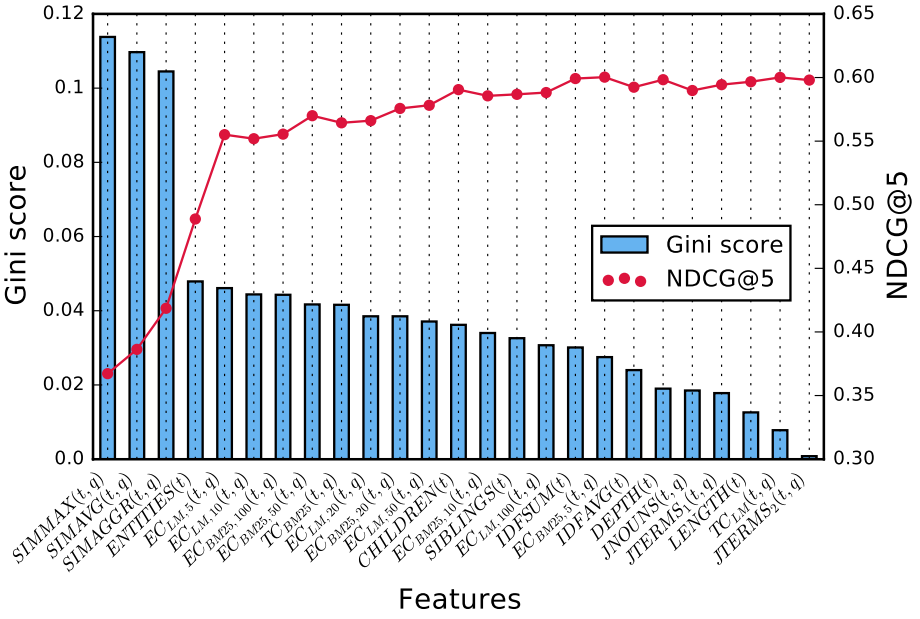

Fig. 8 Performance of our LTR approach, measured by NDCG@5, when incrementally adding features according to their information gain, measured by Gini score

to our observations using oracle types (cf. Sect. 7.2). The *Most specific* representation brings significant improvements only for the LTR method, which is the top performing method in Table 10. Overall, our results show that when target entity types are identified automatically, using hierarchical relationships from ancestor types is the most effective way of representing entity type information; keeping only the most specific types is helpful only when an accurate target entity type identification method is employed.

8.2.2 Type-aware entity retrieval

Revisiting our third research question (RQ3) about type-aware retrieval model, we observe that strict filtering achieves the best performance among all configurations, and the best results are obtained when it is combined with LTR identification model. Specifically, by using this retrieval model with the Along pathrepresentation, we can outperform the text-only baseline by 44% in terms of NDCG@10. The soft filtering and interpolation models perform best when used with *Top-level* representation, significantly outperforming the baseline for LTR model. Similar to the oracle setting for 1TT (Table 7), the λ_t type weight is the highest for *Top-level* representation, showing that the type component contribution to the interpolation model is high in this configuration.

8.2.3 Target entity type identification method

The top row of Fig. 10 shows the performance of interpolation model, when varying the value of λ_t parameter. We observe that assigning high weights (i.e., > 0.2) to type-based information is harmful for the *Most specific* and *Along path* representations. On the other

Table 12 Entity retrieval performance using automatically identified target types from DBpedia. λ_t and k are the best empirically found interpolation and strict filtering parameters, respectively. For reference comparison, the results using human annotated target type collection ("Oracle 2") and the original "Oracle 1"(cf. Sect. 6.2) are also included. Performance is measured in terms of NDCG@10 and NDCG@100

Model	Strict filtering			Soft filtering		Interpolation		
	@10	@100	k	@10	@100	@10	@100	λ_t
Baseline (Metzler and Croft 2005)	0.3036	0.4119	–	0.3036	0.4119	0.3036	0.4119	–
Along path								
EC, BM25	0.3501‡	0.4048	65	**0.3210**†	**0.4154**	**0.3214**†	**0.4204**‡	0.30
TC, LM	0.4304‡	0.5043‡	20	0.2678	0.3805	0.3066	0.4116	0.10
LTR	**0.4378**‡	**0.5151**‡	20	0.2988	0.4029	0.3126†	0.4150	0.15
Oracle 2	0.4245‡	0.4797‡	–	0.3638	0.4406‡	0.3587‡	0.4384‡	0.40
Oracle 1	0.4600‡	0.5079‡	–	0.4353‡	0.4894‡	0.4172‡	0.4746‡	0.65
Top-level								
EC, BM25	0.3501‡	0.4048	65	0.3348†	0.4251†	0.3304	0.4226‡	0.50
TC, LM	0.4295‡	0.5038‡	65	0.3254	0.4159	0.3313	0.4228	0.55
LTR	**0.4371**‡	**0.5157**‡	5	**0.3705**‡	**0.4453**‡	**0.3748**‡	**0.4460**‡	0.80
Oracle 2	0.4245‡	0.4797‡	–	0.3791‡	0.4477‡	0.3732‡	0.4430‡	0.60
Oracle 1	0.4600‡	0.5079‡	–	0.4196‡	0.4779‡	0.4141‡	0.4725‡	0.70
Most specific								
EC, BM25	0.3075	0.3350	65	**0.3048**	**0.4038**	0.3119	**0.4159**	0.15
TC, LM	0.3078	0.3352	65	0.2274	0.3500	0.3036	0.4119	0.00
LTR	**0.3880**‡	**0.4367**†	45	0.2997	0.3980	**0.3161**	**0.4159**	0.20
Oracle 2	0.3064	0.4009	–	0.2792	0.3809	0.3185†	0.4176	0.15
Oracle 1	0.5092‡	0.5393‡	–	0.4309‡	0.4864‡	0.4075‡	0.4696‡	0.55

For a given type representation mode and a given retrieval model, the identification model which leads to the best type-aware entity retrieval performance according to each metric is remarked in bold

Statistical significance, tested using a two-tailed paired t test at $p < 0.05$ and $p < 0.001$, is denoted by † and ‡, respectively

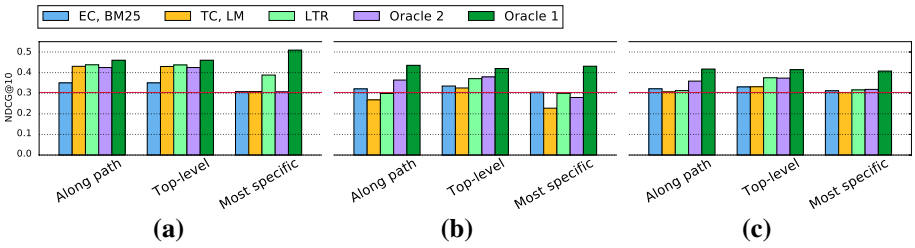

Fig. 9 Entity retrieval performance for all combinations of retrieval models and type representation modes, using automatically identified target types from DBpedia. The red line corresponds to the term-based baseline. Performance is measured by NDCG@10. **a** Strict filtering, **b** soft filtering and **c** interpolation (Color figure online)

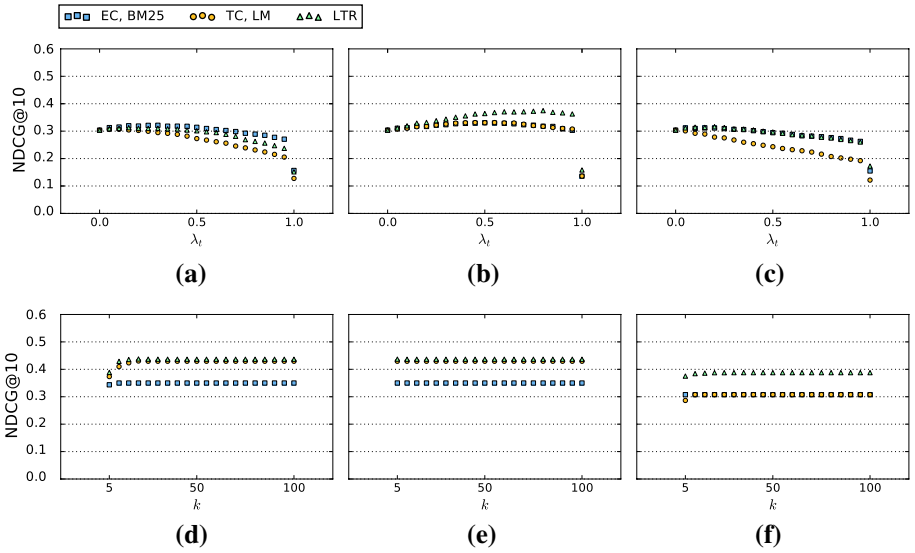

Fig. 10 Retrieval performance using automatically identified target types from DBpedia. Top: interpolation model with different type weights, λ_t; bottom: strict filtering model with different ranking cutoffs k. The leftmost data points ($\lambda_t = 0$) correspond to the term-based baseline. **a** Types along path, **b** Top-level type(s), **c** Most specific type(s), **d** Types along path, **e** Top-level type(s) and **f** Most specific type(s)

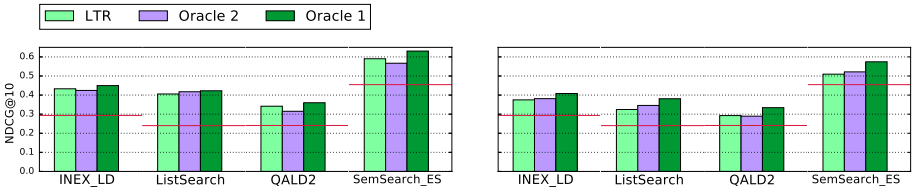

Fig. 11 Entity retrieval performance per query category (cf. Table 4) for the strict filtering (left) and soft filtering (right) retrieval models, using automatically identified top-level target types from DBpedia. The red line corresponds to the term-based baseline. Performance is measured by NDCG@10 (Color figure online)

hand, when using *Top-level* representation, the retrieval performance improves until it reaches a peak (ranging between 0.5 and 0.8), and then it gradually decreases.

The bottom row of Fig. 10 presents the retrieval performance while varying the cut-off value k. We find that retrieval performance for the soft filtering model plateaus quickly by increasing the number of ranked types, especially when using the LTR model. This verifies that an effective target entity type identification method, returning relevant types at top ranks, can bring considerable retrieval improvements using the strict filtering retrieval model.

In order to better understand the effects of automatic target type identification, we break down entity retrieval results into the four query categories present in the DBpedia-Entity v2 collection (cf. Table 4). Figure 11 shows retrieval performance per query category for each of the two filtering retrieval models using top-level DBpedia types. We find that type-aware retrieval using the LTR method significantly and consistently outperforms the term-based baseline for all query categories. Moreover, target types identified by our LTR

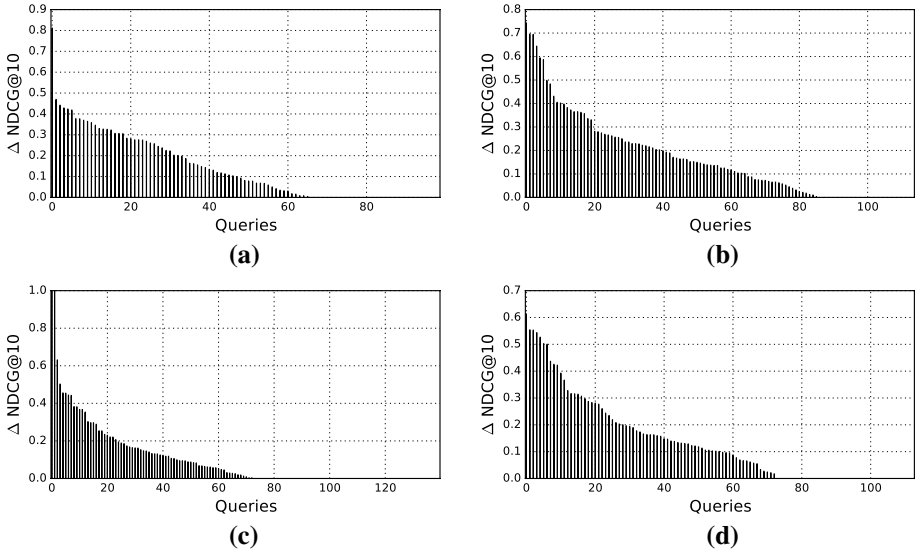

Fig. 12 Differences in NDCG@10 per query between type-aware entity retrieval and its corresponding (term-based) baseline, using the strict filtering model with top-level DBpedia target types automatically detected by LTR, grouped by query categories (cf. Table 4). **a** INEX_LD, **b** ListSearch, **c** QALD2 and **d** SemSearch_ES

method yield better performance than human type annotations, for 3 out of the 4 categories using strict filtering, and for QALD2 (natural language) queries in case of soft filtering.

Furthermore, we conduct an analysis at the query level for each of these query categories. It can be seen in Fig. 12 that every category exhibits a similar distribution, with all ΔNDCG@10 values being positive. Specially, the ListSearch category has the largest proportion of improved queries as well as the largest differences. Finally, Table 13 lists queries with the largest ΔNDCG@10 differences, either positive or negative, along with their respective types using automatic detection (LTR) and human annotators (Oracle 2). Comparing with the top-level ancestor(s) of the target types provided by the human oracle, it is clear that in most of the cases the set of LTR-detected query types contains one or more types than the oracle. The additional type(s) relax the filtering criterion by allowing some non-relevant entities to be kept on the ranking and then hurting the performance. Particularly, when the additional type is Agent, the one with largest coverage in the knowledge base, it potentially introduces many entities including all persons and organizations. In the other hand, the positively affected queries result to have more target types detected by LTR. The more lenient filtering given by the additional type(s) allows to retain relevant entities, then outperforming the contributions of the human oracle target types. The QALD2 category exhibits the distribution of ΔNDCG@10 differences where LTR-detected target types perform the best in comparison with the human oracle. This fact is in line with our observations in Sect. 8.1, since the most important LTR features, the continuous semantic representations, are more effective in these longer, natural language queries.

Table 13 Queries with the largest ΔNDCG@10 differences, using the strict filtering model with top-level DBpedia target types automatically detected by the LTR method, in comparison with target types from human annotators (Oracle 2), grouped per query category. Respective best target type(s), assigned by both the automatic detector and the human annotators, are also shown

Δ	Query	Best target type(s)	
		LTR	Oracle 2
Strict filtering, top-level, INEX_LD			
+ 1.0	INEX_LD-2012335 ("US president authorise nuclear weapons against Japan")	Agent, Device, Event	Agent
+ 0.55	INEX_LD-20120312 ("tango culture countries")	Place, Topical Concept	Place
+ 0.5	INEX_LD-20120322 ("tango music instruments")	Topical Concept, Agent, Work	Work
+ 0.38	INEX_LD-2010057 ("Einstein Relativity theory")	Agent, Work, Place	Work
+ 0.32	INEX_LD-20120122 ("Vietnamese food blog")	Food, Work, Agent	Work
− 0.18	INEX_LD-2012369 ("most famous civic military airports")	Place, Agent, Work	Place
− 0.19	INEX_LD-2012347 ("seat Florida country Dade")	Place, Agent	Place
− 0.23	INEX_LD-20120311 ("tango culture movies")	Work, Agent, Topical Concept	Work
− 0.4	INEX_LD-2010043 ("List of films from the surrealist category")	Work, Agent	Work
− 0.57	INEX_LD-2010004 ("Indian food")	Food, Agent, Place	Food
Strict filtering, top-level, ListSearch			
+ 0.52	INEX_XER-96 ("Pure object oriented programing languages")	Work, Agent	–
+ 0.35	SemSearch_LS-7 ("Branches of the US military")	Agent, Place	Place
+ 0.32	INEX_XER-99 ("Computer systems that have a recursive acronym for the name")	Work, Device, Agent	Device
+ 0.29	INEX_XER-125 ("countries have won the FIFA world cup")	Event, Place, Agent	Place
+ 0.26	INEX_XER-121 ("US presidents since 1960")	Agent, Event	Agent
− 0.27	SemSearch_LS-13 ("five great epics of Tamil literature")	Work, Agent, Topical Concept	Work
− 0.28	SemSearch_LS-49 ("invented the python programming language")	Work, Agent, Device	Agent
− 0.41	SemSearch_LS-3 ("astronauts landed on the Moon")	Agent, Place	Agent
− 0.63	SemSearch_LS-14 ("gods dwelt on Mount Olympus")	Agent, Place	Agent
− 0.64	TREC_Entity-5 ("Products of Medimmune, Inc")	Agent, Unit Of Work, Chemical Substance	Chemical Substance

Table 13 (continued)

Δ	Query	Best target type(s)	
		LTR	Oracle 2
Strict filtering, top-level, QALD2			
+ 1.0	QALD2_tr-42 (“are the official languages of the Philippines”)	Place, Agent	–
+ 0.7	QALD2_te-98 (“country does the creator of Miffy come from”)	Place, Agent, Work	Place
+ 0.63	QALD2_te-43 (“all breeds of the German Shepherd dog”)	Agent, Species	Species
+ 0.63	QALD2_te-89 (“In city was the former Dutch queen Juliana buried”)	Agent, Species	Place
+ 0.63	QALD2_te-97 (“painted The Storm on the Sea of Galilee”)	Place, Work, Agent	Agent
− 0.18	QALD2_te-17 (“all cars that are produced in Germany”)	Mean Of Transportation, Work, Agent	Mean Of Transportation
− 0.18	QALD2_tr-24 (“mountain is the highest after the Annapurna”)	Place, Agent	Place
− 0.22	QALD2_te-100 (“produces Orangina”)	Agent, Food, Work	Agent
− 0.28	QALD2_te-63 (“all Argentine films”)	Work, Agent	Work
− 0.33	QALD2_te-40 (“List all boardgames by GMT”)	Agent, Activity, Event	Activity
Strict filtering, top-level, SemSearch_ES			
+ 1.0	SemSearch_ES-3 (“Bookwork”)	Agent, Place	–
+ 0.74	SemSearch_ES-26 (“Disney Orlando”)	Place, Agent	Place
+ 0.42	SemSearch_ES-41 (“Joan of Arc”)	Agent, Place, Work	Agent
+ 0.39	SemSearch_ES-18 (“Canasta cards”)	Activity, Work	Activity
+ 0.34	SemSearch_ES-80 (“Sonny and Cher”)	Work, Agent	Agent
− 0.37	SemSearch_ES-67 (“Ovguide movies”)	Work, Agent	Work
− 0.39	SemSearch_ES-79 (“Shobana masala”)	Agent, Work, Food	Agent
− 0.41	SemSearch_ES-125 (“Nokia e73”)	Device, Place, Work	Device
− 0.43	SemSearch_ES-58 (“Mason Ohio”)	Agent, Place	Place
− 0.48	SemSearch_ES-4 (“NAACP Image Awards”)	Award, Agent, Work	Award

9 Conclusions

In this paper we have furthered our understanding on the usage of target type information for entity retrieval over structured data sources. A main contribution of this work is the systematic comparison of four well-known type taxonomies (DBpedia, Freebase, Wikipedia, and YAGO) across three dimensions of interest: the representation of hierarchical entity type information, the way to combine term-based and type-based information, and the impact of choosing a particular type taxonomy. Using an idealized "oracle" setting for identifying the target entity types, we have found that type information can significantly and substantially improve a strong text-only entity retrieval baseline.

For realistic scenarios, where the target entity types are not provided, we have developed methods for identifying target entity types automatically. Our experiments have shown that using automatic target entity types not only improves entity retrieval performance, but also brings the same, or even higher, level of performance achievable by human target type annotations.

We identify the following directions for future work. First, we plan to detect `NIL`-types for queries that cannot be assigned any type. Second, we observed that performance is hindered by missing entity type assignments. We aim to address this issue by automatically identifying (missing) entity types in knowledge bases. Third, when addressing the task of automatic target type identification, our analysis has suggested that using a subset of the full feature set may be sufficient (cf. Fig. 8). As a follow-up, we wish to investigate whether the same observation would also hold when using the detected types for the end-to-end entity ranking task. Fourth, as we have noted in Sect. 3.2, the particular term-based component we have employed as our baseline is not the focus of this work, and any other approach could be plugged in instead. Another line of further investigation is then to carry out evaluations analogous to the ones conducted here, but utilizing a different baseline (possibly a non-text-only method, e.g., Hulpuş et al. 2015; Hasibi et al. 2016). Finally, we plan to investigate continuous representations of text and type information, and their integration (e.g., recent work on mixture of text and structural graph embeddings Wang et al. 2017; Subramanian and Chakrabarti 2018; Jain et al. 2018), for entity retrieval.

References

Balog, K. (2018). *Entity-oriented search* (Vol. 39)., The information retrieval series Berlin: Springer.

Balog, K., Bron, M., & De Rijke, M. (2011). Query modeling for entity search based on terms, categories, and examples. *ACM Transactions on Information Systems*, *29*(4), 22:1–22:31.

Balog, K., de Vries, A .P., Serdyukov, P., Thomas, P., & Westerveld, T. (2010) Overview of the TREC 2009 entity track. In *Proceedings of the twentieth text retrieval conference, TREC '10*.

Balog, K., & Neumayer, R. (2012). Hierarchical target type identification for entity-oriented queries. In *Proceedings of the 21st ACM international conference on information and knowledge management, CIKM '12* (pp. 2391–2394).

Balog, K., & Neumayer, R. (2013). A test collection for entity search in DBpedia. In *Proceedings of the 36th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 737–740).

Balog, K., Serdyukov, P., & De Vries, A. P. (2012) Overview of the TREC 2011 entity track. In *Proceedings of the twentieth text retrieval conference, TREC '11*.

Bron, M., Balog, K., & de Rijke, M. (2010). Ranking related entities: Components and analyses. In *Proceedings of the 19th ACM conference on information and knowledge management, CIKM '10* (pp. 1079–1088).

Demartini, G., Firan, C. S., & Iofciu, T. (2008) L3S at INEX 2007. In *Focused access to XML documents, 7th international workshop of the initiative for the evaluation of XML retrieval, INEX '08* (pp. 252–263).

Demartini, G., Firan, C. S., Iofciu, T., Krestel, R., & Nejdl, W. (2010a). Why finding entities in Wikipedia is difficult, sometimes. *Information Retrieval*, *13*(5), 534–567.

Demartini, G., Iofciu, T., & De Vries, A. P. (2010b) Overview of the INEX 2009 entity ranking track. In *Focused retrieval and evaluation, 8th international workshop of the initiative for the evaluation of XML retrieval, revised and selected papers, INEX '09* (pp. 254–264).

Fleischman, M., & Hovy, E. (2002). Fine grained classification of named entities. In *Proceedings of the 15th international conference on computational linguistics, COLING '02* (pp. 1–7).

Fossati, M., Kontokostas, D., & Lehmann, J. (2015). Unsupervised learning of an extensive and usable taxonomy for DBpedia. In *Proceedings of the 11th international conference on semantic systems, SEMANTICS '15* (pp. 177–184).

Gangemi, A., Nuzzolese, A. G., Presutti, V., Draicchio, F., Musetti, A., & Ciancarini, P. (2012). Automatic typing of DBpedia entities. In *Proceedings of the semantic web: 11th international semantic web conference, ISWC '12* (pp. 65–81).

Garigliotti, D., & Balog, K. (2017). On type-aware entity retrieval. In *Proceedings of the ACM SIGIR international conference on theory of information retrieval, ICTIR '17* (pp. 27–34). ACM.

Garigliotti, D., Hasibi, F., & Balog, K. (2017). Target type identification for entity-bearing queries. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, SIGIR '17* (pp. 845–848).

Giuliano, C. (2009) Fine-grained classification of named entities exploiting latent semantic kernels. In *Proceedings of the thirteenth conference on computational natural language learning, CoNLL '09* (pp. 201–209).

Hasibi, F., Balog, K., & Bratsberg, S. E. (2016). Exploiting entity linking in queries for entity retrieval. In *Proceedings of the ACM SIGIR international conference on theory of information retrieval, ICTIR '16* (pp. 209–218).

Hasibi, F., Nikolaev, F., Xiong, C., Balog, K., Bratsberg, S. E., Kotov, A., & Callan, J. (2017). DBpedia-entity v2: A test collection for entity search. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, SIGIR '17* (pp. 1265–1268).

Hulpuş, I., Prangnawarat, N., & Hayes, C. (2015). Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In *Proceedings of the 14th international conference on the semantic web—Volume 9366, ISWC '15* (pp. 442–457).

Jain, P., Kumar, P., Mausam, & Chakrabarti, S. (2018). Type-sensitive knowledge base inference without explicit type supervision. In *Proceedings of the 56th annual meeting of the association for computational linguistics, ACL '18* (pp. 75–80).

Jämsen, J., Näppilä, T., & Arvola, P. (2008) Entity ranking based on category expansion. In *Focused access to XML documents, 7th international workshop of the initiative for the evaluation of XML retrieval, INEX '08* (pp. 264–278).

Kaptein, R., & Kamps, J. (2009). Finding entities in Wikipedia using links and categories. In *Advances in focused retrieval, 7th international workshop of the initiative for the evaluation of XML retrieval, INEX '09* (pp. 273–279).

Kaptein, R., & Kamps, J. (2013). Exploiting the category structure of Wikipedia for entity ranking. *Artificial Intelligence*, *194*, 111–129.

Kaptein, R., Serdyukov, P., De Vries, A. P., & Kamps, J. (2010). Entity ranking using Wikipedia as a pivot. In *Proceedings of the 19th ACM conference on information and knowledge management, CIKM '10* (pp. 69–78).

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., et al. (2015). DBpedia: A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, *6*(2), 167–195.

Lin, T., Mausam, & Etzioni, O. (2012). No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, EMNLP-CoNLL '12* (pp. 893–903).

Ling, X., & Weld, D. S. (2012). Fine-grained entity recognition. In *Proceedings of the thirty-second AAAI conference on artificial intelligence, AAAI '12* (pp. 94–100).

Lopez, V., Unger, C., Cimiano, P., & Motta, E. (2013). Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, *21*, 3–13.

Metzler, D., & Croft, W. B. (2005). A Markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '05* (pp. 472–479). ACM.

Mika, P. (2013). Entity search on the web. In *Proceedings of the 22nd international world wide web conference, WWW '13* (pp. 1231–1232).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems 26: Proceedings of the 27th annual conference on neural information processing systems, NIPS '13* (pp. 3111–3119).

Nakashole, N., Tylenda, T., & Weikum, G. (2013). Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st annual meeting of the association for computational linguistics, volume 1: Long papers, ACL '13* (pp. 1488–1497).

Neumayer, R., Balog, K., & Nørvåg, K. (2012). On the modeling of entities for ad-hoc entity search in the web of data. In *Advances in information retrieval: Proceedings of the 34th European conference on IR research, ECIR '12* (pp. 133–145).

Nuzzolese, A. G., Gangemi, A., Presutti, V., & Ciancarini, P. (2012). Type inference through the analysis of Wikipedia links. In *Proceedings of workshop on linked data on the web (LDOW), WWW '12*.

Pehcevski, J., Thom, J. A., Vercoustre, A. M., & Naumovski, V. (2010). Entity ranking in Wikipedia: Utilising categories, links and topic difficulty prediction. *Information Retrieval*, *13*(5), 568–600.

Pound, J., Mika, P., & Zaragoza, H. (2010). Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international world wide web conference, WWW '10* (pp. 771–780).

Rahman, A., & Ng, V. (2010). Inducing fine-grained semantic classes via hierarchical and collective classification. In *Proceedings of the 23rd international conference on computational linguistics, COLING '10* (pp. 931–939).

Raviv, H., Carmel, D., & Kurland, O. (2012). A ranking framework for entity oriented search using Markov random fields. In *Proceedings of the 1st joint international workshop on entity-oriented and semantic search, JIWES '12* (pp. 1:1–1:6).

Sawant, U., & Chakrabarti, S. (2013). Learning joint query interpretation and response ranking. In *Proceedings of the 22nd international world wide web conference, WWW '13* (pp. 1099–1109).

Subramanian, S., & Chakrabarti, S. (2018). New embedded representations and evaluation protocols for inferring transitive relations. In *The 41st international ACM SIGIR conference on research & development in information retrieval, SIGIR '18* (pp. 1037–1040).

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). YAGO: A core of semantic knowledge. In *Proceedings of the 16th international world wide web conference, WWW '07* (pp. 697–706).

Tonon, A., Catasta, M., Demartini, G., Cudré-Mauroux, P., & Aberer, K. (2013). TRank: Ranking entity types using the web of data. In *Proceedings of the semantic web: 11th international semantic web conference, Part I, ISWC '13* (pp. 640–656).

Tonon, A., Catasta, M., Prokofyev, R., Demartini, G., Aberer, K., & Cudré-Mauroux, P. (2016). Contextualized ranking of entity types based on knowledge graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, *37–38*, 170–183.

Vallet, D., & Zaragoza, H. (2008). Inferring the most important types of a query: A semantic approach. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 857–858).

Vercoustre, A. M., Pehcevski, J., & Thom, J. A. (2008). Using Wikipedia categories and links in entity ranking. In *Focused access to XML documents, 6th international workshop of the initiative for the evaluation of XML retrieval, INEX '07* (pp. 321–335).

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(12), 2724–2743.

Weerkamp, W., Balog, K., & Meij, E. J. (2009). A generative language modeling approach for ranking entities. In *Advances in focused retrieval, 7th international workshop of the initiative for the evaluation of XML retrieval, INEX '09* (pp. 292–299).

Yosef, M. A., Bauer, S., Spaniol, J. H. M., & Weikum, G. (2012). HYENA: Hierarchical type classification for entity names. In *Proceedings of the 25th international conference on computational linguistics, COLING '12* (pp. 1361–1370).

Zhang, S., & Balog, K. (2017). Design patterns for fusion-based object retrieval. In *Advances in information retrieval: Proceedings of the 39th European conference on IR research, ECIR '17* (pp. 684–690).

Zhu, J., Song, D., & Rüger, S. (2008). Integrating document features for entity ranking. In *Focused access to XML documents, 7th international workshop of the initiative for the evaluation of XML retrieval, INEX '08* (pp 336–347).